



WebApp Secure

Administrator Guide

Release

5.5



Published: 2014-06-27

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Copyright © 2014, Juniper Networks, Inc. All rights reserved.

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

WebApp Secure [Administrator Guide]

Copyright © 2014, Juniper Networks, Inc.
All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

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

Table of Contents

	About the Documentation	xi
	Documentation and Release Notes	xi
	Documentation Conventions	xi
	Documentation Feedback	xiii
	Requesting Technical Support	xiv
	Self-Help Online Tools and Resources	xiv
	Opening a Case with JTAC	xiv
Part 1	WebApp Secure Administration	
Chapter 1	Overview	3
	WebApp Secure Overview	3
	Methodology	4
	Features and Benefits	4
	Key Components	6
	Anatomy and Flow of an HTTP Request / Response	8
	Limitations	10
Chapter 2	Deployment	11
	Appliance Deployment Overview	11
	Placement Between Firewall and Web Servers	11
	Options for Load-Balanced Environments	13
	SSL Traffic Considerations	13
	EC2 Deployment	14
	Deploying Using the Command Line	14
	Deploying Using the Web Interface	15
	Assigning the Instance and IP Using the CLI	19
	Assigning the Instance and IP Using the Web Interface	19
	Verify the Instance is Running	20
	Clustering Overview	20
	High Availability Overview	21
Chapter 3	Installing the Appliance	23
	WebApp Secure Appliance Terminology	23
	First Time Configuration	24
	Initial Appliance Configuration	24
	Changing the Password	25
	Resetting the Password	25
	Configure Network Interfaces	26
	Set the Hostname	27

	Set DNS	27
	Initializing the System	27
	Verify Connectivity	28
	Install the License	29
	Configuring High Availability	30
	High Availability Settings	32
	Configuring Clustering	34
	Performing Initial Updates	35
	Updating the Cluster	35
	About the Configuration Wizard	35
	Using the Configuration Wizard	36
	Using WebApp Secure with Third-Party Load Balancer	41
	Verify the Installation	42
Chapter 4	Configuring WebApp Secure	43
	Web Interface Configuration Overview	44
	Edit Web UI User Preferences	44
	View Online Help and Product Documentation from the Web UI	45
	Basic Configuration Mode	46
	Expert Configuration Mode	47
	Import/Export (Web UI)	48
	Security Engine Configuration	48
	Configure Support for Akamai Dynamic Site Accelerator	49
	Security Engine Incident Monitoring	50
	Security Engine Server Identity and Cloaking	52
	Security Engine Traffic	52
	Security Engine Whitelist Settings	53
	Proxy/Backends	54
	Applications Overview	55
	Create a New Application	56
	Edit Applications	58
	Application Patterns	58
	Application Backend Overrides	60
	Enable SSL to the Client	60
	Pages	62
	NTP Service	62
	Alert Service	63
	Integration with SRX Series Overview	63
	Filters and Terms Configuration Summary for SRX Series Integration	64
	Creating SRX Series Filters and Terms	65
	Configure the SRX Series Integration	66
	Testing the SRX Series Integration Configuration	68
	Role-Based Administrator Access Control	69
	Configuring Role-Based Access Control	70
Chapter 5	Managing the Appliance	73
	Overview	74
	Navigating the CLI	74
	The CLI: The Set Command	75
	The CLI: General and Base Commands	77

	The CLI: Configuration Level Commands	81
	The CLI: System Level Commands	84
	CLI: Config Example	86
	CLI: Config: Setting a Configuration Parameter	87
	CLI: Config: Initializing the Configuration	88
	CLI: Config: Import/Export	88
	CLI: Config: Configure a Proxy Exclusion	89
	System Updates	90
	Statistics	92
	High Availability Network Failure Detection, Actions, and Monitoring	95
	Unblock Web UI Login Ban	97
	Health Check URL	97
	Self-Monitoring	98
	Self-Monitoring Configuration Variables	98
	Managing and Viewing Logs	103
	Log File Destination	104
	Backup and Recovery Overview	105
	Database Backup and Restore	107
	About Security Intelligence	107
	Enable the Spotlight Connector Service	109
	Spotlight Connector Session Cookies and Locations	110
	About Spotlight Secure	112
	Enable Spotlight Secure	113
Chapter 6	The Processors	117
	Processors Overview	122
	Complexity Rating Definitions	122
	Session Cookie Spoofing	123
	Session Cookie Tampering	123
	Hostname Spoofing Attempt	124
	Security Processors	124
	HoneyPot Processors: Access Policy Processor	125
	HoneyPot Processors: Access Policy Processor: Incidents - Malicious Service Call	126
	HoneyPot Processors: Access Policy Processor: Incidents - Service Directory Indexing	126
	HoneyPot Processors: Access Policy Processor: Incidents - Service Directory Spider	127
	HoneyPot Processors: AJAX Processor	128
	HoneyPot Processors: AJAX Processor: Incidents - Malicious Script Execution	129
	HoneyPot Processors: AJAX Processor: Incidents - Malicious Script Introspection	130
	HoneyPot Processors: Basic Authentication Processor	130
	HoneyPot Processors: Basic Authentication Processor: Incidents - Apache Configuration Requested	132
	HoneyPot Processors: Basic Authentication Processor: Incidents - Apache Password File Requested	133
	HoneyPot Processors: Basic Authentication Processor: Incidents - Invalid Credentials	133

HoneyPot Processors: Basic Authentication Processor: Incidents - Protected Resource Requested	134
HoneyPot Processors: Basic Authentication Processor: Incidents - Password Cracked	135
HoneyPot Processors: Basic Authentication Processor: Incidents - Basic Authentication Brute Force	136
HoneyPot Processors: Cookie Processor	137
HoneyPot Processors: Cookie Processor: Incident - Cookie Parameter Manipulation	138
HoneyPot Processors: File Processor	139
HoneyPot Processors: File Processor: Incident - Suspicious Filename	139
HoneyPot Processors: File Processor: Incident - Suspicious File Exposed	140
HoneyPot Processors: File Processor: Incident - Suspicious Resource Enumeration	141
HoneyPot Processors: Hidden Input Form Processor	141
HoneyPot Processors: Hidden Input Form Processor: Incident - Hidden Parameter Manipulation	142
HoneyPot Processors: Hidden Input Form Processor: Incident - Parameter Type Manipulation	143
HoneyPot Processors: Hidden Link Processor	144
HoneyPot Processors: Hidden Link Processor: Incident - Link Directory Indexing	145
HoneyPot Processors: Hidden Link Processor: Incident - Link Directory Spidering	145
HoneyPot Processors: Hidden Link Processor: Incident - Malicious Resource Request	146
HoneyPot Processors: Query String Processor	146
HoneyPot Processors: Query String Processor: Incident - Query Parameter Manipulation	147
HoneyPot Processors: Robots Processor	148
HoneyPot Processors: Robot Processor: Incident - Malicious Spider Activity	148
Activity Processors	149
Activity Processors: Custom Authentication Processor: Incident - Auth Input Parameter Tampering	150
Activity Processors: Custom Authentication Processor: Incident - Auth Query Parameter Tampering	151
Activity Processors: Custom Authentication Processor: Incident - Auth Cookie Tampering	151
Activity Processors: Custom Authentication Processor: Incident - Authentication Brute Force	152
Activity Processors: Custom Authentication Processor: Incident - Auth Invalid Login	152
Activity Processors: Cookie Protection Processor	153
Activity Processors: Cookie Protection Processor: Incident - Application Cookie Manipulation	154
Activity Processors: Error Processor	154
Activity Processors: Error Processor: Incident - Illegal Response Status	159
Activity Processors: Error Processor: Incident - Suspicious Response Status	160
Activity Processors: Error Processor: Incident - Unexpected Response Status	160

Activity Processors: Error Processor: Incident - Unknown Common Directory Requested	161
Activity Processors: Error Processor: Incident - Unknown User Directory Requested	161
Activity Processors: Error Processor: Incident - Common Directory Enumeration	162
Activity Processors: Error Processor: Incident - User Directory Enumeration	162
Activity Processors: Error Processor: Incident - Resource Enumeration	163
Activity Processors: Header Processor	164
Activity Processors: Header Processor: Incident - Duplicate Request Header	165
Activity Processors: Header Processor: Incident - Duplicate Response Header	166
Activity Processors: Header Processor: Incident - Illegal Request Header	166
Activity Processors: Header Processor: Incident - Illegal Response Header	167
Activity Processors: Header Processor: Incident - Missing All Headers	167
Activity Processors: Header Processor: Incident - Missing Host Header	168
Activity Processors: Header Processor: Incident - Missing Request Header	168
Activity Processors: Header Processor: Incident - Missing Response Header	169
Activity Processors: Header Processor: Incident - Missing User Agent Header	169
Activity Processors: Header Processor: Incident - Request Header Overflow	169
Activity Processors: Header Processor: Incident - Unexpected Request Header	170
Activity Processors: Method Processor	170
Activity Processors: Method Processor: Incident - Illegal Method Requested	171
Activity Processors: Method Processor: Incident - Unexpected Method Requested	172
Activity Processors: Method Processor: Incident - Missing HTTP Protocol	173
Activity Processors: Method Processor: Incident - Unknown HTTP Protocol	173
Tracking Processors: Etag Beacon Processor	174
Tracking Processors: Etag Beacon Processor: Incident - Session Etag Spoofing	174
Tracking Processors: Client Beacon Processor	175
Tracking Processors: Client Beacon Processor: Incident - Beacon Parameter Tampering	176
Tracking Processors: Client Beacon Processor: Incident - Beacon Session Tampering	177
Tracking Processors: Client Fingerprint Processor	177
Tracking Processors: Client Fingerprint Processor: Incident - Fingerprint Directory Indexing	180
Tracking Processors: Client Fingerprint Processor: Incident - Fingerprint Directory Probing	181
Tracking Processors: Client Fingerprint Processor: Incident - Fingerprint Manipulation	181
Tracking Processors: Client Classification Processor	182
Response Processors	184
Response Processors: Block Processor	185
Response Processors: Request Captcha Processor	186
Response Processors: Request Captcha Processor: Incident - Captcha Answer Automation	189

Response Processors: Request Captcha Processor: Incident - No Captcha Answer Provided	190
Response Processors: Request Captcha Processor: Incident - Multiple Captcha Request Overflow	190
Response Processors: Request Captcha Processor: Incident - Unsupported Audio Captcha Requested	191
Response Processors: Request Captcha Processor: Incident - Bad Captcha Answer	192
Response Processors: Request Captcha Processor: Incident - Mismatched Captcha Session	193
Response Processors: Request Captcha Processor: Incident - Expired Captcha Request	193
Response Processors: Request Captcha Processor: Incident - Captcha Request Tampering	194
Response Processors: Request Captcha Processor: Incident - Captcha Signature Tampering	195
Response Processors: Request Captcha Processor: Incident - Captcha Signature Spoofing	196
Response Processors: Request Captcha Processor: Incident - Captcha Cookie Manipulation	196
Response Processors: Request Captcha Processor: Incident - Captcha Image Probing	197
Response Processors: Request Captcha Processor: Incident - Captcha Request Size Limit Exceeded	198
Response Processors: Request Captcha Processor: Incident - Captcha Disallowed MultiPart	198
Response Processors: Request Captcha Processor: Incident - Captcha Directory Indexing	199
Response Processors: Request Captcha Processor: Incident - Captcha Directory Probing	200
Response Processors: Request Captcha Processor: Incident - Captcha Parameter Manipulation	201
Response Processors: Request Captcha Processor: Incident - Captcha Request Replay Attack	202
Response Processors: Request Captcha Processor: Incident - Multiple Captcha Replays	203
Response Processors: Request Captcha Processor: Incident - Multiple Captcha Disallow Multipart	204
Response Processors: Request Captcha Processor: Incident - Multiple Captcha Parameter Manipulation	204
Response Processors: CSRF Processor	205
Response Processors: CSRF Processor: Incident - CSRF Parameter Tampering	208
Response Processors: CSRF Processor: Incident - Multiple CSRF Parameter Tampering	209
Response Processors: CSRF Processor: Incident - CSRF Remote Script Inclusion	209
Response Processors: CSRF Processor: Incident - HTTP Referers Disabled	210
Response Processors: Header Injection Processor	211

	Response Processors: Force Logout Processor	211
	Response Processors: Strip Inputs Processor	212
	Response Processors: Slow Connection Processor	212
	Response Processors: Warning Processor	213
	Response Processors: Warning Processor: Incident - Warning Code Tampering	214
	Response Processors: Application Vulnerability Processor	214
	Response Processors: Application Vulnerability Processor: Incident - App Vulnerability Detected	215
	Response Processors: Support Processor	215
	Response Processors: Cloppy Processor	217
	Response Processors: Login Processor	218
	Response Processors: Login Processor: Incident - Site Login Invalid	224
	Response Processors: Login Processor: Incident - Site Login Multiple IP	224
	Response Processors: Login Processor: Incident - Site Login Multiple Usernames	225
	Response Processors: Login Processor: Incident - Site Login User Sharing	225
	Response Processors: Login Processor: Incident - Site Login User Pooling	226
	Response Processors: Login Processor: Incident - Site Login User Brute Force	226
	Response Processors: Login Processor: Incident - Site Login Brute Force	227
	Response Processors: Login Processor: Incident - Site Login Username Scan	227
	Response Processors: Google Map Processor	228
Chapter 7	Response Rule Configuration	229
	Response Overview	229
	Using the Editor	232
	List Of Incident Methods	233
Chapter 8	Reporting	237
	Reporting Overview	237
	Information for Report Types	238
	Scheduling a Report Overview	240
	Schedule a Report	241
	Report History	243
	Report Details	244
	Report Types	246
Chapter 9	Using the Web UI	251
	Web UI Overview	251
	The Dashboard	252
	Attackers	257
	Attacker Profile Page	258
	Incidents	262
	Incident Details	263
	Counter Responses	264
	Sessions	265
	Session Details	265
	Search	266
	Reports	268
	Configuration	269

	System Status	269
	Updates	271
Part 2	Appendices	
Appendix A	CAPTCHA Template	275
	CAPTCHA Template	275
Appendix B	Log Formats	279
	Access Log Format	279
	Security Log Format	281
	Audit Log Format	283
	Firewall Log Format	284
	Postgres Log Format	285
	mws Log Format	286
Appendix C	RBAC Groups and Roles	289
	RBAC Groups and Roles	289
Part 3	Index	
	Index	295

About the Documentation

- Documentation and Release Notes on page xi
- Documentation Conventions on page xi
- Documentation Feedback on page xiii
- Requesting Technical Support on page xiv

Documentation and Release Notes

To obtain the most current version of all Juniper Networks® technical documentation, see the product documentation page on the Juniper Networks website at <http://www.juniper.net/techpubs/>.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at <http://www.juniper.net/books>.

Documentation Conventions

Table 1 on page xii defines notice icons used in this guide.

Table 1: Notice Icons

Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.
	Tip	Indicates helpful information.
	Best practice	Alerts you to a recommended use or implementation.

Table 2 on page xii defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the configure command: user@host> configure
Fixed-width text like this	Represents output that appears on the terminal screen.	user@host> show chassis alarms No alarms currently active
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces or emphasizes important new terms. Identifies guide names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS CLI User Guide</i> RFC 1997, <i>BGP Communities Attribute</i>
<i>Italic text like this</i>	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# set system domain-name <i>domain-name</i>

Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none">To configure a stub area, include the stub statement at the [edit protocols ospf area area-id] hierarchy level.The console port is labeled CONSOLE.
< > (angle brackets)	Encloses optional keywords or variables.	stub <default-metric <i>metric</i>>;
(pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	broadcast multicast (<i>string1</i> <i>string2</i> <i>string3</i>)
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	rsvp { # Required for dynamic MPLS only
[] (square brackets)	Encloses a variable for which you can substitute one or more values.	community name members [<i>community-ids</i>]
Indentation and braces ({ })	Identifies a level in the configuration hierarchy.	<pre>[edit] routing-options { static { route default { nexthop <i>address</i>; retain; } } }</pre>
;(semicolon)	Identifies a leaf statement at a configuration hierarchy level.	
GUI Conventions		
Bold text like this	Represents graphical user interface (GUI) items you click or select.	<ul style="list-style-type: none">In the Logical Interfaces box, select All Interfaces.To cancel the configuration, click Cancel.
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select Protocols>Ospf .

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can provide feedback by using either of the following methods:

- Online feedback rating system—On any page at the Juniper Networks Technical Documentation site at <http://www.juniper.net/techpubs/index.html>, simply click the stars to rate the content, and use the pop-up form to provide us with information about your experience. Alternately, you can use the online feedback form at <https://www.juniper.net/cgi-bin/docbugreport/>.

- E-mail—Send your comments to techpubs-comments@juniper.net. Include the document or topic name, URL or page number, and software version (if applicable).

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at <http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf>.
- Product warranties—For product warranty information, visit <http://www.juniper.net/support/warranty/>.
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

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

- Find CSC offerings: <http://www.juniper.net/customers/support/>
- Search for known bugs: <http://www2.juniper.net/kb/>
- Find product documentation: <http://www.juniper.net/techpubs/>
- Find solutions and answer questions using our Knowledge Base: <http://kb.juniper.net/>
- Download the latest versions of software and review release notes: <http://www.juniper.net/customers/csc/software/>
- Search technical bulletins for relevant hardware and software notifications: <http://kb.juniper.net/InfoCenter/>
- Join and participate in the Juniper Networks Community Forum: <http://www.juniper.net/company/communities/>
- Open a case online in the CSC Case Management tool: <http://www.juniper.net/cm/>

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: <https://tools.juniper.net/SerialNumberEntitlementSearch/>

Opening a Case with JTAC

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

- Use the Case Management tool in the CSC at <http://www.juniper.net/cm/>.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see <http://www.juniper.net/support/requesting-support.html>.

PART 1

WebApp Secure Administration

- [Overview on page 3](#)
- [Deployment on page 11](#)
- [Installing the Appliance on page 23](#)
- [Configuring WebApp Secure on page 43](#)
- [Managing the Appliance on page 73](#)
- [The Processors on page 117](#)
- [Response Rule Configuration on page 229](#)
- [Reporting on page 237](#)
- [Using the Web UI on page 251](#)

CHAPTER 1

Overview

- [WebApp Secure Overview on page 3](#)
- [Methodology on page 4](#)
- [Features and Benefits on page 4](#)
- [Key Components on page 6](#)
- [Anatomy and Flow of an HTTP Request / Response on page 8](#)
- [Limitations on page 10](#)

WebApp Secure Overview

WebApp Secure protects websites from would-be attackers, fraud, and theft. Its Web intrusion prevention system uses deception to detect, track, profile, and block attackers in real time by inserting detection points into your webserver's output to identify attackers before they do damage. WebApp Secure then tracks detected attackers, profiling their behavior and deploying countermeasures.

WebApp Secure sits between your webservers and the outside world. It inspects HTTP and HTTPS traffic and functions as a reverse proxy. WebApp Secure seeks out potential attack attempts or probes by adding detection points to outbound web traffic and removing detection points from inbound Web traffic. These detection points are transparent to common, legitimate users. It then monitors and strips these points from the requests coming back from the user's browser. Any change to a detection point is an indicator of an attempted attack. The system logs incidents to a database of attacker profiles, and exposes them to the security administrators through a web-based interface. System administrators can then apply automated abuse-prevention policies, or respond manually.

Related Documentation

- [Methodology on page 4](#)
- [Features and Benefits on page 4](#)
- [Key Components on page 6](#)
- [Anatomy and Flow of an HTTP Request / Response on page 8](#)

Methodology

WebApp Secure protects web sites from attackers using a proprietary blend of intrusion deception techniques. Its real-time protection system detects, tracks, profiles, and defends against hackers. By dynamically inserting detection points, honeypots, and honeytokens into the code of your web application traffic, WebApp Secure identifies and stops attackers before they can do damage.

WebApp Secure protects and defends using the following methodology:

- **Detect**—Detection points are inserted into web application code, creating a virtual minefield that detects hackers when they manipulate these detection points during pre-attack reconnaissance.
- **Track**—WebApp Secure goes beyond the IP address and tracks attackers based on client fingerprinting techniques.
- **Profile**—WebApp Secure's tracking techniques enable attacker profiling. Every attacker is assigned a name and each incident is recorded, along with a threat level based on the attacker's intent and skill.
- **Respond and Understand**—Once an attacker is detected, an appropriate response can be deployed manually or automatically in real-time.

Related Documentation

- [WebApp Secure Overview on page 3](#)
- [Features and Benefits on page 4](#)
- [Key Components on page 6](#)
- [Anatomy and Flow of an HTTP Request / Response on page 8](#)

Features and Benefits

WebApp Secure detects attackers before they have the chance to successfully establish an attack vector, and blocks them with client-level tracking that does not impact legitimate users. It works out-of-the-box, so there are no rules to write, and no signatures to update. It continually profiles attackers as they come onto the scene, and it maintains a profile of known application abusers and all of their malicious activity.

Ease-of-use deployment

- Acts as a reverse proxy with load balancing
- Available as a hardware appliance with high availability
- Available as a VMware or Amazon Machine Image
- Support for alternate ports (other than 80 and 443)

Secure management

- Updates are automatically downloaded and available within the Web UI
- Kernel is hardened, ports are locked-down, and backups are encrypted
- Monitoring is web-based
- Configuration is done through a Web UI with setup wizards or from the robust Command Line Interface
- Access control is role-based to allow for multiple administrators
- External authentication providers are supported, including LDAP and RADIUS
- Traffic for multiple applications or domains can be secured without the need for additional licenses

High Performance

- High availability for hardware version
- Higher throughput using master/slave clustering
- Low latency
- Link aggregation

Alerts, Reporting, Logging

- Alerts are sent by e-mail when specific incidents, incident patterns, or system failures occur
- SNMP traps are available for system logging
- Reports of attacker activity can be scheduled using the Web UI or retrieved on-demand
- Audit logs track changes to the system made by administrators and potential management interface break-in attempts
- Remote system logging, including a custom Device Support Module (DSM) for the Juniper STRM Series Security Threat Response Manager, is available
- RESTful API for third-party software integration is available

Abuse Recording and Analysis

- Full HTTP Capture—Captures and displays all HTTP traffic for security incidents
- Abuse Profiles—Maintain a profile of known application abusers and all of their malicious activity against the application
- Tracking and Re-identification—Enables application administrators to re-identify abusive users and to apply persistent responses, over time, and across sessions—Enhances tracking capabilities and fingerprinting of detected attackers
- Abuse Responses—Enables administrators to respond to application abuse with session-specific warnings, blocks, and additional checks; includes one-click automation of responses during configuration

- Related Documentation**
- [WebApp Secure Overview on page 3](#)
 - [Methodology on page 4](#)
 - [Key Components on page 6](#)
 - [Anatomy and Flow of an HTTP Request / Response on page 8](#)

Key Components

WebApp Secure includes the following components:

Core Data Types

- **Attackers**—Attackers, also called hackers and profiles, are the core data type within WebApp Secure. Attackers can have multiple sessions, locations, environments, and can trigger multiple Incidents. All data on an attacker is tied back to the same place for the same malicious behavior. Based on a combination of incidents, or the manual intervention of the administrator, counter responses are applied to an attacker. Each attacker is assigned a severity level based on the highest-complexity incident the attacker has triggered.
- **Sessions**—Sessions are groups of HTTP requests performed by a user on a protected web site. There is a one-to-one mapping between a session and a fingerprint. If a session turns malicious, an attacker profile is created. These sessions can also be consolidated over time to perform fuzzy-logic matching on traffic patterns.
- **Fingerprint**—WebApp Secure fingerprints incoming HTTP requests using a proprietary combination of traditional and non-traditional methods. These fingerprints ensure that all traffic from the same source is consolidated under the same session and / or attacker.
- **Incidents**—An incident is a recorded instance of a particular security threat. The full HTTP request and response of each incident is recorded and related back to an attacker, session, environment, and location. For convenience's sake, similar incidents may be grouped within a 24-hour time period. Incidents have five complexity levels: informational, suspicious, low, medium, and high. An attacker profile is automatically created once a session triggers at least one low, medium, or high incident.
- **Locations**—A location is created by parsing an IP address. Each IP address that is implicated in an attack, either because it is the attacker's IP address or because it is the IP address of a proxy server that acted as an intermediary, is geo-coded and the results, including the IP address itself, are stored as a location.
- **Environments**—An environment is created by parsing a user-agent string. Every web browser, and many common tools and scripts, emit a user-agent header as part of their request. WebApp Secure parses this header and stores the results as an environment.
- **Counter Responses**—A counter response is an alternate HTTP response that will be served to an attacker. Such responses can be triggered manually by an administrator who is logged into the Web UI, or automatically by way of customizable autoresponse rules that ship with WebApp Secure.

Security Engine

The Security Engine is the core of WebApp Secure. It consists of several components that work together to secure your web sites and applications. See [“Security Engine Configuration” on page 48](#).

- **Location Manager**—The location manager parses IP addresses, determines which ones should be implicated in attacks, and stores this information for later use. It also powers integration with Content Delivery Networks, such as Akamai, which use alternate headers for IP address information.
- **Environment Manager**—The environment manager parses user-agent headers and stores this information for later use.
- **Session Manager**—The session manager keeps track of activity from each user profile and maintains fingerprints used to identify unique users, enabling tracking beyond the IP address.
- **Processors**—Processors are pluggable modules that provide specific security-related functionality. A full description of each processor, including all available configuration options, is available the processors section which begins here [“Processors Overview” on page 122](#).

Support Services

Services run in the background, performing tasks such as sending alerts, generating reports, or performing maintenance tasks.

- **Alert Service**—WebApp Secure can send e-mail alerts to administrators when an incident of a specified severity is detected. For instance, the system can send out an e-mail notification to a specific administrator who is on-call if an incident level of critical is detected, allowing the administrator to respond quickly to the threat. It can also send alerts to e-mail addresses on a defined schedule, and/or send SNMP traps to one or more SNMP servers. The initial configuration of the alert service is performed with the configuration wizard, and these settings are also available through the Services tab in the Web UI. See [“Using the Configuration Wizard” on page 36](#).
- **Backup Service**—Configure the frequency, type, and destination for system backups. These settings can be configured through the configuration wizard, or through the Backups section of the Web UI. See [“Using the Configuration Wizard” on page 36](#).
- **Counter Response Service**—The counter response service handles the deployment of responses triggered by incidents. This service is configured to use the response rules. See [“Response Overview” on page 229](#).
- **Database Cleanup Service**—This service purges expired data from the database on a regular schedule. See [“CLI: Config: Setting a Configuration Parameter” on page 87](#).
- **External Response Service**—The external response service manages the connection to the Juniper Networks SRX Series Service Gateway, which can be used to perform external counter responses. See [“Configure the SRX Series Integration” on page 66](#).
- **NTP Servers**—To keep your appliance clock synchronized to the correct time, WebApp Secure allows the configuration of NTP servers. The appliance can use suggested

publicly-available NTP server pools, or it can be configured to use an internal NTP server for timekeeping. See [“NTP Service” on page 62](#).

- Self-Monitoring—WebApp Secure can monitor its own status and in the case of a failure, automatically recover or notify you when there is a non-recoverable event. See [“Self-Monitoring” on page 98](#) for alert configuration options.
- Session Aggregation Service—The session aggregation service attempts to aggregate sessions on your system, based on common fingerprint attributes.
- SMTP Settings—Various parts of the system can send emails. WebApp Secure can function as its own SMTP server, or you can configure it to use an existing MTA on your network. See [“Using the Configuration Wizard” on page 36](#).
- The Spotlight Connector—The Spotlight Connector provides feeds to the SRX series for automatically filtering traffic on both network and application layers. WebApp Secure detects threats and periodically submits data to a component called the Spotlight Connector. Spotlight Connector publishes the submitted threat data as a standard feed to the SRX series device. See [“Enable the Spotlight Connector Service” on page 109](#).
- Spotlight Secure—Spotlight Secure is designed to provide additional intelligence. If enabled, a two-way communication process shares information about attackers and attacks to and from a Spotlight server run by Juniper Networks. This allows WebApp Secure to positively identify attackers that have attacked other Juniper customers. This service also provides additional details about sessions which allows Juniper to make more informed decisions on how to respond to threats. By default, the service is turned off. See [“Enable Spotlight Secure” on page 113](#).

**Related
Documentation**

- [WebApp Secure Overview on page 3](#)
- [Methodology on page 4](#)
- [Anatomy and Flow of an HTTP Request / Response on page 8](#)

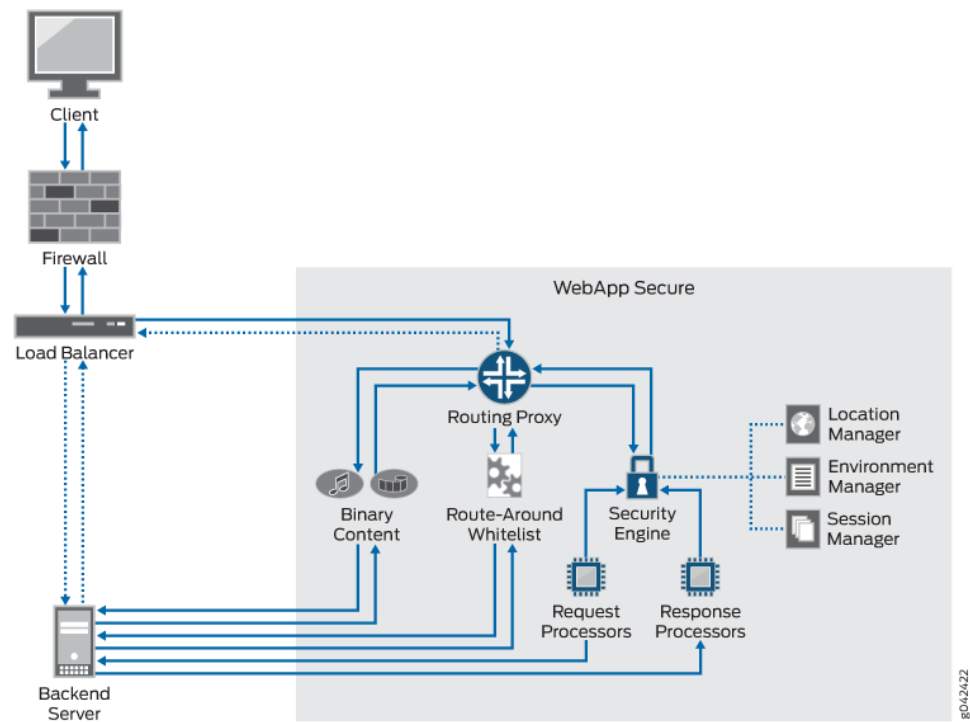
Anatomy and Flow of an HTTP Request / Response

1. The client issues an HTTP request, either from a web browser, script or tool, to an IP or hostname that resolves to a WebApp Secure system (it may pass through firewalls and/or load balancers on the way).
2. This request is initially handled by a routing proxy that either:
 - In the case of static files, such as CSS, Javascript, or binary content like images, proxies directly to the backend server and returns the resource.
 - In the case of URIs added to the proxy exclusions list (engine.proxy.exclusions parameter), proxies directly to the backend server and returns the page or resource.
 - In the case of all other URIs, proxies to the security engine.

This proxy also performs a regular-expression-based match on the hostname to determine which configuration options to apply to the request, based on your application settings. In the event of no match, the global defaults are used.

3. If the request reaches the security engine, several important managers parse it:
 - a. Location Manager—The location manager parses the incoming IP address, as well as IP addresses present in alternate tracking headers, such as the X-Forwarded-For header.
 - b. Environment Manager—The environment manager parses the incoming user-agent header, if present.
 - c. Session Manager—The session manager attempts to determine if the request contains the security engine tracking cookie in order to associate this request with an existing session. If this cookie is not present, the request will be fingerprinted later.
4. Once these managers parse the request, the counter response service determines if any counter responses are active on this session and they are attached accordingly.
5. Next, the processors are applied to the request, and incidents are triggered accordingly. For full information about each processor, consult the processor section that begins here [“Processors Overview” on page 122](#).
6. Next, a process runs to determine if the request was a URL fuzzing attempt, and incidents are triggered accordingly.
7. Then, the request is made to the backend server itself.
8. After the backend server returns a response, it passes through security engine again, where the processors are applied to the response, and incidents are triggered accordingly. For full information about each processor, consult the processor section that begins here [“Processors Overview” on page 122](#).
9. Finally, various headers are attached to the response, such as set-cookie (to set the WebApp Secure tracking cookie) and the response is returned to the client.

Figure 1: HTTP Request/Response Flow



- Related Documentation**
- [WebApp Secure Overview on page 3](#)
 - [Methodology on page 4](#)
 - [Features and Benefits on page 4](#)
 - [Key Components on page 6](#)

Limitations

WebApp Secure has the following limitations:

- As a reverse proxy, WebApp Secure is a Layer-7 device, and must be assigned an IP address. Therefore, Layer-2 bridging functionality (hardware fail-safe) capabilities are not available, and WebApp Secure should not be physically in-line with protected application servers; see the high availability section that begins here [“High Availability Overview” on page 21](#) for information on how to reduce the risk of downtime.
- WebApp Secure only accepts HTTP and HTTPS 1.0 and 1.1 traffic because it is the only traffic it monitors.
- WebApp Secure is not a network firewall and should not be an edge device. While firewall capabilities exist on the device, they are not available to the user and are there to protect the WebApp Secure device itself.
- WebApp Secure does not support NTLM authentication.

CHAPTER 2

Deployment

- [Appliance Deployment Overview on page 11](#)
- [Placement Between Firewall and Web Servers on page 11](#)
- [Options for Load-Balanced Environments on page 13](#)
- [SSL Traffic Considerations on page 13](#)
- [EC2 Deployment on page 14](#)
- [Deploying Using the Command Line on page 14](#)
- [Deploying Using the Web Interface on page 15](#)
- [Assigning the Instance and IP Using the CLI on page 19](#)
- [Assigning the Instance and IP Using the Web Interface on page 19](#)
- [Verify the Instance is Running on page 20](#)
- [Clustering Overview on page 20](#)
- [High Availability Overview on page 21](#)

Appliance Deployment Overview

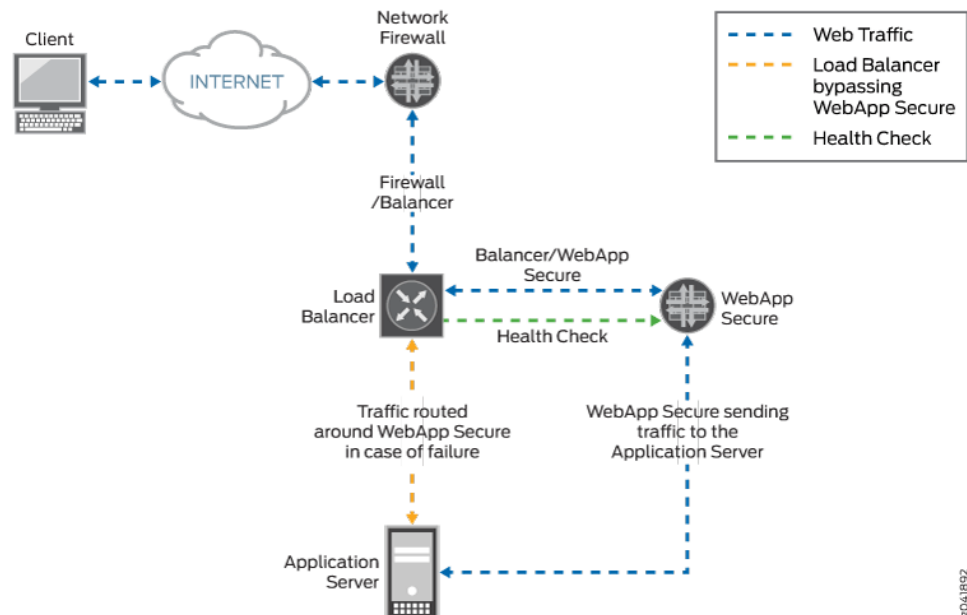
The WebApp Secure appliance (The appliance is the software/hardware system. It is synonymous with WebApp Secure in most contexts.) processes all inbound web requests and outbound web responses. Outbound responses are modified in ways that are invisible to the average user; inbound requests are checked to see if the modified responses have been altered in any way. Any alterations are suspicious and indicate a possible hacker. Due to its focus on web applications, WebApp Secure only accepts HTTP/HTTPS traffic and is normally placed between a load balancer and your web applications. Topologically, you should think of WebApp Secure as a web reverse proxy server.

Placement Between Firewall and Web Servers

WebApp Secure acts as a reverse proxy and actively manipulates traffic between the protected web application and the Internet. It is deployed between the protected webserver and the last system which can alter user-facing traffic. This location gives WebApp Secure full visibility into the HTTP traffic destined for the web servers (including any errors caused by authentication failures), and lets it inject and strip out any code it uses in protecting the application. This topology has the added benefit of minimally

impacting internal network bandwidth. The following figure shows the WebApp Secure deployed in its most simple form as a reverse proxy connected to a load balancer.

Figure 2: WebApp Secure Placement in the Network - Between Firewall and Web Servers

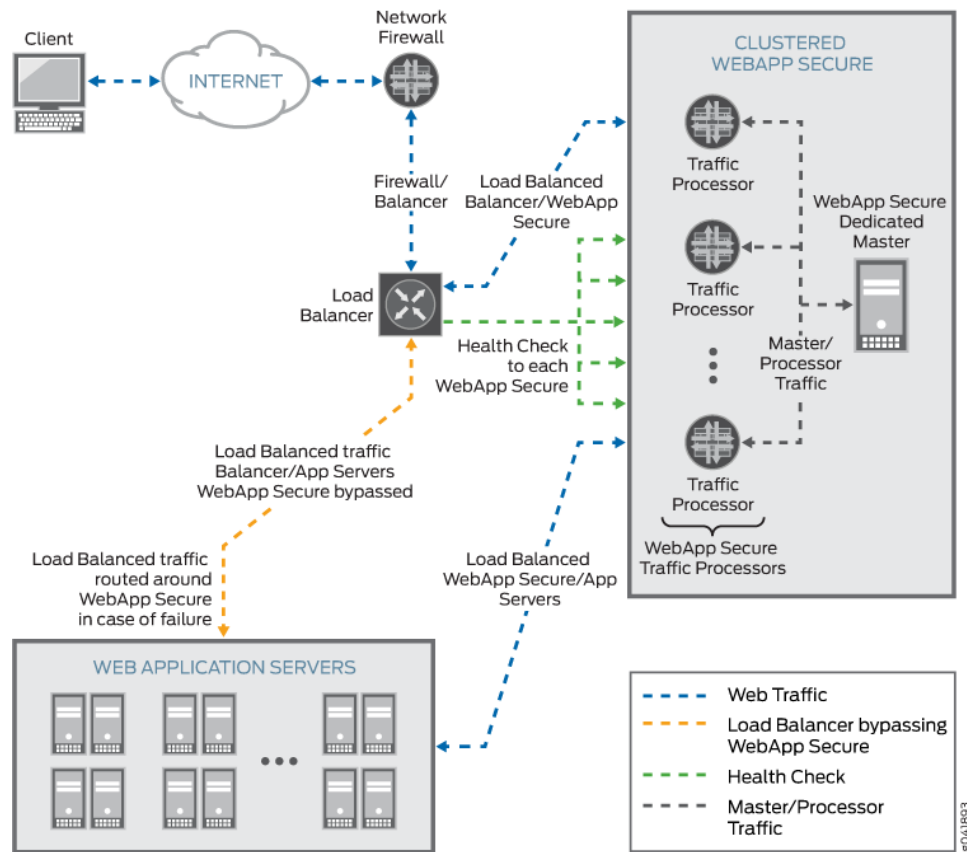


Network placement requirements for WebApp Secure are as follows:

- Because WebApp Secure only processes HTTP and HTTPS traffic, it must live behind a device that can separate Application Layer (Layer 7) traffic.
- In order to prevent a WebApp Secure issue from impacting a protected application, the upstream device (that is, the router or load balancer) must perform Health Check monitoring on WebApp Secure over HTTP. If the Health Check fails, the load balancer or Layer 7 router should pass traffic directly to the protected application servers, rather than to WebApp Secure.

The actual implementation depends on the user's specific network topology. The following figure shows a more complex environment with clustered web servers and clustered appliances.

Figure 3: WebApp Secure Deployment - Connected to Load Balancer



Options for Load-Balanced Environments

WebApp Secure can serve as a load balancer for HTTP and HTTPS web traffic, but we recommend that a dedicated hardware solution be used in that capacity. Dedicated load balancers are optimized for that role and will provide higher overall performance.

SSL Traffic Considerations

WebApp Secure includes SSL decryption capabilities to give it visibility into all of the protected application's traffic. It supports two modes: Passive Decryption and SSL Termination. In Passive Decryption mode, WebApp Secure decrypts requests for processing, then re-encrypts them before sending them on to the application server. HTTPS responses to the user follow the same process, where they are decrypted, processed, and re-encrypted before returning to the user. In SSL Termination mode, the appliance serves as an SSL termination point. It decrypts incoming HTTPS traffic, processes them, then proxies the decrypted requests on to the application. Responses to the user are received unencrypted from the application server, processed, encrypted, then passed to the user.

EC2 Deployment

The WebApp Secure instance is a private instance. For access, you must provide Juniper your account ID. You can find your account ID when you log into your AWS account and select **Account Activity**. Your account ID is displayed in the top right under your account name.

Related Documentation

- [Deploying Using the Command Line on page 14](#)
- [Deploying Using the Web Interface on page 15](#)
- [Assigning the Instance and IP Using the CLI on page 19](#)
- [Assigning the Instance and IP Using the Web Interface on page 19](#)
- [Verify the Instance is Running on page 20](#)

Deploying Using the Command Line

You must have an ec2tools environment setup prior to deploying the instance from the CLI. Please refer to the Amazon documentation for assistance in setting up your environment.

Next, make sure you have access to the image by entering: `./ec2-describe-images -x self -o "969756132034"`

The output should look similar to the following: **IMAGE ami-4d3df524 969756132034/Mykonos Appliance 969756132034** available If you do not see any output at all then you most-likely don't have access to our instance. Please contact Juniper Networks support to get help with this issue.

To deploy using the CLI, do the following:

1. Create a security group for the instance.
 - `./ec2-add-group Mykonos -d "Mykonos Appliance"`
 - `./ec2-authorize Mykonos -p 2022`
 - `./ec2-authorize Mykonos -p 80`
 - `./ec2-authorize Mykonos -p 443`
 - `./ec2-authorize Mykonos -p 5000`
 - `./ec2-authorize Mykonos -p 8080`



NOTE: It is recommended that you only allow ports 5000 and 8080 from known good IPs.

2. You can now deploy the new instance as follows:

- `./ec2-run-instances 'AMI ID' -k 'KEY PAIR' -t m1.large -g Mykonos`

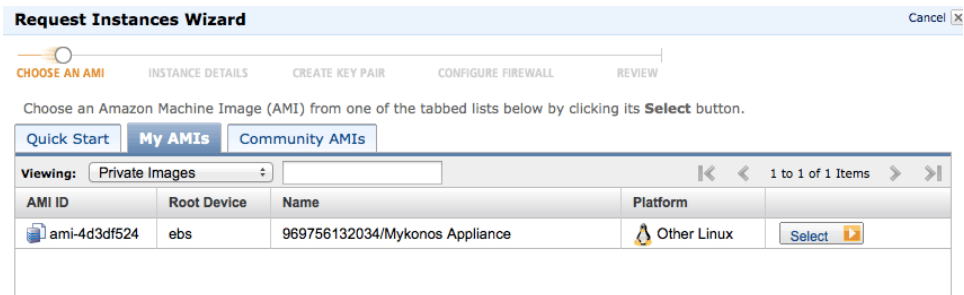
You must replace the AMI ID with the AMI ID listed in **Describe Image**. In this example, it would be **ami-4d3df524**. You must also replace **KEY PAIR** with the name of the key pair you are using to access the system.

Deploying Using the Web Interface

To deploy using the Web Interface, do the following:

1. Log into the AWS management console.
2. Select the **Amazon EC2** tab.
3. Click the **Launch Instance** button under the My Instances section.
4. Select **Launch Classic Wizard** then click **continue**.
5. Select the **My AMIs** tab.
6. Next to Viewing, select **Private Images**.
7. To the right of the appliance, click the **Select** button.

Figure 4: AWS management console



8. Change the Instance Type to **Large (m1.large)**.

Figure 5: Instance Type

Request Instances Wizard Cancel

CHOOSE AN AMI **INSTANCE DETAILS** CREATE KEY PAIR CONFIGURE FIREWALL REVIEW

Provide the details for your instance(s). You may also decide whether you want to launch your instances as "on-demand" or "spot" instances.

Number of Instances: 1 **Instance Type:** Large (m1.large, 7.5 GB)

Launch Instances

EC2 Instances let you pay for compute capacity by the hour with no long term commitments. This transforms what are commonly large fixed costs into much smaller variable costs.

Launch into: EC2

Availability Zone: No Preference

Request Spot Instances

[< Back](#)
[Continue >](#)

- Click the **Continue** button. Then click the **Continue** button again on the next screen.

Figure 6: Configure Instance continued

Request Instances Wizard Cancel

CHOOSE AN AMI INSTANCE DETAILS **CONFIGURE FIREWALL** CREATE KEY PAIR REVIEW

Number of Instances: 1 **Availability Zone:** No Preference

Advanced Instance Options

Here you can choose a specific kernel or RAM disk to use with your instances. You can also choose to enable CloudWatch Detailed Monitoring or enter data that will be available from your instances once they launch.

Kernel ID: Use Default **RAM Disk ID:** Use Default

Monitoring: ☐ Enable CloudWatch detailed monitoring for this instance (additional charges will apply)

User Data:

☒ as text

☐ as file ☐ base64 encoded

Termination Protection: ☐ Prevention against accidental termination.

Shutdown Behavior: Stop Choose the behavior when the instance is shutdown from within the instance.

[< Back](#)
[Continue >](#)

- Add a **Name** to make the instance easily recognizable and click **Continue**.

Figure 7: Instance, Configure Name

Request Instances Wizard Cancel

CHOOSE AN AMI **INSTANCE DETAILS** CREATE KEY PAIR CONFIGURE FIREWALL REVIEW

Add tags to your instance to simplify the administration of your EC2 infrastructure. A form of metadata, tags consist of a case-sensitive key/value pair, are stored in the cloud and are private to your account. You can create user-friendly names that help you organize, search, and browse your resources. For example, you could define a tag with key = Name and value = Webserver. You can add up to 10 unique keys to each instance along with an optional value for each key. For more information, go to [Using Tags](#) in the *EC2 User Guide*.

Key (127 characters maximum)	Value (255 characters maximum)	Remove
Name	Mykonos	

[Add another Tag.](#) (Maximum of 10)

< Back Continue >

11. Select or create your **Key Pair** and click **Continue**.

Figure 8: Instance, Create Key Pair

Request Instances Wizard Cancel

CHOOSE AN AMI **INSTANCE DETAILS** **CREATE KEY PAIR** CONFIGURE FIREWALL REVIEW

Public/private key pairs allow you to securely connect to your instance after it launches. To create a key pair, enter a name and click **Create & Download your Key Pair**. You will then be prompted to save the private key to your computer. Note, you only need to generate a key pair once - not each time you want to deploy an Amazon EC2 instance.

☒ **Choose from your existing Key Pairs**

Your existing Key Pairs*: Main

☐ **Create a new Key Pair**

☐ **Proceed without a Key Pair**

< Back Continue >

12. Select **Create a New Security Group** and enter the following information:

- Group Name: Mykonos
- Group Description: Mykonos Appliance

- Port Range: 2022
- Port Range: 80
 - Add Rule
- Port Range: 443
 - Add Rule
- Port Range: 5000
 - Add Rule

Figure 9: Instance Create Security Group

☐ Choose one or more of your existing Security Groups

☒ Create a new Security Group

Group Name

Group Description

Inbound Rules


Create a new rule:

Port range:

(e.g., 80 or 49152-65535)

Source:

(e.g., 192.168.2.0/24, sg-47ad482e, or 1234567890/default)

 Add Rule

TCP	Port (Service)	Source	Action
	2022	0.0.0.0/0	Delete
	80 (HTTP)	0.0.0.0/0	Delete
	443 (HTTPS)	0.0.0.0/0	Delete
	5000	0.0.0.0/0	Delete
	8080 (HTTP*)	0.0.0.0/0	Delete

[< Back](#)
[Continue >](#)

13. Click the **Continue** button.

14. Click the **Launch** button.

Figure 10: Instance, Review and Launch

Request Instances Wizard Cancel

CHOOSE AN AMI | INSTANCE DETAILS | CREATE KEY PAIR | CONFIGURE FIREWALL | **REVIEW**

Please review the information below, then click **Launch**.

AMI: Other Linux AMI ID ami-4d3df524 (x86_64) [Edit AMI](#)

Number of Instances: 1

Availability Zone: No Preference

Instance Type: Large (m1.large)

Instance Class: On Demand [Edit Instance Details](#)

Monitoring: Disabled **Termination Protection:** Disabled

Tenancy: Default

Kernel ID: Use Default **Shutdown Behavior:** Stop

RAM Disk ID: Use Default

User Data: [Edit Advanced Details](#)

Key Pair Name: Main [Edit Key Pair](#)

Security Group(s): sg-7ae70112 [Edit Firewall](#)

[< Back](#) [Launch](#)

Assigning the Instance and IP Using the CLI

1. Request a new public IP address: `./ec2-allocate-address` Note the IP it returns.
2. Get the Instance ID of the WebApp Secure Instance: `./ec2-describe-instances` Locate the Instance ID of the Appliance.
3. Now you can associate the IP with the WebApp Instance: `./ec2-associate-address -i 'Instance ID'`

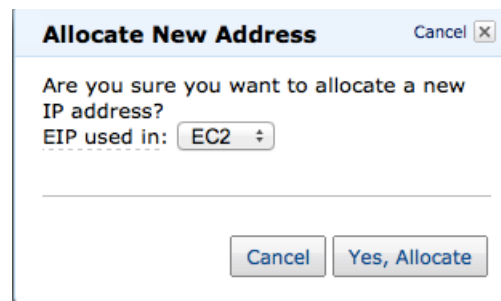


NOTE: Replace the 'IP' with your IP address and the 'Instance ID' with the ID of your instance.

Assigning the Instance and IP Using the Web Interface

1. Select the **Amazon ec2** tab.
2. Select **Elastic IPs** under **Navigation** on the left.
3. Click the **Allocate New Address** button under the **Address** section.
4. Click **Yes, Allocate**.

Figure 11: Allocate New Address



5. Click **Associate Address**.
6. Select the **Appliance**.
7. Click **Yes, Associate**.

Verify the Instance is Running

- Purpose** At this point, your instance should be up and running. You can use the web interface or the CLI tools to verify this.
- Action** To access your instance, you must have a copy of the key pair you used for the instance.
`ssh -i 'PATH TO KEY PAIR' mykonos@'IP' -p 2022`
- Upon successful login, you will be sent to the mykonos-shell where you can further configure the appliance

Clustering Overview

Individual WebApp Secure appliances have the ability to work together as one system in a cluster. Clustering allows traffic to be divided among multiple appliances, effectively reducing the per-system load. In a clustered network configuration, the master node holds the database that is populated by one or more traffic processors. In order to successfully utilize a WebApp Secure cluster, a load-balancer must properly segregate traffic to each of the defined traffic processing nodes. Each of these traffic nodes must maintain connectivity with the master in order to operate.



NOTE: Clustering should not be confused with High Availability. Clustering is used to increase throughput (by utilizing multiple processing nodes), and can reduce the chance that the whole system will fail. Clustering does not protect the master node from failure as in a High Availability setup; only HA configurations are set up to include failsafe procedures to designate a new master when the first one is unavailable.

In a traditional WebApp Secure deployment (one system), the appliance is responsible for holding its own database as well as processing the traffic. In a clustered deployment, you have the ability to segregate the database from those systems which will process

incoming requests. During cluster configuration, you will have the ability to designate a node type for each system. At a minimum, the cluster must have a way to process traffic and a way to store the relevant information.

Node types are as follows:

- **Master:** A master node is similar to a single-system deployment in that it holds the database, and also processes incoming traffic. This satisfies both requirements for a cluster (database and traffic processor). It is possible to set up a cluster with only one master node (no additional processing nodes). Additional traffic processing nodes can be added at a later point in time if desired.
- **Dedicated Master:** A dedicated master node holds the database similar to a master node, but it does not have the ability to process traffic. Using a dedicated master in a clustered configuration requires the addition of at least one traffic node.
- **Traffic Processor:** A traffic node is only responsible for processing incoming requests. It does not contain a database, so a master or a dedicated master node must accompany a traffic node. The number of traffic nodes you can add to a cluster is dependent on (1.) the hardware specifications of the master, (2.) the amount of incoming traffic on protected web application, and (3.) the number of additional traffic nodes in the cluster. For optimal stability, be sure to monitor the cluster's performance as you add each traffic node.

High Availability Overview

To minimize the risk of downtime, WebApp Secure deployments have the ability to be placed in a Highly Available (HA) configuration. In this setup, an additional appliance is on stand-by in the event that the currently-active appliance goes offline. If this happens, the passive appliance is able to become the new active appliance automatically - without needing to restart the system. An HA configuration is similar to clustering, with the major exception being that the passive system has a copy of the services needed to take over when the master fails. WebApp Secure uses a Virtual IP (VIP) to float between the currently active system and the current passive system.



NOTE: An HA configuration is only available on WebApp Secure dedicated hardware systems. It is not available in a Virtual Machine installation.

Related Documentation

- [High Availability Settings on page 32](#)
- [Configuring High Availability on page 30](#)
- [High Availability Network Failure Detection, Actions, and Monitoring on page 95](#)

CHAPTER 3

Installing the Appliance

- [WebApp Secure Appliance Terminology on page 23](#)
- [First Time Configuration on page 24](#)
- [Initial Appliance Configuration on page 24](#)
- [Changing the Password on page 25](#)
- [Resetting the Password on page 25](#)
- [Configure Network Interfaces on page 26](#)
- [Set the Hostname on page 27](#)
- [Set DNS on page 27](#)
- [Initializing the System on page 27](#)
- [Verify Connectivity on page 28](#)
- [Install the License on page 29](#)
- [Configuring High Availability on page 30](#)
- [High Availability Settings on page 32](#)
- [Configuring Clustering on page 34](#)
- [Performing Initial Updates on page 35](#)
- [Updating the Cluster on page 35](#)
- [About the Configuration Wizard on page 35](#)
- [Using the Configuration Wizard on page 36](#)
- [Using WebApp Secure with Third-Party Load Balancer on page 41](#)
- [Verify the Installation on page 42](#)

WebApp Secure Appliance Terminology

The following terminology is used in this section:

- **Appliance**—The software/hardware system. It is synonymous with WebApp Secure in most contexts.
- **Web UI**—WebApp Secure graphical user interface. In cases referencing the WebApp Secure system, this term refers to the web-accessible graphical user interface.
- **HA**—High Availability, a configuration that aims to reduce the chance full system failures.

- **Backend**—In almost all cases (except where explicitly mentioned otherwise) the backend is defined as the server which houses the web application that is being protected by WebApp Secure.
- **Application**—A group of configuration settings that are applied to HTTP traffic based on a regular-expression-based match on the domain component of the URL. Adding an application allows you to override global configuration values.
- **Page**—A group of configuration settings that are applied to HTTP traffic based on a regular-expression-based match on the path component of the URL. Pages exist logically under an application. Adding a page allows you to override application configuration values.

First Time Configuration

Basic configuration of WebApp Secure is a straightforward process utilizing the steps listed below.

- [Configure Network Interfaces on page 26](#)
- [Set the Hostname on page 27](#)
- *Initialize the Appliance*

At this point, the Web UI should be active. From the Web UI, you will:

- [Install the License on page 29](#)
- See “About the Configuration Wizard” on page 35

Initial Appliance Configuration

Initial configuration is done through the console in a limited shell. Once WebApp Secure has been initialized for the first time, you can log into a Web console to finish the initial setup. You must use the direct console interface to configure the appliance IP address. Once the system has an IP address, you can use SSH to connect through port 2022.

Use the SSH command.

```
ssh <machine-ip-address> -p 2022 -l mykonos
```

- User: mykonos
- Password: mykonosadmin



NOTE: You should immediately change these defaults after you login.



NOTE: The credentials used here are the same ones used to access the Web UI.

- Related Documentation**
- [Changing the Password on page 25](#)
 - [Resetting the Password on page 25](#)
 - [Configure Network Interfaces on page 26](#)

Changing the Password

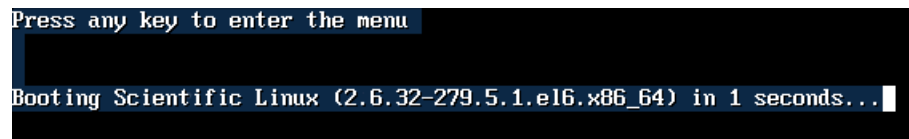
The system password can only be changed from the underlying Linux command line. To do this, connect through the console or SSH. You will see the setup utility screen. Navigate to **Quit** to exit to the shell. Type **passwd** and follow the prompts.

- Related Documentation**
- [Resetting the Password on page 25](#)

Resetting the Password

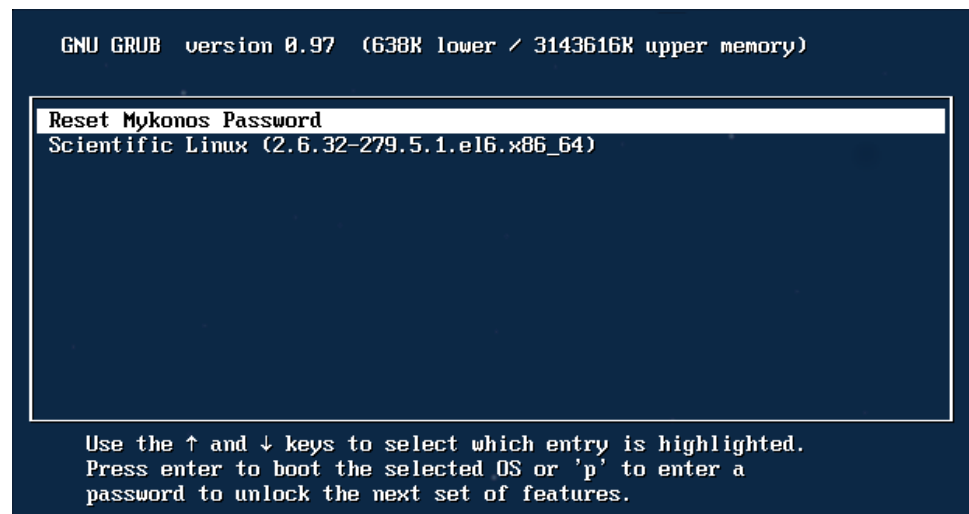
To reset the password, an appliance reboot is required. A boot menu option exists to reset the user credentials. By default, the appliance will boot normally, but by pressing any key before the operating system starts booting, you can get to the boot menu.

Figure 12: Boot Menu



Then you can select the option to reset the password.

Figure 13: Reset Password



NOTE: Resetting the password is only necessary if you have completely forgotten your password. It is not meant for day-to-day password changes.

- Related Documentation**
- [Changing the Password on page 25](#)
 - [Configure Network Interfaces on page 26](#)

Configure Network Interfaces

The first step in setting up your appliance is to configure the network interface(s). The number of interfaces that are required depends on the mode in which the appliance will be operating. If you are using hardware and setting up a HA node, a minimum of 2 network interfaces must be configured. For all other modes, a minimum of 1 interface is required.

When performing the initial setup of a hardware appliance, you will need to connect via the serial console port. You will need a terminal emulation application to enter commands through the console.

Serial Port Settings:

- Baud Rate: 115,200 Bd
- Bit Format: 8-N-1

Once logged in, use the WebApp Secure CLI to configure the network interfaces by running the command `cli` from the bash prompt. The CLI is also available through SSH on port 2022 after WebApp Secure is operating.

Then you can configure the interfaces by entering the 'system' level of the CLI, and typing **set interface <interface> <configuration>**. For more information on the syntax to configure interfaces, see [“The CLI: The Set Command” on page 75](#).

Once you have configured your network interface(s) you must set the management interface that will be used for the appliance. By default, the management interface is set to eth0. If this interface is correct then it is already set and you can move on the next step. If this is not the correct interface to use as the management interface, then you must run the following command from the root of the CLI.

system set management-interface <interface>

The command above sets the system management interface to the given interface. Once initialized, the management interface is used for all access to the appliance and for all node communications with the exception of data replication between HA nodes. The data replication occurs over the interconnect (10GbE interface) only for performance reasons.

Dedicated Management Interface—Optionally, you can designate an interface to be used strictly for management of the appliance. This modifies the firewall rules to force all data services to be sent through the configured interface.

Next, you will set the system hostname. See [“Set the Hostname” on page 27](#).

- Related Documentation**
- [The CLI: The Set Command on page 75](#)

Set the Hostname

From within the CLI, set the hostname for the local system by running the following command:

```
system set hostname <hostname>
```

Next, you will review the DNS settings. See “Set DNS” on page 27.

Related Documentation

- [Configure Network Interfaces on page 26](#)
- [The CLI: General and Base Commands on page 77](#)

Set DNS

Use the Network Level CLI Set command to configure DNS settings. If your appliance is using DHCP, the DNS settings will be inherited from the DHCP request and explicitly setting the DNS settings can be skipped. If all interfaces will be assigned static IP addresses, the DNS settings must be set.

Add the IP address of a name server or name servers to use for DNS resolution. You can pass all name servers as a comma separated list in their order of priority. For example,

```
set dns nameservers 192.168.0.10,192.168.0.11
```

Refer to the standard Linux documentation for all parameters you can supply to the set command.

https://access.cisco.com/Documentation/US/Red_Hat/Enterprise_Linux/6/html/Deployment_Guide/NetworkSetupInterface.html

Next you will initialize the appliance. See “Initializing the System” on page 27.

Related Documentation

- [Initial Appliance Configuration on page 24](#)
- [The CLI: The Set Command on page 75](#)

Initializing the System

After the network interface(s) have been configured and the hostname is set, you are ready to initialize the appliance.

1. From within the CLI, enter the following command:

```
system initialize
```

2. Next, you are prompted to confirm that the management interface is set correctly and then a warning screen appears, letting you know that the initialization process will erase all traffic related data on the local appliance. Once you have confirmed that you understand the warning, the local system begins the reset process which wipes out all local data and attempts to get the local system into a clean state.

3. Once the reset process completes, you are presented with a dialog box that allows you to select the mode that the appliance will be operating in. The mode selections are as follows:
 - Standalone, Master, Dedicated Master—If you are setting up a Standalone appliance or a regular Master appliance, simply select the appropriate mode and hit enter. The initialization process will run and that's all that is required. See [“Clustering Overview” on page 20](#) and *Node Types* and for more information.
 - High Availability Modes—For HA modes, you are asked the addresses of the two nodes that will be part of the HA pair. The addresses given should be the IP addresses assigned to the interconnect interfaces of the two HA nodes. These are the addresses that will be used for data replication. Once you have given the two addresses a quick validation process will run to make sure the two nodes are suitable to run in a HA pair together. If all validation passes then you will be asked for the virtual IP address and netmask to use as the floating address between the HA nodes. This virtual IP and netmask should be within the network of another interface already configured on the system. Otherwise it will not know what physical interface to associate to. Basic validation will be done on the provided information. If validation passes, then the HA connection will be established and the reset and initialization process will be run on both participating appliances. Once both nodes have completed the initialization process, they will go into a sync period where they will attempt to sync the replication partition between the two nodes. The time this takes can vary depending on the speed of the interconnect, but on average it should take approximately 30 minutes to complete. See [“High Availability Overview” on page 21](#) and [“High Availability Settings” on page 32](#) for more information.
 - Traffic Processor—If you are configuring a traffic processor, you will be asked for the address of the master node to use for the traffic processor. Provide the IP address of the management interface on the master node and that's all that is required. The initialization process will validate that the local appliance and the given master address will work together, and if so, the initialization will proceed as normal.

At the end of the initialization process, you are presented with a dialog window that tells you whether or not the process was successful, and under certain circumstances, you may be given recommendations for other settings to change within the configuration. At this point, you should have a fully set-up appliance.

- Related Documentation**
- [Configure Network Interfaces on page 26](#)
 - [Set the Hostname on page 27](#)
 - [Verify Connectivity on page 28](#)

Verify Connectivity

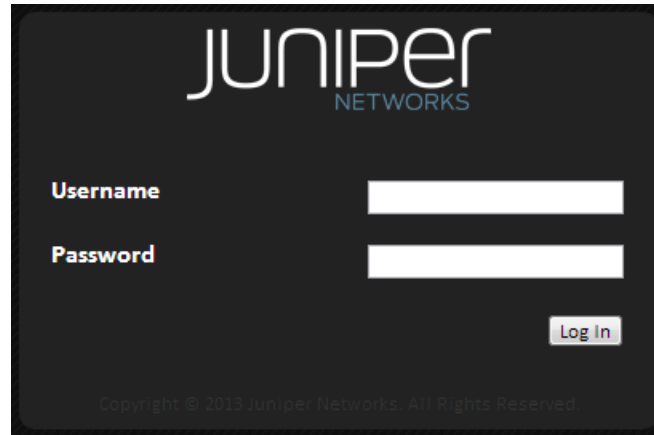
- Purpose** After all initialization steps have been performed, verify that all network settings are correct, and that the appliance can be reached from the network. Navigate to the IP or hostname of the management interface (on SSL), and specify port 5000.

Action For example:

- `https://10.10.10.104:5000`
- `https://my-hostname:5000`

If you see the appliance login screen, network settings are correctly configured.

Figure 14: Appliance Login Screen



Related Documentation • [Install the License on page 29](#)

Install the License

In order to complete WebApp Secure configuration, you must install the license for the product. Use a web browser to connect to your appliance on port 5000 and log in using the Administrator credentials. These are the same credentials used for SSH / console access and for the Web UI.



NOTE: The configuration URL is: `https://<IP address or hostname>:5000`

Examples: `https://10.11.12.13:5000` and
`https://webappsecure.mydomain.com:5000`

Go to the licensing section and follow the prompts:

1. Enter the license key provided to you in the **Add a New License** field.
2. Click **Add**.
3. Review the Terms of Service.
4. Click **Yes** on "I agree to the terms of service" to enable your WebApp Secure product.

Figure 15: License Terms

Licensing

License Key: MSAX-D3ED-17AD-5F3C-18AD

Please read the licensing terms before proceeding.

Juniper Networks License Terms

1. Definitions

In addition to terms defined elsewhere in this Agreement, this Agreement uses the following defined terms.

1.1 **"Approved Hardware"** means any physical device, system, sub-system, or component thereof that is certified by Juniper for the purposes of operating Licensed Software and/or Juniper Technology

1.2 **"Bug Fix"** means any modification or addition to any Licensed Software by Juniper, which is intended to correct Errors or other unwanted or unintended conditions that cause the Licensed Software to fail, malfunction, or operate in a manner other than as anticipated or desired.

1.3 **"Documentation"** means the technical manuals, user guides, and other information, if any, which are made available by Juniper to Licensee with respect to the Juniper Technology or other Licensed Software, together with modifications and updates thereto, whether in printed or machine-readable form.

1.4 **"Enhancement"** means any technology that enhances, improves, or otherwise modifies any Licensed Software.

1.5 **"Error"** means an error in any Licensed Software which significantly degrades the Licensed Software as

☒ I agree to the terms of service

☒ Send hardware details to Juniper customer support?

Save License

If the license validation step fails, check the network settings, particularly proxy settings for the network. WebApp Secure must reach the outside world to contact the licensing server.

Configuring High Availability

Setting up HA Networking

Before you can initialize the system as a HA pair you must first set up the network to handle the appliances.

1. Login to the appliance as the mykonos user.
2. Set the hostname of the appliance.

```
cli system set hostname <hostname>
```

3. To configure the interconnect bond, configure eth4 and eth5 (the 10G ports) to be slaves to bond0.

```
cli system set interface eth4 slave true master bond0 onboot yes
```

```
cli system set interface eth5 slave true master bond0 onboot yes
```

4. Set up the bond.

```
cli system set interface bond0 bootproto static ipaddr <interconnect_ip> netmask  
<interconnect_netmask> onboot yes
```

5. Configure the management interface on eth0, eth1, eth2 or eth3.

```
cli system set interface <interface> bootproto static ipaddr <ip> netmask <netmask>
onboot yes
```

6. Configure the DNS domain.

```
cli system set dns domain <dns_domain>
```

7. Configure the DNS search domain.

```
cli system set dns search <search_path>
```

8. Configure the DNS nameservers.

```
cli system set dns nameservers <dns1>,<dns2>
```

9. Assign the interface configured previously as the management interface.

```
cli system set management-interface <interface>
```

10. Restart networking.

```
cli system services restart network
```



NOTE: Repeat this process for each of the two appliance instances.

Pair the Nodes

Now that the networking is setup on the appliances, they are ready to be paired into a HA system.

1. Enter **cli system initialize** to enter the initialization process.
2. A prompt asks you to verify **<interface>** as the management interface. Confirm, and wait for the process to finish.
3. When the Select System Mode menu appears, select **HA Master** as the mode. Click **OK**.
4. When the Select Interface menu appears, select the previously configured management interface as the interface (**<interface>** in the previous section).
5. When the Select Interface IPs menu appears, enter the two appliance IPs (**<ip>** in the previous section).
6. When the initialization prompts you for the VIP address, enter the VIP address and netmask. Wait for the process to finish.
7. Set the NTP server.

```
cli config set services.ntp.servers <servers>
```

The HA system should now be complete.

To update an HA system, navigate to the management interface (**https://VIP:5000** where VIP is the Virtual IP) and update as described in the System Updates section. The update will be applied to both systems in the HA pair.



NOTE: While both the active and passive machines must be on the same WebApp Secure version to be initially configured in HA mode, appliances already in HA mode will successfully update together.



NOTE: In the event that one node of a High Availability pair becomes unusable, it might be necessary to tell WebApp Secure to rejoin the two instances. For example, if one node in an HA pair gets initialized (effectively severing the HA setup) you can run the command `sudo mykonos-join <master-ip>` on the initialized machine. This will re-link the two WebApp Secure instances, enabling the HA pair to become functional again.

**Related
Documentation**

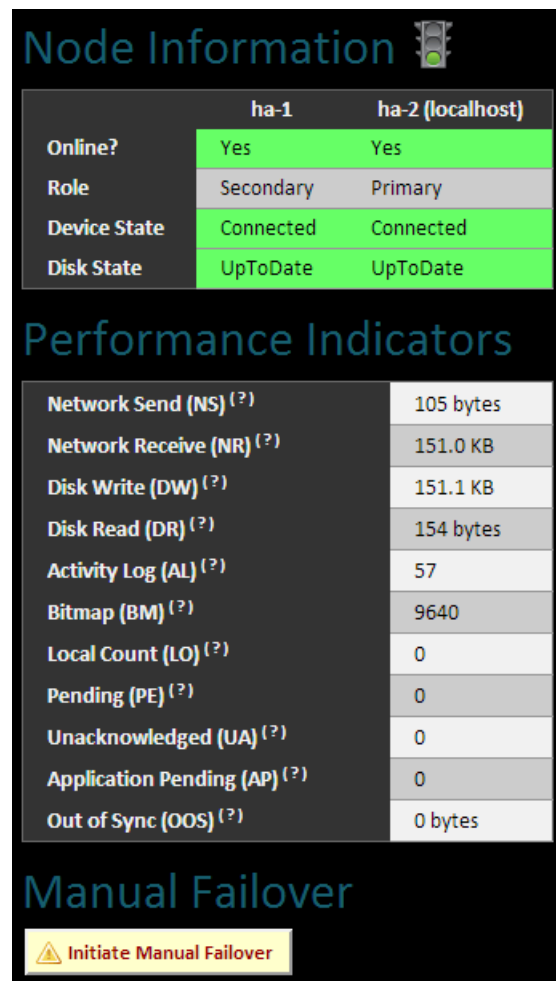
- [High Availability Overview on page 21](#)
- [High Availability Settings on page 32](#)
- [High Availability Network Failure Detection, Actions, and Monitoring on page 95](#)

High Availability Settings

You configure HA settings during the initialization process. See [“Initializing the System” on page 27](#). You can use the CLI to change HA settings.

After allowing the Initialization process to complete, you can verify proper HA setup by navigating to the management interface **`https://VIP:5000`** where VIP is the Virtual IP. Navigate to **High Availability** on the left-side menu to observe the status of the HA pair.

Figure 16: HA Pair Status



WARNING: Since the various HA appliances in a configuration need to interface with the database, port 5432 will be open. Be sure to restrict access to this port with your firewall to prevent unwanted incoming connections. WebApp Secure is not intended to be used as an edge device.



NOTE: If the interconnect between an HA pair drops at any point, it is possible that both systems will try to assume the active system role. This leads to a condition known as split-brain, where data is not properly routed through the pair. To mitigate this, we recommend that you bond the pair using the 10Gb ports on the front of the appliance. See [“The CLI: The Set Command” on page 75](#) for information on setting up the bond interface.



NOTE: You must use the VIP to access the configuration interface. If you attempt to use the management interface on the passive appliance, you will see a notification indicating "The Administrative interface is not accessible on this host because it is the secondary host in a High Availability cluster."

**Related
Documentation**

- [High Availability Overview on page 21](#)
- [Configuring High Availability on page 30](#)
- [High Availability Network Failure Detection, Actions, and Monitoring on page 95](#)

Configuring Clustering



NOTE: Unlike a traditional cluster, a WebApp Secure cluster does not automatically balance traffic between each of the nodes. For this reason, a load balancer is required to be configured to send traffic to each traffic processing node.

Setting up a cluster is as easy as configuring multiple stand-alone boxes. The first step is to set up the master. You must set up the master first because you will need to supply the master's IP when initializing the traffic nodes.

Initialize the appliance like you would any stand-alone appliance - by typing **system initialize** from the WebApp Secure CLI. When the prompt for Select System Mode appears, select **Master** from the list of available appliance modes. Wait for the process to complete.

The setup process will initialize the master, and once the initialization is complete, you should be able to navigate to the management interface at **https://HOSTNAME:5000**. Once the master is initialized, you can initialize the other appliances as traffic processing nodes. The steps are similar to the master setup, however you will be prompted to enter the IP of the master node after selecting **Traffic Processor** as the appliance mode on the additional appliances.

When using a clustered appliance configuration, it is important to configure a valid listening IP address on the Traffic Processor for every application that will use SSL. You can configure a listening interface by navigating to the Traffic Processor's terminal, and using **system set interface** from within the WebApp Secure CLI. For more information on how to configure an interface, see "[The CLI: The Set Command](#)" on page 75. The application must also be bound to the IP address assigned to that interface. This can be achieved through the application configuration settings in the Web UI. To learn how to create SSL applications, see "[Enable SSL to the Client](#)" on page 60.

Once the traffic node is initialized, you can verify the cluster by navigating to the management interface (**https://HOSTNAME:5000**) and clicking on **System Stats**. There should be a separate tab for each node in the cluster, and an additional tab for the aggregate cluster data.



NOTE: Remember that you must use an external load balancing solution to point to each traffic processing node, as the cluster will not do this for you.

Related
Documentation

- [Updating the Cluster on page 35](#)

Performing Initial Updates

Before you perform any additional configuration steps, you should navigate to the **Updates** window of the Web UI. Click the **Check for Updates** link and install any available updates for your system. While reasonable efforts are taken to support updates on a running system, you will always have less risk performing system updates before your system goes in to production.

Related
Documentation

- [Updates on page 271](#)

Updating the Cluster

Updating a cluster is similar to updating a stand-alone box. Navigate to **Updates** in the management interface on the master node (the traffic nodes have no management interface) and apply the updates as you would on an individual appliance. The master node will automatically apply the same update to each of its traffic processing nodes in the cluster. There is no need to individually update each appliance.



NOTE: The process for updating a cluster will take longer than updating a single appliance because the same update must be applied to each node.

About the Configuration Wizard

The configuration wizard helps you configure the most commonly used features on the WebApp Secure appliance, including global backend servers, SMTP settings, alerts, and backups. A banner at the top of the Web UI will advise you to run the wizard until your global backend servers are set.



WARNING: Upon completion of the wizard, make a note of your backup encryption key. If you lose this key, no one - including Juniper Support, can retrieve the information contained in your backups.



NOTE: The wizard has approximately 6 steps. The actual number of steps can increase or decrease depending on your choices within the wizard.

**Related
Documentation**

- [Using the Configuration Wizard on page 36](#)

Using the Configuration Wizard

Use the configuration wizard to set up the following:

Global Backend Servers—WebApp Secure can protect an unlimited number of web applications, each with their own backend server(s). In a separate section, the process for over-riding backend servers for each application is described. However, you must define at least one server at the global level. This server will service requests that reach WebApp Secure but do not match a configured application.

- **Server Name**—A unique name that WebApp Secure uses to identify this server. The name can include any alphanumeric character, "-", and "_", with no white space. Do not use the server's Fully Qualified Domain Name (FQDN) or a URL. If you are using VMware, you can use the same name here as you assigned in VMware to avoid confusion. But that is not required.
- **Server Address**—Specify the server's IP. WebApp Secure does not support IPv6 addressing at this time.
- **HTTP Port**—Usually port 80.
- **HTTPS Port**—Usually port 443.
- **Weight**—The default is 1. This value is used when WebApp Secure is serving as a software load balancer and represents the relative weight the server has for balancing purposes.
- **Backup**—The default is NO. This only applies if you are using WebApp Secure as a software load balancer, and you are designating this server as a backup.

SMTP Server—WebApp Secure can e-mail alerts to your administration team. While the appliance can serve as its own mail server, we recommend that you use a valid mail server for your network.

The following SMTP server configurations are supported for e-mail alerts.

- No Security by SSL=False and blank User/Password
- SASL, No TLS: by SSL=False and specified User/Password
- SASL and TLS: by SSL=True and specified User/Password
- No-SASL but TLS: by SSL=True and unspecified User/Password

Use the Wizard to configure SMTP servers, or in the main Web UI, navigate to **Configuration > Services > SMTP settings** and configure the following:



NOTE: After you enter the server information, click the **Test SMTP Connection Settings** link to make sure the server can be reached.

- SMTP Server Address—Defaults to localhost. Set it to the IP address or FQDN of your mail server if you are using an off-board mail server as recommended.
- SMTP Server Port Number—Defaults to 25. Set it to the port your mail server is listening on.
- SMTP Username—Defaults to blank, and can remain blank if you are using the on-board server. Set it to a user with valid access to the mail server.
- SMTP Password—Defaults to blank, and can remain blank if you are using the on-board server. Set it to the password for the SMTP username.
- SMTP Server SSL—True or False—Set whether or not the SMTP server uses SSL for connections.
- SMTP Server Timeout—This field defaults to 3000. This indicates how long to try and send a message before the connection to the SMTP server times out in milliseconds.
- SMTP Server Debug Mode—True or False—Whether to log all SMTP connection and traffic details. This is very verbose. It is recommended that you only enable this if you're having issues with emails not being issued.
- Sender Email Address—The email address that emails will use in the from field.

Figure 17: Wizard, Configure SMTP Settings, Step 3

STEP 3 OF 5: SMTP SETTINGS

Various components of Mykonos Web Security send email. If you'd like to use the built-in SMTP server, you can just click 'Next', but if you have an MTA on your network already, please specify the details below.

SMTP Default Sender:	<input type="text" value="support@mykonossoftware.com"/> <i>Will be used as the default 'from' address for email sent by the server. Address should be a valid sender for your domain to avoid being miscategorized as spam.</i>
SMTP Server Address:	<input type="text" value="localhost"/> <i>Enter the IP address or FQDN of your SMTP server.</i>
SMTP Server Port:	<input type="text" value="25"/> <i>If your SMTP server is on a non-standard port, enter it here.</i>
SMTP Username:	<input type="text"/> <i>If your SMTP Server requires authentication, enter the username here.</i>
SMTP Password:	<input type="password"/> <i>If your SMTP Server requires authentication, enter the password here.</i>

Alert Service—WebApp Secure can send alerts to an SNMP server or by e-mail to appropriate personnel. The alert service is optional, and defaults to No. If you choose not to activate alerts, the Wizard skips to the next section.

Figure 18: Wizard, Configure Alert Service , Step 4

STEP 4 OF 6: ALERT SERVICE

Mykonos Web Security can be configured to send email alerts or SNMP traps to administrators when an incident of a specified severity is detected. For instance, we could send out an email notification to a specific admin who is on-call if an incident level of "critical" is detected, allowing the admin to respond quickly to the threat.

Send Alerts? ☒ YES

The alert service can send SNMP traps or email system administrators when certain activity is detected. If you leave Alerts turned off, these will be logged *only*.

←back next→

Figure 19: Wizard, Configure Alert Service , Step 5

STEP 5 OF 6: ALERT SERVICE

If you'd like to receive SNMP traps when alerts happen, enter the number of SNMP servers that will receive these traps, or '0'. If you'd like to have administrators receive email notifications when alerts happen, enter the number of administrators who will receive these alerts, or '0'.

Number of SNMP Servers:
If you'd like to send SNMP traps, how many servers will receive them?

Number of email contacts:
If you'd like to send alerts via email, how many administrators will receive them?

←back next→

If you choose to activate alerts, you have the option of setting up the number of SNMP servers to alert and the number of e-mail addresses to which messages are sent. The default values to both are 0.

Figure 20: Wizard, Configure Alert Service , SNMP, Step 6

STEP 6 OF 8: ALERT SERVICE SNMP SETTINGS

For each SNMP server, please enter an IP address or FQDN, and a port, if other than the default.

Server Address:
Specify a hostname or IPv4 address.

Port:

←back next→

If you activate SNMP Alerts, the wizard prompts you for the server address and the port to which alerts are sent.

If you are configuring e-mail alerts, the following fields are required:

- **Name:** A common name for referencing this e-mail address.
- **Email Address:** Email address.
- **Minimum Severity:** Minimum severity level to trigger an e-mail alert to this address.
- **Shift Start:** Start time for this address in 24 hour format.
- **Shift End:** End time for this address in 24 hour format.

You are also given the option of having alerts sent on the weekend. You can build complex schedules by creating multiple entries for the same person. For example, admin@yourcompany.com could have an entry named admin-weekday that specifies 8 AM to 5 PM, M-F, and a second entry named adminweekend that specified 6 AM to 6 PM.

Figure 21: Wizard, Configure Alert Service , Email Contacts, Step 7

STEP 7 OF 8: ALERT SERVICE EMAIL CONTACTS

Please specify the details for each Alert Contact, including their approximate on-call schedule. This information can be changed at any time, after the wizard is completed.

Name:	<input type="text"/>
Email Address:	<input type="text"/>
Minimum Severity:	High <input type="button" value="v"/> <small>The minimum severity that the contact will receive alerts for.</small>
Shift Start:	<input type="text"/> <small>Earliest time that alerts will be sent to this contact (use 24-hour time).</small>
Shift End:	<input type="text"/> <small>Latest time that alerts will be sent to this contact (use 24-hour time).</small>
Sunday:	<input checked="" type="checkbox"/> YES <small>Whether or not alerts will be sent to this contact on Sundays.</small>



NOTE: Configuration of advanced features, such as encryption keys, are not available in the wizard.

Backups—WebApp Secure can perform regular, scheduled backups of all data. You can select backups using FTP or SSH.

Figure 22: Wizard, Configure Backup Service

STEP 7 OF 7: BACKUP SERVICE

Please specify how you would like backups to be conducted.

Frequency: How often would you like backups to be performed?

Retention: Number of days that backups will be kept before being deleted.

FTP: ☐ **NO** Would you like to push backups to an FTP Server?

FTP Server:

Username:

Password:

SSH: ☐ **NO** Would you like to push backups to an SSH Server (via SCP)?

SSH Server:

Username:

Password:

[<-back](#) [next->](#)

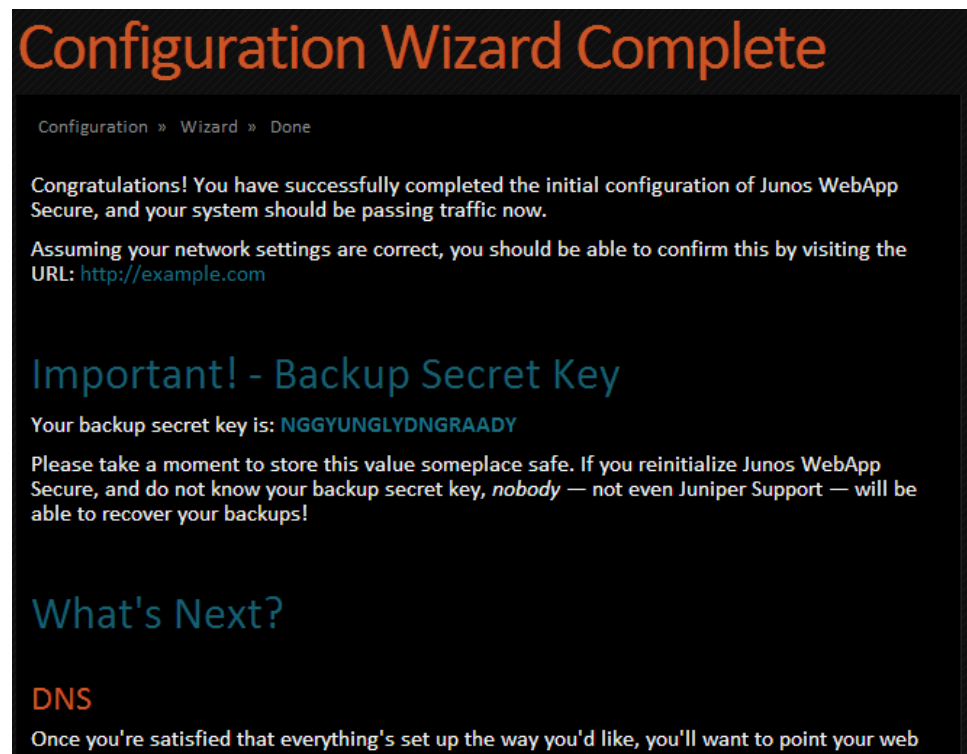
The backup service lets you specify the following fields:

- **Frequency:** How often backups are sent off-board.
- **Retention:** Number of days to keep off-board backups.
- **FTP Service:** Whether to use FTP. If set to YES, the server, username, and password fields are required.
- **SSH Service:** Whether to use SSH. If set to YES, the server, username, and password fields are required.

Spotlight Secure—Spotlight Secure provides a way to import malicious profiles from other subscribers to the service. The service is licensed separately and is enabled by default, but you can choose to disable it.

Wizard Confirmation Page—Once you have completed the wizard's main steps, you will see the confirmation page. Here, there is a URL you can use to confirm that the appliance is performing correctly. You will also see the secret key the appliance generated for your backups. Whether WebApp Secure is storing backups locally or off-site, you must have this key.

Figure 23: Wizard, Confirmation Page



NOTE: The key is actually a link. You may change the value of the key by clicking this link.



NOTE: Record the secret key and keep it someplace safe. If you run through System Initialization again, it will create a new key and you will lose access to your backups if you haven't recorded the old key. If you lose this key, Juniper Support will not be able to recover it or your backups.



NOTE: It is also worthwhile to record other configuration entries in the event that you perform a configuration re-initialization. `engine.session.encryption_key` and `engine.session.initialization_vector` are entries needed to maintain the data of currently active users on the protected application. It is best practice to write these down, as well. Once configuration initialization is done, these old values can be set again.

Using WebApp Secure with Third-Party Load Balancer

If you are using the appliance with a third-party load balancer, you must make sure to tell WebApp Secure to accept the X-Forwarded-For header from the load balancer. If

this is not set, all IPs in the appliance will seem to be coming from the load balancer directly.

To trust the X-Forwarded-For header, SSH into the appliance, and enter the following:

```
sudo mykonos-shell config set engine.exclude_forward_addresses
<IP_of_Loadbalancer>
```

This tells WebApp Secure to trust the header of your load balancer.

Verify the Installation

In order to verify that your WebApp Secure appliance is processing traffic, use the following URL to access the appliance honeypot and confirm that it replies with a fake .htaccess.

http://<IP or Hostname>/.htaccess

The appliance should reply with something similar to the following. (Note that the actual fake .htaccess file might not look exactly like the example.)

```
<files "server_logs.txt">
  AuthUserFile /www/root/.htpasswd
  AuthType Basic
  AuthName "Error logs"
  Require valid-user
</files>
```

Your initial WebApp Secure configuration is complete. The appliance is ready to start protecting your applications.



NOTE: At this point, WebApp Secure is configured to secure one webserver application.

CHAPTER 4

Configuring WebApp Secure

- [Web Interface Configuration Overview on page 44](#)
- [Edit Web UI User Preferences on page 44](#)
- [View Online Help and Product Documentation from the Web UI on page 45](#)
- [Basic Configuration Mode on page 46](#)
- [Expert Configuration Mode on page 47](#)
- [Import/Export \(Web UI\) on page 48](#)
- [Security Engine Configuration on page 48](#)
- [Configure Support for Akamai Dynamic Site Accelerator on page 49](#)
- [Security Engine Incident Monitoring on page 50](#)
- [Security Engine Server Identity and Cloaking on page 52](#)
- [Security Engine Traffic on page 52](#)
- [Security Engine Whitelist Settings on page 53](#)
- [Proxy/Backends on page 54](#)
- [Applications Overview on page 55](#)
- [Create a New Application on page 56](#)
- [Edit Applications on page 58](#)
- [Application Patterns on page 58](#)
- [Application Backend Overrides on page 60](#)
- [Enable SSL to the Client on page 60](#)
- [Pages on page 62](#)
- [NTP Service on page 62](#)
- [Alert Service on page 63](#)
- [Integration with SRX Series Overview on page 63](#)
- [Filters and Terms Configuration Summary for SRX Series Integration on page 64](#)
- [Creating SRX Series Filters and Terms on page 65](#)
- [Configure the SRX Series Integration on page 66](#)
- [Testing the SRX Series Integration Configuration on page 68](#)

- [Role-Based Administrator Access Control on page 69](#)
- [Configuring Role-Based Access Control on page 70](#)

Web Interface Configuration Overview

The Web interface is used for system configuration, as well as monitoring and reporting. The initial installation required you to access this interface to license WebApp Secure and to bring your appliance on-line. You will use this interface for nearly all configuration options.

The web interface URL is: **https://<IP address or hostname>:5000**

For example:

- **https://10.11.12.13:5000**
- **https://webappsecure.mydomain.com:5000**

Edit Web UI User Preferences

User Preferences control the appearance of the user interface and how certain information is displayed. Click the **Edit Preferences** link at the top right of the Web UI to access the User Preferences screen. The following preference settings are available:



NOTE: Changes only apply to the currently logged in user.

- **Skin:** Change the color and overall look of the Web UI.
- **Language:** At this time, only English is supported.
- **Timezone:** Change the timezone setting. Note that this field defaults to UTC.
- **Prompt Level:** Change the amount of help text displayed for each field. If you are familiar with the product, you might prefer abbreviated help text to lessen the amount of information on the screen.
- **Spotlight Name Preference:** If Spotlight is enabled, you can select to have Spotlight global names displayed in attacker lists and reports. You can also choose to display only local names or to display both local and global names.
- **Auto Refresh:** You can enable or disable this setting. Note that Auto refresh affects all security related screens, including the dashboard, lists of hackers, sessions, locations, and incidents.
- **Refresh Interval:** Change the refresh interval. The minimum value you can set here is 10 seconds.
- **Records Per Page:** Change the number of records to display on a per page basis.
- **Debug Mode:** You can enable or disable this setting. Certain Web UI items are hidden by default. For debugging purposes, you can enable this checkbox to reveal all hidden items.

Figure 24: User Preferences

User Preferences

Skin	Dark (Default)	To accommodate aesthetic preferences, you may select an alternate skin.
Language	English (US)	Where available, we will use localized text throughout the system.
Timezone	UTC	By default, items in the User Interface are shown in UTC. To change this, set this value to your preferred timezone.
Prompt Level	Abbreviated	By default, help text is shown inline. If you are familiar with the product, you may wish to have an abbreviated prompt level.
Spotlight Name Preference	Display Global Names If Available	If spotlight is enabled, we will use this setting to determine how to display attacker profile names to you, throughout the UI.
Auto Refresh	<input checked="" type="checkbox"/>	Enable auto refreshing of datasets displayed on pages. (Requires JavaScript)
Refresh Interval	300	How often, in seconds, automatic refreshes will occur, if 'auto refresh' is enabled.
Records Per Page	15	The default number of records returned per page. Used for all lists of data throughout the entire UI.
Debug Mode	<input type="checkbox"/>	Certain items in the user interface can be usable for troubleshooting, but are hidden by default. To see them, set this value.

[Save](#)

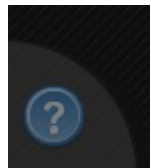
View Online Help and Product Documentation from the Web UI

Online Help (question mark)—When you click the question mark in the bottom left corner of the Web UI, a help screen for the selected configuration item appears (context-sensitive help). By default, the help screen displays within in the Web UI itself. You can make the help appear in a new browser window, with a TOC, Index, and Search capabilities, by holding down the **Shift** key when you **click** the question mark.



NOTE: Context-sensitive help is not available for all tasks at this time. When it's unavailable, the question mark does not appear.

Figure 25: View Online Help



Product Guides (Help book icon)—When you click **Help** at the bottom of the navigation menu in the Web UI, a window containing links to the product documentation appears. From this window, you can view HTML and PDF versions of the following:

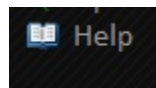
- Administrator Guide (this guide)—Contains configuration and administration information required by most administrators.

- **Developer Guide**—For developers who want to programmatically integrate with WebApp Secure. This guide contains information about the REST API and writing custom autoresponse rules.
- **Third-Party Attributions**—This document contains license information for all third-party or open-source code contained within WebApp Secure.



NOTE: The Developer Guide and Third-Party Attributions are also available in plain text.

Figure 26: View Product Guides



**Related
Documentation**

Basic Configuration Mode

By default, any configuration page navigated to will result in the Basic Configuration page for that particular section. You can view the various sections of Configuration underneath the **Configuration** page on the left side navigation. The available sections are as follows:

- **Security Engine:** Core Security Engine options, such as health checks, and whitelisting.
- **Processors:** Security Processors are pluggable modules that process HTTP traffic and perform actions.
- **Services:** Services run in the background, performing tasks such as sending alerts, generating reports, or performing maintenance tasks.
- **Proxy / Backends:** Core proxy settings, such as backend servers and SSL.
- **Applications:** By default, the system will secure only one application. Adding multiple profiles will enable you to protect multiple applications with their own separate settings.
- **Backups:** Configure backup frequency, retention, and pushes to FTP or SSH servers.
- **Logging:** Options for logging access on the management interfaces, as well as logging the various security incidents triggered by WebApp Secure.
- **Response Rules:** Configure how the system responds to threats, or create custom response rules.
- **Licensing:** Add or update licensing information to ensure operation of your system.
- **Users and Groups:** Add or update user roles and permissions.
- **Rest API:** The WebApp Secure REST API provides programmatic access to the data available in the system. More information can be found in the WebApp Secure Developer Guide.

Related Documentation • [Expert Configuration Mode on page 47](#)

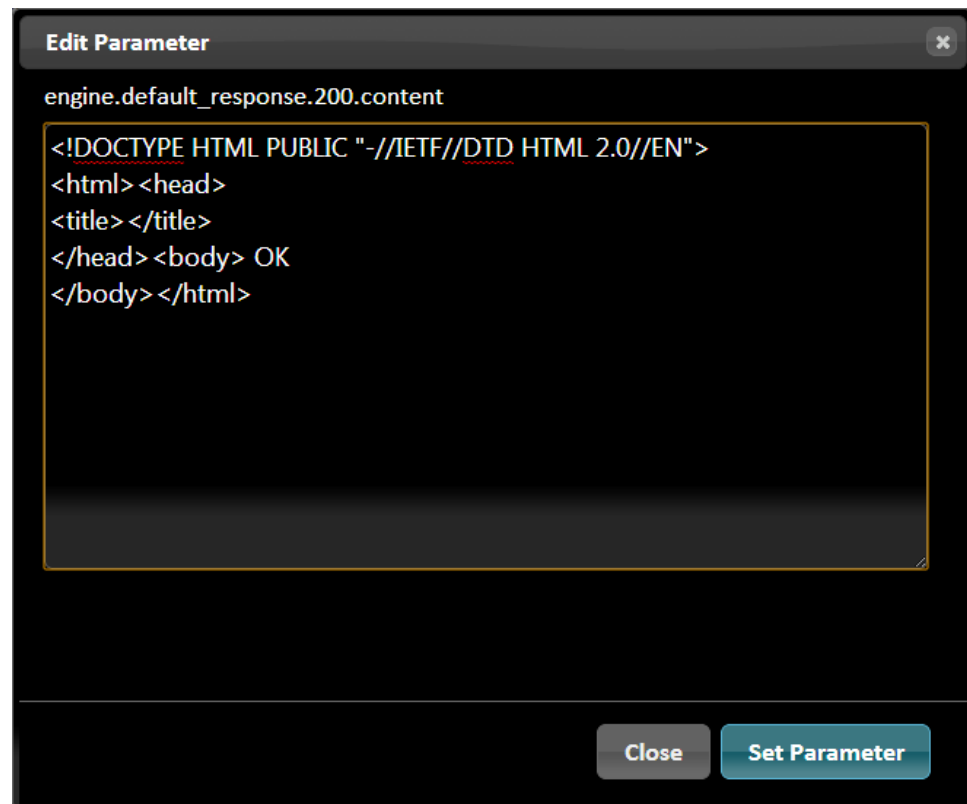
Expert Configuration Mode

In most cases, using the standard configuration interface should be sufficient. Some users might prefer editing the configuration parameters at the key-value level. Expert Mode is a way to view all the configuration attributes from the Web UI. You can reach Expert Mode window by clicking the **Expert Mode** button on the upper right side of the **Configuration** window in the Web UI.

To edit any configuration parameter, first navigate to the correct parameter name. The table is ordered alphabetically, and you can browse through the help documentation for various parameters by using the **help** keyword in the CLI (mykonos-shell).

Once you find the entry, you can edit it, remove it, or reset it to its default value using the icons on the left side of the table. When editing a parameter, you are given a text box in which to make the edit. Some parameters are Base64 encoded (like HTML responses), but will be displayed in an un-encoded form. Make your changes and click **Set Parameter** to save the changes.

Figure 27: Edit Parameter





WARNING: Even after you save the parameter, the changes to configuration have not been saved and set until you click the **Save** button at the bottom of the page. If you navigate away from the page before saving the entire configuration, parameters are not saved.

Related Documentation

- [Basic Configuration Mode on page 46](#)

Import/Export (Web UI)

You can export an XML file containing all configured parameters. Navigate to **Configuration** in the Web UI and click the **Import/Export** button. Then click the **Download** button to begin downloading the XML file.

You can also import an XML configuration file by clicking the **Import** button and browsing to the file. Note that the imported file will overwrite your existing configuration.



NOTE: A configuration export does not contain system-level settings, such as networking configuration and license information. Custom autoresponse rules are also excluded from a configuration export.

Related Documentation

- [CLI: Config: Import/Export on page 88](#)

Security Engine Configuration

The security engine is the core of WebApp Secure. It is responsible for parsing incoming HTTP requests, tracking session and attacker information, and running the processors.

You can disable the entire security engine if necessary for troubleshooting by clicking the **Disable** button.

Related Documentation

- [Configure Support for Akamai Dynamic Site Accelerator on page 49](#)
- [Security Engine Incident Monitoring on page 50](#)
- [Security Engine Server Identity and Cloaking on page 52](#)
- [Security Engine Traffic on page 52](#)
- [Security Engine Whitelist Settings on page 53](#)

Configure Support for Akamai Dynamic Site Accelerator

You can configure WebApp Secure to work with a site that utilizes Akamai Dynamic Site Accelerator. You will need to make minor changes to your site's configuration in the Akamai Luna Control Center and in the Content Delivery Network section of the Security Engine configuration screen in the Web UI.

To make the necessary changes, do the following:

1. Log into Luna Control Center and select the **Configure** tab.
2. Click the link corresponding to the desired site configuration under **Configuration Name**.
3. On the next screen, find the currently-active configuration and click **Create Version from...** in the right-hand column. Make the following changes:

Table 3: Luna Control Center Configuration Changes

Configuration Section	Parameter	Value
Honor HTTP Cache-Control and Expires Headers	Cache Control Headers	false (uncheck)
Honor HTTP Cache-Control and Expires Headers	HTTP Expires Headers	false (uncheck)
Browser Cache Control Headers	Pass through the origin's Cache-Control headers to the browser	true (select)
Browser Cache Control Headers	Pass through all origin cache control headers	true (select)
Edge Services - General	Enable True Client IP Header	true (check)
Edge Services - General	True Client IP Header Name	True-Client-IP (or other; see below)
Edge Services - General	Enable Edge Server Identification	false (uncheck)



NOTE: Choosing a name for the True-Client-IP header other than the default can provide additional security by preventing malicious users from spoofing this header. Make a note of the value chosen for the header. You will need to configure it on the WebApp Secure side.

4. After making these changes, scroll to the bottom of the page and activate the new Akamai configuration as you normally would.
5. Once you have verified that your new Akamai configuration has gone live, log into the WebApp Secure Web UI. If you are configuring Akamai support for an application, browse to that application's configuration page. Otherwise, browse to the **Content**

Delivery Network section of the **Security Engine** configuration (or use the Configuration CLI). Make the following changes:

Table 4: WebApp Secure Configuration Settings for Akamai Support

Parameter ID	Parameter Name	Value
engine.cdn.akamai.enabled	Akamai: Enabled	true
engine.cdn.akamai.true_client_ip	Akamai: True-Client-IP Header	(value specified in Akamai configuration)
engine.cdn.akamai.incidents.spoofing.enabled	Akamai: Spoofing Incident Enabled	true or false

- Set **Akamai Enabled** to **true** and **True-Client-IP Header** to the value that you configured in the Luna Control Center.



NOTE: If you want a security incident to be triggered when a client attempts to spoof a request through Akamai, you can enable the **Akamai Spoof Attempt** incident. This incident carries a severity of Medium and can be incorporated into custom Autoresponse rules.



NOTE: If WebApp Secure is configured to function alongside Akamai and a direct request comes in to the web server's backend, a warning will appear in mws.log, indicating "Unexpected direct access to origin server. This could be malicious or it could be origin site maintainers doing checkout." While this could be malicious, it could also be an indication that the site maintainer is doing work directly with the backend. It is always safe to confirm these direct backend requests with the webmaster.

Related Documentation

- [Security Engine Configuration on page 48](#)

Security Engine Incident Monitoring

While most incidents are triggered by processors, the security engine itself is responsible for several low-level incidents. These will be found in the Web UI under **Session Management** in the **Response Rules** page, and can be enabled or disabled through **Configuration > Security Engine > Incident Monitoring**.

The following settings are available from Security Engine Incident Monitoring window:

- **Session Tampering**—True or False. WebApp Secure uses an HTTP cookie as one of the components of its fingerprinting technology. Because the cookie has its own embedded digital signature, any attempt to fabricate or modify a session cookie will almost always result in a corrupted signature. If WebApp Secure detects that a cookie being provided does not have a valid signature, and does not follow the correct format, it will trigger a Session Cookie Tampering incident.

Default Response: Session Tampering (0004): 1x = Logout User, 2x = 1 Day Clear Inputs, 3x = 5 Day Clear Inputs

- **Session Spoofing**—True or False. WebApp Secure uses an HTTP cookie as one of the components of its fingerprinting technology. Because the cookie has its own embedded digital signature, any attempt to fabricate or modify a session cookie will almost always result in a corrupted signature. If WebApp Secure detects that a cookie being provided has an invalid signature, but otherwise uses the correct format, it will trigger a Session Cookie Spoofing incident.

Default Response: Session Spoofing (0001): 1x = Logout User, 2x = 1 Day Clear Inputs, 3x = 5 Day Clear Inputs

- **URL Path Fuzzing**—True or False. Whether or not to detect attempted fuzzing attacks by monitoring the URL Path for characters defined as invalid in RFC 3869.

Default Response: URL Fuzzing (0005): 3x = Slow Connection 2-6 seconds, 6x = Slow Connection 4-15 seconds, 10x = Escalated Fuzzing Attack Escalated URL Fuzzing Attack (0006): 1x = 1 Day Block

- **URL Fragment Fuzzing**—True or False. Whether or not to detect attempted fuzzing attacks by monitoring the URL Path for URL fragments incorrectly submitted to the server.

Default Response: Same as URL Path Fuzzing



NOTE: Both URL Path Fuzzing and URL Fragment Fuzzing incidents contribute to the count for the response.



NOTE: WebApp Secure is typically used to protect outward facing web sites on the public Internet. These resources all have fully qualified domain names to allow them to be reached by any client on the Internet. But in some cases, WebApp Secure may be used to protect an internal resource that does not have a fully qualified domain name. For example, when you are testing WebApp Secure on an internally available version of your web site which is soon to be released to the wide world. In this case, you should also include the parameter `engine.incidents.url_fuzzing.allow_locals` to your configuration through the use of Expert Mode. Set the value of `engine.incidents.url_fuzzing.allow_locals` to true and save the configuration. This will prevent false alarms coming from legitimate hits on your internally facing site.

Related Documentation • [Security Engine Configuration on page 48](#)

Security Engine Server Identity and Cloaking

One of the most important aspects of WebApp Secure's "Intrusion Deception" philosophy is in blending in with the protected web application. If attackers were aware of the presence of the product, its efficacy would be negatively impacted.

- **Fake Web Root**—Several processors use fake exposed configuration files. Where relevant, this directory will be interpolated into these resources.
- **Fake Server Name**—This value will be used to generate the "Server" HTTP header and can be used to mask the actual technology used. For example, if your backend server runs Apache, you can tell WebApp Secure to identify as Microsoft IIS, and an attacker will end up trying exploits for Microsoft IIS, which, of course, will not work against Apache.

Related Documentation • [Security Engine Configuration on page 48](#)

Security Engine Traffic

This configuration section contains several basic parameters that control how WebApp Secure processes HTTP traffic and parses HTML, as well as health checks and fingerprinting.

- **Default Character Encoding**—Backend servers should specify a character encoding via the Content-Type HTTP header or a byte-order mark. However, if this does not happen, or if the security engine does not recognize the character encoding as valid, the default character encoding is used to parse the response from the backend server.
- **Resolve Host Names**—Whether or not to perform DNS lookups on IP addresses present in the X-Forwarded-For header. While these lookups do happen out-of-band, they nonetheless may affect performance.
- **Track X-Forwarded-For Addresses**—Whether or not to track and record the addresses provided in the X-Forwarded-For header. These addresses will be added as "Proxy" locations for a session.
- **Health Check URL**—WebApp Secure provides for a unique URL, located on the root directory of any domain proxied through the security engine, to return a 200 response. The purpose of this is to provide a health check for the security engine itself, in such a way that the health check will not proxy through to the backend servers.
- **Traffic Fingerprinting Enabled**—For clients that do not accept HTTP cookies, WebApp Secure can fingerprint raw HTTP traffic. This will dramatically improve the association of traffic generated by non-browser clients like scripts or bots.

- **Session Cookie Expires**—One of the ways WebApp Secure tracks clients is with a standard HTTP session cookie. This parameter controls the expiration date of the cookie and should be a random date several years in the future.
- **Additional Address Tracking Headers**—Most proxy servers use the X-Forwarded-For header to send the IP addresses of each "hop" in the chain of proxies. If you are using a non-standard proxy server that uses an alternate header, you may specify it here.

**Related
Documentation**

- [Security Engine Configuration on page 48](#)

Security Engine Whitelist Settings

There are various types of whitelists to which you can add any valid IP address or CIDR block. To access the Whitelist screen, do the following:

Navigate to **Configuration > Security Engine > Whitelists**. The following types of whitelists are available:

- **Trusted IP Addresses:** The IP addresses in this list will not trigger incidents. Click **Add New** to enter IP addresses to be added to this list.
- **X-Forwarded-For Address Exclusions:** The IP addresses in this list are stripped of the X-Forwarded-For header. This effectively trusts that the next IP address in the chain is a genuine address. Click **Add New** to enter IP addresses to be added to this list.



TIP:

- If you only provide an IP address without a subnet mask, a /32 mask (i.e. a single IP address) is implied. If you provide any other mask, it will be used instead.
- You should enter values in a x.x.x.x/y format (the standard, dotted quad, ipv4 format).
- Any CIDR block entered in dotted quad notation with a decimal prefix 1-32 is valid (x.x.x.x/x).

Figure 28: Whitelists



Related
Documentation

- [Security Engine Configuration on page 48](#)

Proxy/Backends

The Proxy/Backends window let you customize how to handle proxying content to and from WebApp Secure. Request and response timeouts, listening ports, and compression settings can be set here in addition to other settings.

Configure or use the defaults for the following fields:

- **Load Balancer Method**—When using WebApp Secure as a load balancer for multiple application servers, this setting defines how the appliance segregates traffic. The pulldown list offers these selections:
 - Round Robin (default)—Attempt to contact all servers in a rotational fashion.
 - Sticky—Act 'Fair' for the first request of each incoming session, but route to the same server for each subsequent request using that session.
 - IP Hash—Act 'Fair' for the first request of each incoming IP, but route to the same server for each subsequent request using the same IP.
- **HTTP Server Port**—The port would you like exposed for incoming HTTP connections. It is recommended to leave this as the default :80
- **Max Request Size**—The maximum total size (in MB) for a request that will be allowed through WebApp Secure to the backend.
- **Request Timeout**—The timeout limit for client requests. Any request taking longer than this to transfer data to the server is terminated.
- **Response Timeout**—The timeout limit for server responses. Any response that takes longer than this timeout to transfer data to the client is terminated.

- Connection Timeout—The maximum time WebApp Secure will wait to get a connection from the backend server(s).
- HTTP Compression Enabled—True or False—Whether or not to compress (using gzip) responses as they pass through WebApp Secure

Health Check—Use the fields here to customize how health checks are sent to the servers.

- Request—The raw HTTP request to send to the server.
- Enabled—True or False—Whether or not to enable the Health Check function.
- HTTPS Enabled—True or False—Whether or not health checks should be performed over HTTPS.



NOTE: The health checks will still be performed over plain HTTP.

- Delay—The time WebApp Secure should wait between Health Checks.
- Fail Count—How many health checks fail before the server is marked as unavailable.
- Timeout—How long for WebApp Secure to wait for a response from the server while issuing a health check.

Servers—see [“Application Backend Overrides” on page 60](#).

Related Documentation

- [Application Backend Overrides on page 60](#)

Applications Overview

You may want to secure multiple web applications or web sites using the same deployment of WebApp Secure. To facilitate this, you can create multiple applications using the Web UI or Configuration CLI, each with their own configuration options.

All configuration items in the main sections of the Web UI are known as global parameters, except for the **Applications** section. Each configured application will inherit these global settings unless you specify that they should be overridden. Likewise, if you change a setting on the global level, it will be changed in every configured application, unless you have specifically overridden it.

Adding an application using the **Applications** configuration menu allows you to override certain parameters within the context of a given application. When you add an application, you tell WebApp Secure the following: Any requests whose hostname matches the regular expression you have specified should be subject to the values you have overridden within the context of the application configuration. To override a value, click **Applications** in the left navigation of the Web UI, and then click the corresponding link to edit the desired application. Not all parameters can be overridden; for example, backups-related configuration is global-only.

When using the configuration CLI, any parameter that does not start with **applications** is a global parameter. You may execute the **info** command to view information about

whether or not a given parameter can be overridden. To override a parameter for an application in the CLI, you use the application's slug, or short name, to create a prefix. For example, overriding "processors.basic_auth.enabled" for the application whose slug is "myapp", can be accomplished by executing the command **set applications.myapp.processors.basic_auth.enabled false**.

Since most discrete web applications are on different application servers, the most commonly overridden parameter is the collection of backend servers.

If your desired configuration requires further granularity, WebApp Secure also allows you to configure Pages which apply a regular expression to the path component of the URL, rather than the domain component, and allow a subset application of parameters to be overridden further.

As an example, let's say that you want to use WebApp Secure to protect `www.example.com` and `blogs.example.com`. You would set up two applications. The first would specify `www\example\.com` as the pattern, and you would specify your web server as the backend. The second would specify `blogs\example\.com` as the pattern, and you would specify your blog system's application server as the backend. Globally, since most of your infrastructure uses Apache, you have left the Basic Authentication Processor enabled on the global level. Your blog site, however, uses Nginx, and so a honeypot of a fake Apache configuration file would be a dead giveaway. Using the **Applications** configuration screen, you can override the setting to disable this processor for the blog site. Additionally, since the blog site uses WordPress, you might want to enable the Application Vulnerability Processor -- but not for your `www` site -- so you can use the **Applications** configuration screen to enable the Application Vulnerability Processor for the blog site.

Continuing the example, let's say that your webmaster is running a campaign using Google Analytics, and it is found that the Query String Processor is injecting a fake parameter that is conflicting with the tracking of this campaign, but only under the "fribjatz" directory of the `www` site. Adding a **Page** with a pattern of `"^/fribjatz(/.*)?$"` would enable you to turn off the Query String Processor for any pages on the site that are in the "fribjatz" directory.

Related Documentation

- [Pages on page 62](#)
- [Create a New Application on page 56](#)

Create a New Application

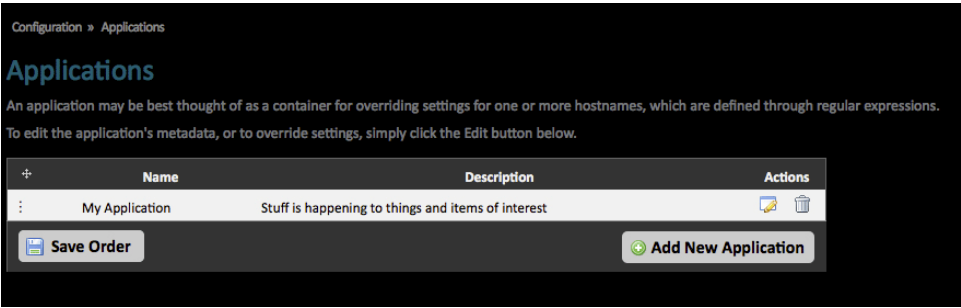
To create a new application within WebApp Secure, do the following:

1. In the Web UI, click **Configuration > Applications**. Here you can view applications that have already been configured.



NOTE: You can reorder applications in the applications list by dragging and dropping them. Click **Save Order** to preserve your newly ordered list.

Figure 29: Configured Applications



2. Enter information into the form to create a new application.
3. Click the **Add Application** button. This launches the Application Wizard. Fill in the available fields, and continue to click the **Next** button until you have completed the wizard. Based on your entries, the wizard can require three to six steps to complete.

Figure 30: Application Wizard

STEP 1 OF 3: APPLICATION METADATA

Please specify basic application metadata, including a friendly name, description, and a slug (short name used in the Configuration system).

Name:	<input type="text"/> <i>Please enter a human-readable "friendly name", to be used throughout the interface.</i>
Slug / Short Name:	<input type="text"/> <i>Please enter a unique, URL-friendly, short identifier comprised of lower-case letters, numbers, and / or underscores ONLY.</i>
Description:	<input type="text"/> <i>Please (optionally) enter a brief, human-readable description.</i>
Number of Patterns:	<input type="text" value="1"/> <i>Most applications only need one pattern. If you are unsure, just leave this '1' here. You can always add more later.</i>
HTTP Port:	<input type="text" value="80"/> <i>The TCP port this application should listen on for HTTP traffic (suggested value: 80).</i>
Configure SSL?	<input type="checkbox"/> <i>If you would like your application to be available over SSL, check this box.</i>

next→



NOTE: If you have already claimed the IP address for SSL, but wish to share it with other applications that are not using SSL, you need to go into the Proxy/SSL Settings section in the Applications window and select the listen IP address on the application that does not have SSL.

Related Documentation

- [Edit Applications on page 58](#)

Edit Applications

You can edit an existing application by doing the following. If you want to change any configured application settings, click the **Edit/Override Settings** icon for the application in question.

1. Navigate to **Configuration > Applications**. Existing applications are listed in the Application screen.
2. To edit an application, click the **Edit/Override Settings** icon under **Actions** for the application in question.
3. This takes you to the dashboard for the application. The dashboard displays which application settings are overridden and which are inherited. From here, you modify each setting individually.

Figure 31: Application Dashboard

Applications — My Application [Edit Name](#)

Application Description: Stuff is happening to things and Items of Interest [Edit Description](#)

PROXY / SSL SETTINGS [Edit these settings](#)

Listening IP Addresses (None)

HTTP Port 80 (inherited value (I))

SSL Not Configured

HEALTH CHECK [Override](#)

This application is using the health check defined at the global level.

URL PATTERNS

WebApp Secure uses regular expressions to determine which application settings apply to web traffic. You must have at least one pattern configured for an application to actually do anything.

HTTP	HTTPS	CI?	Host Regex	Actions
✓	✓	✓	^.*\$	Edit Delete

[Add New](#)

BACKEND SERVERS [Override](#)

You must specify which backend servers host your application if they are different from the ones you have defined globally.

This application is using the backend servers defined at the global level.

PAGES

If you need control over settings at a level even more granular than the application context permits, you may configure settings on a per-page level, based on regular expression URI matching. You currently have no pages configured in this application.

[Add New Page](#)

APPLICATION-LEVEL SECURITY ENGINE PARAMETER OVERRIDES

Parameter	Overridden?	Value	Actions
Default character encoding		ISO-8859-1	Edit
Session Spoofing	✓		Edit
Session Tampering	✗		Edit
Track X-Forwarded-For Addresses	✓		Edit

Related Documentation

- [Create a New Application on page 56](#)

Application Patterns

Application patterns determine which requests get routed to which applications. You can change each application you've added later by navigating to **Configuration > Applications**. Url patterns follow standard Perl Compatible Regular Expressions (PCRE) syntax.

For example:

- Any traffic: `^.*$`
- Any subdomain: `^.*\.domain\.com$`
- Multiple, or no, subdomains: `^((www|shop)\.)*domain\.com$`

Figure 32: URL Pattern

URL Pattern

The patterns used to check the URL to see if it belongs to this application

SUGGESTIONS

Catch All

Use Suggestion

Clear Fields

HTTPS Enabled

true

Whether to apply this pattern to HTTPS traffic

HTTP Enabled

true

Whether to apply this pattern to HTTP traffic

Host Regex

`^.*$`

Host Case Insensitive

true

Cancel

Save



NOTE: WebApp Secure processes applications in order. Therefore conflicting regular expressions will only be processed on the application or page where it first appears. There are also some suggestions that cover some common use-cases such as catch-all, subdomains, and so on.

Figure 33: Application Patterns

APPLICATION PATTERNS

Junos WebApp Secure uses regular expressions to determine which application settings apply to web traffic. You must have at least one pattern configured for an application to actually do anything.

https	http	host.regex	host.case_insensitive	Actions
true	true	<code>^111[.]222[.]111[.]222\$</code>	true	

Add



WARNING: URL patterns and profiles are observed in the order they are created.



NOTE: If SSL is required for this application, you will also need to enable SSL and ensure that all the required certificates are uploaded and configured properly.

Application Backend Overrides

When separating applications, one application can reside on a different physical server. You can define a backend server for this application here. For convenience, WebApp Secure inherits the global backend server that was used in the global context.



NOTE: You do not have to define a backend server at the global level. For example, you can define backend servers using only Applications. However, if there is a backend server defined globally, you should not unset it. You can only change it. There should always be at least one global backend server. Deleting the last global backend server can cause instability. (If you want to define backend servers using only Applications, you can skip this wizard setup step.)

Figure 34: Servers

BACKEND SERVERS						
You may specify the backend servers for this application here, or in the context of the "Security Engine" configuration screen. For convenience, we have inherited the value from the "Global" context. These may be deleted if desired.						
name	address	ports.http	ports.https	weight	backup	Actions
ALogicalNameHere	10.10.20.99	80	443	1	false	
Add						

Related Documentation

- [Proxy/Backends on page 54](#)

Enable SSL to the Client

To enable SSL between WebApp Secure and the client, do the following:

1. In the Web UI, navigate to the application for which you want to enable SSL or switch to the desired application's context.
2. Navigate to **Configuration > Applications > My App > Proxy/SSL Settings** and enable SSL to the backend.
3. Upload your SSL certificate and key file.
4. Select a listening interface IP address and HTTP and HTTPS ports.



NOTE: The combination of port/IP must be unique for the system. If the system is clustered, an IP must be selected for each node.

- When you save the SSL configuration in a deployment containing multiple appliances, the certificate is propagated from the master system to all subsequent systems.

Figure 35: Proxy / Backends

Proxy / Backends

HTTP Port
The TCP port this application should listen on for HTTP traffic (suggested value: 80).

HTTPS Port
The TCP port this application should listen on for HTTPS traffic (suggested value: 443).

IP Addresses
The IP address to listen on for HTTP and HTTPS. This list is populated automatically based on the IP addresses you currently have assigned to interfaces.

SSL Certificate

Since the system needs to decrypt SSL traffic to work, you will need to copy-paste your SSL Certificate (in PEM format) into this box.

SSL Key

Since the system needs to decrypt SSL traffic to work, you will need to copy-paste your SSL Private Key (in PEM format) into this box.



WARNING: To safeguard against inheriting SSL certificates, WebApp Secure does not allow SSL at the global level. Therefore, you must configure an application in order to enable SSL.



WARNING: Your certificate and key files cannot be password protected. If they are, WebApp Secure will be unable to read them. You can remove passwords on your existing certificate by using the openssl program. For example, `openssl rsa -in mykey.pem -out newkey.pem`.



NOTE: Certificates must be in valid PEM (Privacy Enhanced Mail) format. You can verify the SSL certificate by using the command, `openssl verify <sslcert.crt>`. WebApp Secure is only concerned with the validity of the format. openssl verify might allude to other problems with the certificate, but other issues do not come into play when used within WebApp Secure.

Pages

WebApp Secure supports different configurations for different pages within a protected application. Fill out the required information and click **Add Page** to create a new page.



NOTE: The page nomenclature is used for simplicity. Much like applications, page contexts can define a set of pages using a RegEx. They aren't restricted to one actual page on the application.

Figure 36: Add New Page

ADD NEW PAGE

Page Metadata

Name *
Please enter a human-readable "friendly name", to be used throughout the interface.

Slug / Short Name *
Please enter a unique, URL-friendly, short identifier comprised of letters, numbers, underscores, and / or dashes.

Description *
Please enter a brief, human-readable description.

URL Pattern

To add a page, you must define a URL pattern to match traffic against.

Suggestions:

HTTP? ☒
Whether to apply this pattern to HTTP traffic

HTTPS? ☒
Whether to apply this pattern to HTTPS traffic

URI Regex *
The regular expression to match on the URI

URI Case Insensitive? ☒
Whether the URI regex should be case-insensitive or not

[Related
Documentation](#)

NTP Service

To keep your appliance clock synchronized to the correct time, WebApp Secure allows the configuration of NTP servers. The appliance can use suggested publicly-available NTP server pools, or it can be configured to use an internal NTP server for timekeeping.

To configure the NTP service, do the following:

1. In the Web UI, navigate to **Configuration > Services > NTP servers**.
2. Click the **Add New** button.

3. A list of servers is displayed with a suggested server selected. Click the **Use Suggestion** button to use the recommended server. Otherwise, enter a valid hostname or IP address of an NTP server in the Server field.
4. Click **Test NTP Server Connection**. If the test is successful, click **Save**.

Alert Service

WebApp Secure can e-mail alerts to your administration team. While the appliance can serve as its own mail server, we recommend that you use a valid mail server for your network.

To configure e-mail alerts, do the following:

1. In the Web UI, navigate to **Configuration > Services > Alert Service**.
2. Set Alerts Enabled to **True**.
3. Enter the sender e-mail address. This is the address that e-mails will use in the from field.
4. In the Contacts section, click the **Add New** button to enter contact information for e-mail alerts. You can also select the severity of events that can trigger e-mails as well as the time-frame for sending e-mail alerts to the contact in question.
5. In the SNMP Addresses section, click the **Add New** button to enter the address and port of SNMP managers. The address field defaults to localhost. Set it to the IP address or FQDN of your mail server if you are using an off-board mail server as recommended. The port defaults to 25. Set it to the port your mail server is listening on. **Test your settings** and click **Save**.

Related Documentation

Integration with SRX Series Overview

The SRX series by Juniper is an enterprise-level Secure Gateway for networks. WebApp Secure has the ability to integrate with this solution, which means it can send IPs to the SRX series to achieve a block (or other configurable response) at the gateway level. This effectively allows the SRX series to tap into the identifying metrics produced by WebApp Secure.

Related Documentation

- [Filters and Terms Configuration Summary for SRX Series Integration on page 64](#)
- [Creating SRX Series Filters and Terms on page 65](#)
- [Configure the SRX Series Integration on page 66](#)
- [Testing the SRX Series Integration Configuration on page 68](#)

Filters and Terms Configuration Summary for SRX Series Integration

The SRX series uses a pipeline of filters to be applied to incoming packets. Each filter contains any number of terms that can apply actions to these incoming packets. The first step in configuring WebApp Secure to work with the SRX series is to configure the filters and terms required. WebApp Secure will require a valid IPv4 filter. This can be named anything and can be a filter already set up prior to WebApp Secure integration. Remember this filter name, because you will input it into the WebApp Secure appliance once the SRX series configuration has been completed.

Along with a filter, you must create two terms. Unlike the filter, these terms cannot be modified by any other service. The first term is the term that IP addresses are added to in the event of an External Counter Response activation, and whose name will be supplied to configuration. The second term must be added as a safeguard which will determine what action to take when no IPs are in the first term. It is recommended that the second term be similar to the following:

```
term jwas_default {  
  then {  
    accept;  
  }  
}
```

This should be placed after the **blocking term**. It allows all traffic through once the previous term's action has been changed to **next term**. Consult the SRX series documentation for more information on the SRX series and its filters.



NOTE: Because the SRX series will drop packets when **next term** is the action and no actual next term exists, it is important to have this additional term below the term which will contain the actual IPs.



WARNING: Any IPs added to the WebApp Secure term through the SRX series CLI, the SRX series GUI, or any other external service besides WebApp Secure, are not guaranteed to remain in the term.

Related Documentation

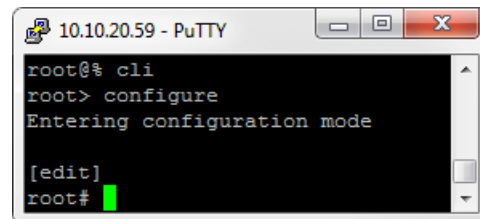
- [Integration with SRX Series Overview on page 63](#)
- [Creating SRX Series Filters and Terms on page 65](#)
- [Configure the SRX Series Integration on page 66](#)
- [Testing the SRX Series Integration Configuration on page 68](#)

Creating SRX Series Filters and Terms

To initialize a filter for use with WebApp Secure do the following:

1. Log into the SRX series through SSH. Then enter **cli** and next enter **configure** to put the cli into configuration mode.

Figure 37: Initialize Filter



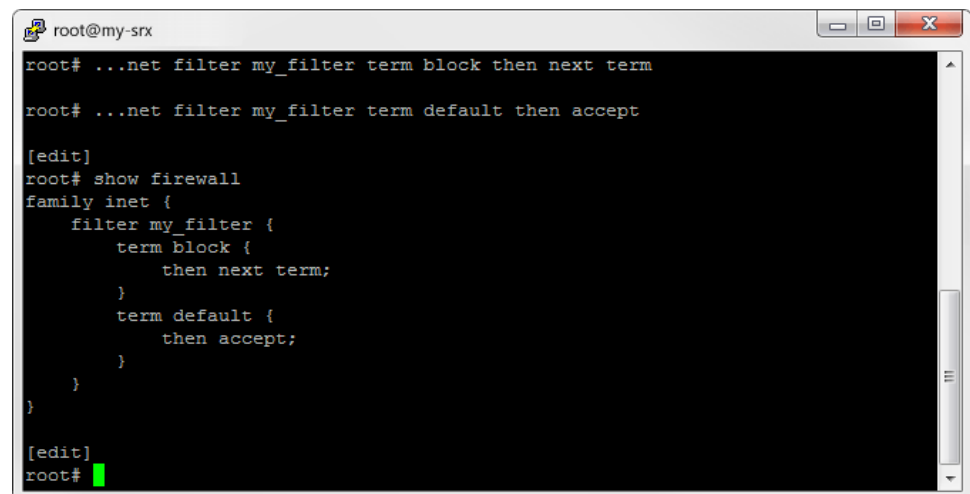
```

10.10.20.59 - PuTTY
root@% cli
root> configure
Entering configuration mode

[edit]
root#
  
```

2. Next create the filter, term, and a placeholder action. Because each term must have some sort of action, choose the **next term** action. This passes the packet on to the next term in the filter. Although the inside of the term will be replaced by WebApp Secure, it will allow the filter to be created. To do this enter **set firewall family inet filter my_filter term block then next term**. You can enter **show firewall** to see your newly-created filter.

Figure 38: Create Filter Term



```

root@my-srx
root# ...net filter my_filter term block then next term

root# ...net filter my_filter term default then accept

[edit]
root# show firewall
family inet {
  filter my_filter {
    term block {
      then next term;
    }
    term default {
      then accept;
    }
  }
}

[edit]
root#
  
```

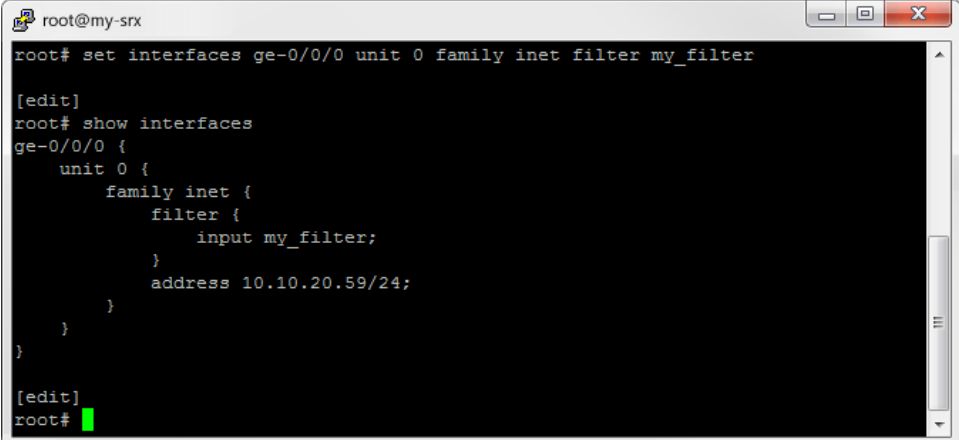


NOTE: The filter name **my_filter** and term **block** are example names. You can choose any names you like, but remember them because you will need to inform \WebApp Secure of your name choices later on in the configuration.

3. Although the filter is created, it is not set to intercept incoming packets. You must now bind the filter to an interface. The interface and unit names will be different depending on your network implementation, but an example is: **set interfaces ge-0/0/0**

unit 0 family inet filter input my_filter. After binding to an interface, you should see the newly created filter appear under the appropriate interface when you enter **show interfaces**.

Figure 39: Bind Filter to Interface



```

root@my-srx
root# set interfaces ge-0/0/0 unit 0 family inet filter my_filter

[edit]
root# show interfaces
ge-0/0/0 {
  unit 0 {
    family inet {
      filter {
        input my_filter;
      }
      address 10.10.20.59/24;
    }
  }
}

[edit]
root#
  
```

4. Save the changes by entering **commit**. Exit the CLI by entering **exit** twice (once to exit configure mode, and once to exit the CLI).



WARNING: If the **blocking** term is misplaced after the default (accept) term, the filter will not commit. Make sure that the accepting term is placed after the blocking term. Remember: next term needs a next term to switch to.

Related Documentation

- [Integration with SRX Series Overview on page 63](#)
- [Filters and Terms Configuration Summary for SRX Series Integration on page 64](#)
- [Creating SRX Series Filters and Terms on page 65](#)
- [Configure the SRX Series Integration on page 66](#)
- [Testing the SRX Series Integration Configuration on page 68](#)

Configure the SRX Series Integration

To configure the integration of an SRX series appliance with WebApp Secure, you must enable the External Counter Response Service, found within the configuration of the WebApp Secure web interface. Navigate to **Configuration > Services > External Response Server (SRX Integration)** and configure the fields described in the table below.



NOTE: After you enter the information, click the **Test SRX Connection Settings** link to make sure the system can be reached.

The External Counter Response Service allows the SRX series to send filter requests to the Appliance, and can be found under the **Global** section of the WebApp Secure configuration. It is an **Advanced** configuration set, so you will need to show the advanced configuration entries to see the External Counter Response Service configuration category.



WARNING: The configuration category will validate if there is an IP address or hostname in the corresponding configuration entry, and a filter name along with a term name, but this does not mean the service is properly working. Always test the counter response after changing the configuration entries, explained in the next section.

Be sure to examine the configuration entries available for this service, and fill out all necessary fields, outlined in the following table.

Table 5: External Counter Response Service Configuration Parameters

Parameter	Type	Default Value	Description
External Counter Responses Enabled	Boolean	False	Whether or not to enable this service.
Network Address	IP (or DNS name)	[Not Set]	Required. Either the IP address or the DNS name of the device.
SRX series Password	String	[Not Set]	The password to log into the SRX series.
SRX series Username	String	[Not Set]	The username to log into the SRX series.
Filter Name	String	[Not Set]	Provide a filter name that WebApp Secure will use.
Term Name	String	[Not Set]	The term in the configured filter that WebApp Secure should add the IPs to. It should not be currently in-use by any other service, and should only be used for WebApp Secure.
Action(s) to Apply	Collection (Strings)	[collection:1]	Choose the actions for the SRX series to take on IPs sent to it by WebApp Secure. When no IPs are blocked on the SRX series through WebApp Secure, these terms will be changed to Evaluate Next Term , which will continue to the next term in the filter. By default, this is set to a collection of 1, consisting of only discard .



WARNING: When configuring multiple actions to take, be careful not to populate the collection with conflicting actions. An example of two conflicting actions are **reject** and **accept** (You cannot reject a connection and then accept a connection!). WebApp Secure has no protection for conflicting actions. The system will overwrite older actions with newer ones (further down the collection). An example of non-conflicting actions are **log** and **discard**. In this case, the packets will be logged, and then discarded. For more information on actions to take, consult the SRX series documentation.



NOTE: If the External Counter Response Service is disabled or otherwise configured incorrectly, blocking a profile through the External Block response will not work, but will still be shown in the User Interface as a valid Counter Response.

**Related
Documentation**

- [Integration with SRX Series Overview on page 63](#)
- [Filters and Terms Configuration Summary for SRX Series Integration on page 64](#)
- [Creating SRX Series Filters and Terms on page 65](#)
- [Testing the SRX Series Integration Configuration on page 68](#)

Testing the SRX Series Integration Configuration

Purpose To verify the configuration of the WebApp Secure portion of the SRX series integration, do the following

- Action**
1. Create a profile by accessing the .htaccess file (explained in “Verify the Installation”).
 2. Navigate to the WebApp Secure web interface and find the newly created profile.
 3. Manually activate the **Filter on SRX** Counter Response.
 4. Log into the SRX series CLI, and run the command **show configuration firewall** (or **show firewall** if in).

You should see a new filter created with the name you gave in configuration, and a new term within that filter called that you also named within configuration. It should appear similar to the following (depending on how you set up your filter and actions):

```
family inet {
  filter my_filter {
    term block {
      from {
        address {
          10.10.10.10/32;
        }
      }
    }
    then {
      reject;
    }
  }
  term default {
    then {
      accept;
    }
  }
}
```

In the example above, 10.10.10.10 is the IP of the profile you activated the Counter Response on. This is telling the SRX series to reject the IP of the profile at the gateway

level. Note the default term below the block term which will act as an accept-all in the case that the block term's action has been changed to **next term**.

You can also verify the line with the IP address gets deleted when deactivating the Counter Response.



NOTE: When there are no IPs to block, the SRX series defaults to * or All Traffic. This would effectively block all traffic from that interface! To counter this, WebApp Secure changes the action from your configured entry to next term, essentially letting the next term within the filter deal with the traffic. Because you set up a default term to handle this case (see Configuration), the next term simply accepts all traffic.

This filter should now look as follows:

```
family inet {
  filter my_filter {
    term block {
      then next term;
    }
    term default {
      then {
        accept;
      }
    }
  }
}
```

This is indicating that all traffic will be sent through this term, but the action is simply passing the packet onto the next term in the filter, which is our default term that will accept all traffic.

Related Documentation

- [Integration with SRX Series Overview on page 63](#)
- [Filters and Terms Configuration Summary for SRX Series Integration on page 64](#)
- [Creating SRX Series Filters and Terms on page 65](#)
- [Configure the SRX Series Integration on page 66](#)

Role-Based Administrator Access Control

Role-Based Access Control (RBAC) is a way to assign different levels of administrator functionality to different users. You can assign roles to various users that exist on a configured LDAP or RADIUS server. The first step in integrating with your existing LDAP or RADIUS service is to give WebApp Secure the connection information. In the Web UI, navigate to **Configuration > Users and Groups** and click on **Manage Authentication Settings**. On the resulting page, input all information relating to your LDAP or RADIUS server and click **Save**. You should now see the corresponding service as "Enabled" under the

Authentication section of Users and Groups. Once the server has been connected to WebApp Secure, the next step is to configure roles for various users. By default, the user "mykonos" is enabled and given the role "Super Administrator". To add additional users, click the **Add User** link. You will be prompted to enter a Username and will be given a choice of which groups you want that user to inherit. A complete description of all roles is available by clicking on **View Role Descriptions** beneath the Roles dropdown. A more simplistic table of roles and their corresponding permissions are given in Appendix D, RBAC Groups and Roles.

- Related Documentation**
- [Configuring Role-Based Access Control on page 70](#)
 - [RBAC Groups and Roles on page 289](#)

Configuring Role-Based Access Control

1. In the Web UI, go to **Configuration > Users and Groups**.
2. Click **Manage Authentication Settings**.
3. Enter all information relating to your RADIUS or LDAP server as follows:

RADIUS
 - RADIUS Enabled—True or False
 - RADIUS Server—Enter the FQDN, hostname, or IP address of the RADIUS server.
 - RADIUS Secret—Enter the RADIUS shared secret.
 - RADIUS Timeout—Enter the timeout, in seconds, for RADIUS operations.
LDAP
 - LDAP Enabled—True or False
 - LDAP Server—Enter the FQDN, hostname, or IP address of the LDAP server.
 - LDAP Base DN—Enter the base Distinguished Name (DN) of the highest level tree on which you wish to support LDAP.
 - LDAP TLS Enabled—True or False—Whether or not to use Transport Layer Security (TLS) when making LDAP connections.
 - LDAP TLS CA Certificate—Enter the CA certificate used to authenticate the certificate provided by the LDAP server.
 - Use LDAP for Authentication—True or False—Whether or not to use LDAP for Authentication or just for user information.
 - LDAP Bind DN—Enter the bind distinguished name (DN) for connecting to the LDAP server.
 - LDAP Bind Password—Enter the password to be used when binding to the LDAP server.
4. Click **Save**. You should now see the corresponding service as Enabled under the Authentication section of Users and Groups.

- 5. The next step is to configure roles for various users. By default, the user *mykonos* is enabled and given the role *Super Administrator*. To add additional users, click the **Add User** link.
- 6. You are prompted to enter a **Username** and you are given a choice of which groups you want the user to inherit. A complete description of all roles is available by clicking **View Role Descriptions** beneath the **Roles** drop down list. A more simplistic table of roles and their corresponding permissions can be found in Appendix D, RBAC Groups and Roles.

Figure 40: Users and Groups, Add User

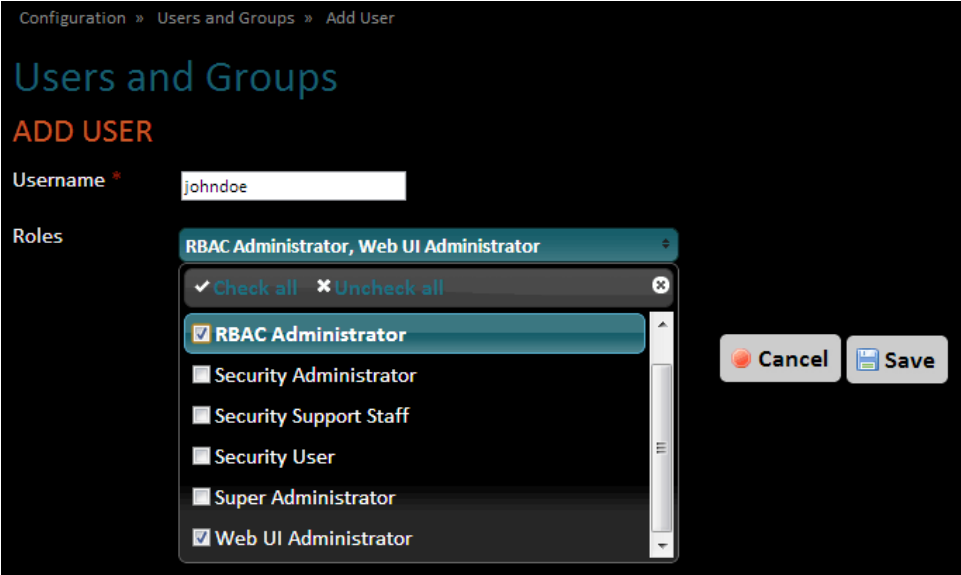


Figure 41: Assigned Roles

Username	Assigned Role(s)	Actions
johndoe	RBAC Administrator (System Group: rbacadmin), Web UI Administrator (System Group: webuiadmin)	
mykonos	Super Administrator (System Group: superadmin)	

Add User



NOTE: Because WebApp Secure doesn't actually create users on the appliance itself but merely maps the username to the given permissions, the only way to effectively remove the user is to strip them from all roles. After removing roles and saving, the entry in the Authorization table is removed.



NOTE: WebApp Secure doesn't allow the last RBAC Administrator role to be deleted. It is possible to remove your own permissions, though, essentially locking you out of the system. Similarly, re-initializing the configuration settings will wipe out all user-role mappings, and the *mykonos* user will be the only one able to assign roles.



NOTE: Any violations of access control (a user trying to access some part of the system they aren't configured to access) will be logged to the audit log.

**Related
Documentation**

- [Role-Based Administrator Access Control on page 69](#)
- [RBAC Groups and Roles on page 289](#)

CHAPTER 5

Managing the Appliance

- Overview on page 74
- Navigating the CLI on page 74
- The CLI: The Set Command on page 75
- The CLI: General and Base Commands on page 77
- The CLI: Configuration Level Commands on page 81
- The CLI: System Level Commands on page 84
- CLI: Config Example on page 86
- CLI: Config: Setting a Configuration Parameter on page 87
- CLI: Config: Initializing the Configuration on page 88
- CLI: Config: Import/Export on page 88
- CLI: Config: Configure a Proxy Exclusion on page 89
- System Updates on page 90
- Statistics on page 92
- High Availability Network Failure Detection, Actions, and Monitoring on page 95
- Unblock Web UI Login Ban on page 97
- Health Check URL on page 97
- Self-Monitoring on page 98
- Self-Monitoring Configuration Variables on page 98
- Managing and Viewing Logs on page 103
- Log File Destination on page 104
- Backup and Recovery Overview on page 105
- Database Backup and Restore on page 107
- About Security Intelligence on page 107
- Enable the Spotlight Connector Service on page 109
- Spotlight Connector Session Cookies and Locations on page 110
- About Spotlight Secure on page 112
- Enable Spotlight Secure on page 113

Overview

The Command Line Interface is a standard tool to perform various management tasks in the WebApp Secure system. Anything from configuration settings, service statuses and network settings can be controlled through the CLI. To enter the CLI, SSH into WebApp Secure and type `cli` at the bash prompt. You should see the prompt `>` along with a welcome message indicating you have successfully entered the CLI.

```
[mykonos@webappsecure ~]$ cli
```

Welcome to the Junos WebApp Secure CLI

```
> _
```



NOTE: Note: The commands `sudo mykonos-shell` and `cli` are analogous; the latter is a bash alias to the former.

Related Documentation

- [Navigating the CLI on page 74](#)
- [The CLI: General and Base Commands on page 77](#)
- [The CLI: Configuration Level Commands on page 81](#)
- [The CLI: System Level Commands on page 84](#)
- [The CLI: The Set Command on page 75](#)

Navigating the CLI

The CLI uses different contexts to segregate commands. Currently, there are three contexts that are available from the root context:

- **config**

Allows you to display or manipulate the WebApp Secure configuration. The CLI provides an alternate way to edit configurations without the need to navigate through the Web UI.

- **system**

Any command that deals with the system as a whole is contained in the **system** context. Tasks such as configuration of network interfaces, checking on the appliance services or executing backups can be done through this context. The **system** context also contains the **services** subcontext.

- **support**

Enter the support level of the CLI. This level provides support features meant for troubleshooting system problems.

To exit to the last context used, simply type `exit` or use the keyboard shortcut `Ctrl + D`.



NOTE: At any point, you can enter ? or help to view the available commands at the current context or help <command> to get contextual help on the specified command.

```
[mykonos@webappsecure ~]$ cli
```

```
Welcome to the Junos WebApp Secure CLI
```

```
> config
```

```
config> help show
```

```
Summary: display configuration parameters
```

```
Syntax:
```

```
show [parameter]
```

```
show|compare
```

Display the value of the given parameter. If no parameter name is given the entire configuration tree will be displayed.

You can also compare what is currently in memory to what was loaded from disk using the "compare" flag. To do this use the syntax:

```
show|compare
```

```
config> exit
```

```
> exit
```

```
[mykonos@webappsecure ~]$ _
```

Related Documentation

- [The CLI: General and Base Commands on page 77](#)
- [The CLI: Configuration Level Commands on page 81](#)
- [The CLI: System Level Commands on page 84](#)
- [The CLI: The Set Command on page 75](#)

The CLI: The Set Command

The set command allows you to set the various local system network settings.



NOTE: The following commands imply you are already in the system context in the cli.

- Configure Network Interfaces—The syntax for setting network interface attributes is as follows:

set interface <interface> <attr1> <value1> <attr2> <value2> ... <attrN> <valueN>

The settings will not take affect immediately and will require a restart of the interface to actually take affect. You may set multiple attributes on a single interface at one time. The available attributes can be see by using the tab completion when setting an interface.

- Configure an Alias Interface—Specify the interface using the following format:

set interface <physical interface>:<alias number>

For example, if you are configuring the first alias on physical device eth0, then the interface you would give would be eth0:0.

- Configure a Vlan Tagged Interface—Specify the interface using the following format:

set interface <physical interface>:<vlan id>

You must also need to set the vlan attribute to true.

- Configure a Bonded Interface—Specify the interface using the following format:

set interface bond<bond number>

Configure the bond interface itself as you would any other interface , where <bond_number> is the number to assign to the bond name. For each physical interface that will be a part of the bond you will need to set the slave attribute to true and set the master attribute to the bond interface. For example:

cli system set interface bond0

cli system set interface eth4 slave true master bond0

cli system set interface eth5 slave true master bond0

- Configure Hostname Resolution—Use the following syntax:

set dns nameservers <servers>

set dns domain <domain>

set dns search <search>

These settings let you configure how hostnames are resolved by the local system. If you provide multiple nameservers to use for DNS resolution, they should be separated by a command and given in the order of priority.

- Configure System HTTP Proxy—Use the following syntax:

set proxy <protocol>://[<username>:<password>@]<host>:<port>

To setup a HTTP or HTTPS proxy to use for the local system you would provide the proxy information in the above format. The <protocol> will determine which one the proxy is used for, and you can optionally specify a username and password to be used for connecting to the proxy.

- Configure the Management Interface—Specify the interface using the following format:

set management-interface <interface>

Sets the management interface for the local system. All management traffic and internode communications will be restricted to the given interface. The only exception to this is HA replication traffic. The replication traffic will only happen on the interconnect.

- Configure the HA Management VIP—Use the following syntax:

set management-vip <ip>/<netmask>

Set the floating IP address to be used by the management interfaces of the HA pair. The floating IP address must be a valid IPV4 address and should include the netmask to be used.

- Configure the HA Traffic VIP(s)—Use the following syntax:

set traffic-vip <name> <ip>/<netmask>

Add a floating IP address to be used by the HA pair for handling of application traffic.

- Configure the Interface Monitor—Use the following syntax:

set ethmonitor <interface>

Set a network interface to be monitored for availability. If the given interface is detected to be down, then a failover event will occur with the HA pair.

- Specify a custom IP route and/or rule - Use the following format:

system set interface <interface> route <route definition>**system set interface <interface> rule <rule definition>**

To allow complete IP routing customizability, WebApp Secure allows administrators to define custom routing rules and routes. Using the above format, you can apply specific routing behavior to any interface so you can better handle any routing issue that may arise. Once added, they should be listed with the interface when running **system show interfaces**. The routes/rules do not take affect immediately; they require a restart of the network service the same as any other network interface change. After restarting the network service you can verify that the routes/rules have taken affect with the commands **ip route list** and **ip rule list**. For information on how to define route and rule definitions, exit the CLI and type **ip route help** and **ip rule help** from the WebAppSecure bash prompt

**Related
Documentation**

- [The CLI: General and Base Commands on page 77](#)
- [The CLI: Configuration Level Commands on page 81](#)
- [The CLI: System Level Commands on page 84](#)

The CLI: General and Base Commands

When you enter **CLI** at the bash prompt, this puts you at the base level of the CLI. You can also jump directly to a specific level or even run a command directly without entering the interactive mode of the CLI. You do this by providing the full path to what you are

trying to access. For example, to show the current network configuration for interface eth0 you could run the command **cli system show eth0** directly from the bash prompt.

Command Flags

- Some commands available in the CLI can take various flags to alter the default behavior of the command. The normal format for these flags is in the form: **command | flags**
- If you want to pass multiple flags to a command, do so by separating the flags with a space as follows: **command | flag1 flag2 flag3**



NOTE: Any unknown or invalid flags passed are ignored.

Special Characters—When passing a value to the CLI that contains special characters you should wrap the value in quotes. You should also do this for any value or argument that contains spaces. For example if you are setting the `engine.server_name` parameter in config and you want the value to be Microsoft IIS, run the following command: **cli config set engine.server_name "Microsoft IIS"** If you do not quote the value the example, then the parameter would be incorrectly set to the value Microsoft.

The following sections provide lists of commands you can reference when using the CLI.

Table 6: General CLI Commands

Command	Syntax	Description
color	color	<p>Changes the display of color in the CLI. By default, only the default terminal foreground color is used for all output. If enabled then output of certain commands is syntax highlighted to make it easier to distinguish certain types of information.</p> <p>Example:</p> <pre>> color</pre> <p>Color Enabled</p> <pre>> color</pre> <p>Color Disabled</p>
echo	echo [string]	<p>Echos the string passed to it back to the screen. The main purpose of this command is to give you the ability to provide a status message when writing automated scripts against the CLI.</p> <p>Example:</p> <pre>> echo Hello, World!</pre> <p>Hello, World!</p>

Table 6: General CLI Commands (*continued*)

Command	Syntax	Description
exit	exit	<p>Exits the current level of the CLI and returns you to the entry point of that level.</p> <p>Example:</p> <pre>> system system> exit > _</pre>
help	help [topic]	<p>Displays help information about the various levels and command available within the CLI.</p> <p>Example:</p> <pre>> help echo</pre> <p>Summary: echo the given string back to the terminal</p> <p>Syntax: echo [string]</p> <p>This is a simple implementation of a echo command.</p>
pager	pager	<p>Toggles use of the screen pager for output longer than your current screen size. When enabled, the output of commands are paginated according to the size of your current terminal window. When disabled, the pagination will not take affect which can make it difficult to view long output from some commands.</p> <p>Example:</p> <pre>> pager</pre> <p>Pager Disabled</p> <pre>> pager</pre> <p>Pager Enabled</p>
tee	tee	<p>Toggles teeing of output to a log file. When enabled, all output displayed inside the CLI is also logged to a file in the user's home directory.</p> <p>Example:</p> <pre>> tee</pre> <p>Output logging to /home/mykonos/2014-04-03-19:06:59.697271.log</p> <pre>> tee</pre> <p>Output logging stopped</p>

Table 7: Base Level CLI Commands

Command	Syntax	Description
config	config [subcommand]	<p>Enter the configuration level of the CLI. From here you can manage the various configuration settings used by WebApp Secure. System specific settings such as network configuration, or settings that only affect a limited number of nodes in a cluster/install are not handled through the configuration level. For system specific configuration settings please see the system level of the CLI.</p> <p>Example:</p> <pre>> config show engine.enabled</pre> <p>true</p> <p>Refer to “The CLI: Configuration Level Commands” on page 81 for details.</p>
system	system [subcommand]	<p>Enter the system level of the CLI. The system level is where all local system configuration is done and where local system actions can be performed. This level is also where HA actions/configuration is located.</p> <p>Example:</p> <pre>> system services status</pre> <p>mykonos-security-engine mykonos-security-engine (pid 9150) is running...</p> <p>Refer to “The CLI: System Level Commands” on page 84 for details.</p>
support	support [subcommand]	<p>Enter the support level of the CLI. This level provides support features meant for troubleshooting system problems.</p> <p>Example:</p> <pre>> support bundle</pre> <p>The support bundle has been successfully sent to the support staff.</p>

Related Documentation

- [The CLI: Configuration Level Commands on page 81](#)
- [The CLI: System Level Commands on page 84](#)
- [The CLI: The Set Command on page 75](#)

The CLI: Configuration Level Commands

The following section provides a list of configuration level commands you can reference when using the CLI.

Table 8: Configuration Level CLI Commands

Command	Syntax	Description
commit	commit	<p>Commits any outstanding changes to the configuration settings. If running within a cluster, this command will also publish the new configuration settings out to all nodes in the cluster.</p> <p>Example:</p> <pre>config> set engine.enabled false</pre> <pre>config> commit</pre> <p>1 parameter changed</p>
export	export	<p>Exports the current saved configuration to a time-stamped file in the user's home directory. The exported file is in the correct format for later re-importing the file back into the configuration system.</p> <p>Example:</p> <pre>config> export</pre> <p>Configuration exported to /home/mykonos/jwas-2014-04-03-193700.839423.cfg</p>

Table 8: Configuration Level CLI Commands (*continued*)

Command	Syntax	Description
info	info <parameter>	<p>Display information about the given parameter, such as the level within the configuration it can be set at, whether it inherits from other levels of config, and possible suggested values for the parameter.</p> <p>Example:</p> <pre>config> info logging.audit.local</pre> <pre>-----</pre> <p>Log Audit Locally</p> <pre>-----</pre> <p>Whether to log audit log entries locally</p> <p>Parameter: logging.audit.local</p> <p>Contexts: global</p> <p>Inheritable: True</p> <p>Suggestions:</p> <p>(1) True - Turn on local logging</p> <p>Value: true</p> <p>(2) False - Turn off local logging</p> <p>Value: false</p>
init	init <parameter>	<p>Initializes configuration setting values. If a parameter name is given, then only the value of that parameter is initialized.</p> <p>Available Flag:</p> <ul style="list-style-type: none"> • ifempty - Only initialize parameter if the current value is not set or null. <p>Example:</p> <pre>config> set engine.enabled false</pre> <pre>config> show engine.enabled false</pre> <pre>config> init engine.enabled</pre> <pre>config> show engine.enabled true</pre>

Table 8: Configuration Level CLI Commands (*continued*)

Command	Syntax	Description
set	set <parameter> <value>	<p>Sets the value of the given configuration parameter to the value provided.</p> <p>Available Flags:</p> <ul style="list-style-type: none"> • edit - Opens a text editor to change the value. This is useful when setting parameters that can take large values such as response bodies. • suggestion # - Sets the value of the parameter to the given suggestion number. Use the info command to lookup what the available suggestions are and the number associated with them. • null - Sets the value of the parameter to null if allowed. • ifempty - Only sets the value of the parameter if its current value is not set or null. • json - Parse the given value as json. Useful for setting multiple attributes at one time on a parameter. <p>Example:</p> <pre>config> set processors.block.response suggestion 3 config> show processors.block.response processors.block.response config> show processors.block.response.status 403</pre>
show	show <parameter>	<p>Show the value of the given parameter and any parameters below it. If no parameter is given, then the entire configuration structure is returned. The default format returned is json.</p> <p>Available Flag:</p> <ul style="list-style-type: none"> • flat - Returned data is in dot notation format rather than json. <p>Example:</p> <pre>config> show engine.enabled true</pre>

Table 8: Configuration Level CLI Commands (*continued*)

Command	Syntax	Description
unset	unset <parameter>	<p>Remove a parameter from configuration.</p> <p>Available Flag:</p> <ul style="list-style-type: none"> match(<attribute>=<value>) - Remove all parameters from a collection that have an attribute of the given name that matches the given value. <p>Example:</p> <pre>config> set foo.bar baz config> show foo.bar baz config> unset foo.bar Are you sure you want to unset "foo.bar"? [y N]: y config> show foo.bar config> _</pre>

- Related Documentation**
- [The CLI: General and Base Commands on page 77](#)
 - [The CLI: System Level Commands on page 84](#)
 - [The CLI: The Set Command on page 75](#)

The CLI: System Level Commands

The following section provides a list of system level commands you can reference when using the CLI.

Table 9: System Level CLI Commands

Command	Syntax	Description
set	set <setting> <value> set interface <interface> <attribute> <value> set dns <setting> <value>	<p>Sets local system level properties, including the hostname, network interfaces, DNS servers, proxy servers and more. For more information about the various settings available with this command use help set for usage information.</p> <p>Example:</p> <p>system> set interface eth2 onboot no</p> <p>See “The CLI: The Set Command” on page 75 for details.</p>
show	show [category]	<p>Shows system configuration. When given without a [category], this command will show the entire system configuration.</p> <p>Example:</p> <p>system> show management-interface "eth0"</p>
status	status <item>	<p>Depending on the item supplied, this will look over the local system for the status of the component requested. For a list of all available items to retrieve status information for, type help status.</p> <p>Example:</p> <p>system> status network</p> <p>system> status ha</p>
unset	unset <setting> <value> unset interface <interface> <attribute> <value> unset dns <setting> <value>	<p>Unsets the given local system configuration. If allowed to be empty or unset this command will remove the specified setting completely. Not all system level properties are allowed to be unset, such as hostname and the management interface.</p>
backup	backup	<p>Initiate a system backup. Backups are stored in /home/mykonos/backups/ directory.</p> <p>Example:</p> <p>system> backup</p> <p>Confirm Backup [y N]: y</p>
initialize	initialize	<p>Runs the system initialization process.</p> <p>Example:</p> <p>[mykonos@webappsecure ~]\$ cli system initialize</p>

Table 9: System Level CLI Commands (*continued*)

Command	Syntax	Description
services	services [subcommand]	<p>Enter the services level of the CLI. This level allows you to control the services running on the local system. The subcommand can consist of two parts, the action to take, and the component to take that action.</p> <p>Actions</p> <ul style="list-style-type: none"> • status • start • restart • shutdown <p>Example:</p> <p>system> services status nginx</p> <p>nginx (pid 1614) is running...</p> <p>NOTE: To see all available components, type the action, a space, and then hit the TAB key to see a list of components available for that action. To restart the network for interface configurations to update, type cli system services restart network.</p>

Related Documentation

- [The CLI: General and Base Commands on page 77](#)
- [The CLI: Configuration Level Commands on page 81](#)
- [The CLI: The Set Command on page 75](#)

CLI: Config Example

Typing **config** at the CLI prompt will put the CLI into the configuration context. Configuration values are organized in a hierarchical fashion, with the most general words located at the beginning of the full configuration attribute string. For example:

```
services.cleanup.db.enabled
```

From the entry above, you can see that this parameter is for a service that handles the cleanup of the database. Specifically, this parameter determines whether the service is enabled or not.

Within the config context, you can choose to **show** any portion of the configuration. For example:

```
show services.cleanup.db.enabled
```

In the entry above, the value of the parameter in question is shown. If you want to see all of the configuration for the DB Cleanup Service, you can enter **show services.cleanup.db**

to return a JSON object representation of the entire configuration that relates to the DB Cleanup Service. Likewise, entering only **show** displays the entire configuration as one large object.

- Related Documentation**
- [Overview on page 74](#)
 - [The CLI: Configuration Level Commands on page 81](#)

CLI: Config: Setting a Configuration Parameter

To set a configuration parameter, enter **set** **PARAMETER.TO.SET** **VALUETOSETTO**. For example, to enable the DataBase Cleanup Service (which allows you to delete profiles from WepApp Secure), enter the following:

```
set services.cleanup.db.enabled true
```

For more advanced users, you can edit configuration entries with an actual editor. If you do this, you can append **| edit** to the end of the set command where your value would be. The shell will put you into a text editor (VIM by default) where you can make changes to the configuration values. This is convenient when editing the JSON representation of a set of configuration entries, such as `services.cleanup.db`. Make any changes and, in VIMs Normal Mode, enter **wq** to write and quit the editor. For more information about VIM and how to use it, consult the VIM Documentation.



NOTE: You can choose to show a portion of the configuration without setting it by using the keyword **show** rather than **set**. For example: **show services.cleanup.db** This displays all configurations related to the DataBase Cleanup Service.

After making any changes, you can compare the new configuration with the last-saved version by typing entering **| compare**. A diff is printed to screen, with **-** indicating original settings, and **+** indicating modified settings. For example, changing the DB Cleanup Service from true to false will yield:

```
- - - Original Settings
+++ Modified Settings
@@ -2635,7 +2635,7 @@
services.backups.retention: 7
services.backups.secret: WIB25IkIsbMM3wOR
services.backups.ssh.enabled: false
-services.cleanup.db.enabled: true
+services.cleanup.db.enabled: false
services.cleanup.db.expiration.history: 2592000
services.cleanup.db.expiration.malicious: 10368000
services.cleanup.db.expiration.session: 2592000
```

- Related Documentation**
- [Overview on page 74](#)
 - [The CLI: Configuration Level Commands on page 81](#)

CLI: Config: Initializing the Configuration

At some point, it might be necessary to reset all configuration entries to default values. To do this, you enter **config init** at the root context, or **init** and **commit** at the config context. Once you do that, all entries are reset to their factory defaults.



WARNING: Because configuration initialization resets every parameter to a default, you might want to record some entries before doing this. Specifically, `engine.session.encryption_key` and `engine.session.initialization_vector`. Those two entries are needed to maintain the correct session data for currently-active users. If these values change, you might see false positives of Session Etag Spoofing, Session Tampering, and Application Cookie Manipulation incidents, because the corresponding key values have indeed been manipulated.



WARNING: You should also record the backups encryption key `services.backups.secret`. If you reinitialize WebApp Secure, and do not know your backup secret key, nobody — not even Juniper Support — will be able to recover your backups.

Initializing the configuration will not reset the autoreponse rule activation or deactivation state. The state of autoreponse rules is stored in the database. The easiest way to reset them is to do it manually.

Related Documentation

- [Overview on page 74](#)
- [The CLI: Configuration Level Commands on page 81](#)

CLI: Config: Import/Export

You can export a configuration image containing all configured parameters. This image can be imported by the system, letting you make a backup of your system configuration before making major changes, or to aid in some types of deployment. If you execute a system initialization from the cli, you can select to keep your configuration upon re-initialization. However, historical traffic information is not part of the exported configuration and is therefore not recoverable.

To access the Configuration Import / Export feature, enter **cli** in an SSH session on the appliance, and at the prompt enter **config export <filename>**. The configuration is saved using the filename given. Similarly, import the configuration by entering **config import <filename>**.



NOTE: Because configurations can change from version to version of the product, importing a configuration exported from an older version of WebApp Secure can fail.

Related Documentation

- [Import/Export \(Web UI\) on page 48](#)
- [Overview on page 74](#)
- [The CLI: Configuration Level Commands on page 81](#)

CLI: Config: Configure a Proxy Exclusion

WebApp Secure functions as a reverse proxy. Therefore, all web traffic goes through WebApp Secure so that it can analyze traffic for attacks. To improve performance, you can configure WebApp Secure to not process certain types of resources, such as images, or zip files, for example. To configure an exclusion for certain file extensions that you want routed around WebApp Secure (as opposed to through it), you can use the configuration setting described here.

For example, to have WebApp Secure not process any zip files, you can add the zip file extension to the proxy exclusions list by entering the following command at the WebApp Secure terminal:

```
cli set engine.proxy.exclusions zip
```

Now WebApp Secure will not process zip files.



NOTE: Google Web Toolkit utilizes specialized cache files that can conflict with WebApp Secure. If your protected site utilizes Google Web Toolkit, you will need to add the file extension of these cache files (typically `.cache.html`) to the `engine.proxy.exclusions` parameter.

Exclusions are not limited to file extensions. You can pass in any regex or standard string to match the URI. For example, to exclude all xml files from Security Engine processing, do the following:

1. Navigate to **Configuration > Expert Mode**.
2. Click the **Add Parameter** button.
3. For Key enter `engine.proxy.exclusions`.
4. For Value enter `xml`.
5. Click **Add Parameter**. This value is now added to the exclusion list.



NOTE: Using the cli, type the following to add the same exclusion: `cli config set engine.proxy.exclusions xml`

As another example, to exclude all html files that end in 6 digits, do the following:

1. Navigate to **Configuration > Expert Mode**.
2. Click the **Add Parameter** button.
3. For Key enter **engine.proxy.exclusions**.
4. For Value enter **[0-9]{6}\.html**.
5. Click **Add Parameter**. This value is now added to the exclusion list.



NOTE: Using the cli, type the following to add the same exclusion: `cli config set engine.proxy.exclusions "[0-9]{6}\.html"`

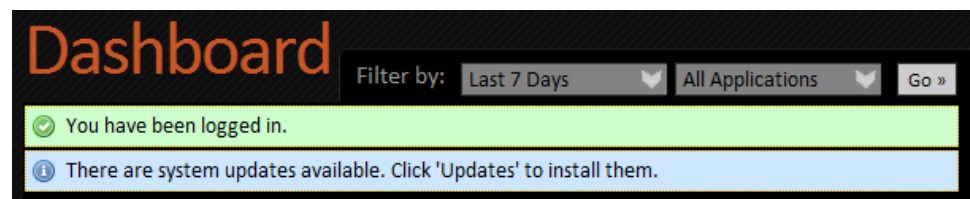
Related
Documentation

- [Overview on page 74](#)

System Updates

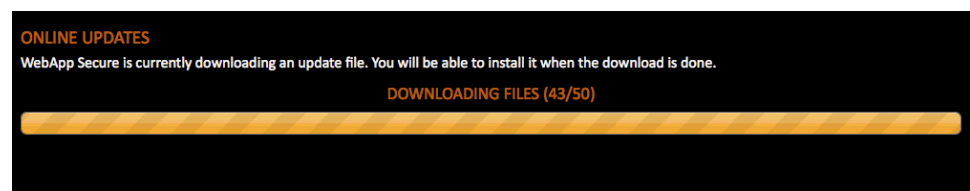
Provided WebApp Secure has Internet access, either direct or through a configured proxy, it will automatically check for software updates every night and download them when new ones are available. However, the appliance will not automatically apply updates. For security and stability, an administrator must manually apply updates. The Web UI informs the you that there is an update by a banner indicator at the top of the page.

Figure 42: Dashboard, Updates



While WebApp Secure checks for updates every night, you can force the appliance to check for updates at any time by clicking the **check for updates** link under **Online Updates**. WebApp Secure will find any available online updates at this time. You can see progress of the download through a status bar.

Figure 43: Downloading Update



WebApp Secure can also upload updates manually, without an Internet connection. After uploading the package to the appliance (through the Web UI's **Updates** page), it

will become available to the updates system, and you will be able to apply the update as described here.



WARNING: While WebApp Secure is uploading offline updates, you should stay on the Updates pane until the upload is complete.

If an update is available (either an uploaded offline update or an automatically downloaded one), you can view the available update package along with any information about it, including the package name, version, whether or not a reboot is required after installing the update, a description, and list of changes. After reviewing the changes you can choose to apply the update by clicking the **Update Selected** button at the bottom of the package table.

Figure 44: Update Description

ONLINE UPDATES

check for updates

You have 1 update available.

Update?	Name	Version	Reboot (?)
<input checked="" type="checkbox"/>	Junos WebApp Secure Core	4.2.2-96	No

DESCRIPTION

Upscaling the resurgent networking exchange solutions, achieving a breakaway systemic electronic data interchange system synchronization, thereby exploiting technical environments for mission critical broad based capacity constrained systems. Fundamentally transforming well designed actionable information whose semantic content is virtually null. Empowerment in information design literacy demands the immediate and complete disregard of the entire contents of this cyberspace communication.

CHANGELOG

9/26/2012
User Guide updated with section explaining offline vs. online updates.

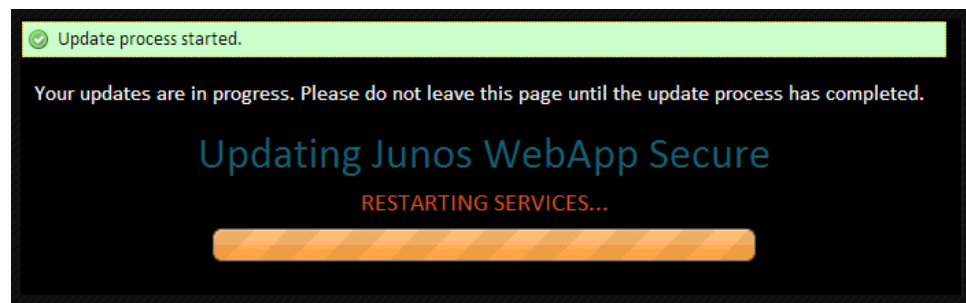
9/23/2012
Administrator can now configure the maximum login attempts on a page before forcing a captcha response.

9/19/2012
Reports system overhauled. Reports can now be exported from many pages in the security monitor.

Update Selected

The system will update and you of its progress through a status bar.

Figure 45: Updating the Application



NOTE: At this time, it is not possible to roll back to earlier versions of the appliance software.

Applying Updates Using the CLI

You can also update WebApp Secure using the CLI. Use the following commands to download, unpack, and install the update.

mykonos-update -d

mykonos-update -l /srv/updates/updates/<updatefile.meta>

mykonos-update -u



NOTE: When you use **mykonos-update -u**, it only updates the system on which the command is run. It will not update members in a cluster or HA pair. To update those systems, you must run the update command on all members.

Statistics

WebApp Secure software allows for standard SNMP system monitoring. All statistics available on a typical Linux system would be available to WebApp Secure through standard system SNMP mibs. In addition, WebApp Secure currently offers six types of systems statistics in a form of graphs. They include CPU Utilization, CPU Load Average, Memory Utilization, Network Traffic, Proxy Connection and Proxy Requests. They can be accessed through **System Status** button in the main menu of the Configuration management interface. Depending on the desired level of details, the statistics can be viewed for the Last Hour, Last 12 Hours, Last Day, Last Week and, finally, Last Month (always last 30 days).

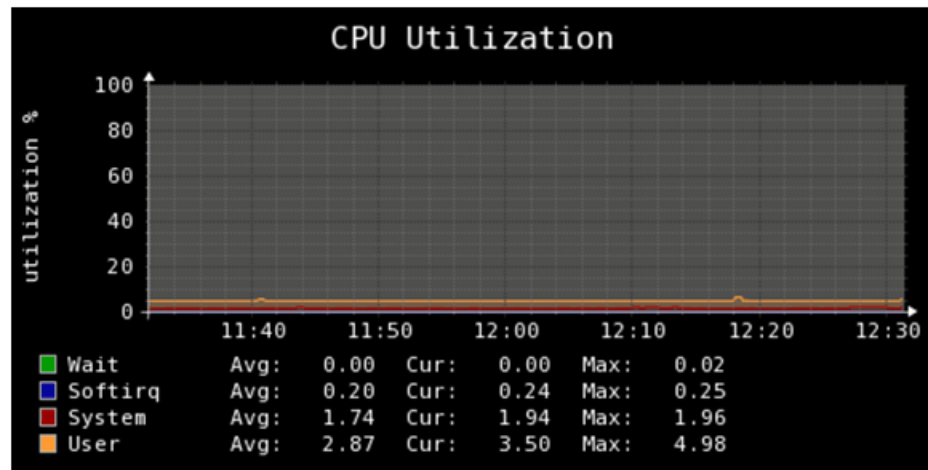
Below are the details of the statistics that are available for each type:

CPU Utilization

- **Wait** - Percentage of CPU time spent in wait (on disk)
- **Softirq** - Percentage of CPU time spent handling software interrupts

- **System** - Percentage of CPU time spent in kernel space
- **User** - Percentage of CPU time spent in user space

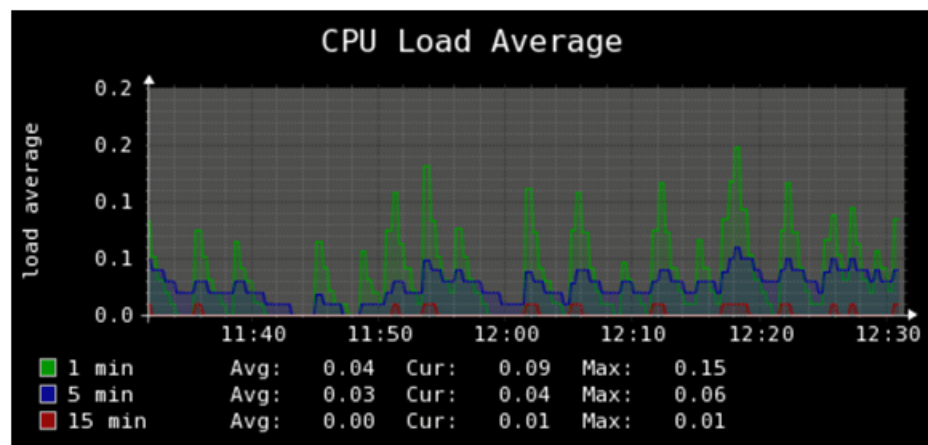
Figure 46: CPU Utilization



CPU Load Average

- 1 min - CPU Load for the last minute
- 10 min - CPU Load for the last 10 minutes
- 15 min - CPU Load for the last 15 minutes

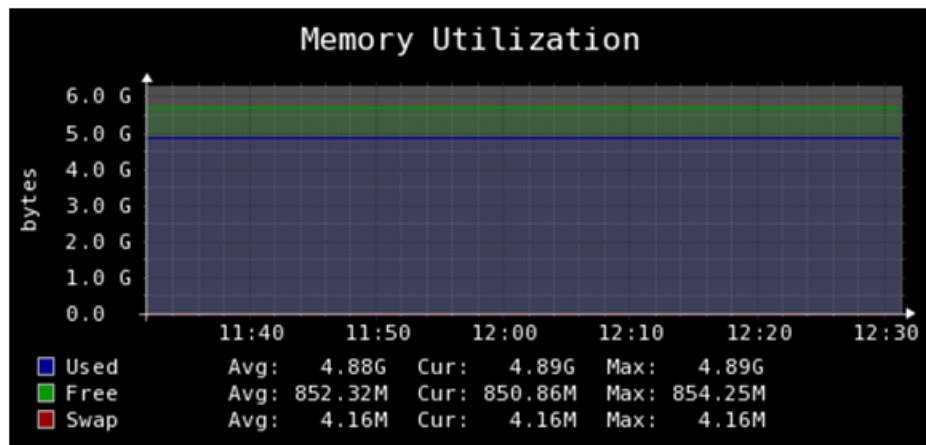
Figure 47: CPU Load Average



Memory Utilization

- Used - Amount of memory used
- Free - Amount of memory free
- Free - Amount of memory free

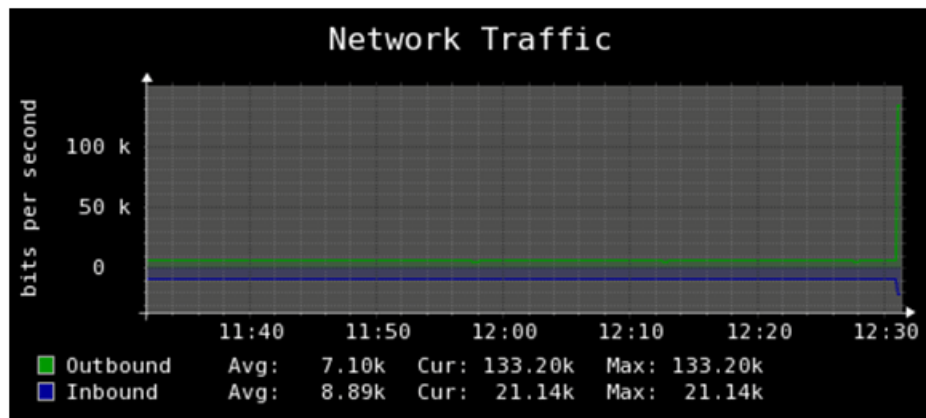
Figure 48: Memory Utilization



Network Traffic

- Outbound - Amount of traffic leaving the box
- Inbound - Amount of traffic entering the box

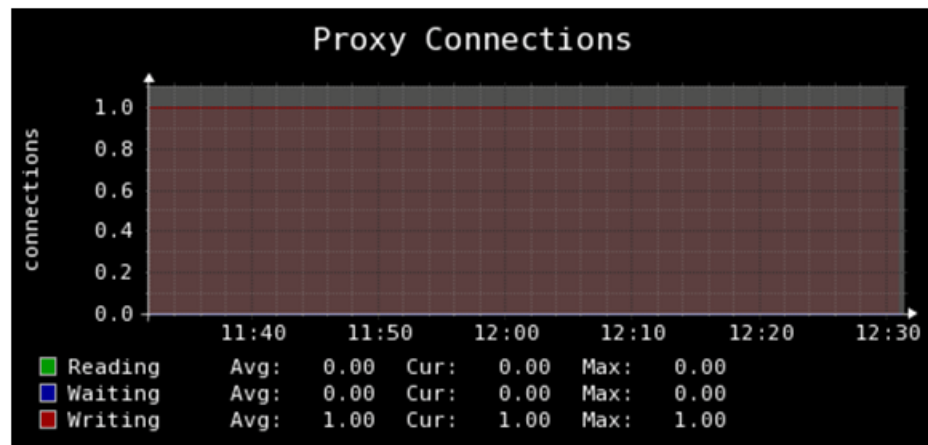
Figure 49: Network Traffic



Proxy Connections

- Reading - Number of TCP connections reading data
- Waiting - Number of TCP connections waiting
- Writing - Number of TCP connection writing Proxy

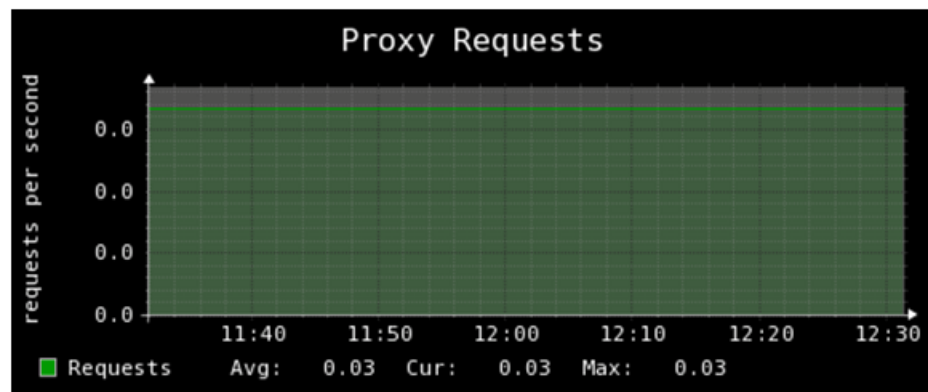
Figure 50: Proxy Connections



Proxy Requests

- Requests - Current number of HTTP/HTTPS requests being processed

Figure 51: Proxy Requests



Master - Slave Mode—In the case of the appliance running in multi-server mode, the systems statistics will show details for each node in the cluster as well as the key cumulative data across the entire cluster. Each system will be presented as a tab in the Web UI's System Status page with the **Aggregate** tab being first. The **Aggregate** tab always shows CPU Utilization, Network traffic, Proxy connections and Proxy requests collected from the entire active WebApp Secure cluster.

High Availability Network Failure Detection, Actions, and Monitoring

WebApp Secure high availability systems now have the ability to detect network card and/or interface failures. When traffic interfaces (or the HA interconnect) goes down, WebApp Secure can react by failing over to the other system, sending an alert to a specified contact person, performing both actions, or performing no action.

To instruct WebApp Secure to listen on a certain interface, enter the CLI and type:

```
cli system set ethmonitor <interface>
```

Once the command is run, it will add a monitor on the interface and also add it into the colocation group along with the rest of the data services. Technically, you could add multiple interfaces to monitor, and the monitored interfaces are the same across both system in the pair.

In the case of a failure condition described below, the appropriate failover or other action will take place.

Table 10: Failure Scenarios

Symptom	Log	Alert	Failover	Split Brain	Recommended Action	Notes
Loss of interconnect	yes	yes	no	no	none	No alerting is sent out directly from the failure but an alert will be sent from monitoring based on DRBD falling out of sync.
Loss of Monitored Interface on Slave	yes	yes	no	no	none	none
Loss of Monitored Interface on Master	yes	yes	yes	no	none	none
Loss of Monitored Interfaces on Master and Slave	yes	yes	no	no	Connectivity of traffic interfaces should be fixed immediately.	No traffic will be processed.
Loss of Monitored Interface on Master and Interconnect	yes	yes	yes	no	Connectivity of interconnect interface should be fixed immediately.	Some data may be lost in this state.
Loss of Interconnect and Management Interfaces	yes	yes	no	yes	Connectivity of interconnect and/or management interfaces should be fixed immediately.	None

- Related Documentation**
- [High Availability Overview on page 21](#)
 - [High Availability Settings on page 32](#)
 - [Configuring High Availability on page 30](#)

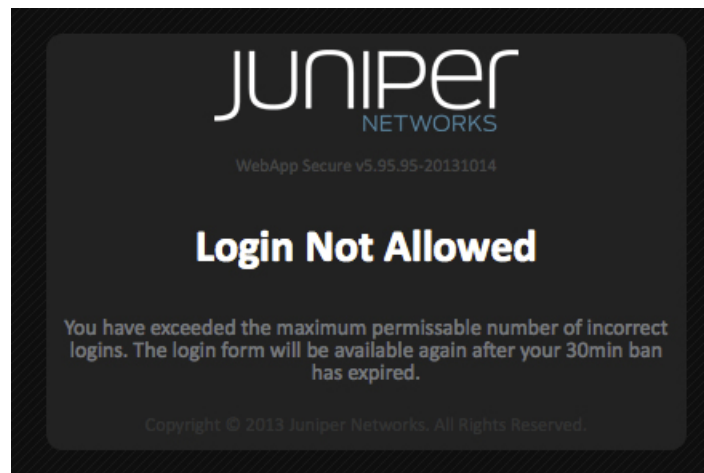
Unblock Web UI Login Ban

After six failed login attempts, the IP address in question is blocked from further login attempts for a period of thirty minutes.



NOTE: Once you reach four failed login attempts, you are warned that you will be banned after two more failed attempts.

Figure 52: Blocked Login



A system administrator can remove this block before the thirty minute time-frame has expired by using the following command:

```
[root@johnsmith-vm mykui]# cli
Welcome to the WebApp Secure CLI
> system
system> unlock_webui
Confirm Unlock [y|N]: y
system> exit
```



NOTE: This command unlocks all locked accounts.

Health Check URL

The Health Check URL lets an external system (typically a load balancer) that confirms the WebApp Secure system is operating properly. The system will generate a file name consisting of an arbitrary string of characters; make a note of it. If an HTTP request is

sent to WebApp Secure for this file name, it will return **200 OK**, with a code in the body of the message. The responses are as follows.

Table 11: Health Check responses and corresponding meanings.

Response	Meaning
No response	WebApp Secure is offline
200 OK, plus OK	WebApp Secure is fully functional and is protecting your websites
200 OK, plus DISABLED	WebApp Secure is running, but has been disabled or the license has expired
200 OK, plus STAND-BY [...]	WebApp Secure is waiting on an external resource. The contents of [...] will provide additional information

The format of the HTTP request should be: `http://jws_fullyqualifieddomainname_or_IPaddress/filenamegeneratedbyjws`

Self-Monitoring

WebApp Secure can monitor its own status and in the case of a failure automatically recover or notify you when there is a non-recoverable event. Optionally, you change thresholds and settings for self-monitoring alerting parameters. See “[Self-Monitoring Configuration Variables](#)” on page 98 for details.

Related Documentation

- For more information about Alerts, see [System Status on page 269](#).

Self-Monitoring Configuration Variables

Monitoring and alerting on key systems when there is a failure helps you to better understand what is occurring and why. This also helps customer support to troubleshoot issues. The following categories are checked as part of self-monitoring:



NOTE: Self-monitoring alerts are only sent to mws.log if the alert service is enabled and there is at least one contact configured. If the alert service is not enabled, self-monitoring alerts are not sent. See “[Alert Service](#)” on page 63 for instructions to turn this service on.

- disk—Stores the percentage of free disk space per mount point. It runs every 5 minutes.
- load—Stores the five minute load average. It runs every 5 minutes.
- swap—Stores swapcached, swaptotal, and swapfree in KB. It also stores percentused as a percentage. It runs every 5 minutes.
- raid—Stores the raid status as running or failed. It only runs on hardware appliances, and it runs every 10 minutes.

- **sessions**—Stores number of sessions in last 24 hours as `last24hours`, largest session group in last 24 hours as `maxgroupsize`, and session with the most request in last 24 hours as `maxrequests`. It runs once an hour.
- **incidents**—Stores incidents with the percentage of sessions in the last 24 hours as the incident name. It runs once an hour.
- **logsize**—Stores the log file size as `log filename`. It runs every 5 minutes.
- **logging.resources.TERM**—Stores the number of times the regex is found since the last time it ran. It runs every 5 minutes.
- **appliance**—Stores the status of each script returned by `/etc/init.d/mykonos-appliance` status. This will also attempt to restart failed services up to the `max_restart` config variable. This script runs every 1 minute.
- **services**—Stores the status of `nginx`, `pyro`, `postgres`. This will also attempt to restart any of these if they fail up to the `max_restart` config variable. These scripts run every minute.
- **ha**—Stores the ring status and sync status of the disks in HA mode. This will also store the current master in HA mode.
- **interconnect** —Stores the latency and packet loss of the interconnect in HA mode.

Configuration variables for monitored categories are as follows:

- Syntax: **system.monitor.alert.interval [INTEGER]**
Default Value: 7200 - Sets the re-alert interval for non-acknowledged alerts in seconds.
- Syntax: **system.monitor.alert.enabled [true|false]**
Enable or disable all alerts.
- Syntax: **system.monitor.CATEGORY.collect_stats [true|false]**
Enables historic stats to be collected for that category.

Setting	Default
<code>system.monitor.appliance.collect_stats</code>	true
<code>system.monitor.disk.collect_stats</code>	true
<code>system.monitor.load.collect_stats</code>	true
<code>system.monitor.services.collect_stats</code>	true
<code>system.monitor.logging.resources.clientaborted.collect_stats</code>	true
<code>system.monitor.logging.resources.outofmemory.collect_stats</code>	true
<code>system.monitor.ha.oos.collect_stats</code>	true
<code>system.monitor.ha.ring.interconnect.collect_stats</code>	true

Setting	Default
<code>system.monitor.ha.ring.management.collect_stats</code>	true
<code>system.monitor.interconnect.latency.collect_stats</code>	true
<code>system.monitor.interconnect.loss.collect_stats</code>	true

- Syntax: `system.monitor.CATEGORY.threshold [THRESHOLD]`

Sets the threshold for the alert to trigger. The threshold is based on the type of check. For status type checks, there are two options [stopped|failed]. For other checks it should be a numeric value.

Setting	Default
<code>system.monitor.appliance.threshold</code>	failed
<code>system.monitor.disk.threshold</code>	85
<code>system.monitor.incidents.threshold</code>	70
<code>system.monitor.load.threshold</code>	10
<code>system.monitor.sessiongroup.threshold</code>	2000
<code>system.monitor.sessions.threshold</code>	10
<code>system.monitor.swap.percentused.threshold</code>	30
<code>system.monitor.services.threshold</code>	failed
<code>system.monitor.ha.oos.threshold</code>	10240
<code>system.monitor.ha.ring.interconnect.threshold</code>	failed
<code>system.monitor.ha.ring.management.threshold</code>	failed
<code>system.monitor.interconnect.latency.threshold</code>	10
<code>system.monitor.interconnect.loss.threshold</code>	25
<code>system.monitor.raid.0.threshold</code>	failed

- Syntax: `system.monitor.CATEGORY.description [STRING]`

Sets the description used in the alerts. If not set, the system will use the CATEGORY name.

Setting	Default
<code>system.monitor.logging.resources.clientaborted.description</code>	Client aborted Errors
<code>system.monitor.logging.resources.outofmemory.description</code>	Out of Memory Errors
<code>system.monitor.ha.oos.description</code>	The amount of data out of sync between HA nodes in Kibibytes
<code>system.monitor.ha.ring.interconnect.description</code>	The status of the interconnect connection between HA nodes
<code>system.monitor.ha.ring.management.description</code>	The status of the management connection between HA nodes

- Syntax: `system.monitor.CATEGORY.alert.severity [1.0|2.0|3.0|4.0]`

Sets the severity for the alert. These are used when determining who to send the alert to. You can set your minimum alert level to 3.0. Then for checks with a severity under 3.0, you would not receive an alert.

Setting	Default
<code>system.monitor.appliance.alert.severity</code>	4.0
<code>system.monitor.disk.alert.severity</code>	3.0
<code>system.monitor.incidents.alert.severity</code>	3.0
<code>system.monitor.load.alert.severity</code>	2.0
<code>system.monitor.logging.resources.outofmemory.alert.severity</code>	2.0
<code>system.monitor.sessiongroup.alert.severity</code>	2.0
<code>system.monitor.sessions.alert.severity</code>	3.0
<code>system.monitor.services.alert.severity</code>	4.0
<code>system.monitor.ha.oos.alert.severity</code>	3.0
<code>system.monitor.ha.ring.interconnect.alert.severity</code>	4.0
<code>system.monitor.ha.ring.management.alert.severity</code>	4.0
<code>system.monitor.interconnect.latency.alert.severity</code>	3.0
<code>system.monitor.interconnect.loss.alert.severity</code>	4.0
<code>system.monitor.raid.0.alert.severity</code>	4.0

- Syntax: **system.monitor.CATEGORY.alert.below_threshold** [true|false]

Flips the check so that its a less than (Threshold < Value) check vs the default of a greater than (Value < Threshold) check.

Setting	Default
system.monitor.sessions.last24hours.alert.below_threshold	true

- Syntax: **system.monitor.CATEGORY.alert.enabled** [true|false]

Enable or disable the specific alert.

- Syntax: **system.monitor.logging.resources.CATEGORY.filename** [PATH TO FILE]

Sets the path to the filename for errors to be searched for.

Setting	Default
system.monitor.logging.resources.clientaborted.filename	/var/log/mws/mws.log
system.monitor.logging.resources.outofmemory.filename	/var/log/mws/mws.log

- Syntax: **system.monitor.logging.resources.CATEGORY.regex** [REGEX]

Sets the regex for strings to search for in logfile.

Setting	Default
system.monitor.logging.resources.clientaborted.regex	Client aborted
system.monitor.logging.resources.outofmemory.regex	OutOfMemory

- Syntax: **system.monitor.CATEGORY.alert.on_change** [true|false]

If true, threshold is not used, and an alert is sent if the value changes between checks.

Setting	Default
system.monitor.ha.master.alert.on_change	true
system.monitor.logging.resources.outofmemory.alert.on_change	true

- Syntax: **system.monitor.CATEGORY.alert.on_each_change** [true|false]

If true, and on_change is set to true, an alert is sent every time the value changes during checks.

Setting	Default
system.monitor.ha.master.alert.on_each_change	true

- Related Documentation
- [Self-Monitoring on page 98](#)
 - [System Status on page 269](#)

Managing and Viewing Logs

WebApp Secure keeps its log files in the `/var/log/mws` directory. The log files often prove useful for troubleshooting if there is ever a problem with the Appliance. WebApp Secure uses the following logs.

- **mws.log**: includes all of the systems operational logs. These entries each include a header that states which service created the log entry. See [“mws Log Format” on page 286](#).
- **access.log**: includes details of HTTP transactions that are passing between the outside user, WebApp Secure, and the protected Application Server. See [“Access Log Format” on page 279](#).
- **audit.log**: contains the systems auditing information on who has logged into the system and might include actions they performed. See [“Audit Log Format” on page 283](#).
- **firewall.log**: stores information about dropped packets from the iptables firewall. For various reasons (intentional and/or unintentional) iptables might drop a particular packet. If this happens, the event's information is logged to firewall.log. See [“Firewall Log Format” on page 284](#).
- **postgres.log**: contains logs of manipulations on the schema of the database, as well as any errors that occurred during database operations. See [“Postgres Log Format” on page 285](#).
- **security.log**: all security alerts are sent to the security log file. There are different types of security incidents that are part of this log: new profiles, security incidents, new counter responses. See [“Security Log Format” on page 281](#).

By navigating to **Configuration > Logging** in the Web UI, you can adjust logging levels for Access Logging and Security Logging.



NOTE: To change the default destination of log files, click the **Log Destinations** link at the top of the Logging Configuration page.

Set the following for Access Logging:

- Log Level: Off, Basic, Basic with Headers, Basic with Headers and body
- Log requests before processing: True or False
- Log requests to access log after processing: True or False
- Log responses to access log before processing: True or False
- Log responses to access log after processing: True or False

Set the following for Security Logging:

- Log incidents to the syslog: True or False
- Incident severity log level: Informational, Suspicious, Low, Medium, High
- Log Profile Creation: True or False
- Log Response Activation: True or False

The information logged here is usually used for troubleshooting, allowing an administrator to see exactly what the requests look like before and after processing by WebApp Secure.

Log Retention

Log Retention is located in the **Logging** section of the Web interface. You can set values for the following:

- Log File Rotation: The number of logs files to keep. (Log file rotation runs every 10 minutes. If the log file size is over the threshold, then it will be rotated. If not, a check will occur again in 10 minutes. Note that if the Log File Size is set very low, the logs can grow to be larger than the maximum log size setting.)
- Log File Size: The maximum size of each file in MB. (Note that changes made to this field are applied immediately.)



NOTE: Log File Rotation and Log File Size parameters can be set to 0 or to any positive integer value. There are no hard upper limits on these values. Such limits are wholly dependent on the amount of disk space that is available for log storage. If you set these values to 0, for Log File Rotation, no copies are saved when a file is rotated. The file is deleted. If Log File Size is set to 0, the log file is truncated every 30 minutes.

Related Documentation

- [Log File Destination on page 104](#)

Log File Destination

WebApp Secure keeps its log files in the **/var/log/mws** directory. You can change the destination to a remote server for each log file type by clicking the **Log Destinations** link at the top of the **Configuration > Logging** window.

In the Log Destinations window, you can set the following:

- Log Access Locally: True or False. If you select False, click the **Add** button beside Access at the bottom of the window to add the Server IP and Port for the destination of the log file. Click **Test Server Connection** to ensure the remote server can be reached.
- Log Security Locally: True or False. If you select False, click the **Add** button beside Security at the bottom of the window to add the Server IP and Port for the destination of the log file. Click **Test Server Connection** to ensure the remote server can be reached.

- Log Audit Locally: True or False. If you select False, click the **Add** button beside Audit at the bottom of the window to add the Server IP and Port for the destination of the log file. Click **Test Server Connection** to ensure the remote server can be reached.
- Log Default Locally: True or False. If you select False, click the **Add** button beside Default (Fallback) at the bottom of the window to add the Server IP and Port for the destination of the log file. Click **Test Server Connection** to ensure the remote server can be reached.

**Related
Documentation**

- [Managing and Viewing Logs on page 103](#)

Backup and Recovery Overview

System backups are archives of WebApp Secure system information, configuration, and licensing information, along with a copy of the latest database state. You can configure backup settings in the Web UI's by navigating to **Configuration > Backups**. Enter the following information in the Backups Configuration window:

- Encryption Key—Enter the user-defined segment of the encryption key that is used to encrypt backups.
- Retention—Enter the number of days that backups will be kept before being deleted.
- Frequency—Select the frequency for performing backups: Hourly, Daily, Weekly, Monthly

Push via SSH

WebApp Secure can push backups to an SSH server (via SCP). Configure the following fields to push via SSH.

- Push to SSH—Select True or False
- SSH Server—Enter the SSH server to which backups will be transferred
- SSH Username—Enter the username used for the transfer.
- SSH Password—Enter the password used for the transfer.
- SSH Retries—Enter the times to attempt the transfer before aborting.



NOTE: Click the **Test SSH Connection Settings** link to make sure the connection works.

Push via FTP

WebApp Secure can push backups to an FTP server. Configure the following fields to push via FTP.

- Push to FTP—Select True or False
- FTP Server—Enter the FTP server to which backups will be transferred

- FTP Username—Enter the username used for the transfer.
- FTP Password—Enter the password used for the transfer.
- FTP Retries—Enter the times to attempt the transfer before aborting.



NOTE: Click the [Test FTP Connection Settings](#) link to make sure the connection works.

Click **Save**.

You can invoke a backup from the command line mykonos-shell, by entering **system backup**. You will be prompted to confirm, and a file will be created in `/home/mykonos/backups/`.



NOTE: The file will be named `mykonos-<version>-<hostname>-<timestamp>.myk`



NOTE: WebApp Secure stores its backups in the `/home/mykonos/backups` directory.



WARNING: If you change the host name of WebApp Secure, backups made while using the old host name will no longer be valid. You can, however, revert back to the old host name, restore the backup, and change the host name back.

Restoring from a System Backup

To restore from a previously-exported system backup (`.myk` file), invoke the CLI by typing `cli` and then type **system restore** `<path/to/myk_file>`.

System backups do include a database backup, but only system settings are restored upon restoring the backup. To subsequently restore a database backup that was part of a system backup, you must separately restore the database using the bash command **sudo mykonos-db restore latest**.



NOTE: After restoring from a system backup, supplying the latest keyword to the `mykonos-db restore` command will restore the database state at the time of the system backup, and not necessarily the most up-to-date database backup (chronologically), that is, until another database backup is performed.

Database Backup and Restore

To restore the data that is displayed in the Monitoring Console from a back up, you must use the command line utility specialized for the database backups. This does not include configuration or other system settings, only database information.



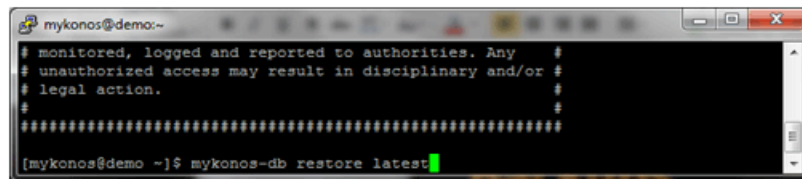
NOTE: If the data is being restored to the console, a database backup will need to be specified from `/usr/share/msa/database` or use the `latest` option to restore from the last valid backup.

A restore is run with the following command: **sudo mykonos-db <option>**

The options for the command above are as follows:

- backup
- restore (filename)
- restore latest
- clean

Figure 53: Restore Backup



Related Documentation

- Backup and Recovery Overview on page 105

About Security Intelligence

Security Intelligence describes a security solution comprised of several Juniper Networks security products. An essential part of the solution is the Spotlight Connector which is a virtual appliance. The Spotlight Connector is an on-premise component which serves as an intermediary between the SRX series and various sources of security intelligence feeds. The Spotlight Connector publishes the submitted threat data as a standard feed to the SRX series device for automatically filtering traffic on both network and application layers.

WebApp Secure contributes to the effectiveness of Security Intelligence by publishing attacker information to the Spotlight Connector. The Spotlight Connector can then determine SRX series security actions to take against known attackers and publish these actions to SRX series devices.

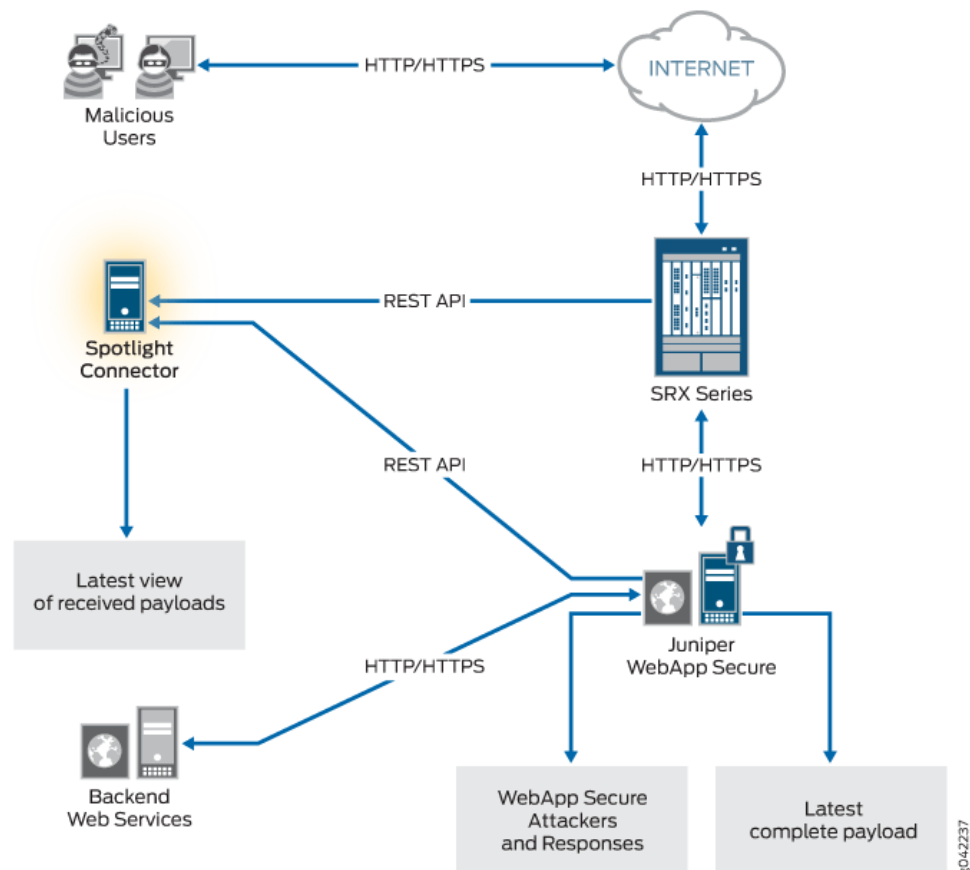


NOTE: In the WebApp Secure Web UI, under the Juniper Spotlight menu item, you can choose to enable Spotlight Secure and/or Spotlight Connector if you are using these services. Both are part of Security Intelligence, although they are different types of services. Spotlight Secure provides a database of known attackers to WebApp Secure for use throughout the appliance. See [“About Spotlight Secure” on page 112](#) for more information.

In overview, the flow of information between components is as follows:

- WebApp Secure sends IP addresses and session cookies to the Spotlight Connector.
- For each IP address or cookie, WebApp Secure suggests a threat level (1-10) and a time-frame (TTL) in seconds.
- The SRX series reads the updated feeds from the Spotlight Connector, including the WebApp Secure attacker feed, and takes the configured actions.

Figure 54: WebApp Secure-Spotlight Connector Data Flow



Related Documentation

- [Enable the Spotlight Connector Service on page 109](#)
- [Spotlight Connector Session Cookies and Locations on page 110](#)

- [About Spotlight Secure on page 112](#)
- [Enable Spotlight Secure on page 113](#)

Enable the Spotlight Connector Service

The Spotlight Connector service is disabled by default. To enable this service so that WebApp Secure can send data to the Spotlight Connector, do the following:


1. In the Web UI, go to **Juniper Spotlight > Spotlight Connector**.
2. In the Spotlight Connector window, click the **Configure** button.
3. Enter values for following fields:
 - Service Enabled—Select **True** from the pulldown menu.
 - Spotlight Connector URL—This is the URL for the Connector.
 - Spotlight Connector Auth Token—A secret token string configured on the Connector REST API for a WebApp secure appliance authorized to access the Connector. Refer to your Spotlight Connector documentation for details.
 - Spotlight Connector Group Name—A named container for attackers from this particular WebApp Secure appliance. Every WebApp Secure instance that publishes to the Connector adds the attacker cookies or IPs to their named group.
 - Spotlight Connector SSL Server Certificate— A PEM-formatted SSL certificate from the Spotlight Connector's REST API server. If the Spotlight Connector URL is HTTPS, access to the Connector from WebApp Secure will take place using HTTP over SSL. Note that if the connector administrator wants to use a self-signed server certificate, the certificate may be exported as a .pem file, and the contents of the file can be entered into the server certificate configuration value.

Figure 55: Spotlight Connector Configuration

Configuration

Spotlight Connector » Configuration

Spotlight Connector

 Test Connection Settings

Service Enabled ☐

The state of the SecurityIntelligence Integration service.

Spotlight Connector URL

The entry point URL for JWAS connections to the Connector server. This is available from the Connector server configuration. Suggested Values: http://connector2.us-west-1.amazonaws.com/jwas/manifest.json

Spotlight Connector Auth Token

The authorization key for making requests to the Connector service. This is available from the Connector server configuration.

Spotlight Connector Group Name

The group name (identifier) on the Connector server that should receive published attacker information. The group is configured on the Connector server. Suggested Values: default

Spotlight Connector SSL Server Certificate

The PEM-format SSL certificate of the Connector server, if desired. It is recommended that a self-signed certificate is set here, if the Connector server is not configured with an authority-validated certificate.

4. Click the **Save** button to save your configuration.

Related Documentation

- [About Security Intelligence on page 107](#)
- [Spotlight Connector Session Cookies and Locations on page 110](#)
- [About Spotlight Secure on page 112](#)
- [Enable Spotlight Secure on page 113](#)

Spotlight Connector Session Cookies and Locations

As part of the overall Security Intelligence solution, WebApp Secure sends information on malicious cookies and IP session to Spotlight Connector. WebApp Secure recommends a threat level for the session cookie based on a set of criteria and how malicious the associated attacker is deemed to be. Note that all sessions are not sent to the Spotlight Connector, only malicious items.

- Low threat levels (4-5) incorporate IP addresses and hosts where the threat is not as severe, the malicious activity has not been seen for a long period of time, or there is evidence of both malicious and non-malicious activity on the same host.
- Medium threat (6-7) levels represent a moderate threat and are unlikely to be non-malicious.
- High threat levels (8-10) represent severe threats at a very high level of certainty.

To view session cookies and locations sent to the Spotlight Connector, in the WebApp Secure Web UI, navigate to **Juniper Spotlight > Spotlight Connector**. There you will find a Session Cookies tab and a Locations tab.

Figure 56: Spotlight Connector Session Cookies

Spotlight Connector

Clear Local Cache Configure

Status: Enabled
Last Cookie Sync Time: 7m, 31s ago
Last Cookie Sync Status: Succeeded
Last IP Address Sync Time: 2h, 9m ago
Last IP Sync Status: Succeeded

Session Cookies (1,806) Locations (1)

Refresh this tab

SESSION COOKIES 1 - 15 of 1,806

Description	Threat	Data	Last Update	Expires
Gina 5288	8.0	rid=25nxMfQ9/jG6KvzFbQdlPg	7m, 43s ago	In 21h, 49m
Lorrie 4748	8.0	rid=t10uuNltM1wGK/SmwYONgQ	29m, 12s ago	In 20h, 43m
Agnes 5416	8.0	rid=+6fm0bG0XnmoCIYhE4PyrQ	2h, 9m ago	In 21h, 50m
Mitchell 1585	8.0	rid=p3y5vaVbLnoDoq7TiWfgZw	2h, 9m ago	In 2h, 55m
Gale 6105	8.0	rid=R8w0DGa9hrrUptrQdnsoOQ	2h, 9m ago	In 2h, 56m
Clyde 8519	8.0	rid=4oHZ9sDaC1fmr9zeKqxMSg	2h, 9m ago	In 19h, 52m
Marcy 5086	8.0	rid=pihHnp/y1c3z4xmO1oE2Yg	2h, 9m ago	In 2h, 56m
Mabel 571	8.0	rid=hpOsqbPI3Tel9/PmKoVQZA	2h, 9m ago	1h, 4m ago

Figure 57: Spotlight Connector Locations

Spotlight Connector

Clear Local Cache Configure

Status: Enabled
Last Cookie Sync Time: 7m, 31s ago
Last Cookie Sync Status: Succeeded
Last IP Address Sync Time: 2h, 9m ago
Last IP Sync Status: Succeeded

Session Cookies (1,806) Locations (1)

Refresh this tab

LOCATIONS 1 - 1 of 1

Description	Threat	Data	Last Update	Expires
10.10.0.123	4.0	168427643	2h, 9m ago	In 21h, 47m

- Related Documentation**
- [About Security Intelligence on page 107](#)
 - [Enable the Spotlight Connector Service on page 109](#)
 - [About Spotlight Secure on page 112](#)
 - [Enable Spotlight Secure on page 113](#)

About Spotlight Secure

Spotlight Secure provides a database of known attackers to WebApp Secure for use throughout the appliance. If enabled, a two-way communication process shares information about attackers and attacks to and from a Spotlight server run by Juniper Networks. This allows WebApp Secure to positively identify attackers that have attacked other Juniper customers. This service also provides additional details about sessions which allows Juniper to make more informed decisions on how to respond to threats. By default, the service is turned off.

The two-way link enables WebApp Secure to block attackers based only on a unique and specialized fingerprint gathered by a completely different WebApp Secure installation. It also provides a mechanism for reporting attacker information gathered on the local installation to the Global Attacker Database. Because your local WebApp Secure appliance is relaying information to a central data store, the ability to recognize attacker quickly and effectively increases as the database grows.

Here is an overview of how Spotlight Secure works:

1. A user gets profiled by WebApp Secure.
2. WebApp Secure sends a unique client fingerprint that is unique to that user.
3. The Spotlight service searches its Global Attacker Database for an attacker with the same fingerprint.
4. If a match is found, Spotlight feeds all identifying information on that user to the WebApp Secure appliance automatically.
5. If the user is not doing anything malicious, and is not found currently within Spotlight's database, the fingerprint for the user is still stored within the local session.
6. If at any point the user becomes malicious and is flagged by WebApp Secure, the appliance will submit the fingerprint and other data to the Spotlight service for inclusion in the Global Attacker Database.

- Related Documentation**
- [Enable Spotlight Secure on page 113](#)
 - [About Security Intelligence on page 107](#)
 - [Enable the Spotlight Connector Service on page 109](#)
 - [Spotlight Connector Session Cookies and Locations on page 110](#)

Enable Spotlight Secure

WebApp Secure builds attacker fingerprints from characteristics of attacker web requests. This information can then be queried against the Spotlight Secure attacker database to help identify and report malicious activity. To use this service, you must enable it.

1. In the WebApp Secure Web UI, navigate to **Juniper Spotlight > Spotlight Secure**.
2. From the Spotlight Enabled pulldown list, select **True** to enable the service.



NOTE: Navigate to **Configuration > Processors** and scroll down to **Tracking Processors** to check that the **Client Fingerprint processor** is enabled. It is required for this service.

3. In the **Server Address** field, enter the address of the Spotlight server. Once you enter the address, you click the **Test Connection to Spotlight Server** link to make sure the server can be reached.
4. For the remaining fields, it's recommended that you use the default values.
5. Click the **Save** button.

Once an attacker from another site visits a page on your site, a Spotlight profile will be created for that user. Having attackers from other sites consolidated in the Spotlight window in the Web UI does allow you to keep close tabs on them. You can view the Spotlight profiles from the Spotlight page. Each Spotlight profile will be displayed in a row, with information such as their Local Profile name, Global (Spotlight) profile name, and the first and last times seen both locally and globally.

Figure 58: Recent Attackers: Global and Local Names

Profile	Threat	Public ID	Last IP	Req	Pg	Err	First Time	Last Time	Actions
Elmer 3842 / Brown 0428	Low	EkCtla5zhQ889U19W	10.10.10.242	8	1	0	1mo, 8d ago	5d, 20h ago	
Ernestine 5614 / Taupe 2634	Low	JL2kQwXSTODmV1mBfmg	10.10.10.218	9	1	0	18d, 16m ago	3h, 24m ago	
Madge 5129	Low	aC5GNbejN8PtuEKQT4	10.10.10.134	17	1	13	6d, 20h ago	6d, 20h ago	
Mollie 3064 / Cranberry 1332	Low	AlpAlMrS24A9RPEBkdKI	10.10.10.134	12	3	0	1mo, 11d ago	4d, 1h ago	
Queen 1447	Medium	Lx46ocalWIHfKIHUCNw	10.10.10.149	7	3	0	7d, 3h ago	4d, 2h ago	
Tami 2366	Medium	6IAyASDl9jt9sVvzMRcz	10.10.10.134	7	7	0	4d, 1h ago	4d, 1h ago	
Thomas 4024	Medium	yUAJgoPaS4P7J7W3tyG	10.10.10.149	3	3	0	4d, 2h ago	4d, 2h ago	

You can view the Spotlight attackers' activities on your system on the **Sessions and Attackers** page. They are displayed with the same information as local attackers, and are indicated by the Spotlight icon next to their name.

Figure 59: Recent Attackers: Global Names

Profile	Threat	Public ID	Last IP	Req.	Pg.	Err.	First Time	Last Time	Actions
Brown 0428	Low	EkCtla5zhQU8B9U119W	10.10.10.242	8	1	0	1mo, 8d ago	5d, 20h ago	
Cranberry 1332	Low	AlgAlMrS4A9RPEBkdKI	10.10.10.134	12	3	0	1mo, 11d ago	4d, 1h ago	
Madge 5129	Low	aCt5GNbejN8PEtuEKQT4	10.10.10.134	17	1	13	6d, 20h ago	6d, 20h ago	
Queen 1447	Medium	Lz46ocalWlHfbKIHjCNw	10.10.10.149	7	3	0	7d, 3h ago	4d, 2h ago	
Tami 2366	Medium	6IAyASDt9J9sVvzMRcz	10.10.10.134	7	7	0	4d, 2h ago	4d, 2h ago	
Taupe 2634	Low	JL2kQwXSTODmV1mBfzmg	10.10.10.218	9	1	0	18d, 21m ago	3h, 29m ago	
Thomas 4024	Medium	yUAJgoPaS4PJ7W3tyG	10.10.10.149	3	3	0	4d, 2h ago	4d, 2h ago	

On the far left side of the Spotlight Attackers table is a small icon representing the local threat of the attacker, as it pertains to your site. This is a fast way to scan through the spotlight profiles and determine which ones might pose an immediate threat to your system. The severities range from Low to High.



NOTE: Throughout the Web UI, you can start to see Spotlight profiles, indicated by the Spotlight icon next to their Profile name. You can choose to display either Local or Global (Spotlight) names (or both) through the User Preferences screen.

Figure 60: User Preferences: Select Spotlight Name Preference

The screenshot shows the 'User Preferences' page with a dark theme. The 'Spotlight Name Preference' dropdown menu is open, showing three options: 'Display Global Names if Available' (selected with a checkmark), 'Display Local Names Only', and 'Display Both Names, Side by Side (widescreen only!)'. Other settings visible include Skin (Dark (Default)), Language (English (US)), Timezone (UTC), Prompt Level (Standard), Auto Refresh (checked), Refresh Interval (120), Records Per Page (15), and Debug Mode (unchecked). A 'Save' button is at the bottom left.

Related Documentation

- [About Spotlight Secure on page 112](#)
- [About Security Intelligence on page 107](#)
- [Enable the Spotlight Connector Service on page 109](#)
- [Spotlight Connector Session Cookies and Locations on page 110](#)

CHAPTER 6

The Processors

- [Processors Overview on page 122](#)
- [Complexity Rating Definitions on page 122](#)
- [Session Cookie Spoofing on page 123](#)
- [Session Cookie Tampering on page 123](#)
- [Hostname Spoofing Attempt on page 124](#)
- [Security Processors on page 124](#)
- [Honeypot Processors: Access Policy Processor on page 125](#)
- [Honeypot Processors: Access Policy Processor: Incidents - Malicious Service Call on page 126](#)
- [Honeypot Processors: Access Policy Processor: Incidents - Service Directory Indexing on page 126](#)
- [Honeypot Processors: Access Policy Processor: Incidents - Service Directory Spider on page 127](#)
- [Honeypot Processors: AJAX Processor on page 128](#)
- [Honeypot Processors: AJAX Processor: Incidents - Malicious Script Execution on page 129](#)
- [Honeypot Processors: AJAX Processor: Incidents - Malicious Script Introspection on page 130](#)
- [Honeypot Processors: Basic Authentication Processor on page 130](#)
- [Honeypot Processors: Basic Authentication Processor: Incidents - Apache Configuration Requested on page 132](#)
- [Honeypot Processors: Basic Authentication Processor: Incidents - Apache Password File Requested on page 133](#)
- [Honeypot Processors: Basic Authentication Processor: Incidents - Invalid Credentials on page 133](#)
- [Honeypot Processors: Basic Authentication Processor: Incidents - Protected Resource Requested on page 134](#)
- [Honeypot Processors: Basic Authentication Processor: Incidents - Password Cracked on page 135](#)
- [Honeypot Processors: Basic Authentication Processor: Incidents - Basic Authentication Brute Force on page 136](#)

- [Honeypot Processors: Cookie Processor on page 137](#)
- [Honeypot Processors: Cookie Processor: Incident - Cookie Parameter Manipulation on page 138](#)
- [Honeypot Processors: File Processor on page 139](#)
- [Honeypot Processors: File Processor: Incident - Suspicious Filename on page 139](#)
- [Honeypot Processors: File Processor: Incident - Suspicious File Exposed on page 140](#)
- [Honeypot Processors: File Processor: Incident - Suspicious Resource Enumeration on page 141](#)
- [Honeypot Processors: Hidden Input Form Processor on page 141](#)
- [Honeypot Processors: Hidden Input Form Processor: Incident - Hidden Parameter Manipulation on page 142](#)
- [Honeypot Processors: Hidden Input Form Processor: Incident - Parameter Type Manipulation on page 143](#)
- [Honeypot Processors: Hidden Link Processor on page 144](#)
- [Honeypot Processors: Hidden Link Processor: Incident - Link Directory Indexing on page 145](#)
- [Honeypot Processors: Hidden Link Processor: Incident - Link Directory Spidering on page 145](#)
- [Honeypot Processors: Hidden Link Processor: Incident - Malicious Resource Request on page 146](#)
- [Honeypot Processors: Query String Processor on page 146](#)
- [Honeypot Processors: Query String Processor: Incident - Query Parameter Manipulation on page 147](#)
- [Honeypot Processors: Robots Processor on page 148](#)
- [Honeypot Processors: Robot Processor: Incident - Malicious Spider Activity on page 148](#)
- [Activity Processors on page 149](#)
- [Activity Processors: Custom Authentication Processor: Incident - Auth Input Parameter Tampering on page 150](#)
- [Activity Processors: Custom Authentication Processor: Incident - Auth Query Parameter Tampering on page 151](#)
- [Activity Processors: Custom Authentication Processor: Incident - Auth Cookie Tampering on page 151](#)
- [Activity Processors: Custom Authentication Processor: Incident - Authentication Brute Force on page 152](#)
- [Activity Processors: Custom Authentication Processor: Incident - Auth Invalid Login on page 152](#)
- [Activity Processors: Cookie Protection Processor on page 153](#)
- [Activity Processors: Cookie Protection Processor: Incident - Application Cookie Manipulation on page 154](#)
- [Activity Processors: Error Processor on page 154](#)
- [Activity Processors: Error Processor: Incident - Illegal Response Status on page 159](#)

- [Activity Processors: Error Processor: Incident - Suspicious Response Status on page 160](#)
- [Activity Processors: Error Processor: Incident - Unexpected Response Status on page 160](#)
- [Activity Processors: Error Processor: Incident - Unknown Common Directory Requested on page 161](#)
- [Activity Processors: Error Processor: Incident - Unknown User Directory Requested on page 161](#)
- [Activity Processors: Error Processor: Incident - Common Directory Enumeration on page 162](#)
- [Activity Processors: Error Processor: Incident - User Directory Enumeration on page 162](#)
- [Activity Processors: Error Processor: Incident - Resource Enumeration on page 163](#)
- [Activity Processors: Header Processor on page 164](#)
- [Activity Processors: Header Processor: Incident - Duplicate Request Header on page 165](#)
- [Activity Processors: Header Processor: Incident - Duplicate Response Header on page 166](#)
- [Activity Processors: Header Processor: Incident - Illegal Request Header on page 166](#)
- [Activity Processors: Header Processor: Incident - Illegal Response Header on page 167](#)
- [Activity Processors: Header Processor: Incident - Missing All Headers on page 167](#)
- [Activity Processors: Header Processor: Incident - Missing Host Header on page 168](#)
- [Activity Processors: Header Processor: Incident - Missing Request Header on page 168](#)
- [Activity Processors: Header Processor: Incident - Missing Response Header on page 169](#)
- [Activity Processors: Header Processor: Incident - Missing User Agent Header on page 169](#)
- [Activity Processors: Header Processor: Incident - Request Header Overflow on page 169](#)
- [Activity Processors: Header Processor: Incident - Unexpected Request Header on page 170](#)
- [Activity Processors: Method Processor on page 170](#)
- [Activity Processors: Method Processor: Incident - Illegal Method Requested on page 171](#)
- [Activity Processors: Method Processor: Incident - Unexpected Method Requested on page 172](#)
- [Activity Processors: Method Processor: Incident - Missing HTTP Protocol on page 173](#)
- [Activity Processors: Method Processor: Incident - Unknown HTTP Protocol on page 173](#)
- [Tracking Processors: Etag Beacon Processor on page 174](#)
- [Tracking Processors: Etag Beacon Processor: Incident - Session Etag Spoofing on page 174](#)
- [Tracking Processors: Client Beacon Processor on page 175](#)
- [Tracking Processors: Client Beacon Processor: Incident - Beacon Parameter Tampering on page 176](#)
- [Tracking Processors: Client Beacon Processor: Incident - Beacon Session Tampering on page 177](#)
- [Tracking Processors: Client Fingerprint Processor on page 177](#)

- [Tracking Processors: Client Fingerprint Processor: Incident - Fingerprint Directory Indexing on page 180](#)
- [Tracking Processors: Client Fingerprint Processor: Incident - Fingerprint Directory Probing on page 181](#)
- [Tracking Processors: Client Fingerprint Processor: Incident - Fingerprint Manipulation on page 181](#)
- [Tracking Processors: Client Classification Processor on page 182](#)
- [Response Processors on page 184](#)
- [Response Processors: Block Processor on page 185](#)
- [Response Processors: Request Captcha Processor on page 186](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Answer Automation on page 189](#)
- [Response Processors: Request Captcha Processor: Incident - No Captcha Answer Provided on page 190](#)
- [Response Processors: Request Captcha Processor: Incident - Multiple Captcha Request Overflow on page 190](#)
- [Response Processors: Request Captcha Processor: Incident - Unsupported Audio Captcha Requested on page 191](#)
- [Response Processors: Request Captcha Processor: Incident - Bad Captcha Answer on page 192](#)
- [Response Processors: Request Captcha Processor: Incident - Mismatched Captcha Session on page 193](#)
- [Response Processors: Request Captcha Processor: Incident - Expired Captcha Request on page 193](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Request Tampering on page 194](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Signature Tampering on page 195](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Signature Spoofing on page 196](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Cookie Manipulation on page 196](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Image Probing on page 197](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Request Size Limit Exceeded on page 198](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Disallowed MultiPart on page 198](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Directory Indexing on page 199](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Directory Probing on page 200](#)

- [Response Processors: Request Captcha Processor: Incident - Captcha Parameter Manipulation on page 201](#)
- [Response Processors: Request Captcha Processor: Incident - Captcha Request Replay Attack on page 202](#)
- [Response Processors: Request Captcha Processor: Incident - Multiple Captcha Replays on page 203](#)
- [Response Processors: Request Captcha Processor: Incident - Multiple Captcha Disallow Multipart on page 204](#)
- [Response Processors: Request Captcha Processor: Incident - Multiple Captcha Parameter Manipulation on page 204](#)
- [Response Processors: CSRF Processor on page 205](#)
- [Response Processors: CSRF Processor: Incident - CSRF Parameter Tampering on page 208](#)
- [Response Processors: CSRF Processor: Incident - Multiple CSRF Parameter Tampering on page 209](#)
- [Response Processors: CSRF Processor: Incident - CSRF Remote Script Inclusion on page 209](#)
- [Response Processors: CSRF Processor: Incident - HTTP Referers Disabled on page 210](#)
- [Response Processors: Header Injection Processor on page 211](#)
- [Response Processors: Force Logout Processor on page 211](#)
- [Response Processors: Strip Inputs Processor on page 212](#)
- [Response Processors: Slow Connection Processor on page 212](#)
- [Response Processors: Warning Processor on page 213](#)
- [Response Processors: Warning Processor: Incident - Warning Code Tampering on page 214](#)
- [Response Processors: Application Vulnerability Processor on page 214](#)
- [Response Processors: Application Vulnerability Processor: Incident - App Vulnerability Detected on page 215](#)
- [Response Processors: Support Processor on page 215](#)
- [Response Processors: Cloppy Processor on page 217](#)
- [Response Processors: Login Processor on page 218](#)
- [Response Processors: Login Processor: Incident - Site Login Invalid on page 224](#)
- [Response Processors: Login Processor: Incident - Site Login Multiple IP on page 224](#)
- [Response Processors: Login Processor: Incident - Site Login Multiple Usernames on page 225](#)
- [Response Processors: Login Processor: Incident - Site Login User Sharing on page 225](#)
- [Response Processors: Login Processor: Incident - Site Login User Pooling on page 226](#)
- [Response Processors: Login Processor: Incident - Site Login User Brute Force on page 226](#)
- [Response Processors: Login Processor: Incident - Site Login Brute Force on page 227](#)

- [Response Processors: Login Processor: Incident - Site Login Username Scan on page 227](#)
- [Response Processors: Google Map Processor on page 228](#)

Processors Overview

WebApp Secure uses a modular approach to securing your application. Each module is responsible for monitoring, detecting and securing a particular aspect of the application and/or individual HTTP request/ response. These logical entities are referred to as Security Processors. Processors are the configurable operators that implement an additional layer of security between the application/web servers and the end user. They are responsible for analyzing the request and response data sent to and from the server and they monitor anything from the state of injected honey pots to contents of the headers and the body of the HTTP/HTTPS requests and responses.

Processors can be managed through the system configuration user interface. While some of the operations can be as simple as incrementing a counter, others are far more sophisticated and can alter the request and response data so it is important that you configure processors correctly to ensure web application's security and functionality.

Each processor is monitoring the HTTP stream for particular alterations from what is considered typical traffic. These alterations are called "triggers". Each security processor can have several triggers they are responsible for detecting. If matched, the processor responsible for handling it will generate a security incident. Incident varies by its complexity, which is explained in the section below.

Complexity Rating Definitions

Complexity is a rating of the skill, effort, and experience necessary to trigger a specific incident. The following is a description of the rating system:

- **Informational (0.0):** Informational incidents represent information about the client that might or might not indicate malicious activity, but are not common. Informational incidents are used to identify more complex abuse patterns that cannot be identified from a single request. An example of an informational incident is when the user has disabled the Referer header.
- **Suspicious (1.0):** Suspicious incidents represent activity that is abnormal but not guaranteed to be malicious. This is similar to an informational incident, except that the event is borderline malicious, not just unusual. Just like informational incidents, suspicious incidents are used to identify more complex abuse patterns that cannot be confirmed as malicious from just one request. An example of a suspicious incident is when the user requests a file that does not exist (404 error).
- **Low (2.0):** Low complexity incidents represent malicious activity that does not require any special tools, does not require a deep understanding of application architecture, and generally can be executed by an unsophisticated threat. An example of a low complexity incident is when the user modifies a query string parameter in the URL.
- **Medium (3.0):** Medium complexity incidents represent malicious activity that would require special tools, advanced browser configuration, scripting, or a understanding of how web applications are designed and implemented. These types of attacks are

generally not executed by unsophisticated attackers, and are more likely to be targeted at the protected site, rather than at an arbitrary IP range. An example of a medium complexity incident is when the user requests the robots.txt spider configuration file from a browser or a script spoofing its identity as a browser.

- **High (4.0):** High complexity incidents represent malicious activity that is highly advanced and requires a deep understanding of web application architecture, implementation, security features, and multi request workflows. High complexity incidents are generally far too advanced for an average attacker and usually have a specific target. An example of a high complexity incident is when a user is able to break the encryption used on basic authentication password files.

Session Cookie Spoofing

Complexity: Low (2.0)

Default Response: 1x = Logout User, 2x = 1 Day Clear Inputs, 3x = 5 Day Clear Inputs

Cause: WebApp Secure uses an HTTP cookie as one of the components of its fingerprinting technology. The session cookie is comprised of an AES-encrypted and base64-encoded numerical ID and a validation signature. Because the cookie has its own embedded digital signature, any attempt to fabricate or modify a session cookie will almost always result in a corrupted signature. If WebApp Secure detects that a cookie being provided has an invalid signature, but otherwise uses the correct format, it will trigger a "Session Cookie Spoofing" incident.

Behavior: Session cookies are commonly used by a web application order to facilitate state. HTTP, by itself, is not a stateful protocol, and without technologies like cookies, a web application would be unable to correlate requests made by the same user. When an attacker attempts to modify a cookie, especially when they are careful to follow the same format constraints as the original value (22 letters and numbers, or 16 hex characters, and so on), they are attempting to modify their state. If for example, an attacker were able to successfully guess the session cookie value of another actively logged in user, they would be able to assume that user's state (including their authentication and authorization levels). This is referred to by the WASC as a "Credential and Session Prediction" attack (see [Credential and Session Prediction](#) for information.)

Session Cookie Tampering

Complexity: Medium (3.0)

Default Response: 1x = Logout User, 2x = 1 Day Clear Inputs, 3x = 5 Day Clear Inputs

Cause: WebApp Secure uses an HTTP cookie as one of the components of its fingerprinting technology. The session cookie is comprised of an AES-encrypted and base64-encoded numerical ID and a validation signature. Because the cookie has its own embedded digital signature, any attempt to fabricate or modify a session cookie will almost always result in a corrupted signature. If WebApp Secure detects that a cookie being provided does not have a valid signature, and does not follow the correct format, it will trigger a "Session Cookie Tampering" incident.

Behavior: Session cookies are commonly used by a web application order to facilitate state. HTTP, by itself, is not a stateful protocol, and without technologies like cookies, a web application would be unable to correlate requests made by the same user. However, just like form parameters and query string parameters, cookies represent another type of user-input. Just about any attack that can be accomplished by injecting malicious values into a form input (SQL injection², XSS³, Buffer Overflow⁴, Integer Overflow⁵, and so on.), could also potentially be accomplished by injecting malicious values into the session cookie. An aggressive hacker would likely test for multiple vulnerability types in all form inputs, query parameters, and cookies, because these are the inputs most likely to be insecurely handled.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Hostname Spoofing Attempt

Complexity: Medium (3.0)

Default Response: 1x = 1 Day Block

Cause: WebApp Secure expects certain indexing clients (like Google, Yahoo! and others) to visit protected sites. These clients are identified by their originating hosts, and those hostname patterns are defined in the Client Classification Incident options. This incident is triggered when a client attempts to spoof the hostname of the IP they are originating from. A forward-confirmed reverse DNS lookup is performed to compare the client provided hostname with the actual resolved hostname.

Behavior: Manipulating the hostname requires some manipulation of a local DNS cache and has no legitimate purpose. This type of behavior is possibly related to a user that is attempting to appear to be scanning a website under the guise of another legitimate spider. If the user is knowledgeable about WebApp Secure counter response behavior, they may be attempting to avoid counter responses that are intended for clients which are not legitimate spiders.

Security Processors

The Security Processors are separated into four groups:

- Honeypot Processors
- Activity Processors
- Tracking Processors
- Response Processors

Honeypot processors contain the logic of injecting the fake vulnerabilities and points of interest to the hackers with the goal of exposing the attacker prior to them finding an actual vulnerability on the site. Activity processors are the processors that monitor for

and report any other malicious behavior. These operators watch for malicious activity based on non-injected points of interest. These typically involve monitoring headers, errors, input fields, URL sequences, and so on, with the goal of identifying malicious behavior within the valid application stream.

Activity processors enable monitoring of session traffic. Things like authentication and cookies are among the types of traffic that become introspected by various activity processors.

Tracking processors, allow for more advanced tracking of the attackers. These processors attempt to collect additional data based on behavioral characteristics and unique attacker's environment information. These "fingerprints" become a basis for the "hacker database" used in detecting attackers from the first request they make.

Response processors are the processors that are used for generating response to the end user. If turned on, these can be used to either manually or automatically (depending on the configuration) respond to a hacker as soon as their activity is detected. In case of an automated response, these can be tuned to match more or less any condition including but not limited to frequency of occurrence, complexity, types of incidents triggered.

Honey_pot Processors: Access Policy Processor

This processor injects fake permission data into the `clientaccesspolicy.xml` file of the web application's domain. The fake access policy references a fake service and grants a random domain access to call it. If the service is ever called, or any files are ever requested in the directory the service is supposedly contained in, an incident can be created. Under normal conditions, no user will ever see the `clientaccesspolicy.xml` file, and therefore be unaware of the URL to the fake service or the directory it resides in. In the cases where a Silverlight object is legitimately requesting `clientaccesspolicy.xml` from the protected domain in order to access a known service, it will not create an incident, because the service being called is defined with real access directives.

Table 12: Access Policy Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether or not to enable this process for https traffic.
Advanced			
Fake Service	String	Random	The fake service the user requested.
Incident: Malicious Service Call	Boolean	True	The user manually entered the URL into the browser and accessed the service that way. They did not call the function.
Incident: Service Directory Indexing	Boolean	True	The user asked for a file index on the directory that contains the fake service.

Table 12: Access Policy Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Incident: Service Directory Spider	Boolean	True	The user is issuing requests for resources inside the directory that contains the fake service. Since the directory does not exist, all of these types of requests are unintended and malicious.

HoneyPot Processors: Access Policy Processor: Incidents - Malicious Service Call

Complexity: Medium (3.0)

Default Response: 1x = 5 day Clear Inputs

Cause: WebApp Secure adds a fake cookie to the websites it protects. The cookie is intended to look as though it is part of the applications overall functionality, and is often selected to appear vulnerable (such as naming the cookie 'debug' or 'admin' and giving it a numerical or Boolean value). The "Cookie Parameter Manipulation" incident is triggered whenever the fake cookie value changes its value.

Behavior: Modifying the inputs of a page is the foundation of a large variety of attack vectors. Basically, if you want to get the backend server to do something different, you need to supply different input values (either by cookie, query string, URL, or form parameters). Depending on what value the user chose for the input, the attack could fall under large number of vectors, including "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", and "SQL injection" among many others. A common practice is to first spider the website, then test every single input on the site for a specific set of vulnerabilities. For example, the user might first index the site, then visit each page on the site, then test every exposed input (cookie, query string, and form inputs) with a list of SQL injection tests. These tests are designed to break the resulting page if the input is vulnerable. As such, the entire process (which can involve thousands of requests) can be automated and return a clean report on which inputs should be targeted. Because a WebApp Secure cookie looks just like a normal application cookie, a spider that tests all inputs will eventually test the fake cookie as well. This means that if there is a large volume of this incident, it is likely due to such an automated process. It should be assumed that the values tested against the fake cookie, have also been tested against the rest of the cookies on the site.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

HoneyPot Processors: Access Policy Processor: Incidents - Service Directory Indexing

Complexity: Medium (3.0)

Default Response: 1x = 5 day Block

Cause: Originally, embedded HTML technologies such as Flash and Java, were not able to communicate with third party domains. This was a security constraint to prevent a malicious Java or Flash object from performing unwanted actions against a site other than the one hosting the object (for example, a Java applet that brute forces a Gmail login in the background). This limitation was eventually decreased in order to facilitate more complex mash-ups of information from a variety of sources. However to prevent any untrusted websites from abusing this new capability, a resource called the "clientaccesspolicy.xml" was introduced. Now, when a plugin object wants to communicate with a different domain, it will first request "clientaccesspolicy.xml" from that domain. If the file specifies that the requesting domain is allowed to access the specified resource, then the plugin object will be given permission to communicate directly with the third party. The clientaccesspolicy.xml therefore provides a convenient reference for hackers when trying to scope the attack surface of the website. For example, there can be a vulnerable service listed in clientaccesspolicy.xml, but that service cannot be referenced anywhere else on the site. So unless the hacker looks at clientaccesspolicy.xml, they would never even know the service existed. WebApp Secure will inject a fake service definition into the clientaccesspolicy.xml file in order to identify which users are manually probing the file for information. The "Service Directory Indexing" incident will be triggered if the user attempts to get a file listing from the directory the fake service is supposedly located in.

Behavior: Attempting to get a file listing from the directory where the potentially vulnerable service is located is likely in an effort to identify other unreferenced vulnerable services, or possibly even data or source files used by the service. Such a request represents a "Directory Indexing" attack, and is generally performed while attempting to establish a full understanding of a websites attack surface.

Honeygot Processors: Access Policy Processor: Incidents - Service Directory Spider

Complexity: Medium (3.0)

Default Response: 1x = 5 day Block

Cause: Originally, embedded HTML technologies such as Flash and Java, were not able to communicate with third party domains. This was a security constraint to prevent a malicious Java or Flash object from performing unwanted actions against a site other than the one hosting the object (for example, a Java applet that brute forces a Gmail login in the background). This limitation was eventually decreased in order to facilitate more complex mash-ups of information from a variety of sources. However to prevent any untrusted websites from abusing this new capability, a resource called the "clientaccesspolicy.xml" was introduced. Now, when a plugin object wants to communicate with a different domain, it will first request "clientaccesspolicy.xml" from that domain. If the file specifies that the requesting domain is allowed to access the specified resource, then the plugin object will be given permission to communicate directly with the third party. The clientaccesspolicy.xml therefore provides a convenient reference for hackers when trying to scope the attack surface of the website. For example, there can be a vulnerable service listed in clientaccesspolicy.xml, but that service cannot be referenced anywhere else on the site. So unless the hacker looks at clientaccesspolicy.xml, they would never even know the service existed. WebApp Secure will inject a fake service definition into the clientaccesspolicy.xml file in order to identify which users are manually

probing the file for information. The "Service Directory Spidering" incident will be triggered if the user attempts to request a random file inside the directory the fake service is supposedly located in.

Behavior: Requesting a random file from the directory where the potentially vulnerable service is supposedly located is likely in an effort to identify other unreferenced resources. This could include configuration files, other services, data files, and so on. Usually an attacker will first attempt to get a full directory index (which only takes one request), but if that fails, the only other technique is to guess the filenames (which could take thousands of requests). Because guessing the file names can take so many requests, there are several publicly available tools that can enumerate over a large list of common file and directory names in a matter of minutes. This type of behavior is an attempt to exploit a server for "Predictable Resource Location" vulnerabilities, and is generally done while the attack is trying to scope the web applications attack surface.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium Web Site](#) and search for the attack name to learn more about it.

Honeypot Processors: AJAX Processor

A mistake commonly made by web developers is to consolidate every JavaScript file used by their website into a single file. They then reference that one file from every page on the site, regardless of whether it needs all of the code defined in the file. This is an optimization trick that works, but exposes potential vulnerabilities. The goal is to get the browser to cache all of the external JavaScript, so that you don't need to keep downloading additional code as you navigate the site. Consider the case where one of the pages on the site contains an administrative console written with AJAX technology. In the administrative page, there is a JavaScript file that contains code for managing users of the site (creating user, deleting users, getting user details, and so on). Normally only administrators would visit this page, and they would be the only ones who can see this code. Once all JavaScript on the site is consolidated however, these types of sensitive functions tend to get mixed into the rest of the safer functions. Hackers look for these types of functions in order to find both the administrative page that uses them, as well as exploit the function itself. The goal of this trap is to emulate this common mistake and entice hackers into attempting to exploit the "sensitive looking" function.

Table 13: AJAX Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			

Table 13: AJAX Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Inject Script Enabled	Boolean	True	Whether to inject the fake Javascript code into HTML responses.
Service	Configurable	AJAX Service	The fake service to expose.
Incident: Malicious Script Execution	Boolean	True	The user executed the fake JavaScript function.
Incident: Malicious Script Introspection	Boolean	True	The user manually entered the URL into the browser and accessed the service that way. They did not call the function.

Honeypot Processors: AJAX Processor: Incidents - Malicious Script Execution

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection 2-6 seconds and permanent Clear Inputs in 10 minutes.

Cause: WebApp Secure injects a fake JavaScript file into the websites it protects. This fake JavaScript file is designed to look as though it is intended for administrative use only, but has been mistakenly linked in with non administrative pages. The JavaScript file exposes an AJAX function that communicates with a potentially vulnerable fake service. If the user attempts to invoke this function using a tool like Firebug, this incident will be triggered.

Behavior: It is common practice to create a few single JavaScript files that contain the majority of the code your site needs, and then importing that code into all of the pages. This increases the performance of the site, because the user can download and cache all the JavaScript at once, rather than having to re-download all or some of it again on every page change. However in some cases, developers mistakenly include sensitive administrative functions in with common functions needed by unauthenticated users. For example, a developer might include an "addUser" function into a file that also contains a "changeImageOnHover" function. The "addUser" function can only be called from an administrative UI (behind a login), while the hover image effect would be called on a lot of different pages. Hackers often look through all of the various Javascript files being included on the pages of a website in order to find references to other services that might be vulnerable. Once a function has been identified, the hacker will attempt to find a way to exploit the service the function uses. Because the attacker is actually executing the function instead of attempting to directly communicate with the potentially vulnerable service, this is likely a less sophisticated attack. They are more than likely just trying to determine if the service actually exists, and if they can call it without being authenticated, however depending on the values they supplied as arguments to the function, this could be a number of different attack types, including "Abuse of Functionality", "Buffer Overflow", "Denial of Service", "Format String", "Integer Overflows", "OS Commanding", and "SQL Injection."



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Honeypot Processors: AJAX Processor: Incidents - Malicious Script Introspection

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection 2-6 seconds and Captcha. 2x = Slow Connection 4-14 seconds and permanent Block in 10 minutes.

Cause: WebApp Secure injects a fake JavaScript file into the websites it protects. This fake JavaScript file is designed to look as though it is intended for administrative use only, but has been mistakenly linked in with non administrative pages. The JavaScript file exposes an AJAX function that communicates with a potentially vulnerable fake service. If the user manually inspects the code of the function and attempts to exploit the service it uses directly (without calling the function itself), this incident will be triggered.

Behavior: To improve performance of a website, by minimizing the number of HTTP requests (and taking advantage of browser-side caching), web developers commonly combine most of their JavaScript code into just a few files, which are then included in the HTML of the entire site. However, in some cases, developers mistakenly include sensitive administrative functions in with common functions needed by unauthenticated users. For example, a developer might include an "addUser" function into a file that also contains a "changeImageOnHover" function. The "addUser" function can only be called from an administrative UI (behind a login), while the hover image effect would be called on a lot of different pages. Hackers often look through all of the various Javascript files being included on the pages of a website in order to find references to other services that might be vulnerable. Once a function has been identified, the hacker will attempt to find a way to exploit the service the function uses. Unlike the malicious script execution incident, here the attacker has actually dissected the fake AJAX function and attempted to directly exploit the service it uses. This is a more sophisticated attack than actually calling the Javascript function, because it requires that the user understand Javascript logic. Depending on what values they are sending to the service, this could be in an effort to perform any number of exploits, including Abuse of Functionality, "Buffer Overflow", "Denial of Service", "Format String", "Integer Overflows", "OS Commanding", and "SQL Injection."



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Honeypot Processors: Basic Authentication Processor

The basic authentication processor is responsible for emulating a vulnerable authentication mechanism in the web application. This is done by publicly exposing fake

server configuration files (.htaccess and .htpasswd) that appear to be protecting a resource with basic authentication (a part of the HTTP protocol). To the attacker, the site will appear to be exposing a sensitive administrative script on the site, with weak password protection. As the malicious user identifies the availability of such publicly exposed files, they are walked through a series of steps that emulate exposing an additional piece of information. As the final step, if they end up breaking the weakly authenticated password, they will be considered a high threat.



NOTE: This processor should only be used when the site is using Apache as front end webserver due to particular files involved (.htaccess and .htpasswd) being specific to Apache webserver.)



NOTE: Browsers often ignore the body content of HTTP responses if the status code is anything other than 200. For best compatibility with different browser versions, you might want to use a 200 status code when uploading responses such as images or executable code.

Table 14: Basic Authentication Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			Whether traffic should be passed through this processor.
Processor Enabled	Boolean	True	
Advanced			
Authorized Users	Collection	Collection	A list of authorized user accounts.
Protected Resource URL	String	[random resource]	The fake protected resource.
Protected Resource Response Status	String	[random status]	The HTTP status to return when accessing the resource.
Randomization Salt	String	Random	A random set of characters used to salt the generation of code. Any value is fine here.
Incident: Password Cracked	Boolean	True	The user has successfully accessed a fake protected resource using a cracked username and password.
Incident: Apache Configuration Requested	Boolean	True	The user has requested the apache directory configuration file .htaccess.
Incident: Apache Password File Requested	Boolean	True	The user has requested the apache password file .htpasswd

Table 14: Basic Authentication Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Incident: Invalid Credentials	Boolean	True	The user has attempted to login to access the fake file protected by basic authentication, but failed.
Incident: Protected Resource Requested	Boolean	True	The user has requested a fake file which is protected by basic authentication.

Honeypot Processors: Basic Authentication Processor: Incidents - Apache Configuration Requested

Complexity: Low (2.0)

Default Response: none.

Cause: Apache is a webserver used by many websites on the Internet. As a result, hackers will often look for vulnerabilities specific to apache, because there is a good chance any given website is probably running apache. One such vulnerability involves the use of an .htaccess22 file to provide directory level configuration (such as default 404 messages, password protected resources, directory indexing options, and so on...), while not sufficiently protecting the .htaccess file itself. By convention, any resource that provides directory level configuration should not be exposed to the public. This means that if a user requests .htaccess or a related resource, they should get either a 404 or a 403 error. Unfortunately, not all web servers are configured correctly to block requests for these resources. In such a scenario, a hacker could gain valuable intelligence on the way the server is configured.

Behavior: Hackers will often attempt to get the .htaccess file from various directories on a website in an effort to find valuable information about how the server is configured. This is usually done to find a "Server Misconfiguration" weakness that might expose a "Credential/Session Prediction", "OS Commanding", "Path Traversal", or "URL Redirector Abuse" vulnerability among others. The fact that an .htaccess file is even exposed is a "Server Misconfiguration" vulnerability in itself. In this specific case, the attacker is asking for a different resource that is related to .htaccess. They are requesting a user database file for a password protected resource defined in .htaccess. This file is generally named ".htpasswd". The user either opened the .htaccess file and found the reference to .htpasswd, or they simply tried .htpasswd to see if anything came back (with or without asking for .htaccess). Either way, this behavior is involved in the establishment of a "Credential/Session Prediction" vulnerability. The request for .htpasswd is usually performed while attempting to establish the scope of the websites attack surface, although sometimes is not performed until trying to identify a valid attack vector.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

HoneyPot Processors: Basic Authentication Processor: Incidents - Apache Password File Requested

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds.

Cause: Apache is a webserver used by many websites on the Internet. As a result, hackers will often look for vulnerabilities specific to apache, because there is a good chance any given website is probably running apache. One such vulnerability involves the use of an .htaccess file to provide directory level configuration (such as default 404 messages, password protected resources, directory indexing options, and so on...), while not sufficiently protecting the .htaccess file itself. By convention, any resource that provides directory level configuration should not be exposed to the public. This means that if a user requests .htaccess or a related resource, they should get either a 404 or a 403 error. Unfortunately, not all web servers are configured correctly to block requests for these resources. In such a scenario, a hacker could gain valuable intelligence on the way the server is configured. WebApp Secure will automatically block any requests for the .htaccess resource, and return a fake version of the file. The fake version of the file will contain the directives necessary to password protect a fake resource. These directives allude to the existence of a user database file that contains usernames and encrypted passwords. The "Apache Password File Requested" incident will trigger in the event that the user requests the fake user database file (generally named .htpasswd).

Behavior: Hackers will often attempt to get the .htaccess file from various directories on a website in an effort to find valuable information about how the server is configured. This is usually done to find a "Server Misconfiguration" weakness that might expose a "Credential/Session Prediction", "OS Commanding", "Path Traversal", or "URL Redirector Abuse" vulnerability among others. The fact that an .htaccess file is even exposed is a "Server Misconfiguration" vulnerability in itself. In this specific case, the attacker is asking for a different resource that is related to .htaccess. They are requesting a user database file for a password protected resource defined in .htaccess. This file is generally named ".htpasswd". The user either opened the .htaccess file and found the reference to .htpasswd, or they simply tried .htpasswd to see if anything came back (with or without asking for .htaccess). Either way, this behavior is involved in the establishment of a "Credential/Session Prediction" vulnerability. The request for .htpasswd is usually performed while attempting to establish the scope of the websites attack surface, although sometimes is not performed until trying to identify a valid attack vector.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

HoneyPot Processors: Basic Authentication Processor: Incidents - Invalid Credentials

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection 2-6 seconds. 15x = Basic Authentication Bruteforce Incident.

Cause: Apache is a webserver used by many websites on the Internet. As a result, hackers will often look for vulnerabilities specific to apache, because there is a good chance any given website is probably running apache. One such vulnerability involves the use of an .htaccess³⁴ file to provide directory level configuration (such as default 404 messages, password protected resources, directory indexing options, and so on), while not sufficiently protecting the .htaccess file itself. By convention, any resource that provides directory level configuration should not be exposed to the public. This means that if a user requests .htaccess or a related resource, they should get either a 404 or a 403 error. Unfortunately, not all web servers are configured correctly to block requests for these resources. In such a scenario, a hacker could gain valuable intelligence on the way the server is configured. WebApp Secure will automatically block any requests for the .htaccess resource, and return a fake version of the file. The fake version of the file will contain the directives necessary to password protect a fake resource. Should the user request the password protected resource, WebApp Secure will simulate the correct authentication method defined in .htaccess, and simulate the existence of the fake resource. The "Invalid Credentials" incident will trigger in the event that the user requests the fake password protected file and supplies an invalid username and password (as would be the case if they requested the file in a browser and guessed a username and password at the login prompt).

Behavior: Hackers will often attempt to get the .htaccess file from various directories on a website in an effort to find valuable information about how the server is configured. This is usually done to find a "Server Misconfiguration" weakness that might expose a "Credential/Session Prediction", "OS Commanding", "Path Traversal", or "URL Redirector Abuse" vulnerability among others. The fact that an .htaccess file is even exposed is a "Server Misconfiguration" vulnerability in itself. In this specific case, the attacker is asking for a different resource that is referenced only from .htaccess. The fake resource is password protected, and the user has attempted to authenticate with bad credentials. This is most likely in an effort to guess a valid username and password combination, such as "admin:admin", or "guest:guest". It can also be part of a larger brute force attempt, where the attacker tries a long list of possible combinations. This is a poor method for locating valid usernames and passwords, because the user database file .htpasswd is actually exposed (albeit fake). So a brute force attack (represented by a large quantity of this incident type) generally means the attacker is less sophisticated.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

HoneyPot Processors: Basic Authentication Processor: Incidents - Protected Resource Requested

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds.

Cause: Apache is a webserver used by many websites on the Internet. As a result, hackers will often look for vulnerabilities specific to apache, because there is a good chance any given website is probably running apache. One such vulnerability involves the use of an .htaccess file to provide directory level configuration (such as default 404 messages, password protected resources, directory indexing options, and so on), while not sufficiently protecting the .htaccess file itself. By convention, any resource that provides directory level configuration should not be exposed to the public. This means that if a user requests .htaccess or a related resource, they should get either a 404 or a 403 error. Unfortunately, not all web servers are configured correctly to block requests for these resources. In such a scenario, a hacker could gain valuable intelligence on the way the server is configured. WebApp Secure will automatically block any requests for the .htaccess resource, and return a fake version of the file. The fake version of the file will contain the directives necessary to password protect a fake resource. Should the user request the password protected resource, WebApp Secure will simulate the correct authentication method defined in .htaccess, and simulate the existence of the fake resource. The "Protected Resource Requested" incident will trigger in the event that the user requests the fake password protected file and does not supply a username and password (as would be the case if they requested the file in a browser and canceled the login prompt).

Behavior: Hackers will often attempt to get the .htaccess file from various directories on a website in an effort to find valuable information about how the server is configured. This is usually done to find a "Server Misconfiguration" weakness that might expose a "Credential/Session Prediction", "OS Commanding", "Path Traversal", or "URL Redirector Abuse" vulnerability among others. The fact that an .htaccess file is even exposed is a "Server Misconfiguration" vulnerability in itself. In this specific case, the attacker is asking for a different resource that is referenced only from .htaccess. The resource is password protected, but the user has not yet tried to supply credentials. This is most likely in an attempt to see if the password protected file actually exists.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

HoneyPot Processors: Basic Authentication Processor: Incidents - Password Cracked

Complexity: High (4.0)

Default Response: 1x = Permanent Block.

Cause: Apache is a webserver used by many websites on the Internet. As a result, hackers will often look for vulnerabilities specific to apache, because there is a good chance any given website is probably running apache. One such vulnerability involves the use of an .htaccess46 file to provide directory level configuration (such as default 404 messages, password protected resources, directory indexing options, and so on.), while not sufficiently protecting the .htaccess file itself. By convention, any resource that provides directory level configuration should not be exposed to the public. This means that if a user requests .htaccess or a related resource, they should get either a 404 or a 403 error. Unfortunately, not all web servers are configured correctly to block requests for these

resources. In such a scenario, a hacker could gain valuable intelligence on the way the server is configured. WebApp Secure will automatically block any requests for the .htaccess resource, and return a fake version of the file. The fake version of the file will contain the directives necessary to password protect a fake resource. The directives will also allude to the existence of a password database file. If the attacker requests the password database file, and then uses a tool such John The Ripper to crack one of the encrypted passwords, they will be able to authenticate against the fake protected resource successfully. Should the user request the password protected resource, and supply a valid username and password combination (as defined in the password database), the "Password Cracked" incident will be triggered.

Behavior: Hackers will often attempt to get the .htaccess file from various directories on a website in an effort to find valuable information about how the server is configured. This is usually done to find a "Server Misconfiguration" weakness that might expose a "Credential/Session Prediction", "OS Commanding", "Path Traversal", or "URL Redirector Abuse" vulnerability among others. The fact that an .htaccess file is even exposed is a "Server Misconfiguration" vulnerability in itself. In this specific case, the attacker is asking for a different resource that is referenced only from .htaccess. The fake resource is password protected, and the user has supplied valid authentication credentials. The only way to obtain valid credentials is to either brute force the login (which would be the case if there were excessive numbers of "Invalid Credential" incidents), or to access the fake password database file (usually .htpasswd) and crack one of the encrypted passwords using an encryption cracking tool. This represents the final and most complicated step in a successful "Credential/Session Prediction" exploit, and is usually performed long after the attack surface of the site has been fully scoped. Unless there are excessive numbers of "Invalid Credential" incidents, which would be the case in a brute force attack, the user must have also requested ".htpasswd", and therefore should also have an "Apache Password File Requested" incident. If this incident is missing, then the hacker has likely established two independent profiles in WebApp Secure.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Honeypot Processors: Basic Authentication Processor: Incidents - Basic Authentication Brute Force

Complexity: Medium (3.0)

Default Response: 1X - CAPTCHA; 2x = Permanent Block.

Cause: Apache is a very common webserver. As a result, hackers will often look for vulnerabilities specific to Apache, because there is a good chance that any given website is running Apache. One such vulnerability involves the use of an .htaccess file to provide directory-level configuration (password-protected resources, directory indexing options, and so on), while not sufficiently protecting the .htaccess file itself. By convention, configuration files should not be exposed to the public — so if a user requests .htaccess or a related resource, they should get either a "404 Not Found" or "403 Forbidden" error.

Unfortunately, an improperly-configured installation of Apache cannot block requests for these resources. In such a scenario, a hacker could gain valuable knowledge of the way the server is configured. WebApp Secure will automatically block any requests for the .htaccess resource, and instead return a fake version of the file, which contains the directives necessary to password-protect a fake resource. Should the user request the password-protected resource, WebApp Secure will simulate the correct authentication method defined in .htaccess, and simulate the existence of the fake resource. The "Basic Authentication Brute Force" incident will trigger in the event that the user requests the fake passwordprotected file and repeatedly supplies an invalid username and password (as would be the case if the user were guessing various username and password combinations).

Behavior: Hackers will often attempt to get the .htaccess file from various directories on a website in an effort to find valuable information about how the server is configured. This is usually done to find a "Server Misconfiguration" weakness that might expose a "Credential/Session Prediction", "OS Commanding", "Path Traversal", or "URL Redirector Abuse" vulnerability among others.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium Web Site](#) and search for the attack name to learn more about it.

In this specific case, the attacker is requesting a different resource that is referenced only from .htaccess. The fake resource is password-protected, and the user has attempted to authenticate with a large number of bad credentials. This is most likely in an effort to guess a valid username and password combination, such as "admin:admin", or "guest:guest". This is a poor method for locating valid usernames and passwords, because the user database file .htpasswd is actually exposed (albeit fake). So a brute force attack generally means the attacker is less sophisticated. Because the password-protected file is not referenced from anywhere outside of .htaccess, this incident should not happen unless an "Apache Configuration Requested" incident has occurred first. If that is not the case, then the hacker has likely established two independent profiles in WebApp Secure. This type of behavior is generally performed when attempting to establish a successful attack vector.

HoneyPot Processors: Cookie Processor

Cookies are used by web applications to maintain state for a given user. They consist of key/value pairs that are passed around in headers and also stored client side. Each key/value pair has various attributes including which domains it is valid for, what paths within those domains, as well as security restrictions and expiration information. Because this is the primary way for a web application to maintain a session, hackers will often try to manipulate cookie values manually in an effort to escalate access or hijack someone else's session. All of the attacks applicable to modifying form parameters are also applicable to modifying cookie parameters. It can be possible, although unlikely, to find an SQL injection flaw in a cookie parameter.

Table 15: Cookie Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether or not to enable this process for http traffic.
Advanced			
Cookie	String	Cookie	The fake cookie to use.
Incident: Cookie Parameter Manipulation	Boolean	True	The user modified the value of a cookie which should never be modified.

Honeypot Processors: Cookie Processor: Incident - Cookie Parameter Manipulation

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection 2-6 seconds and permanent Clear Inputs in 10 minutes.

Cause: WebApp Secure adds a fake cookie to the websites it protects. The cookie is intended to look as though it is part of the applications overall functionality, and is often selected to appear vulnerable (such as naming the cookie 'debug' or 'admin' and giving it a numerical or Boolean value). The "Cookie Parameter Manipulation" incident is triggered whenever the fake cookie value changes its value.

Behavior: Modifying the inputs of a page is the foundation of a large variety of attack vectors. Basically, if you want to get the backend server to do something different, you need to supply different input values (either by cookie, query string, URL, or form parameters). Depending on what value the user chose for the input, the attack could fall under large number of vectors, including "Buffer Overflow", "XSS5", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", and "SQL injection" among many others. A common practice is to first spider the website, then test every single input on the site for a specific set of vulnerabilities. For example, the user might first index the site, then visit each page on the site, then test every exposed input (cookie, query string, and form inputs) with a list of SQL injection tests. These tests are designed to break the resulting page if the input is vulnerable. As such, the entire process (which can involve thousands of requests) can be automated and return a clean report on which inputs should be targeted. Because WebApp Secure cookie looks just like a normal application cookie, a spider that tests all inputs will eventually test the fake cookie as well. This means that if there is a large volume of this incident, it is likely due to such an automated process. It should be assumed that the values tested against the fake cookie, have also been tested against the rest of the cookies on the site.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Honeypot Processors: File Processor

When developing websites, administrators will often rename files in order to make room for a newer version of the file. They can also archive older files. A common vulnerability is the case where these older files are left in the web accessible directories, and they contain non static resources. For example, consider the case where a developer renames shopping_cart.php to shopping_cart.php.bak. If an attacker looks for php files and tries to access all of them with a .bak extension, they can stumble across the backup file. Because the server is not configured to parse .bak files as php files, it will serve the unexecuted script source code to the client. This technique can yield database credentials, system credentials, as well as expose more serious vulnerabilities in the code itself. The goal of this processor is to detect when a user is attempting to find unreferenced files.

Table 16: File Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Block Response	Configurable	HTTP Response	The response to return when a request is blocked due to a matching suspicious token rule with blocking enabled.
Suspicious Tokens	Collection	Collection	The configured suspicious extensions.
Incident: Suspicious File Exposed	Boolean	True	A file which has a suspicious filename is publicly available.
Incident: Suspicious Filename	Boolean	True	A file with a filename that contains a suspicious token was requested.

Honeypot Processors: File Processor: Incident - Suspicious Filename

Complexity: Suspicious (1.0)

Default Response: 10x = Suspicious Resource Enumeration Incident.

Cause: WebApp Secure has a list of file tokens which represent potentially sensitive files. For example, developers will often rename source files with a ".bck" extension during debugging, and sometimes they forget to delete the backup after they are done. Hackers

often look for these left over source files. WebApp Secure is configured to look for any request to a file with a ".bck" extension (as well as any other configured extensions), and trigger this incident if the file does not exist. An incident will not be triggered if the file does in fact exist, and the extension is not configured to block the response. This is to avoid legitimate files being flagged as suspicious filenames.

Behavior: There are specific files that many websites host, that contain valuable information for a hacker. These files generally include data such as passwords, SQL schema's, source code, and so on. When hackers try to breach a site, they will often check to see if they can locate some of these special files in order to make their jobs easier. For example, if a hacker sees that the home page is called "index.php", they can try and request "index.php.bak", because if it exists, it will be returned as raw source code. This is usually an effort to exploit a "Predictable Resource Location" vulnerability. Because this incident is only created if the file being requested does not actually exist, it does not represent a successful exploit.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium Web Site](#) and search for the attack name to learn more about it.

Honeypot Processors: File Processor: Incident - Suspicious File Exposed

Complexity: Suspicious (1.0)

Default Response: 10x = Suspicious Resource Enumeration Incident.

Cause: WebApp Secure has a list of file tokens which represent potentially sensitive files. For example, developers will often rename source files with a ".bck" extension during debugging, and sometimes they forget to delete the backup after they are done. Hackers often look for these left over source files. WebApp Secure is configured to look for any request to a file with a ".bck" extension (as well as any other configured extensions), and trigger this incident if the extension is configured as illegal. This incident will only be triggered if the file actually exists, and the request reaches the backend server. For example, the user might request "database.sql". If the .sql extension is configured to block, and the file actually exists on the server, this incident will be generated. If "database.sql" does not exist, then only a "Suspicious Filename" incident will be created.

Behavior: There are specific files that many websites host, that contain valuable information for a hacker. These files generally include data such as passwords, SQL schema's, source code, and so on. When hackers try to breach a site, they will often check to see if they can locate some of these special files in order to make their jobs easier. For example, if a hacker sees that the home page is called "index.php", they can try and request "index.php.bak", because if it exists, it will be returned as raw source code. This is usually an effort to exploit a "Predictable Resource Location" vulnerability. This incident is only triggered when the user requested a file that would otherwise have been successfully returned, if it were not blocked by WebApp Secure. For example, the user might request "database.sql" and actually get a 200 response from the server indicating that the file exists and is accessible to everyone. However if the system is configured to

mark the ".sql" extension as illegal, then WebApp Secure will block the request. This prevents the sensitive file from potentially being exposed to an actual malicious user. If this incident occurs, the server administrator should immediately remove the sensitive file or change its permissions so it is no longer publicly accessible.

HoneyPot Processors: File Processor: Incident - Suspicious Resource Enumeration

Complexity: Low (2.0)

Default Response: 1x = 5 day Block.

Cause: WebApp Secure has a list of file tokens which represent potentially sensitive files. For example, developers will often rename source files with a ".bak" extension during debugging, and sometimes they forget to delete the backup after they are done. Hackers often look for these left over source files. WebApp Secure is configured to look for any request to a file with a ".bak" extension (as well as any other configured extensions), and trigger a Suspicious Filename incident if the file does not exist. Should the suspicious filename incident be triggered several times, this incident will then be triggered.

Behavior: There are specific files that many websites host, that contain valuable information for a hacker. These files generally include data such as passwords, SQL schema's, source code, and so on... When hackers try to breach a site, they will often check to see if they can locate some of these special files in order to make their jobs easier. For example, if a hacker sees that the home page is called "index.php", they can try and request "index.php.bak", because if it exists, it will be returned as raw source code. This is usually an effort to exploit a "Predictable Resource Location" vulnerability. The first few times a user requests a filename containing a suspicious token, they will only get "Suspicious Filename" incidents. However if they request a large volume of filenames with suspicious tokens, then the "Suspicious Resource Enumeration" incident is generated. This incident represents a user who is actively scanning the site with very aggressive tactics to find unlinked and sensitive data.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

HoneyPot Processors: Hidden Input Form Processor

Many webmasters create forms which post to a common form handling service; using hidden fields to indicate how the service should handle the data. A common hacking technique is to look for these hidden parameters and see if there is any way to change the behavior of the service by manipulating its input parameters. This processor is responsible for injecting a fake hidden input into forms in HTML responses and ensuring that when those values are posted back to the server, they have not been modified.

Table 17: Hidden Input Form Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Hidden Input Parameter	Collection	Collection	The possible hidden inputs on a page.
Inject Input Enabled	Boolean	True	Whether to inject hidden inputs into HTML forms.
Maximum Injections	Integer	3	The maximum number of fake hidden parameters that will be added to any given URL.
Strip Fake Input	Boolean	True	Whether to remove the fake input value from the posted form results before proxying the request to the backend servers. This should only be turned off if there is some additional security implemented on the form, where its contents are signed on the client and validated on the server.
Incident: Hidden Parameter Manipulation	Boolean	True	The user submitted the form and the value of the injected parameter is not what was expected.
Incident: Hidden Input Type Manipulation	Boolean	True	The user submitted the form and the value of the injected parameter is not what was expected. It was also modified to post a file.

Honeypot Processors: Hidden Input Form Processor: Incident - Hidden Parameter Manipulation

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection 2-6 seconds. 2x = Logout User. 3x = Clear Inputs.

Cause: WebApp Secure inspects outgoing traffic for HTML forms with a "POST" method type. Forms that post to a local URL (within the same domain), will be modified to include a fake hidden input with a defined value. The input is intended to look as though it was always part of the form, and is often selected to appear vulnerable (such as naming the input 'debug' or 'loglevel' and giving it a numerical or Boolean value). The "Hidden Parameter Manipulation" incident is triggered whenever the fake hidden input is modified from its originally assigned value.

Behavior: Modifying the inputs of a page is the foundation of a large variety of attack vectors. Basically, if you want to get the backend server to do something different, you need to supply different input values (either by cookie, query string, URL, or form parameters). Depending on what value the user chose for the input, the attack could fall under large number of vectors, including "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", and

"SQL injection" among many others. A common practice is to first spider the website, then test every single input on the site for a specific set of vulnerabilities. For example, the user might first index the site, then visit each page on the site, then test every exposed input (cookie, query string, and form inputs) with a list of SQL injection tests. These tests are designed to break the resulting page if the input is vulnerable. As such, the entire process (which can involve thousands of requests) can be automated and return a clean report on which inputs should be targeted. Because WebApp Secure injects several fake inputs, a spider that tests all inputs will eventually test the fake input as well. This means that if there is a large volume of this incident, it is likely due to such an automated process. It should be assumed that the values tested against the fake input, have also been tested against the rest of the inputs on the site.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Honeypot Processors: Hidden Input Form Processor: Incident - Parameter Type Manipulation

Complexity: High (4.0)

Default Response: 1x = Permanent Clear Inputs.

Cause: WebApp Secure inspects outgoing traffic for HTML forms with a "POST" method type. Forms that post to a local URL (within the same domain), will be modified to include a fake hidden input with a defined value. The input is intended to look as though it was always part of the form, and is often selected to appear vulnerable (such as naming the input 'debug' or 'loglevel' and giving it a numerical or Boolean value). The input will however, always be assigned a value that can be represented as a string of characters (in other words, not binary data). The "Parameter Type Manipulation" incident is triggered whenever the fake hidden input is modified from its originally assigned value in order to submit a multipart file.

Behavior: Modifying the inputs of a page is the foundation of a large variety of attack vectors. Basically, if you want to get the backend server to do something different, you need to supply different input values (either by cookie, query string, URL, or form parameters). Depending on what value the user chose for the input, the attack could fall under large number of vectors, including "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", and "SQL injection" among many others. Unlike a normal "Hidden Parameter Manipulation" incident, this version is triggered when the user changes the encoding of the form and submits the hidden input as a file post. This is likely in an attempt to either achieve a "Buffer Overflow", or to exploit a filter evasion weakness, that might have otherwise blocked the value being submitted. A common practice is to first spider the website, then test every single input on the site for a specific set of vulnerabilities. For example, the user might first index the site, then visit each page on the site, then test every exposed input (cookie, query string, and form inputs) with a list of SQL injection tests. These tests are designed to break the resulting page if the input is vulnerable. As such, the entire

process (which can involve thousands of requests) can be automated and return a clean report on which inputs should be targeted. Because WebApp Secure injects several fake inputs, a spider that tests all inputs will eventually test the fake input as well. This means that if there is a large volume of this incident, it is likely due to such an automated process. It should be assumed that the values tested against the fake input, have also been tested against the rest of the inputs on the site.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium Web Site](#) and search for the attack name to learn more about it.

Honey_pot Processors: Hidden Link Processor

When trying to exploit a site, hackers will often scan the contents of the site in search of directories and files that are of interest. Because this activity is done at the source level, the hacker finds every file referenced, whereas when a user views a website, they can only see the links that are visible according to the HTML. This processor injects a fake link into documents that references a file that looks interesting. The link is injected in such a way that prevents it from being rendered when the browser loads the page. This means that no normal user would ever find/click on the link, but that a scanner or hacker who is looking at the source code likely will.

Table 18: Hidden Link Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Hidden Links	Configurable	Hidden Links	The set of hidden links that can be injected into the site.
Inject Link Enabled	Boolean	True	Whether to inject the link into HTTP responses.
Incident: Link Directory Indexing	Boolean	True	The user requested a directory index on one of the fake parent directories of the linked file.
Incident: Link Directory Spidering	Boolean	True	The user requested a resource inside the fake directory of the linked file.
Incident: Malicious Resource Request	Boolean	True	The user requested the fake linked resource.

HoneyPot Processors: Hidden Link Processor: Incident - Link Directory Indexing

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds and 1 day Block.

Cause: WebApp Secure injects a hidden link into pages on the protected web application. This link is not exposed visually to users of the website. In order to find the link, a user would need to manually inspect the source code of the page. If a user finds the hidden link code in the HTML, and attempts to get a directory file listing from the directory the link points to, this incident will be triggered.

Behavior: A common technique for hackers when scoping the attack surface of a website is to spider the site and collect the locations of all of its pages. This is generally done using a simple script that looks for URL's in the returned HTML of the home page, then requests those pages and checks for URL's in their source, and so forth. Legitimate search engine spiders will do this as well. But the difference between a legitimate spider and a malicious user, is how aggressively they will use the newly discovered URL to derive other URLs. This incident triggers when the user goes beyond just checking the linked URL, but instead also attempts to get a file listing from the directory the URL points to. A legitimate spider would not do this, because it is considered fairly invasive. This activity is generally looking for a "Directory Indexing" weakness on the server, in an effort to locate unlinked and possibly sensitive resources.

HoneyPot Processors: Hidden Link Processor: Incident - Link Directory Spidering

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds and 5 day Block in 6 minutes.

Cause: WebApp Secure injects a hidden link into pages on the protected web application. This link is not exposed visually to users of the website. In order to find the link, a user would need to manually inspect the source code of the page. If a user finds the hidden link code in the HTML, and attempts to request some other arbitrary file in the same fake directory as the link, this incident will be triggered.

Behavior: A common technique for hackers when scoping the attack surface of a website is to spider the site and collect the locations of all of its pages. This is generally done using a simple script that looks for URL's in the returned HTML of the home page, then requests those pages and checks for URL's in their source, and so forth. Legitimate search engine spiders will do this as well. But the difference between a legitimate spider and a malicious user, is how aggressively they will use the newly discovered URL to derive other URLs. This incident triggers when the user goes beyond just checking the linked URL, but instead also attempts to request one or more arbitrary files inside the same directory as the file referenced by the hidden link. A legitimate spider would not do this, because it is considered fairly invasive. This activity is generally looking for a "Directory Indexing" weakness on the server, or a "Predictable Resource Location" vulnerability, in an effort to locate unlinked and possibly sensitive resources.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Honeypot Processors: Hidden Link Processor: Incident - Malicious Resource Request

Complexity: Suspicious (1.0)

Default Response: 1x = Slow Connection 2-6 seconds and 5 day Block in 6 minutes.

Cause: WebApp Secure injects a hidden link into pages on the protected web application, which is only discoverable through manual source code inspection. If a user discovers the hidden link, and attempts to request the file it references, this incident will be triggered.

Behavior: When scoping the attack surface of a website, hackers commonly spider the site and collect the locations of all pages. Spidering can be performed with the assistance of simple scripts that look for URLs in the returned HTML of the home page, then request those pages and check for URLs in their source, and so forth. Legitimate search engine spiders will do this as well — but the difference between legitimate spiders and malicious users lies in how aggressively they will use the newly discovered URL to derive other URLs. This incident triggers when the user simply requests the hidden link URL. Because this can also be triggered by a legitimate search engine spider, this type of incident is not considered malicious on its own.

Honeypot Processors: Query String Processor

Hackers tend to manipulate the values of query string parameters in order to get the application to behave differently. The goal of this processor is to add fake query string parameters to some of the links and forms in the page, and verify that they do not get modified when accessed by the user.

Table 19: Query String Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Fake Parameters	Collection	Collection	The collection of fake parameters to add to the links which already have parameters.
Inject Parameter Enabled	Boolean	True	Whether to inject query string parameters on URLs in HTTP responses.

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Fake Parameters	Collection	Collection	The collection of fake parameters to add to the links which already have parameters.
Inject Parameter Enabled	Boolean	True	Whether to inject query string parameters on URLs in HTTP responses.

Table 19: Query String Processor Configuration Parameters Parameter Type Default Value Description (*continued*)

Parameter	Type	Default Value	Description
Maximum Injections	Integer	3	Whether to inject query string parameters on URLs in HTTP responses.
Randomization Token	String	[Not Set]	Some websites use complex redirection rules or modify query string parameters of static links using javascript on the client. In these situations, the randomization of fake query parameter values can be problematic. To resolve the issue, you can either update the list of fake parameters so that it does not include randomized tokens, or you can define a randomization token name here. If you define a randomization token, then the data used to randomize which value is selected will be transferred as an additional query string parameter by this name. It is recommended that you leave this field empty unless you experience a lot of fake positives on query parameter manipulation incidents shortly after setting up Webapp Secure to protect a website.
Strip Fake Input	Boolean	True	Whether to remove the fake input value from the query string before proxying the request to the backend servers. This should only be turned off if there is some additional security implemented on the site, where links are signed on the client and validated on the server.
Incident: Query Parameter Manipulation	Boolean	True	The user manually modified the value of a query string parameter.

HoneyPot Processors: Query String Processor: Incident - Query Parameter Manipulation

Complexity: Low (2.0)

Default Response: 3x = Slow Connection 2-6 seconds. 5x = 1 day Clear Inputs.

Cause: WebApp Secure injects a fake query parameter into some of the links of the protected website. This query parameter has a known value, and should never change, because it is not part of the actual web application. If a user modifies the query parameter value, this incident will be triggered.

Behavior: Query parameters represent the most visible form of user input a web application exposes. They are clearly visible in the address bar, and can be easily changed by even an inexperienced user. However most users do not attempt to change values directly in the query string, unless they are trying to perform some action the website does not normally expose through its interface, or does not make sufficiently easy. Because it is so easy for a normal user to accidentally change a query parameter, this incident alone is not considered strictly malicious. However depending on the value that is submitted, this could be part of a number of different exploit attempts, including "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", and "SQL injection".



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Honeypot Processors: Robots Processor

The Robots.txt proxy processor is responsible for catching malicious spiders that do not behave in accordance with established standards for spidering. Hackers often utilize the extra information sites expose to spiders, and then use that information to access resources normally not linked from the public site. Because this activity is effectively breaking established standards for spidering, this processor will also identify hackers who are using the information maliciously.

Table 20: Robots Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled Boolean	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Fake Disallowed Directories	String	Random	The path to a fake directory to add to the disallow rules in the robots.txt file. This path should be completely fake and not overlap with actual directories.
Incident: Malicious Spider Activity	Boolean	True	The user requested a resource which was restricted in the spider rules file, indicating this user is not a good spider, but is spidering the site anyway.

Honeypot Processors: Robot Processor: Incident - Malicious Spider Activity

Complexity: Low (2.0)

Default Response: 1x = Captcha and Slow Connection 2-6 seconds. 6x = 1 day Block.

Cause: One of the standard resources that just about every website should expose is called robots.txt. This resource is used by search engines to instruct them on how to spider the website. Two of the more important directives are "allow" and "disallow". These directives are used to identify which directories a spider should index, and which directories it should stay away from. Good practice for any website is to lock down any resource that should not be exposed. However some web masters simply add a "disallow" statement so that those resources do not get indexed and therefore are never found by users. This technique does not work, because attackers will often access robots.txt and intentionally traverse the "disallow" directories in search of vulnerabilities. So in effect, the listing of such directories is basically pointing hackers in the direction of the most sensitive resources on the site. WebApp Secure will intercept requests for robots.txt and

either generate a completely fake robots.txt file (if one does not exist), or modify the existing version by injecting a fake directory as a disallow directive. The "Malicious Spider Activity" incident is triggered whenever a user attempts to request a resource in the fake disallow directory, or attempts to perform a directory index on the disallow directory.

Behavior: Requesting robots.txt occurs in two different scenarios. The first is where a legitimate spider, such as Google, attempts to index the website. In this case, the robots.txt file will be requested, and no requests from that client will be issued to the disallow directories. In the second scenario, a malicious user requests robots.txt and then indexes some or all of the disallow directories. In this specific case, the user has requested robots.txt to obtain the list of disallow directories, and then started searching for resources in those directories. This activity is performed to find a "Predictable Resource Location" vulnerability. Because spidering a directory tends to be a noisy process (lots of requests), there are likely to be many of these incidents if there are any. The sum of occurrences of this incident represent the type of activity the user is performing to index a directory. The set of URL's for which this incident is triggered, represent the filenames the malicious user is testing for. For example, if they were searching for PDF files that contain stock information, there would be an incident for each filename with a PDF extension they tried to request. There is a very strong chance that if the filename was requested in the disallow directory, it was probably requested in every other directory on the site as well. This type of behavior is generally observed while the client is attempting to establish the overall attack surface of the website (or in the case of a legitimate spider, they are attempting to establish the desired index limitations).

Activity Processors

The custom authentication processor is designed to add strong and secure authentication to any page in the protected application. The authentication processor also logs malicious activity like invalid logins and modifying cookies or query parameters.

Table 21: Custom Authentication Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
User Accounts	Collection	[collection:0]	The list of valid user accounts to use for this processor.
Advanced			
Auth Cookie Name	String	Random	The name of the authentication cookie.
Login Page Timeout	Integer	10 Minutes	The number of seconds a login page can be used before it times out. This is intended to prevent attacks based on watching network traffic. It should be as short as is tolerable.
MD5 Script Name	String	Random	The name of the Javascript resource that contains the MD5 code.

Table 21: Custom Authentication Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Session Timeout	Integer	1 Hour	The number of seconds a session can be idle before it times out.
Incident: Auth Cookie Tampering	Boolean	True	The user has modified the cookie used to manage custom authentication, probably in an attempt to expose sensitive information or bypass access restrictions.
Incident: Auth Input Parameter Tampering	Boolean	True	The user has modified the parameters used to manage custom authentication, probably in an attempt to expose sensitive information or bypass the authentication mechanism.
Incident: Auth Invalid Login	Boolean	True	The user has attempted to login but supplied invalid credentials, this could be perfectly normal, but large numbers of this type of incident would indicate a brute force attack.
Incident: Auth Query Parameter Tampering	Boolean	True	The user has modified the query parameters that were submitted when the user was asked to originally login. This is likely in an attempt to probe the authentication mechanism for exploits.

Activity Processors: Custom Authentication Processor: Incident - Auth Input Parameter Tampering

Complexity: Medium (3.0)

Default Response: 3x = Warn User, 5x = Captcha. 9x = 1 day Clear Inputs.

Cause: WebApp Secure provides the capability of password protecting any URL on the protected site. This means that if a user attempts to access that URL, they will be prompted to enter a username and password before the original request is allowed to be completed. This incident is triggered when a user attempts to manipulate the hidden form parameters used to handle authentication.

Behavior: Manipulating hidden input fields in a form, for whatever reason is generally considered malicious. In this case, because the form is being used to password protect a resource, it is likely that the attacker is trying to bypass the authentication by finding a vulnerability in the authentication mechanism. Depending on the modified value they submit, they could be attempting to launch a "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", or "SQL injection" attack among many others.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Activity Processors: Custom Authentication Processor: Incident - Auth Query Parameter Tampering

Complexity: Low (2.0)

Default Response: 1x = Warn User. 2x = 1 day Clear Inputs.

Cause: WebApp Secure provides the capability of password protecting any URL on the protected site. This means that if a user attempts to access that URL, they will be prompted to enter a username and password before the original request is allowed to be completed. This incident is triggered when a user attempts to manipulate the query parameters that were submitted with the original unauthenticated request, after authentication has been completed.

Behavior: Manipulating query parameters after authenticating is not very easy to do without a third party tool, and has no legitimate purpose. As such, this type of behavior is most likely related to a user who is trying to smuggle a malicious payload through a network or web firewall. Depending on the value the user submits for the modified query string, they could be attempting a "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", or "SQL injection" attack among many others. One interesting note is that the user has actually authenticated in order to cause this incident. As such, it is also likely that the account for which the user authenticated has been compromised and should be updated (with a new password). Although it is possible that the true owner of the account has executed the malicious action, and should therefore potentially be banned.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Activity Processors: Custom Authentication Processor: Incident - Auth Cookie Tampering

Complexity: Medium (3.0)

Default Response: 1x = Warn User, 2x = Captcha. 3x = 1 day Strip Inputs.

Cause: WebApp Secure provides the capability of password protecting any URL on the protected site. This means that if a user attempts to access that URL, they will be prompted to enter a username and password before the original request is allowed to be completed. This incident is triggered when a user attempts to manipulate the cookie used to maintain the authenticated session once the user logs in.

Behavior: Manipulating cookies is not easy to do without a third party tool, and has no legitimate purpose. As such, this type of behavior is most likely related to a user who is trying to perform a "Credential/Session Prediction" attack, or execute an input based attack such as a "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", or "SQL injection" attack among

many others. One interesting note is that the user has actually authenticated in order to cause this incident. As such, it is also likely that the account for which the user authenticated has been compromised and should be updated (with a new password). Although it is possible that the true owner of the account has executed the malicious action, and should therefore potentially be banned.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Activity Processors: Custom Authentication Processor: Incident - Authentication Brute Force

Complexity: Medium (3.0)

Default Response: 1x = Captcha. 2x = 1 day Block.

Cause: WebApp Secure provides the capability of password protecting any URL on the protected site. This means that if a user attempts to access that URL, they will be prompted to enter a username and password before the original request is allowed to be completed. This incident is triggered when a user submits a large volume of invalid username and password combinations.

Behavior: Submitting a single invalid username or password is likely a user typo, and is not necessarily malicious. However it does represent a security event, and a large number of these events can represent a more serious threat such as "Brute Force". It is possible however, that the invalid username or password might also be an attack vector targeted at the authentication mechanism such as a "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", or "SQL injection" attack among many others. This incident is a higher level incident that gets tripped when dozens of "Auth Invalid Login" incidents are created. As such, it does not contain much information about the actual accounts being targeted. If more detail is desired, the underlying "Auth Invalid Login" incidents should be reviewed. These incidents are only suspicious (not considered malicious on their own), so the filtering option will need to be set to show non malicious incidents.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Activity Processors: Custom Authentication Processor: Incident - Auth Invalid Login

Complexity: Suspicious (1.0)

Default Response: 20x = Authentication Brute Force Incident.

Cause: WebApp Secure provides the capability of password protecting any URL on the protected site. This means that if a user attempts to access that URL, they will be prompted to enter a username and password before the original request is allowed to be completed. This incident is triggered when a user submits an invalid username or password. This incident alone is not necessarily malicious, as it is possible for a legitimate user to accidentally type their username or password incorrectly.

Behavior: Submitting a single invalid username or password is likely a user typo, and is not necessarily malicious. However it does represent a security event, and a large number of these events can represent a more serious threat such as "Brute Force". It is possible however, that the invalid username or password might also be an attack vector targeted at the authentication mechanism such as a "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", or "SQL injection" attack among many others. So if the value specified for the username and password does not look like a legitimate username and password (they are too long, or contain unusual characters), then this incident can be more serious. However, even in this case, the user is more likely to submit dozens of invalid credentials (not just one), and there is a different incident for that scenario.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Activity Processors: Cookie Protection Processor

This processor is responsible for protecting a set of application cookies from modification or assignment by the user.

Table 22: Cookie Protection Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Protected Cookies	Collection	Collection	The name of the protected cookie.
Advanced			
Protected Cookie Signature Suffix	String	Random	The suffix to add to the protected cookie names when generating a signature cookie. For example, if the protected cookie is PHPSESSID and the suffix is _MX, then the signature for PHPSESSID would be in a cookie named PHPSESSID_MX.
Incident: Application Cookie Manipulation	Boolean	True	The user either attempted to modify one of the protected cookies, or attempted to assign a new value.

Activity Processors: Cookie Protection Processor: Incident - Application Cookie Manipulation

Complexity: Low (2.0)

Default Response: 1x = Warn User and Logout User. 2x = 5 day Clear Inputs.

Cause: WebApp Secure is designed to provide additional protection to cookies used by the web application for tracking user sessions. This is done by issuing a signature cookie any time the web application issues a "protected cookie" (which cookies to protect is defined in configuration). The signature cookie ties the application cookie (such as PHPSESSID) to the WebApp Secure session cookie. If any of the 3 cookies are modified (WebApp Secure session cookie, signature cookie, or the actual application cookie), then this incident will be triggered, and the application cookie will be terminated (effectively terminating the users session). This prevents any users from manually creating a session cookie, hijacking another users cookie, or manipulating an existing cookie.

Behavior: Manipulation of cookies is generally performed in order to hijack another user's session. However because cookies represent another type of application input, modifications could also be performed to attempt other exploits. If the modified value resembles a legitimate value for the application cookie, then this is likely a session hijacking attempt. If the cookie contains other values that are clearly not valid, then it is more then likely an attack on generic application inputs such as a "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", and "SQL injection" attack among many others.



NOTE: For information on the attack types mentioned here, go to The [Web Application Security Consortium](#) Web Site and search for the attack name to learn more about it.

Activity Processors: Error Processor

Errors and their contents play a big part in hacking a website. When a hacker obtains an error message, it provides useful information, the very least of which is that the attacker found a way to do something unintended in the web application and the server executed code to handle it. As such, when a user attempts to hack a website, they frequently induce and receive error messages. Often these error messages are very unusual and are not common when a normal user visits the site. For example, the error code 400 (Bad Request) is returned when the raw data in a request does not follow the HTTP standards. While it is possible to get a 400 error by typing invalid characters into the URL, the majority of these errors are caused by third party software (usually not a browser), improperly communicating with the server. A hacker might for example, manually construct a malicious request and forget to include the "Host" header. The goal of this processor is to record unusual and unexpected errors as incidents. This processor will also monitor all 404 errors and attempt to identify Common Directory Enumeration and User Directory Enumeration.

Table 23: Error Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Legitimate Error Detection Enabled	Boolean	True	Whether to attempt to identify errors in the protected web applications so that they can be ignored.
Advanced			
Error Cache Expiration	Integer	43200 (12 hours)	The number of seconds to cache an error condition so that subsequent matching error conditions from other users can be identified. The less traffic the site sees on a regular basis, the higher this value must be. The recommended default is for sites that see several thousand users a day or more.
Error Cache Size	Integer	50	The number of error conditions to cache for each level of specificity. If too many error conditions are encountered in a short period of time, this will prevent the tracking code from consuming too much memory. Errors at the full URL with query string specificity will cache this many conditions, at the URL only level it will cache twice this many, and at the filename level, it will cache 3 times as many as this value.
Filename Only Expiration	Integer	259200 (3 days)	The number of seconds that an error must not be encountered on a filename regardless of its location before an ignored error starts being recorded again.

Table 23: Error Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Filename Only Threshold	Integer	70	The maximum number of unique users who can hit a specific filename, regardless of location, and get the same error before it stops being recorded as suspicious (zero = do not track based on filename).
URL With Query Expiration	Integer	259200 (3 days)	The number of seconds that an error must not be encountered on the full URL with query string before an ignored error starts being recorded again.
URL With Query Threshold	Integer	30	The maximum number of unique users who can hit a full URL including query string and get the same error before it stops being recorded as suspicious (zero = do not track based on full URL).
URL Without Query Expiration	Integer	259200 (3 days)	The number of seconds that an error must not be encountered on the URL excluding query string before an ignored error starts being recorded again.
URL Without Query Threshold	Integer	50	The maximum number of unique users who can hit a URL excluding query string and get the same error before it stops being recorded as suspicious (zero = do not track based on URL).
100 Continue	Configurable	HTTP Status Codes	Continue.
101 Switching Protocols	Configurable	HTTP Status Codes	Switching Protocols.
102 Processing	Configurable	HTTP Status Codes	Processing.
300 Multiple Choices	Configurable	HTTP Status Codes	Multiple Choices.
301 Moved Permanently	Configurable	HTTP Status Codes	Moved Permanently.
302 Found	Configurable	HTTP Status Codes	Found.
303 See Other	Configurable	HTTP Status Codes	See Other.

Table 23: Error Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
304 Not Modified	Configurable	HTTP Status Codes	Not Modified
305 Use Proxy	Configurable	HTTP Status Codes	Use Proxy.
306 Switch Proxy	Configurable	HTTP Status Codes	Switch Proxy.
307 Temporary Redirect	Configurable	HTTP Status Codes	Switch Proxy.
400 Bad Request	Configurable	HTTP Status Codes	Bad Request
401 Unauthorized	Configurable	HTTP Status Codes	Unauthorized.
402 Payment Required	Configurable	HTTP Status Codes	Payment Required.
403 Forbidden	Configurable	HTTP Status Codes	Forbidden
404 Not Found	Configurable	HTTP Status Codes	Not Found
405 Method Not Allowed	Configurable	HTTP Status Codes	Not allowed.
406 Not Acceptable	Configurable	HTTP Status Codes	Not acceptable.
407 Proxy Authentication Required	Configurable	HTTP Status Codes	Proxy Authentication Required
408 Request Timeout	Configurable	HTTP Status Codes	Request Timeout.
409 Conflict	Configurable	HTTP Status Codes	Conflict.
410 Gone	Configurable	HTTP Status Codes	Gone.
411 Length Required	Configurable	HTTP Status Codes	Length Required.
412 Precondition Failed	Configurable	HTTP Status Codes	Precondition Failed.
413 Request Entity Too Large	Configurable	HTTP Status Codes	Request Entity Too Large.
414 Request-URI Too Long	Configurable	HTTP Status Codes	Request-URI Too Long.
415 Unsupported Media Type	Configurable	HTTP Status Codes	Unsupported Media Type.
416 Requested Range Not Satisfiable	Configurable	HTTP Status Codes	Requested Range Not Satisfiable.
417 Expectation Failed	Configurable	HTTP Status Codes	Expectation Failed.
418 I'm a teapot	Configurable	HTTP Status Codes	418 I'm a teapot

Table 23: Error Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
422 Unprocessable Entity	Configurable	HTTP Status Codes	Unprocessable Entity.
423 Locked	Configurable	HTTP Status Codes	Locked.
424 Failed Dependency	Configurable	HTTP Status Codes	Failed Dependency.
425 Unordered Collection	Configurable	HTTP Status Codes	Unordered Collection
426 Upgrade Required	Configurable	HTTP Status Codes	Upgrade Required
449 Retry With	Configurable	HTTP Status Codes	Retry With
450 Blocked by Windows Parental Controls	Configurable	HTTP Status Codes	Blocked by Windows Parental Controls.
500 Internal Server Error	Configurable	HTTP Status Codes	Internal Server Error
501 Not Implemented	Configurable	HTTP Status Codes	Not Implemented
502 Bad Gateway	Configurable	HTTP Status Codes	Bad Gateway
503 Service Unavailable	Configurable	HTTP Status Codes	Service Unavailable
504 Gateway Timeout	Configurable	HTTP Status Codes	Gateway Timeout
505 HTTP Version Not Supported	Configurable	HTTP Status Codes	HTTP Version Not Supported
506 Variant Also Negotiates	Configurable	HTTP Status Codes	Variant Also Negotiates
507 Insufficient Storage	Configurable	HTTP Status Codes	Insufficient Storage
509 Bandwidth Limit Exceeded	Configurable	HTTP Status Codes	Bandwidth Limit Exceeded
510 Not Extended	Configurable	HTTP Status Codes	Not Extended
Incident: Illegal Response Status	Boolean	True	The user issued a request that resulted in an error status code that is considered suspicious and possibly malicious.

Table 23: Error Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Incident: Suspicious Response Status	Boolean	True	The user issued a request that resulted in a known error status code generally involved in malicious behavior. On its own this is not enough to classify abuse, but patterns of this indicator can lead to higher level malicious incidents.
Incident: Unexpected Response Status	Boolean	True	The user issued a request that resulted in an unknown error status code and could represent a successful exploit.
Incident: Unknown Common Directory Requested	Boolean	True	The user has requested a directory that does not exist. The directory is in a list of common directory names, so it is likely that this request is in an attempt to find a directory that is not linked from the site.
Incident: Unknown User Directory Requested	Boolean	True	The user has requested a directory for a specific system user that does not exist. The username is in a list of common usernames, so it is likely that this request is in an attempt to identify a user account that is not linked from the site.

Activity Processors: Error Processor: Incident - Illegal Response Status

Complexity: Suspicious (1.0)

Default Response: None.

Cause: WebApp Secure monitors the various status codes returned by the protected website and compares them to a configurable list of known and acceptable status codes. Some status codes are expected during normal usage of the site (such as 200 - OK, or 403 - Not Modified), but some status codes are much less common for a normal user (such as 500 - Server Error, or 404 - File Not Found). When a user issues a request that results in a status code that is marked as Suspicious or Illegal in this parameter, the corresponding incident is triggered. If the code is not in this collection, the Unknown incident is triggered.

Behavior: In the process of attempting to find vulnerabilities on a webserver, hackers will often encounter errors. Just a single error or two is likely not a problem, because even

legitimate users accidentally type a URL incorrectly on occasion. However when excessive numbers of unexpected status codes are returned, the behavior of the user can be narrowed down and classified as malicious. The actual vulnerability an attacker is looking for, can be identified through the status codes they are being returned. For example, if the user is getting a lot of 404 errors, they are likely searching for unlinked files ("Predictable Resource Location"). If the user is getting a lot of 500 errors, they can be trying to establish a successful "SQL Injection" or "XSS150" vulnerability.

Activity Processors: Error Processor: Incident - Suspicious Response Status

Complexity: Suspicious (1.0)

Default Response: 10x 404 = Resource Enumeration Incident.

Cause: WebApp Secure monitors the various status codes returned by the protected website and compares them to a configurable list of known and acceptable status codes. Some status codes are expected during normal usage of the site (such as 200 - OK, or 403 - Not Modified), but some status codes are much less common for a normal user (such as 500 - Server Error, or 404 - File Not Found). When a user issues a request that results in a status code that is marked as Suspicious or Illegal in this parameter, the corresponding incident is triggered. If the code is not in this collection, the Unknown incident is triggered.

Behavior: In the process of attempting to find vulnerabilities on a webserver, hackers will often encounter errors. Just a single error or two is likely not a problem, because even legitimate users accidentally type a URL incorrectly on occasion. However when excessive numbers of unexpected status codes are returned, the behavior of the user can be narrowed down and classified as malicious. The actual vulnerability the attacker is looking for can be identified through the status codes they are being returned. For example, if the user is getting a lot of 404 errors, they are likely searching for unlinked files ("Predictable Resource Location"). If the user is getting a lot of 500 errors, they can be trying to establish a successful "SQL Injection" or "XSS" vulnerability. In the case of this incident, the user is getting an unexpected status code. This is likely because of a bug in the web application which the user has found and is attempting to exploit. The URL this incident is created for, should be reviewed to determine why it would be responding with a non standard status code. If the status code is intentionally non-standard, but is acceptable behavior, then the custom status code should be added to the list of known and accepted status codes in config.

Activity Processors: Error Processor: Incident - Unexpected Response Status

Complexity: Suspicious (1.0)

Default Response: None.

Cause: WebApp Secure monitors the various status codes returned by the protected website and compares them to a configurable list of known and acceptable status codes. Some status codes are expected during normal usage of the site (such as "200 OK" or "304 Not Modified"), but some status codes are much less common for a normal user (such as "500 Internal Server Error" or "404 Not Found"). When a user issues a request

which results in a status code that is not known and does not have any associated configuration, this incident will be triggered.

Behavior: In the process of attempting to find vulnerabilities on a webserver, hackers will often encounter errors. Just a single error or two is likely not a problem, because even legitimate users accidentally type a URL incorrectly on occasion. However when excessive numbers of unexpected status codes are returned, the behavior of the user can be narrowed down and classified as malicious. The actual vulnerability the attacker is looking for can be identified through the status codes they are being returned. For example, if the user is getting a lot of 404 errors, they are likely searching for unlinked files ("Predictable Resource Location"). If the user is getting a lot of 500 errors, they can be trying to establish a successful "SQL Injection" or "XSS" vulnerability. In the case of this incident, the user is getting an unexpected status code. This is likely because of a bug in the web application which the user has found and is attempting to exploit. The URL this incident is created for, should be reviewed to determine why it would be responding with a non standard status code. If the status code is intentionally non-standard, but is acceptable behavior, then the custom status code should be added to the list of known and accepted status codes in config.

Activity Processors: Error Processor: Incident - Unknown Common Directory Requested

Complexity: Suspicious (1.0)

Default Response: 5x = Common Directory Enumeration Incident

Cause: This incident is triggered when a user requests a directory on the server that does not exist, and that directory name is in a list of commonly used directory names (for example: <http://www.example.com/public/> where "public" is not a real directory).

Behavior: Often times, administrators will upload sensitive content onto a webserver in an obscure location and not link to that content anywhere on the site. The assumption is that the content is private because no one will find it. However humans are somewhat predictable, so it's actually quite common for two administrators to pick the same "obscure" location to place sensitive content. As such, hackers have compiled a list of the most commonly chosen directory names where sensitive content is often stored, and they will basically test every name in the list to see if a site has a directory by that name. If it does, the attacker is able to locate and obtain that sensitive content. An example of a tool that allows attackers to quickly identify hidden directories is called "DirBuster" (https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project).

Activity Processors: Error Processor: Incident - Unknown User Directory Requested

Complexity: Suspicious (1.0)

Default Response: 5x = User Directory Enumeration Incident

Cause: Many webservers allow the users on the system to maintain publicly accessible web directories. These directories are generally accessible from the root directory of the website followed by a tilde and the username. For example, if the webserver had a user named 'george', that user could serve content from <http://www.example.com/~george/>. This incident is triggered when an attacker requests a user directory on the server that

does not exist, and that user directory name is in a list of commonly used usernames (for example: <http://www.example.com/~root/> where "root" is not a real user directory).

Behavior: Often times, administrators will upload sensitive content onto a webserver in an obscure location and not link to that content anywhere on the site. The assumption is that the content is private because no one will find it. However humans are somewhat predictable, so it's actually quite common for two administrators to pick the same "obscure" location to place sensitive content. As such, hackers have compiled a list of the most commonly chosen directory names where sensitive content is often stored, and they will basically test every name in the list to see if a site has a directory by that name. If it does, the attacker is able to locate and obtain that sensitive content. In this specific case, the attacker is testing for default user directories for users with predictable names (such as 'root', 'guest', 'nobody', and so on...). An example of a tool that allows attackers to quickly identify hidden user directories is called "DirBuster" (https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project).

Activity Processors: Error Processor: Incident - Common Directory Enumeration

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection 2-6 seconds & Captcha, 2x = Slow Connection 2-6 seconds & 1 day Block

Cause: This incident is triggered when a user requests a directory on the server that does not exist, and that directory name is in a list of commonly used directory names (for example: <http://www.example.com/public/> where "public" is not a real directory). Specifically, this incident is triggered when the user requests many different commonly named directories, as would be the case if they were testing for a large list of possible directory names.

Behavior: Often times, administrators will upload sensitive content onto a webserver in an obscure location and not link to that content anywhere on the site. The assumption is that the content is private because no one will find it. However humans are somewhat predictable, so it's actually quite common for two administrators to pick the same "obscure" location to place sensitive content. As such, hackers have compiled a list of the most commonly chosen directory names where sensitive content is often stored, and they will basically test every name in the list to see if a site has a directory by that name. If it does, the attacker is able to locate and obtain that sensitive content. An example of a tool that allows attackers to quickly identify hidden directories is called "DirBuster" (https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project).

Activity Processors: Error Processor: Incident - User Directory Enumeration

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection 2-6 seconds & Captcha, 2x = Slow Connection 2-6 seconds & 1 day Block

Cause: Many webservers allow the users on the system to maintain publically accessible web directories. These directories are generally accessible from the root directory of the website followed by a tilde and the username. For example, if the webserver had a user

named 'george', that user could serve content from <http://www.example.com/~george/>. This incident is triggered when an attacker requests a user directory on the server that does not exist, and that user directory name is in a list of commonly used usernames (for example: <http://www.example.com/~root/> where "root" is not a real user directory). Specifically, this incident is triggered when an attacker requests many different username directories, as would be the case if they were testing for a large list of possible usernames.

Behavior: Often times, administrators will upload sensitive content onto a webserver in an obscure location and not link to that content anywhere on the site. The assumption is that the content is private because no one will find it. However humans are somewhat predictable, so it's actually quite common for two administrators to pick the same "obscure" location to place sensitive content. As such, hackers have compiled a list of the most commonly chosen directory names where sensitive content is often stored, and they will basically test every name in the list to see if a site has a directory by that name. If it does, the attacker is able to locate and obtain that sensitive content. In this specific case, the attacker is testing for default user directories for users with predictable names (such as 'root', 'guest', 'nobody', and so on). An example of a tool that allows attackers to quickly identify hidden user directories is called "DirBuster" ([https://www.owasp.org/index.php/ Category:OWASP_DirBuster_Project](https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project)).

Activity Processors: Error Processor: Incident - Resource Enumeration

Complexity: Low (2.0)

Default Response: 1x = 5 day Block.

Cause: WebApp Secure has a list of file tokens which represent potentially sensitive files. For example, developers will often rename source files with a ".bck" extension during debugging, and sometimes they forget to delete the backup after they are done. Hackers often look for these left over source files. WebApp Secure is configured to look for any request to a file with a ".bck" extension (as well as any other configured extensions), and trigger a Suspicious Filename incident if the file does not exist. Should the suspicious filename incident be triggered several times, this incident will then be triggered.

Behavior: There are specific files that many websites host, that contain valuable information for a hacker. These files generally include data such as passwords, SQL schema's, source code, and so on. When hackers try to breach a site, they will often check to see if they can locate some of these special files in order to make their jobs easier. For example, if a hacker sees that the home page is called "index.php", they can try and request "index.php.bak", because if it exists, it will be returned as raw source code. This is usually an effort to exploit a "Predictable Resource Location" vulnerability. Automated scanners will generally test all of these types of extensions (.bck, .bak, .zip, .tar, .gz, and so on...) against every legitimate file that is located through simple spidering. The first few times a user requests a filename containing a suspicious token, they will only get "Suspicious Filename" incidents. However if they request a large volume of filenames with suspicious tokens, then the "Suspicious Resource Enumeration" incident is generated. This incident represents a user who is actively scanning the site with very aggressive tactics to find unlinked and sensitive data.

Activity Processors: Header Processor

A useful technique when attacking a site is to determine what software the site is using. This is known as fingerprinting the server. There are many methods used, but the basic idea is to look for signatures that identify various products. For example, it might be a known signature that Apache always lists the **Date** response header before the **Last-Modified** response header. If very few other servers follow this same pattern, then checking to see which header comes first could be used as a means of identifying if Apache is being used or not. Other key methods include looking for **Server** or **X-Powered-By** headers that actually specify the software being used. The goal of this processor is to eliminate headers as a means of fingerprinting a server.

You can allow local machine names (non-FQDN's) in the host header by setting the parameter `engine.incidents.url_fuzzing.allow_locals` to true using Expert Mode or the CLI. By default, any HTTP 1.1 requests without a host header will be considered a URL manipulation because of how nginx handles lack of a host header. The reason for this difference is because the host header is required in HTTP 1.1 requests, but not required in HTTP 1.0 requests. When the nginx proxy sends the request to security engine, it realizes that the HTTP 1.1 request is invalid and adds host: localhost to the request. The URL fuzzing logic considers this malicious, as a host of 'localhost' is suspicious.



NOTE: While the goal of this processor is mainly to prevent fingerprinting, it can also catch some malicious behavior and erroneous behavior in the protected applications (potentially as a result of an exploit). As such, the following incidents are recognized by the processor.

Table 24: Header Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Header Mixing Enabled	Boolean	False	Whether this processor should shuffle the order of response headers to avoid exposing identifiable information.
Request Header Stripping Enabled	Boolean	False	Whether this processor should strip unnecessary headers in request packets to avoid sending malicious data to the server.
Response Header Stripping Enabled	Boolean	False	Whether this processor should strip unnecessary response headers to avoid giving away identifiable information.

Table 24: Header Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Maximum Header Length	Integer	8192	The maximum allowed length of a header in bytes. If header stripping is enabled, then any headers that exceed this length will be removed from the request before proxying.
Known Request Headers	Collection	Collection	A list of known request headers.
Known Response Headers	Collection	Collection	A list of known Response headers.
Incident: Duplicate Request Header	Boolean	False	The application returned multiple instances of the same header, which it is never expected to do.
Incident: Duplicate Response Header	Boolean	False	The user provided multiple instances of the same header, and the header does not usually allow multiples.
Incident: Illegal Request Header	Boolean	False	The user provided a request header which is known to be involved in malicious activity.
Incident: Illegal Response Header	Boolean	False	The application returned a response header which it is never supposed to return.
Incident: Missing All Headers	Boolean	True	The user issued a request which has no headers at all. This incident only triggers on HTTP 1.1 requests (not on HTTP 1.0 requests).
Incident: Missing Host Header	Boolean	True	The application returned a response which is missing a required header. This incident only triggers on HTTP 1.1 requests (not on HTTP 1.0 requests).
Incident: Missing Request Header	Boolean	False	The user issued a request which is missing a required header.
Incident: Missing Response Header	Boolean	False	The application returned a response which is missing a required header.
Incident: Missing User Agent Header	Boolean	False	The user issued a request which is missing a required header.
Incident: Request Header Overflow	Boolean	True	The user issued a request which contained a header that was longer than the allowed maximum.
Incident: Unexpected Request Header	Boolean	False	The user issued a request which contains an unexpected and unknown header.

Activity Processors: Header Processor: Incident - Duplicate Request Header

Complexity: Informational (0.0)

Default Response: None

Cause: WebApp Secure monitors all of the request headers sent from the client to the web application. According to the HTTP RFC, no client should ever provide more than one copy of a specific header. For example, clients should not send multiple Host headers. However there are a few exceptions, such as the Cookie header, which can be configured to allow multiples. If the user sends multiple headers that are not configured explicitly to allow duplicates, then this incident will be triggered.

Behavior: Sending duplicate headers of the same type can be caused by several different things. It is either an attempt to profile the webserver and see how it reacts, an attempt to smuggle malicious data into the headers (because a firewall might not look at subsequent copies of the same header), or possibly just be a poorly programmed web client. In either case, it represents unusual activity that sets the user aside from everyone else. It signifies that the user is suspicious and is doing something average users do not do.

Activity Processors: Header Processor: Incident - Duplicate Response Header

Complexity: Informational (0.0)

Default Response: None

Cause: Secure monitors all of the response headers sent from the server to the client. According to the HTTP RFC, no server should ever provide more than one copy of a specific header. For example, servers should not send multiple "Content-Length" headers. However there are a few exceptions, such as the "Set-Cookie" header, which can be configured to allow multiples. If the server attempts to return multiple headers of the same type, which are not configured explicitly to allow duplicates, then this incident will be triggered.

Behavior: The RFC does not allow for servers to return multiple headers of the same type, with a few exceptions, such as Set-Cookie. If the server does return duplicates for a header that normally does not support duplicates, then there is either a bug in the web application, or the user has successfully executed a "Response Splitting" attack. In either case, the service located at the URL this incident is triggered for should probably be reviewed for response splitting vulnerabilities or bugs that would cause duplicate response headers to be returned.

Activity Processors: Header Processor: Incident - Illegal Request Header

Complexity: Suspicious (1.0)

Default Response: None.

Cause: WebApp Secure monitors all of the request headers included by clients. It has a list of known request headers that should never be accepted. This list is configurable, and by default, includes any headers known to be exclusively involved in malicious activity. Should a user include one of the illegal headers, this incident will be triggered. Because the list of illegal headers is configurable, it cannot be guaranteed that the request that contained the header is strictly malicious, but it does signify that the client is doing something highly unusual.

Behavior: Some HTTP headers can be used in order to get the server to do something it isn't designed to do. For example, the "max-forwards" header can be used to specify how many hops within the internal network the request should make before it is dropped. An attacker could use this header to identify how many network devices are between themselves and the target webserver. Because the list of illegal headers is customizable, the type of behavior the header relates to can vary. However this type of behavior is generally performed when scoping the attack surface of the website.

Activity Processors: Header Processor: Incident - Illegal Response Header

Complexity: Informational (0.0)

Default Response: None.

Cause: WebApp Secure monitors all of the response headers sent to the client from the web application. It has a list of known response headers that should never be returned. This list is configurable, and by default, includes any headers known to compromise the server's identity or security. Should the server return one of the illegal headers, this incident will be triggered. Because the list of illegal headers is configurable, it cannot be guaranteed that the request that contained the header is strictly malicious, but it does signify that something unusual has taken place. This can even represent a hackers successful attempt to exploit a backend service.

Behavior: There is a strict set of HTTP response headers that browsers understand and can actually use. Any headers returned by the server outside of the standard set could potentially expose information about the server or its software. Some headers can even be used to execute more complex attacks. In order to protect the server in the event of a serious issue (such as a "Response Splitting¹⁵⁹" attack), some headers can be configured as illegal. Because the set is configurable, it is not straight forward as to what the actual header means or what vulnerability it might be targeted at.

Activity Processors: Header Processor: Incident - Missing All Headers

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds and Captcha.

Cause: Most legitimate web browsers and tools submit at least a few headers with each HTTP request. Headers are used to provide valuable information to the server when trying to construct a response, such as what type of browser the user is using, or what domain name they are trying to access. If a user submits a request that does not contain any headers at all, this incident will be triggered. Note that this incident only triggers on HTTP 1.1 requests (not on HTTP 1.0 requests). Also note that "X-" headers are not counted as headers for this incident.

Behavior: Not providing any headers at all is generally an activity performed when probing an IP to see if it is running a webserver. The user will submit a minimal request containing 1 line of text, and see if the response given back from the server is an HTTP response. If so, the attacker has confirmed that the IP is hosting a webserver on the given port. In many cases, the attacker will also be able to identify which webserver is running, and if

that webserver has any known vulnerabilities. Such information can then be used to attack the webserver directly.

Activity Processors: Header Processor: Incident - Missing Host Header

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds and Captcha.

Cause: All legitimate web browsers submit a Host header with each HTTP request. The host header contains the value entered into the address bar as the server. This could be either the server IP address or the domain name. In either case, it will always be provided. If a user submits a request that does not contain a Host header, this incident will be triggered. Note that this incident only triggers on HTTP 1.1 requests (not on HTTP 1.0 requests).

Behavior: Not providing a host header is generally an activity performed when trying to scope the attack surface of the website. Some webserver are configured to host different websites from the same IP address, based on which domain name is supplied. Hackers will often attempt to send a request without a host header to see if the server will serve back a default website. If the default website is not the main website, this can provide additional pages the attacker can attempt to exploit. This could be considered a "Server Misconfiguration" weakness, but can also be a legitimate design choice for the webserver and its applications. It does not necessarily expose a vulnerability as long as the default web application is secure. Because all major browsers submit host headers on every request, the user would need to take advantage of a more complex tool, such as a raw data client, or HTTP debugging proxy to manually construct a request that does not have a host header. As such, this activity is almost always malicious. In a few cases, some legitimate monitoring tools can omit this header, but those tools should be added to the trusted IP list in configuration.

Activity Processors: Header Processor: Incident - Missing Request Header

Complexity: Low (2.0)

Default Response: None.

Cause: WebApp Secure monitors all of the request headers sent from the client to the server. It also maintains a list of headers which are required for all HTTP requests (such as Host and User-Agent). If one of the required headers is not included in a request, this incident will be triggered.

Behavior: Every legitimate client will always supply specific headers such as "Host" and "User-Agent". If a client does not provide these headers, then the client is likely not a legitimate user. There are several different cases of not legitimate clients, such as hacking tools, manually crafted HTTP requests using something like Putty, or a network diagnostic tool such as nagios. Because there are a few cases that are not necessarily malicious (such as nagios), the incident itself is not necessarily malicious. It does however exclude the user from being a legitimate web browser doing the intended actions allowed by the web application.

Activity Processors: Header Processor: Incident - Missing Response Header

Complexity: Informational (0.0)

Default Response: None.

Cause: WebApp Secure monitors all of the response headers sent from the server to the client. It also maintains a list of headers which are required for all HTTP responses (such as Content-Type). If one of the required headers is not included in a response, this incident will be triggered.

Behavior: If the server is acting correctly, it should always return all of the required response headers. If it is missing a response header, this is likely due to a bug in the web application, or a successfully executed "Response Splitting" attack. In either case, the service located at the URL this incident is triggered for, should probably be reviewed for either response splitting vulnerabilities, or bugs that would cause abnormal HTTP responses (such as dropping the connection immediately after sending the status code).

Activity Processors: Header Processor: Incident - Missing User Agent Header

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds and Captcha.

Cause: Most legitimate web browsers and tools submit a User-Agent header with each HTTP request. The user agent header contains information that identifies which software the user is using to access the website, whether that software is Googlebot, Firefox, Safari, or another piece of software. If a user submits a request that does not contain a User-Agent header, this incident will be triggered.

Behavior: Not providing a user-agent header is generally an activity performed trying to evade detection. The user agent header provides identifying information that could be used by the webserver to track requests made by the same user. It can also provide information about the user's personal computer. Sometimes, hackers will replace the user agent string with another user agent string that is perfectly legitimate, but for a different environment than the one they are actually using. Some legitimate users also take this measure as a general security practice; therefore, as long as at least some value is submitted for the user-agent, it cannot be guaranteed to be a malicious act. However, in the case of the header being absent, a user would have had to take advantage of a tool or debugging proxy in order to filter the traffic. This is almost always performed during the course of a malicious action. Some tools such as network health monitors can also trigger this incident, because they are doing something normal users should not do, but they are considered trusted. In this case, the IP addresses of those tools should be added to the configuration trusted IP whitelist.

Activity Processors: Header Processor: Incident - Request Header Overflow

Complexity: Suspicious (1.0)

Default Response: 3x = Compound Request Header Overflow Incident.

Cause: WebApp Secure monitors all of the request headers sent from the client to the server. It has a configured limit that defines how long any individual header is allowed to be. After 3 or more headers are submitted that exceed the limit, this incident will be triggered.

Behavior: While not as common as form inputs or query parameter inputs, some web applications actually use the values submitted in headers within their code base. If these values are treated incorrectly, such as not being validated before being used in an SQL statement, they potentially expose the same set of vulnerabilities a form input might. As such a hacker who is attempting to execute a "Buffer Overflow162" attack might do so by attempting to provide an excessively long value in a header. They can also use an excessively long header value to craft a complex "SQL Injection" attack. Because the user submitted multiple headers which exceeded the defined limit, the intentions of the user are more likely to be malicious. It is less likely that a poorly crafted browser plug-in would overflow multiple headers, despite the possibility that it might overflow a single one. Because there is a possibility that a legitimate user with a poorly-written browser plugin can cause a header of unusual length to be submitted, this incident cannot be guaranteed to be malicious from just a single case.

Activity Processors: Header Processor: Incident - Unexpected Request Header

Complexity: Informational (0.0)

Default Response: None

Cause: WebApp Secure monitors all of the request headers included by clients. It has a list of known request headers that should be accepted. This list includes all of the headers defined in the HTTP RFC document, which means that if any additional headers are passed, it is part of some non standard HTTP extension. Should a user include a non standard header, this incident will be triggered. It is not necessarily a malicious action on its own, but it does signify that the client is unusual in some way (and potentially malicious) and therefore warrants additional monitoring.

Behavior: When attackers are trying to exploit a server, one of the techniques is to attempt to profile what software the server is running. This can be partially accomplished by observing how the server reacts to various types of headers. For example, if the attacker knows that a specific third party web application has a feature where it behaves differently if you send a header "X-No-Auth", then a hacker might send "X-No-Auth" to the site just to see what happens. While this could represent a higher level attack on a specific application; sending non standard headers is more likely part of the hacker's effort to scope the attack surface of the website. This incident alone cannot be deemed malicious because some users have browser plug-ins installed that automatically include non standard headers with requests to some sites. Additionally, some AJAX sites also pass around custom headers as part of their expected protocol.

Activity Processors: Method Processor

GET and POST are two very well known HTTP request methods. A request method is a keyword that tells the server what type of request the user is making. In the case of a GET, the user is requesting a resource. In the case of a POST, the user is submitting data

to a resource. There are however, several other supported request methods which include HEAD, PUT, DELETE, TRACE, and OPTIONS. These methods are intended to divide the types of requests into more granular operation. In almost all web application implementations, the PUT, DELETE, TRACE and OPTIONS methods are all left unimplemented. Unfortunately, some systems provide default implementations for things such as TRACE and OPTIONS. As a result, some administrators accidentally expose unprotected services. Hackers often try these different request methods to identify servers which support them, and therefore can be vulnerable.

Table 25: Method Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			Whether traffic should be passed through this processor.
Processor Enabled	Boolean	True	
Advanced			
Block Unknown Methods	Boolean	True	Whether to block requests that contain unknown HTTP methods.
Block Unknown Protocol	Boolean	True	Whether to block requests that contain unknown HTTP protocols.
Known Methods	Collection	Collection	The list of known HTTP methods. Also allows you to customize the action to take for each occurrence of the known HTTP method.
Incident: Illegal Method Requested	Boolean	True	The user issued a request using an HTTP method which is considered illegal.
Incident: Unexpected Method Requested	Boolean	True	The user issued a request using a request method other than GET, POST, and HEAD, which resulted in a server error.
Incident: Missing HTTP Protocol	Boolean	True	No protocol specified in GET line.
Incident: Unknown HTTP Protocol	Boolean	True	Non standard protocol specified in GET line (anything except 0.9, 1.0, 1.1).

Activity Processors: Method Processor: Incident - Illegal Method Requested

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds and 1 Day Clear Inputs in 10 minutes

Cause: HTTP supports several different "methods" of submitting data to a webserver. These methods generally include "GET", "POST", and "HEAD", and less commonly "PUT", "DELETE", "TRACE", and "OPTIONS". WebApp Secure monitors all of the methods used by a user when issuing HTTP requests, and compares them to a configured list of known and allowed HTTP methods. If the user submits a request that uses a method which is not in the list of known methods, this incident will be triggered.

Behavior: HTTP methods allow the webserver to handle user provided data in different ways. However some of the supported methods are somewhat insecure and should not be supported unless absolutely necessary. In a few cases, methods which are not standard to HTTP are used by third party web applications. When an attacker is looking for a known vulnerability, they can issue requests using some of these custom defined HTTP methods to see if the server accepts or rejects the request. If the server accepts the request, then the software is likely installed. This type of activity is generally performed when scoping the attack surface of the web application. It is possible that if a third-party web application is legitimately installed and is using custom HTTP methods, that those methods will need to be added to the list of configured HTTP methods so as not to flag users who are using those applications. In either case, because it is possible for this incident to happen without malicious intent, it is considered only suspicious.

Activity Processors: Method Processor: Incident - Unexpected Method Requested

Complexity: Suspicious (1.0)

Default Response: None.

Cause: HTTP supports several different "methods" of submitting data to a webserver. These methods generally include "GET", "POST", and "HEAD", and less commonly "PUT", "DELETE", "TRACE", and "OPTIONS". WebApp Secure monitors all of the methods used by a user when issuing HTTP requests, and compares them to a configured list of known and allowed HTTP methods. If the user submits a request that uses a method which is not in the list of known methods, this incident will be triggered.

Behavior: HTTP methods allow the webserver to handle user provided data in different ways. However some of the supported methods are somewhat insecure and should not be supported unless absolutely necessary. In a few cases, methods which are not standard to HTTP are used by third party web applications. When an attacker is looking for a known vulnerability, they can issue requests using some of these custom defined HTTP methods to see if the server accepts or rejects the request. If the server accepts the request, then the software is likely installed. This type of activity is generally performed when scoping the attack surface of the web application. It is possible that if a third party web application is legitimately installed and is using custom HTTP methods, that those methods will need to be added to the list of configured HTTP methods so as not to flag users who are using those applications. In either case, because it is possible for this incident to happen without malicious intent, it is considered only suspicious.

Activity Processors: Method Processor: Incident - Missing HTTP Protocol

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection 2-6 seconds & 1 Hour Clear Inputs

Cause: HTTP comes in several different versions. These are specified in each request issued by a client to the webserver. The acceptable standard versions are 0.9, 1.0, and 1.1. Any other protocol represents a nonstandard HTTP request issued by a non-standard HTTP client. Under nearly every legitimate use-case, there is no reason to either omit the protocol or to provide one that is not standard. This incident triggers whenever a user submits a request that is completely missing a protocol version. This would represent a clear violation of the HTTP protocol RFC specifications.

Behavior: This incident is likely to occur whenever the attacker is attempting to create a custom attack script against the website. They can have either forgotten to include a protocol value, or they are intentionally omitting it to prevent intended functionality by one of the devices that processes the request. For example, an attacker can try to submit a request without a protocol in an effort to break security devices protecting the webserver. These security devices might not be able to handle non-standard protocols correctly, and as a result, can allow malicious requests to reach the backend unmodified.

Activity Processors: Method Processor: Incident - Unknown HTTP Protocol

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection 2-6 seconds & 1 Hour Clear Inputs

Cause: HTTP comes in several different versions. These are specified in each request issued by a client to the webserver. The acceptable standard versions are 0.9, 1.0, and 1.1. Any other protocol represents a nonstandard HTTP request issued by a non-standard HTTP client. Under nearly every legitimate use-case, there is no reason to either omit the protocol or to provide one that is not standard. This incident triggers whenever a user submits a request that contains an unknown protocol version. This would represent a clear violation of the HTTP protocol RFC specifications. The only time this should be acceptable behavior, is if the web application intentionally utilizes a non-standard protocol, however this should rarely, if ever, be the case.

Behavior: This incident is likely to occur whenever the attacker is attempting to create a custom attack script against the website. They can have either mistyped the protocol value, or they are intentionally using a non-standard value to prevent intended functionality by one of the devices that processes the request. For example, an attacker can try to submit a request with an invalid protocol of 11.1 in an effort to break security devices protecting the webserver. These security devices might not be able to handle non-standard protocols correctly, and as a result, can allow malicious requests to reach the backend unmodified.

Tracking Processors: Etag Beacon Processor

This processor is not intended to identify hacking activity, but instead is intended to help resolve a potential vulnerability in the proxy. Because session tracking in the proxy is done using cookies, it is possible for an attacker to clear their cookies in order to be recognized by the proxy as a new user. This means that if we identify that someone is a hacker, they can shed that classification simply by clearing their cookies. To help resolve this vulnerability, this processor attempts to store identifying information in the browsers JavaScript persistence mechanism. It then uses this information to attempt to identify new sessions as being created by the same user as a previous session. If successful, a hacker who clears their cookies and obtains a new session will be re-associated with the previous session shortly afterwards.

Table 26: Etag Beacon Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Beacon Resource	Configurable	Random	The resource to use for tracking.
Inject Beacon Enabled	Boolean	True	Whether a reference to the beacon resource should be automatically injected into HTML responses.
Revalidation Frequency	Integer	180 (3 Minutes)	How often in seconds to re-validate the old stored etag and re-associate that session with the current one. This value should not be left too short, because it will cause the browser to constantly re-request the fake resource and make the tracking technique more visible.
Incident: Session Etag Spoofing	Boolean	True	The user has provided a fake ETag value which is not a valid session.

Tracking Processors: Etag Beacon Processor: Incident - Session Etag Spoofing

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection 2-6 seconds, 3x = Slow Connection 4-15 seconds.

Cause: The HTTP protocol supports many different types of client side resource caching in order to increase performance. One of these caching mechanisms uses a special header called "E-Tag" to identify when the client already has a valid copy of a resource. When a user requests a resource for the first time, the server has the option of returning an E-Tag header. This header contains a key that represents the version of the file that was returned (ex. an MD5 hash of the file contents). On subsequent requests for the same resource, the client will provide the last E-Tag it was given for that resource. If the server

identifies that both the provided E-Tag, and the actual E-Tag of the file are the same, then it will respond with a 403 status code (Not Modified), and the client will display the last copy it successfully downloaded. This prevents the client from downloading the same version of a resource over and over again. In the event that the E-Tag value does not match, the server will return a new copy of the resource and a new E-Tag value. WebApp Secure takes advantage of this caching mechanism to store a tracking token on the client. It does this by injecting a fake embedded resource reference (such as an image or a JavaScript file) into some of the pages on the protected site. When the browser loads these pages, it will automatically request the embedded resources in the background. The fake resource that was injected by WebApp Secure, will supply a special E-Tag value that contains a tracking token. As the user continues to navigate around the site, each time they load a page that contains a reference to the fake resource, the browser will automatically transmit the previously received E-Tag to the server. This allows WebApp Secure to correlate the requests, even if other tracking mechanisms such as cookies are not successful. The E-Tag value returned by the fake resource, which contains the tracking token, is also digitally signed and encrypted, much like the WebApp Secure session cookie. This prevents a user from successfully guessing a valid E-Tag token, or attempting to provide an arbitrary value without being detected. If an invalid E-Tag is supplied for the fake resource, a "Session ETag Spoofing" incident is triggered.

Behavior: There are very few cases where the E-Tag caching mechanism is part of an attack vector, so this incident would almost exclusively represent a user who is attempting to evade tracking or exploit the tracking method to their advantage. For example, if a user identifies the E-Tag tracking mechanism, they can provide alternate values in order to generate errors in the tracking logic and potentially disconnect otherwise correlated traffic. They can also attempt to guess other valid values in order to correlate otherwise nonrelated traffic (such as a hacker attempting to group other legitimate users into their traffic). While this is a highly unlikely attack vector, it could loosely be classified as a "Credential and Session Prediction" attack. It is also possible, though unlikely, that once an attacker identifies the dynamic nature of the E-Tag header for the fake resource, they can also launch a series of other attacks based on input manipulation. This could include testing for SQL injection, XSS, Buffer Overflow, Integer Overflow, and HTTP Response Splitting among others. However these would be attacks directly against WebApp Secure, and not against the protected web application.

Tracking Processors: Client Beacon Processor

The client beacon processor is intended to digitally tag users for later identification by for embedding a tracking token into the client. There are configurable parameters that administrators can use to configure each type of storage mechanisms that are used track malicious users.

Table 27: Client Beacon Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.

Table 27: Client Beacon Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Advanced			
Flash Storage Enabled	Boolean	True	Whether to use the flash shared data API to track the user.
IE UserData Storage Enabled	Boolean	True	Whether to use Internet explorers userData storage API to track the user.
Local Storage Enabled	Boolean	True	Whether to use Javascript local storage to track the user.
Private Storage Enabled	Boolean	True	Whether to track users between private browsing mode and normal browsing mode in Firefox. A collection of names to use for the Application session cookie.
Silverlight Storage Enabled	Boolean	True	Whether to use the Silverlight storage api to track the user. The Silverlight storage API is unique in that it is exposed across all browsers. If this beacon is enabled and the user has Silverlight installed, this beacon can track the user even if they switch browsers.
Window Name Storage Enabled	Boolean	True	Whether to use the window.name property of the browser window to track the user.
Resource Extensions	Collection	Collection	A collection of resource extensions to use for the processor.
Script Refresh Delay	Integer	3600 (1 Hour)	The amount of time in seconds to cache the randomly generated set of beacon scripts. After this amount of time, the beacon scripts will change.
Script Variations	Integer	30	The number of random variations of the beacon script to cache, and then to select from on each request.
Incident: Beacon Parameter Tampering	Boolean	True	The user has issued a request to the session tracking service which appears to be manually crafted. This is likely in an attempt to spoof another users session, or to exploit the applications session management. This would never happen under normal usage.
Incident: Beacon Session Tampering	Boolean	True	The user has altered the data stored on the client in an effort to prevent tracking. They have altered the data in such a way as to remain consistent with the same data format. This would never happen under normal usage.

Tracking Processors: Client Beacon Processor: Incident - Beacon Parameter Tampering

Complexity: Medium (3.0)

Default Response: 1x = 5 day Clear Inputs in 10 minutes

Cause: WebApp Secure uses a special persistent token that inserts itself in multiple locations throughout the client. When a user returns to the site later on, these tokens are transmitted back to the server. This allows the server to correlate the traffic issued by the same user, even if the requests are weeks apart. This incident is triggered when the user manipulates the token data being transmitted to the server on a subsequent visit. They manipulated the data in such a way as to break the expected formatting for the token.

Behavior: Attempts to manipulate and spoof the tracking tokens are generally performed when the attacker is trying to figure out what the token is used for and potentially evade tracking. Because the format of the token is completely wrong, this is likely a generic input attack, where the user is attempting to find a vulnerability in the code that handles the token. This could include a "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", or "SQL injection" attack among many others. The content of the manipulated token should be reviewed to better understand what type of attack the user was attempting, however because the tokens are heavily encrypted and validated, this incident does not represent a threat to the security of the system tracking mechanism.

Tracking Processors: Client Beacon Processor: Incident - Beacon Session Tampering

Complexity: Medium (3.0)

Default Response: 1x = 5 day Clear Inputs in 10 minutes.

Cause: WebApp Secure uses a special persistent token that inserts itself in multiple locations throughout the client. When a user returns to the site later on, these tokens are transmitted back to the server. This allows the server to correlate the traffic issued by the same user, even if the requests are weeks apart. This incident is triggered when the user manipulates the token data being transmitted to the server on a subsequent visit. They manipulated the data in such a way as to remain consistent with the correct formatting for the token, but the token itself is not valid and was never issued by the server.

Behavior: Attempts to manipulate and spoof the tracking tokens are generally performed when the attacker is trying to figure out what the token is used for and potentially evade tracking. If they are assuming it's used for session management, this might also be a part of a "Credential/Session Prediction" attack. Because the format of the submitted modified token is still consistent with the format expected, this is not likely a generic input attack. It also does not represent any threat to the system, as the modified token is simply ignored.

Tracking Processors: Client Fingerprint Processor

This processor is designed to collect uniquely identifying information from requests issued by a user. This information is then compared to the information collected about other sessions in the system. If a match is identified, the two sessions are merged. This allows session association to work even if all storage mechanisms used by the other tracking processors are cleared. Some of the uniquely identifying information includes the browser plugin list, the system font list, time skew, time-zone, user-agent, system language, and so on.

Table 28: Client Fingerprint Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	False	Whether traffic should be passed through this processor.
Exclude Rules	Collection	[collection:0]	The fingerprint association rules to ignore.
Excluded Collectors	Collection	[collection:0]	The data points to prevent collection of on the client.
Hash Fingerprint Data	Boolean	False	Whether to hash the raw fingerprint data points before storing them. This prevents the recorded data from being used to obtain the original information about the client and reduces the overall storage size requirements. If collecting PII data is a concern, this is a recommended option, as it will eliminate any PII data in place of hashed versions of that data which cannot be reversed.
Page Injection Enabled	Boolean	True	Whether the fingerprint script should be injected into the requested page.
Advanced			
Binary Resource Directory	String	(randomized)	The fake directory where binary resources required by the fingerprinting script are served from.
Data Obfuscation Key	String	(randomized)	The key used to prevent easy reading of the submitted fingerprint data. This should be alphanumeric and at least 8 unique characters long, duplicate characters are allowed, but do not count toward the total 8.

Table 28: Client Fingerprint Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Fingerprint Scope Key	String	(randomized)	The key used to store fingerprint data. If this key is changed, all previously stored fingerprint data will be lost and the system will begin collecting fresh fingerprint data.
Fingerprint Submission Response	HTTP Response	text/plain 200 OK	The response to return when a user attempts to submit a fingerprint in the background. The user will not see this response unless they are using a debug proxy.
Fingerprint Tracking Cookie Name	String	(randomized)	The name of the cookie used on the client to ensure we don't submit multiple copies of the same fingerprinting data. This can be anything, but should not overlap with a legitimate cookie being used on the site.
Hash Fingerprint Data	Boolean	False	Whether to hash the raw fingerprint data points before storing them. This prevents the recorded data from being used to obtain the original information about the client and reduces the overall storage size requirements. If collecting PII data is a concern, this is a recommended option, as it will eliminate any PII data in place of hashed versions of that data which cannot be reversed.
Script Filename	String	(randomized)	The filename to use when serving the fingerprint script to the client.
Submission Filename	String	(randomized)	The filename where fingerprint data should be submitted back to the server

Table 28: Client Fingerprint Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Incident: Fingerprint Directory Indexing	Boolean	True	The user requested a directory index listing on the fake directory used to serve binary resources required by the fingerprinting script. Since this is a fake directory, the request represents a malicious action.
Incident: Fingerprint Directory Probing	Boolean	True	The user requested a random file within the fake directory used to serve binary resources required by the fingerprinting script. Since only files we specifically reference in the fingerprinting script should be requested, this represents a malicious action.
Incident: Fingerprint Manipulation	Boolean	True	The user submitted fingerprint data to the server which was not properly formatted. This likely means that the user was manipulating the fingerprinting data or spoofed it entirely.

Tracking Processors: Client Fingerprint Processor: Incident - Fingerprint Directory Indexing

Complexity: Low (2.0)

Default Response: n/a

Cause: The client fingerprint processor is designed to obtain a semi-unique identifier from the clients rendering engine. The fingerprint is a hash of data obtained through JavaScript such as the plugin list, time zone, and screen resolution. In order to calculate a fingerprint, some binary resources such as flash objects might be required. These resources will be served from a known fake directory. This incident is triggered if the user attempts to get a directory index listing from the known fake resource directory.

Behavior: If an attacker discovers the script being used to collect and submit the fingerprint data, they might be interested to know what else is in the directory where fingerprint binary resources are served. As such, they can request a directory index listing from the fake directory. Because the directory is fake, there are no files to list, but the simply action of attempting to get the list is indicative of abusive behavior. If an attacker is able to obtain a directory index listing, they can attempt to exploit some of the other resources in the directory, or gain information about the website that can otherwise not be available. Any attempts to index the directory will result in a 403, which will yield no useful

information to the attacker. This is usually part of a spidering effort and targets "Predictable Resource Location" vulnerabilities.

Tracking Processors: Client Fingerprint Processor: Incident - Fingerprint Directory Probing

Complexity: Low (2.0)

Default Response: n/a

Cause: The client fingerprint processor is designed to obtain a semi-unique identifier from the clients rendering engine. The fingerprint is a hash of data obtained through JavaScript such as the plugin list, time zone, and screen resolution. In order to calculate a fingerprint, some binary resources such as flash objects might be required. These resources will be served from a known fake directory. This incident is triggered if the user attempts to request a file in the fake directory that does not exist. In other words, they are looking for a specific file that does not exist within a fake directory.

Behavior: If an attacker discovers the script being used to collect and submit the fingerprint data, they might be interested to know what else is in the directory where fingerprint resources are served from. As such, they can request specific files they think they be inside the fake directory. Because the directory is fake, there are no actual files available, but the simply action of attempting to get a resource that does not exist in a fake directory is indicative of abusive behavior. This type of attack is generally targeted at "Predictable Resource Location" vulnerabilities.

Tracking Processors: Client Fingerprint Processor: Incident - Fingerprint Manipulation

Complexity: Medium (3.0)

Default Response: n/a

Cause: The client fingerprint processor is designed to obtain a semi-unique identifier from the clients rendering engine. The fingerprint is a hash of data obtained through JavaScript such as the plugin list, time zone, and screen resolution. This incident is triggered when the user attempts to submit an invalid fingerprint.

Behavior: Normally, the fingerprinting code will be allowed to execute on the client without any problems. However if an attacker discovers the fingerprinting code, they might try to spoof fingerprints of other users, or simply try to exploit the fingerprint service. To do this, they can create a fake fingerprint value and submit it to the server in the same way that legitimate fingerprints are submitted. It likely would not be clear to the attacker as to what the value is used for, or how the value is consumed by the server, so this type of activity would be purely exploratory. Once the attacker identifies a valid fingerprint that was not generated from their rendering engine, they will likely continue to statically submit that same fingerprint on all transactions. Once that happens, it will not be possible to identify the manipulated fingerprint. So this incident should come early in the attack, but will stop once the attacker has reached their goal. In such a case, the attacker is simply trying to disguise their true identity. If the modified fingerprint is not alpha numeric and contains special characters, then the attacker is probably attempting to launch a targeted

attack against the way the service consumes the data, such as a "SQL Injection", "XSS", or "Buffer Overflow" attack.

Tracking Processors: Client Classification Processor

The client classification processor is designed to detect popular legitimate search engine bots. These types of bots are notorious for performing aggressive spider activity on websites, and often this activity can trigger security related incidents. Using this processor to define the conditions used to identify such bots, allows the system to ignore security incidents from those clients. This will remove search engine related false positives, as well as prevent errors in indexed and cached results. The popular search engines are included by default, but if additional search engines should be allowed, new rules can be created. Be careful not to define a rule that will match clients other than the targeted search engine bot. The less specific the conditions of a rule, the easier it will be for an attacker to spoof the search engine and circumvent detection. It is critical that DNS be enabled on WebApp Secure to achieve effective classification of search engines. Not enabling DNS and leaving this processor turned on, can result in some attackers not being identified.

If a client is classified as a search engine based on one of the defined rules, then that client will not be able to generate incidents, and additionally:

- Query String Processor will be turned off for that user (no query param injections)
- Hidden Link Processor will be turned off for that user (no hidden link injections)

This is done to ensure that the results cached by the search engine bot do not include fake code that can change in the future, and thus end up flagging clients who are following legitimate search engine links. Classification rules are made up of a series of patterns to run against various attributes of the client:

- IP Address
- Hostname
- User Agent
- Country Code
- City
- Region
- Header Name and Value

At least one pattern must be specified on at least one attribute, however you can specify patterns for as many attributes as the bot will allow. For example, if the bot changes its IP address constantly, then you should not define a pattern for the IP. However if the hostname always ends in google.com, then a pattern of `[.]google[.]com$` could be assigned to the "Hostname" attribute. If the user agent always contains "googlebot", then "googlebot" could be assigned as the user agent pattern. Here is an example of a complete pattern for the Googlebot search engine spider:

Hostname Pattern: `[.]google(bot)?[.]com$`

User Agent Pattern: (adsbot.google|googlebot|Google[]Web[]Preview|Mediapartners-Google)
 Country Pattern: US
 Region Pattern: (California|Georgia)



NOTE: It would be extremely difficult for an attacker to spoof values for all of those attributes which would match the patterns. For example, spoofing the reverse DNS lookup to end in “.google.com” would require serious effort, and would require insecure DNS configuration on behalf of the WebApp Secure administrator. Ideally every rule should include either an “ip” or “hostname” pattern.

Table 29: Client Classification Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	False	Whether traffic should be passed through this processor.
Classification Rules			
Client Type	String	(none)	The name of the type of client being identified.
IP Pattern	String	(none)	The IP address pattern to require (if any).
Hostname Pattern	String	(none)	The hostname pattern to require (if any) if DNS is enabled.
User Agent Pattern	String	(none)	The user agent pattern to require (if any).
Country Pattern	String	(none)	The country pattern to require (if any).
City Pattern	String	(none)	The city pattern to require (if any).
Region Pattern	String	(none)	The region pattern to require (if any).
Header Name Pattern	String	(none)	A pattern used to identify a required header name (if any).
Header Value Pattern	String	(none)	A pattern used to verify the value of a header that matches the header name pattern (if any).

Response Processors

The processors in this section are responsible for issuing the various counter responses to malicious users on a server protected by WebApp Secure. A response is activated when WebApp Secure believes intervention is required between the profiled user and the webserver. This response can manifest into any of the types fully explained below.

Response Methodology: When WebApp Secure believes a response is required, the type of response issued depends on the type of behavior the malicious user exhibited to receive the response. For example, users that WebApp Secure think are automated tools will likely get issued a CAPTCHA response, whereas it is obvious that a real malicious user (not a bot) will be able to solve a CAPTCHA. In the second case, adding a 2 to 6 second slow might be more effective at wasting the hacker's time. Another factor that comes into play when issuing counter responses is risk level. If WebApp Secure believes a user is of no immediate risk to the system, it might only activate those responses which still allow the user to browse the site somehow, such as the Warning response or Slow Connection response. This way, WebApp Secure can monitor that user and gather additional information to properly assess their risk level. If WebApp Secure believes the user is a danger to the system, it will issue a more severe response, such as stripping out all inputs on every request or outright blocking the profile. Some responses might not get issued right away. For example, an incident can produce "a permanent block in 20 minutes". The reason for this delay in the counter response is that WebApp Secure uses this buffer time to gather some last-minute information on the profile before issuing the final response. WebApp Secure will respond instantly if it perceives immediate threat to the integrity of the system, but instances where this is not the case allow WebApp Secure to profile the attacker for a bit longer. The end result will be a more complete look at the attacker and his/her habits.

Types of Responses: Certain response processors are self-explanatory, such as the Block Processor (the user will see that they are blocked). Other responses are "invisible" in that there are no manifestations of the response visible to the user. An example of an invisible response processor is the Strip Inputs Processor. This processor will simply Block Processor 125 remove all values from all inputs on any form submitted because WebApp Secure has determined that the user's input can no longer be trusted. On the user-end, they will see nothing that will indicate to them that this response is active (until they figure out that all inputs are not being recognized).

Response Activation: Responses get automatically activated according to rules set forth within WebApp Secure. These rules are outlined for each incident a user can trigger, and are described in the documentation for each processor. The default response for each incident is documented in the User Guide, and will look something like, "Default Response: 1x = Warn User. 2x = 1 Day Block". The '1x' or '2x' indicate the number of incidents of that type triggered. For this example, triggering this incident once results in the Warning Processor being activated. If the same incident is triggered again on the same profile, the user then gets a 1 day block through the Block Processor.



NOTE: You might want to disable automatic counter responses entirely. If this is the case, changing the configuration parameter **Auto Response Activation Enabled** to **False** will prevent any new automatic activations, but will not hinder your ability to manually activate responses on profiles. (Configuration > Global Configuration > Auto Response Service > Auto Response Activation Enabled = False)

Compounding and Overriding Responses

- Warning - There is no need to warn someone when they are already blocked.
- Captcha - If the user is ever unblocked (or the block expires), they will be prompted to solve the captcha.
- Cloppy - If they are ever unblocked (or the block expires) Cloppy will appear.
- Google Maps - If the user is ever unblocked (or the block expires), they will be shown the Google map.

Captcha overrides:

- Warning - WebApp Secure will warn after they solve the captcha.
- Cloppy - Cloppy will appear after they solve the captcha.
- Google Maps - The Google map will be shown after they solve the captcha.

Strip Inputs overrides:

- Break Authentication - It is redundant, as WebApp Secure is already stripping login credentials.

Response Processors: Block Processor

The block processor is actually a form of auto response. When this processor is enabled, it will allow the security system to block a response with "Blocked!" message sent back to the user.



NOTE: There are no actual triggers for this processor; it is a form of response.

Table 30: Block Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			

Table 30: Block Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Block Response	Configurable	HTTP Response	The response to return to the user when they are blocked.

Response Processors: Request Captcha Processor

The Captcha processor is designed to protect specific pages in a web application from automation. This is done by using a "Captcha" challenge, where the user is required to transcribe random characters from an obscured image or muffled audio file in order to complete the request. The intent is that a human would be capable of correctly answering the challenge, while an automated script with no human intervention would be unable to do so. This assumes that the image is obscured enough that text recognition software is not effective, and the audio file significantly distorted to defeat speech-to-text software. Requiring such user interaction is somewhat disruptive, so it should be utilized only for pages that are prime automation targets (such as contact forms, registration pages, login pages, and so on.). Furthermore, these captcha challenges can be customized to fit the style of the application it is protecting.

Table 31: Request Captcha Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Protected Pages	Collection	None	A collection of protected pages.
Advanced			
Bad Request Block Response	HTTP Response	400 HTTP Response	The response to return if the user issues a request that either is too large, or uses multipart and multi-part is disabled.
Blocked Replay Response	String	Random Value	The response to return if the user attempts to submit the validated request multiple times using the same captcha answer, and that behavior is not allowed.
Captcha Binary Directory	String	Random Value	The name of the directory where captcha images and audio files will be served from. This should not conflict with any actual directories on the site.
Captcha Characters	String	Random Value	The characters to use when generating a random captcha value. Avoid using characters that can be easily mixed up. This set of characters is case sensitive.
Captcha State Cookie Name	String	Random Value	The name of the cookie to use to track the active captchas that have not yet been solved. The cookie is only served to the captcha binary directory.

Table 31: Request Captcha Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Captcha Validation Input Name	String	Random Value	The name of the form input used to transmit the captcha validation key. This should be obscure so that users who have not been required to enter a captcha cannot supply bad values to this input to profile the system.
Maximum Active Captchas	Integer	7	The maximum number of captchas any given user can be solving at any given time. This limit can be overcome, but the majority of users will not be able to. This is primarily for performance, as the more active captchas that are allowed, the larger the state cookie becomes.
Support Audio Version	Boolean	True	Whether an audio version of the captcha is provided to the user. This can be a requirement for accessibility, as vision impaired users would otherwise be unable to solve the captcha.
Watermark	String	Random Value	The text to watermark the captcha with. This can be used to prevent the captcha from being used in a phishing attack. For example, an abuser would not be able to simply display the captcha on a different site and ask a user to solve it. The watermark would tip the user off that the captcha was not intended for the site they are visiting. Use %DOMAIN to use the domain name as the watermark.
Cancel URL	String	None	The URL to redirect the user to if they cancel the captcha. This should not be to the same domain, because the domain is being blocked using a captcha, and therefore, canceling would only redirect to a new captcha. An empty value will hide the cancel button.
Captcha Expiration	Integer	2 minutes	The maximum number of seconds the user has to solve the captcha before the request is no longer possible.
Expired Captcha Response	HTTP Response	400 HTTP Response	The response to return if the user submits a validated request after the captcha has expired. This can happen if the user refreshes the results of the captcha long after they have solved it.
Maximum Request Size	Integer	500kb	The maximum number of bytes in a request before it is considered not acceptable for captcha validation, and will be blocked.
Incident: Bad Captcha Answer	Boolean	False	The user was asked to solve a captcha and entered the wrong value. This could be a normal user error, or it could be the results of failed abuse.
Incident: Captcha Cookie Manipulation	Boolean	True	The user submitted a request and was asked to solve a captcha. They then modified the state cookie used to track captchas, making it invalid. This is likely in an attempt to find a way to bypass the captcha validation mechanism.
Incident: Captcha Directory Indexing	Boolean	True	The user has requested a directory index in the directory that serves the captcha images and audio files. This is likely in an attempt to get a list of all active captchas or to identify how the captchas are generated.

Table 31: Request Captcha Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Incident: Captcha Directory Probing	Boolean	True	The user has requested a random file inside the directory that serves the captcha images and audio files. This is likely in an attempt to find an exploitable service or sensitive file that can help bypass the captcha validation mechanism.
Incident: Captcha Disallowed MultiPart	Boolean	True	The user has submitted a multipart form post to the protected page, which has been configured as a disallowed option. This is likely in an attempt to find an edge case the captcha validation mechanism is not expecting.
Incident: Captcha Image Probing	Boolean	True	The user is probing the directory used to serve captcha images. This is likely in an attempt to find hidden files or a way to invoke errors from the captcha serving logic.
Incident: Captcha Parameter Manipulation	Boolean	True	The user has submitted a request with a valid captcha, but they modified the query string parameters. This could be in an attempt to change the output of executing the request without requiring the user to re-validate with another captcha.
Incident: Captcha Request Replay Attack	Boolean	True	The user has attempted to submit the same request multiple times with the same captcha answer. In other words, they solved the captcha once and issued the resulting request multiple times.
Incident: Captcha Request Size Limit Exceeded	Boolean	True	The user has submitted a request to the protected page which contains more data than is allowed. This might be an attempt to reduce system performance by issuing expensive requests, or it can be an indicator of a more complex attack.
Incident: Captcha Request Tampering	Boolean	True	The user submitted a request and was asked to solve a captcha. They introspected the page containing the captcha and altered the serialized request data (the data from the original request before the captcha prompt). They then submitted a valid captcha using the modified request data. This is likely in an attempt to abuse the captcha system and identify a bypass technique.
Incident: Captcha Signature Spoofing	Boolean	True	The user submitted a request and was asked to solve a captcha. They introspected the page containing the captcha and provided a validation key from a previously solved captcha. This is likely in an attempt to submit multiple requests under the validation of the first.
Incident: Captcha Signature Tampering	Boolean	True	The user submitted a request and was asked to solve a captcha. They introspected the page containing the captcha and provided a fake validation key. This is likely in an attempt to bypass the captcha validation mechanism.
Incident: Expired Captcha Request	Boolean	True	The user submitted a request and was given a set window of time to solve a captcha. The user solved the captcha and submitted the request for final processing after the window of time expired. This is likely an indication of a packet replay attack, where the user attempts to invoke the business logic of the protected page multiple times under the same captcha validation.

Table 31: Request Captcha Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Incident: Mismatched Captcha Session	Boolean	True	The user submitted a request and was asked to solve a captcha. They solved the captcha, but upon submitting the request for final processing, they did so under a different session ID. This is likely due to multiple machines participating in the execution of the site workflow and can indicate a serious targeted automation attack.
Incident: No Captcha Answer Provided	Boolean	True	The user attempted to validate a captcha but did not supply an answer to validate. There is no interface that allows the user to do this, so they must be manually executing requests against the captcha validation API in an attempt to evade the mechanism.
Incident: Unsupported Audio Captcha Requested	Boolean	True	The user has requested an audio version of the captcha challenge, but audio is not supported and there should not be an interface to ask for the audio version. The user is likely trying to find a way to more easily bypass the captcha system.

Response Processors: Request Captcha Processor: Incident - Captcha Answer Automation

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds and 1 Day Clear Inputs

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user provides an abnormal volume of bad solutions to the captcha image. For example, the image might have said "Hello", but the user attempted 30 different values all of which did not match "Hello". Because the images can be somewhat difficult to read at times (in order to ensure a script cannot break them), it is not uncommon for a legitimate user to enter the wrong value a few times before getting it right, especially if they are unfamiliar with this type of technique, but after dozens of failed attempts, it is more likely a malicious user.

Behavior: Simply providing a bad solution to the captcha image is not necessarily malicious. Legitimate users are not always able to solve the captcha on the first try. However if a large volume of invalid solutions are provided, then it is more likely that a script is attempting to crack the captcha image through educated guessing and "Brute Force".

Response Processors: Request Captcha Processor: Incident - No Captcha Answer Provided

Complexity: Medium (3.0)

Default Response: 1x = Warn User. 2x = 1 Day Block

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user forces the captcha interface to submit the request without a valid captcha solution. There is no way to do this without manipulating the logic that controls captcha protected requests.

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the captcha, they can use various different techniques. One of these techniques is to try changing various values used by the web application in the captcha mechanism in an effort to see if an error can be generated, or an unexpected outcome can be achieved. This type of probing and reverse engineering is generally performed by advanced hackers. In this specific case, the attacker attempted to submit the captcha protected page without actually solving the captcha. Instead they provided an empty value for the solution parameter. It is not possible to submit an empty solution using the provided captcha interface, so this is almost guaranteed to be a malicious attempt at generating an error and obtaining additional details about the captcha implementation through an "Information Leakage" weakness.

See <http://projects.webappsec.org> for information on attack types.

Response Processors: Request Captcha Processor: Incident - Multiple Captcha Request Overflow

Complexity: Low (2.0)

Default Response: 1x = 1 Day Clear Inputs.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user attempts to submit dozens of captcha protected requests that exceed the configured maximum for protected request sizes.

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the captcha, they can use various different techniques. One of these techniques is to try changing various values used by the web application in the captcha mechanism in an effort to see if an error can be generated, or an unexpected outcome can be achieved. This type of probing and reverse engineering is generally performed by advanced hackers. In this specific case, the attacker submitted dozens of extremely large requests, probably in an effort to find a "Buffer Overflow" vulnerability, which would produce useful error data and potentially open the server up to further exploitation. They might also be attempting to overload the server and execute a "Denial of Service" attack.

Response Processors: Request Captcha Processor: Incident - Unsupported Audio Captcha Requested

Complexity: Medium (3.0)

Default Response: 3x = Slow Connection 2-6 seconds and Warn User. 5x = 1 Day Block.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve

"Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user attempts to request the audio version of a captcha challenge when support for audio captchas has been explicitly disabled.

Behavior: Solving an image based captcha is exceptionally difficult and requires a great deal of time and research. Solving an audio captcha however is far less difficult. There are already multiple open source libraries available for translating speech to text. As such, it is often necessary to disable the support of "audio" captchas for critical workflows (such as administrative login dialogs), unless absolutely necessary for accessibility reasons. This incident occurs when the audio captcha has been disabled, but a user is attempting to manually request the audio version of the captcha challenge anyway. The captcha interface does not expose a link to the audio version unless it is explicitly enabled in configuration, so this would require that the user knows where to look for the audio version, they understand the filename conventions, and they know how to make the request manually to download the file. In either case, if audio captchas are not enabled (through configuration), then this effort will not be successful.

Response Processors: Request Captcha Processor: Incident - Bad Captcha Answer

Complexity: Suspicious (1.0)

Default Response: 10x = Captcha Answer Automation Incident.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user provides a bad solution to the captcha image. For example, the image might have said "Hello", but the user typed "hfiiO" instead. Because the images can be somewhat difficult to read at times (in order to ensure a script cannot break them), it is not uncommon for a legitimate user to enter the wrong value a few times before getting it right, especially if they are unfamiliar with this type of technique.

Behavior: Simply providing a bad solution to the captcha image is not necessarily malicious. Legitimate users are not always able to solve the captcha on the first try. However if a large volume of invalid solutions are provided, then it is more likely that a script is attempting to crack the captcha image through educated guessing and "Brute Force".

Response Processors: Request Captcha Processor: Incident - Mismatched Captcha Session

Complexity: High (4.0)

Default Response: 1x = Warn User, 2x = 5 Day Clear Inputs.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user provides a solution to a captcha that was issued for a different session than their own, as might be the case in a script that uses minimal human interaction to solve the captcha's, but everything else is automated

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the captcha, they might use various different techniques. One of these techniques is to try and harvest successfully solves captchas from other users on the site. This can be done either by infecting those machines with a virus, or by implanting script into some of the sites pages (possibly through XSS). If this technique is used, then the captcha that is being solved might not have originated from the same session as the user who is submitting the solution. This is a dead giveaway that the user is attempting to defeat the captcha system to automate a specific task.

Response Processors: Request Captcha Processor: Incident - Expired Captcha Request

Complexity: Suspicious (1.0)

Default Response: None.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways

by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user provides a solution to a captcha after the allotted time for solving the captcha has elapsed.

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the captcha, they can use various different techniques. One of these techniques is to run expensive image processing algorithms on the captcha image in order to identify what the represented value might be. Additionally, a user might attempt to send the captcha to a warehouse of human captcha solvers. These warehouses specialize in solving large volumes of captchas at a fairly low price (less than a penny per captcha). In either case, it can take several minutes to get the correct captcha answer, and will likely run out the amount of time the user is allowed for solving the captcha. If using a browser, the input would flat out stop accepting answers, but in a scripted scenario, the script will likely try and submit the value anyway, because it is unaware of the expiration. It is possible that this incident would be triggered by a legitimate user, if they were to refresh the page that was produced after the captcha was solved. This would effectively cause the captcha to be reprocessed after the expiration time had been exceeded. As such, this incident on its own is not considered malicious.

Response Processors: Request Captcha Processor: Incident - Captcha Request Tampering

Complexity: High (4.0)

Default Response: 1x = Warn User. 2x = 5 Day Clear Inputs.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user provides a solution to a captcha which is correct, but they have modified the parameter containing the original request (which is heavily encrypted to prevent tampering).

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the captcha, they can use various different techniques. One of these techniques is to try changing various values used by the web application in the captcha mechanism in an effort to see if an error can be generated, or an unexpected outcome can be achieved. This type of probing and reverse engineering is generally performed by advanced hackers. The parameter that was modified contained the original request data (before the captcha was issued), it is likely that the attacker is attempting to smuggle a malicious payload through the system without being detected by any network or web firewalls. Because this parameter uses heavy encryption and validation, this type of activity will not produce any useful information or expose any vulnerabilities. Depending on the value they submitted for the original request data, this can also fall under one of the other attack categories involving manipulating general inputs, such as a "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", or "SQL injection" attack among many others.

Response Processors: Request Captcha Processor: Incident - Captcha Signature Tampering

Complexity: High (4.0)

Default Response: 1x = Warn User. 2x = 5 Day Clear Inputs.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user provides a solution to a captcha which is correct, but they have modified the integrity checking signature passed along with the captcha solution.

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the captcha, they can use various different techniques. One of these techniques is to try changing various values used by the web application in the captcha mechanism in an effort to see if an error can be generated, or an unexpected outcome can be achieved. This type of probing and reverse engineering is generally performed by advanced hackers. Depending on the value they submitted for the original request data, this can also fall under one of the other attack categories involving manipulating general inputs, such as a "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", or "SQL injection" attack among many others.

Response Processors: Request Captcha Processor: Incident - Captcha Signature Spoofing

Complexity: High (4.0)

Default Response: 1x = Warn User. 2x = 5 Day Clear Inputs.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user provides a solution to a captcha which is correct, but they have replaced the integrity checking signature passed along with the captcha solution to one that was used in a previous captcha solution.

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the captcha, they can use various different techniques. One of these techniques is to try changing various values used by the web application in the captcha mechanism in an effort to see if an error can be generated, or an unexpected outcome can be achieved. This type of probing and reverse engineering is generally performed by advanced hackers. This specific incident generally reflects the behavior of a user who is trying to submit a request that would normally be protected by a captcha, but they are trying to trick the system into thinking the captcha was solved correctly, even though it was not. This is generally looking for a "Insufficient Anti- Automation" weakness in the captcha handling mechanism.

Response Processors: Request Captcha Processor: Incident - Captcha Cookie Manipulation

Complexity: Medium (3.0)

Default Response: 1x = Warn User. 2x = 5 Day Clear Inputs.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the

deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user alters the cookies used to maintain captcha state.

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the captcha, they can use various different techniques. One of these techniques is to try changing various values used by the web application in the captcha mechanism in an effort to see if an error can be generated, or an unexpected outcome can be achieved. This type of probing and reverse engineering is generally performed by advanced hackers. In this specific case, the attacker modified a cookie that is used to maintain the state of the captcha. The cookie is heavily encrypted, but the attacker might be attempting to establish a way of either identifying what the value of the captcha is algorithmically (by analyzing the cookie value), or they can be attempting to assign a value to the captcha. In either case, this activity generally indicates a user who is trying to find a way to bypass the captcha. Depending on the value they submitted for the original request data, this can also fall under one of the other attack categories involving manipulating general inputs, such as a "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", or "SQL injection" attack among many others.

Response Processors: Request Captcha Processor: Incident - Captcha Image Probing

Complexity: Low (2.0)

Default Response: 1x = Warn User. 2x = 5 Day Block.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user attempts to request a captcha image file for a request that is not being protected by a captcha.

Behavior: In order to find a way to bypass the captcha mechanism, attackers will often attempt to collect a large number of captcha images for offline analysis. If the attacker can find a pattern in how the captcha images are issued, or how the filename relates to the value in the image, then they can effectively bypass the captcha mechanism at will. In this case, the attacker is guessing arbitrary captcha image filenames, but is attempting to keep the format of the names consistent with known captcha image URL's. Because the filename used and the values in the image have no correlation, this technique will not be successful and will simply waste the attacker's time and resources.

Response Processors: Request Captcha Processor: Incident - Captcha Request Size Limit Exceeded

Complexity: Suspicious (1.0)

Default Response: 10x = Multiple Captcha Request Overflow Incident.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user attempts to submit a captcha protected request that contains a request body larger than the configured maximum.

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the captcha, they can use various different techniques. One of these techniques is to try changing various values used by the web application in the captcha mechanism in an effort to see if an error can be generated or if an unexpected outcome can be achieved. This type of probing and reverse engineering is generally performed by advanced hackers. In this specific case, the attacker submitted an extremely large request, probably in an effort to find a "Buffer Overflow" vulnerability, which would produce useful error data and potentially open the server up to further exploitation. This incident is not necessarily malicious on its own, as it is possible for a normal user to submit a value that is larger than the configured maximum, especially if the configured maximum is small, or if the form protected by the captcha allows file posts.

Response Processors: Request Captcha Processor: Incident - Captcha Disallowed MultiPart

Complexity: Suspicious (1.0)

Default Response: 10x = Multiple Captcha Disallow Multipart Incident.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user attempts to submit a captcha protected request that contains a binary file, and the captcha is explicitly configured to not allow binary file submission (it has been configured to disallow multi-part form submissions).

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the captcha, they can use various different techniques. One of these techniques is to try changing various values used by the web application in the captcha mechanism in an effort to see if an error can be generated, or an unexpected outcome can be achieved. This type of probing and reverse engineering is generally performed by advanced hackers. In this specific case, the attacker submitted a binary file in the request that is being protected. The captcha in this case has been explicitly configured to not allow Multi-Part form submissions, so this represents unexpected and undesired activity. Using Multi-Part forms, the attacker can more easily accomplish a "Buffer Overflow" attack, which would produce useful error data and potentially open the server up to further exploitation. Additionally, some web applications do not handle the encoding used for multi-part forms gracefully, so error information can also be obtained from conflicts arising from the submission type. This is not necessarily a malicious incident on its own, because it is possible that the user is legitimately submitting a multi-part form, and just happened to have the captcha activated during the submission. However this is a very rare case, and still represents a somewhat suspicious client.

Response Processors: Request Captcha Processor: Incident - Captcha Directory Indexing

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds and 1 Day Block.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed.

Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user attempts to request a directory index from the same fake directory as the captcha images are being served from.

Behavior: When attempting to either bypass the captcha mechanism, or find a vulnerability in the server, attackers will often try finding unlinked resources throughout the website. The captcha mechanism uses a fake directory in order to serve the images and audio files that contain the captcha challenge. If the attacker is requesting an arbitrary file within the same fake directory, they are likely trying to find a "Predictable Resource Location" vulnerability. In this specific case, the attacker is attempting to get a full file listing of everything inside the captcha directory. This could potentially be used to get a massive list of all active captcha URL's, or to find resources that are used in the creation of captcha challenges. The directory index will not be allowed, so this does not actually provide the attacker with any useful information.

Response Processors: Request Captcha Processor: Incident - Captcha Directory Probing

Complexity: Low (2.0)

Default Response: 1x = Warn User. 2x = Slow Connection 2-6 seconds and 5 Day Block.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user attempts to request an arbitrary file (not a captcha image, but something else) from within the same fake directory as the captcha images are being served from.

Behavior: When attempting to either bypass the captcha mechanism, or find a vulnerability in the server, attackers will often try finding unlinked resources throughout the website. The captcha mechanism uses a fake directory in order to serve the images and audio

files that contain the captcha challenge. If the attacker is requesting an arbitrary file within the same fake directory, they are likely trying to find a "Predictable Resource Location" vulnerability. For example, the attacker might be trying to find a source file in the captcha serving directory in hopes of actually being able to get the source code behind how captcha images are generated. Because the directory is fake, the attacker will never find any of the resources they are looking for.

Response Processors: Request Captcha Processor: Incident - Captcha Parameter Manipulation

Complexity: Suspicious (1.0)

Default Response: 5x = Multiple Captcha Parameter Manipulation Incident.

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user attempts to submit multiple solutions for multiple captchas, but they keep modifying the query parameters that were submitted with the original requests. For example, if the user submitted a "add product to cart" request, and one of the query parameters was the item to add, this incident would be triggered if after solving the captcha, the value of that query parameter was modified to some other value, and this modification happened dozens of times.

Behavior: Because captcha's prevent automation, attackers will sometimes try and find ways to abuse the technique used to request the captcha in order to exploit the site. For example, if the attacker can find a way to submit the same solution over and over again, but have the web application perform a different action each time, they might be able to solve the captcha once and still automate the resulting workflow. In this case, the attacker changed a query parameter that was submitted with the original request. They submitted the original request, solved the captcha, changed the query parameter, and then resubmitted the solved captcha request. In some cases, this might cause the web application to execute a different operation based on the difference in query parameter values. For example, if the protected workflow is "add product to cart" on a shopping site, then the attacker might attempt to submit the same solved captcha repeatedly, but change the product ID that is being added on each request. This might allow them to automate the addition of products to a shopping cart, after solving only one captcha

challenge. The captcha mechanism does not allow the modification of query parameters after the original request has been submitted, so this type of activity will not be successful.

Response Processors: Request Captcha Processor: Incident - Captcha Request Replay Attack

Complexity: Suspicious (1.0)

Default Response: 5x = Multiple Captcha Replay Incident

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user attempts to submit a captcha solution multiple times and "replay" is explicitly disabled for the captcha being used.

Behavior: Because captcha's prevent automation, attackers will sometimes try and find ways to abuse the technique used to request the captcha in order to exploit the site. For example, if the attacker can find a way to submit the same solution over and over again, they might be able to solve the captcha once and still automate the resulting workflow. This is sometimes considered legitimate behavior (as would be expected if the user refreshed the browser after submitting a successful captcha), however in many cases, such functionality would make the captcha significantly less effective at preventing automation. In this case, the attacker resubmitted a request that had already been successfully validated through a captcha, and "replay" was explicitly disabled for the captcha. This is not necessarily a malicious incident on its own, because the user can have accidentally refreshed the browser, however multiple attempts would definitely represent malicious intent. An example of where a captcha's "replay" could cause a problem is on a gaming site, where the user is adding fake "money" to their account. In order to add the fake money, they must solve the captcha. This workflow is protected with a captcha, because if a user could automate the process, they would be able to add unlimited funds to their account. If an attacker were able to solve the captcha once, and continuously resubmit the resulting request, they could effectively add funds over and over again without resolving a new captcha. This would then allow for automation. Replay attackers are less of a problem if the web application being protected already has a method of preventing the same request from being submitted accidentally multiple times. Such would be the case if the web application maintained state information for

the given session, and recorded the operation after it was successful, then used that state information to prevent a future occurrence of the operation.

Response Processors: Request Captcha Processor: Incident - Multiple Captcha Replays

Complexity: Low (2.0)

Default Response: 1x = Warn User, 2x = 1 Day Clear Inputs

Cause: A captcha is a special technique used to differentiate between human users, and automated scripts. This is done through a Turing test, where the user is required to visually identify characters in a jumbled image and transcribe them into an input. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. Additionally, an audio version is optionally available to allow users who have a visual handicap to complete the captcha successfully. Captchas are used in two different ways by the system. They can be explicitly added to any workflow within the protected web application (such as requiring a captcha to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). Captchas are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of captcha is being used, this incident is generated when the user attempts to submit a captcha solution multiple times and "replay" is explicitly disabled for the captcha being used.

Behavior: Because captcha's prevent automation, attackers will sometimes try and find ways to abuse the technique used to request the captcha in order to exploit the site. For example, if the attacker can find a way to submit the same solution over and over again, they might be able to solve the captcha once and still automate the resulting workflow. This is sometimes considered legitimate behavior (as would be expected if the user refreshed the browser after submitting a successful captcha), however in many cases, such functionality would make the captcha significantly less effective at preventing automation. In this case, the attacker resubmitted a request that had already been successfully validated through a captcha, and "replay" was explicitly disabled for the captcha. This is not necessarily a malicious incident on its own, because the user can have accidentally refreshed the browser, however multiple attempts would definitely represent malicious intent. An example of where a captcha's "replay" could cause a problem is on a gaming site, where the user is adding fake "money" to their account. In order to add the fake money, they must solve the captcha. This workflow is protected with a captcha, because if a user could automate the process, they would be able to add unlimited funds to their account. If an attacker were able to solve the captcha once, and continuously resubmit the resulting request, they could effectively add funds over and over again without resolving a new captcha. This would then allow for automation. Replay attackers are less of a problem if the web application being protected already has a method of preventing the same request from being submitted accidentally multiple times. Such would be the case if the web application maintained state information for the given session, and recorded the operation after it was successful, then used that state information to prevent a future occurrence of the operation.

Response Processors: Request Captcha Processor: Incident - Multiple Captcha Disallow Multipart

Complexity: Low (2.0)

Default Response: 1x = 1 Day Clear Inputs

Cause: A CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a special technique used to differentiate between human users, and automated scripts. The user is required to visually identify characters in a jumbled image and transcribe them into a text box. An audio version is also available, for users with a visual handicap. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. CAPTCHAs are used in two different ways by the System. They can be explicitly added to any workflow within the protected web application (such as requiring a CAPTCHA to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). CAPTCHAs are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of CAPTCHA is being used, this incident is generated when the user attempts to submit dozens of CAPTCHA-protected requests that contain binary files, and the CAPTCHAs are explicitly configured to not allow binary file submission (it has been configured to disallow multi-part form submissions).

Behavior: When a hacker is attempting to establish an automated script that is capable of defeating the CAPTCHA, they can use various techniques. One of these techniques is to try changing various values used by the web application in the CAPTCHA mechanism in an effort to see if an error can be generated, or an unexpected outcome can be achieved. This type of probing and reverse-engineering is generally performed by advanced hackers. In this specific case, the attacker submitted dozens of binary files in the requests that are being protected. The CAPTCHA in this case has been explicitly configured to not allow Multi-Part form submissions, so this represents unexpected and undesired activity. Using Multi-Part forms, the attacker can more easily accomplish a "Buffer Overflow" attack, which would produce potentially sensitive error data and possibly open the server up to further exploitation. Additionally, some web applications do not handle the encoding used for multi-part forms gracefully, so error information can also be obtained from conflicts arising from the submission type. Because this is happening so frequently from the same user, it is also possible that the user is attempting to execute a "Denial of Service" attack.

Response Processors: Request Captcha Processor: Incident - Multiple Captcha Parameter Manipulation

Complexity: Low (2.0)

Default Response: 1x = Warn User, 2x = 1 Day Clear Inputs

Cause: A CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a special technique used to differentiate between human users, and automated scripts. The user is required to visually identify characters in a jumbled image and transcribe them into a text box. An audio version is also available, for users with a visual handicap. If the user is unable to complete the challenge in a reasonable amount of time, they are not allowed to proceed with their original request. Because it is nearly-impossible to script the deciphering of the image, automated scripts generally get stuck and cannot proceed. CAPTCHAs are used in two different ways by the System. They can be explicitly added to any workflow within the protected web application (such as requiring a CAPTCHA to login, or checkout a shopping cart), and they can be used to test a suspicious user before allowing them to continue using the site (similar to blocking the user, but with a way for the user to unblock themselves if they can prove they are not an automated script). CAPTCHAs are generally used to resolve "Insufficient Anti-Automation" weaknesses in the protected web application. Regardless of which type of CAPTCHA is being used, this incident is generated when the user attempts to submit multiple solutions for multiple CAPTCHAs, but they keep modifying the query parameters that were submitted with the original requests. For example, if the user submitted a "add product to cart" request, and one of the query parameters was the item to add, this incident would be triggered if, after solving the CAPTCHA, the value of that query parameter was modified to some other value, and this modification happened dozens of times.

Behavior: Because CAPTCHAs prevent automation, attackers will sometimes try to find ways to abuse the technique used to request the CAPTCHA in order to exploit the site. For example, if the attacker can find a way to submit the same solution over and over again, but have the web application perform a different action each time, they might be able to solve the CAPTCHA once, and still automate the resulting workflow. In this case, the attacker changed many query parameters on many different requests that were protected with a CAPTCHA. They submitted the original request, solved the CAPTCHA, changed the original query parameters, and then resubmitted the solved CAPTCHA request. In some cases, this might cause the web application to execute a different operation based on the difference in query parameter values. For example, if the protected workflow is "add product to cart" on a shopping site, then the attacker might attempt to submit the same solved CAPTCHA repeatedly, but change the product ID that is being added on each request. This might allow them to automate the addition of products to a shopping cart, after solving only one CAPTCHA challenge. The CAPTCHA mechanism does not allow the modification of query parameters after the original request has been submitted, so this type of activity will not be successful. This is not considered malicious activity right away, because it is possible that a user can accidentally modify a query parameter; however, when this incident is triggered, it represents a user who has modified dozens of different query parameters on different CAPTCHA-protected pages.

Response Processors: CSRF Processor

The CSRF processor is responsible for ensuring that the protected website does not allow a cross site request forgery attack. CSRF attacks are a type of session hijacking, where a malicious website redirects a user to a sensitive service call on the target website. For example, a user might visit a malicious website that has an image tag pointed to the "deleteAccount" service running on a target website. When a user visits the malicious

website, they are unknowingly calling the "deleteAccount" operation. If they had an active session on the target site, their account would be deleted.

This processor works by intercepting any request that could potentially be part of a CSRF attack. This is determined by looking at the referer header being passed in by the client. The referer header tells the server where the user came from. If the user is navigating around the actual website legitimately, they will have a referer header on nearly all requests they make which will match the domain of the site they are navigating. If the user types the URL in manually, or follows a link from another site, they will not have a referer. If it's a CSRF attack, there will either be no referer, or a third party domain in the referer.

In all cases where the referer does not match the domain of the protected site, a special redirection page will be returned to the client instead of the request they actually asked for. The redirection page will check to make sure the user is not a victim of a CSRF attack, and if they are not, it will automatically redirect the user to the original page they requested.

This processor only protects clients that have "user-agent" headers matching that of a known browser. This is because CSRF attacks are specifically targeted at average web users, and they generally stick to the major browsers. So spiders and scripts will bypass the CSRF processors detection/protection mechanism. This processor also detects the case where a user has turned off referers (and thus, no requests will contain a referer), and in that case, will turn off CSRF protection for the client. As such, a user who has disabled referers will still be susceptible to CSRF, but that should be a very small percentage (if not zero) of the overall user pool.

In the event that a user issues a request that cannot be validated as not a CSRF attack, the user will not be automatically redirected. Instead, they will be presented a "This page has moved" response, and will be asked to click a link to continue to the page they actually wanted. The link to proceed is randomly positioned on the page to prevent Click Jacking attacks (where a malicious site overlays legitimate content on top of the target site and gets the user to click the legitimate content, while also hijacking the click to transparently activate the content underneath). A special case involves when a third party website opens the target site in a new window or tab. If the third party site retains ownership of the newly opened window or tab, the user will be asked to click the "continue" link so that the original window can be closed and a new window can be opened in its place. This action breaks the ownership and prevents the third party website from performing actions on the window (such as closing or redirecting it).

Because it is sometimes expected that a third party site will be making calls into the target site, it is possible to configure a list of "trusted" third party sites. Any requests issued from a trusted domain will not be protected against CSRF. This allows the trusted site to host the target site in an IFRAME or make service calls unimpeded. Be careful who you add to the trusted domain list, because if the trusted domain is susceptible to XSS or CSRF itself, then it can be used as a proxy to launch a CSRF attack against your protected sites. This trust does not apply if the hosting domain is running over SSL, and the target domain is not running over SSL. If the third party page hosting an IFRAME of the target site is running in SSL, it must load the SSL version of the target site, otherwise

the CSRF protection will still be applied. It is however fine if the third party site is not SSL protected and the target site is SSL protected.

Table 32: CSRF Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Block Response	Configurable	HTTP Response	The response to return if the CSRF mechanism cannot complete the request due to errors or tampering.
CSRF Nonce Salt	String	Random	A 256 character random string used to ensure that CSRF nonce tokens are generated differently between different deployments.
CSRF Token Name	String	Random	The name of the query string parameter used to indicate a successfully validated request after it has been determined that it is not a CSRF attack. Select a name that will not conflict with a real query parameter used by the site.
Ignore Scripts	Boolean	True	CSRF is largely a browser based attack, so to ensure that scripts such as legitimate spiders are not treated as potential CSRF victims, this option can be enabled to ignore all non browsers for CSRF protection.
Ignored Extensions	Collection	.xap, .xaml	A list of file suffixes (extensions) that will not be protected by CSRF. By default, Silverlight binaries are included, because some browsers will remove the referer for Silverlight embedded content, which can interfere with CSRF protection and prevent the Silverlight content from loading.
Remote Script Resource	Configurable	CSRF Script Inclusion Resource	The fake resource to request if the page is being loaded as a remote script on a third party domain. This is primarily for detection of the attack and can be any fake resources as long as it does not actually exist on the server.
Trusted Domains	Collection	None	The list of domains that are allowed to display the web application in a frame, reference resources such as images or scripts, or are allowed to make remote API calls using techniques that are similar to a CSRF attack. If the trusted domain starts with a period, then it will match any subdomain before the designated period. For example, .site.com will match www.site.com, my.new.site.com and site.com.
CSRF Extra JavaScript	String	None	Since CSRF protection can cause the referer to be removed from the request, it can be necessary to add any analytic code to the JavaScript used to detect and stop CSRF attacks. As such, if you use a third party analytics script, you should put that code in this parameter to capture the unmodified original request details. The code will be injected into a script tag, so it must be valid JavaScript or the CSRF protection can stop functioning correctly.

Table 32: CSRF Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Incident: CSRF Parameter Tampering	Boolean	True	The user tampered with the parameters used by the security engine to prevent CSRF on requests that have an untrusted third party referer. This is likely in an attempt to find a vulnerability in the CSRF protection mechanism.
Incident: CSRF Remote Script Inclusion	Boolean	False	The user has accessed an untrusted third party website which contains an embedded script reference to the protected application. While the user might not be malicious, this represents a CSRF attack from the untrusted website against the protected application. Because the attack was not successful, it is likely being executed by the user who is attempting to construct the attack vector.
Incident: HTTP Referers Disabled	Boolean	True	The user is using what looks like a browser, but they have HTTP referers disabled. This is not a malicious incident, but it does indicate an unusual client.

Response Processors: CSRF Processor: Incident - CSRF Parameter Tampering

Complexity: Suspicious (1.0)

Default Response: 10x = Multiple CSRF Parameter Tampering Incident.

Cause: WebApp Secure protects against CSRF attacks by using a special interception technique. When a request comes in to WebApp Secure, the referer is checked. In the event that there is a third party referer (the user was following a link from another site), the interception mechanism kicks in. This involves returning a special page to the user that validates that the user is intentionally requesting the resource. If the validation is successful, the user is transparently redirected to the original resource they requested. If the validation fails, the user is then instructed to manually confirm their intentions, or return to the page they came from (to prevent the CSRF attack from working). In most cases, a valid CSRF attack would function in such a way as to hide this manual confirmation step, so the user would probably never see it (for example if the URL was loaded using an image HTML tag, then the resulting HTML confirmation step would not render, because its HTML, not an image). This incident is triggered when a user submits a request with a third party referer, and then manipulates the code of the CSRF interception page to alter the original data that was submitted. For example, they submit a request that looks like a CSRF attack (has a third party referer), and then use a tool like Firebug to edit the query string parameters that would be sent to the server after they manually allowed the request on the CSRF intercept page.

Behavior: CSRF attacks are generally two-phase. The first phase involves the attacker establishing a functional CSRF attack. This could take quite a while and involves the attacker making requests to the protected site, trying all different types of CSRF techniques. The second phase is when the attacker injects the successful CSRF vector into a public website. In the second phase, legitimate users are visiting the public website and unknowingly executing the CSRF attack in the background. It is not useful to flag the victims of the CSRF attack as hackers, because they might not even know what is going

on. However it is useful to flag the original attack vector establishment, because it can shed light on who created the "CSRF243" attack. This incident reflects a user who is manipulating the CSRF prevention mechanism, likely in an attempt to find a way to get around it. As such, if a user has this incident, they are probably trying to establish a CSRF attack, and careful attention should be paid to the values they are changing the parameters to and which URL is being requested (this will help identify what the user is trying to attack).

Response Processors: CSRF Processor: Incident - Multiple CSRF Parameter Tampering

Complexity: Low (2.0)

Default Response: 1x = Captcha, 2x = 1 Day Clear Inputs

Cause: WebApp Secure protects against CSRF attacks by using a special interception technique. When a request comes in to WebApp Secure, the referer is checked. In the event that there is a third party referer (the user was following a link from another site), the interception mechanism kicks in. This involves returning a special page to the user that validates that the user is intentionally requesting the resource. If the validation is successful, the user is transparently redirected to the original resource they requested. If the validation fails, the user is then instructed to manually confirm their intentions, or return to the page they came from (to prevent the CSRF attack from working). In most cases, a valid CSRF attack would function in such a way as to hide this manual confirmation step, so the user would probably never see it (for example if the URL was loaded using an image HTML tag, then the resulting HTML confirmation step would not render, because its HTML, not an image). This incident is triggered when a user submits dozens of requests with a third party referers, and then manipulates the code of the CSRF interception page to alter the original data that was submitted. For example, they submit a bunch of requests that look like CSRF attacks (they have third party referers), and then use a tool like Firebug to edit the query string parameters that would be sent to the server after they manually allowed the requests on the CSRF intercept page.

Behavior: CSRF attacks are generally two-phase. The first phase involves the attacker establishing a functional CSRF attack. This could take quite a while and involves the attacker making requests to the protected site, trying all different types of CSRF techniques. The second phase is when the attacker injects the successful CSRF vector into a public website. In the second phase, legitimate users are visiting the public website and unknowingly executing the CSRF attack in the background. It is not useful to flag the victims of the CSRF attack as hackers, because they might not even know what is going on. However it is useful to flag the original attack vector establishment, because it can shed light on who created the "CSRF" attack. This incident reflects a user who is manipulating the CSRF prevention mechanism, likely in an attempt to find a way to get around it. As such, if a user has this incident, they are probably trying to establish a CSRF attack, and careful attention should be paid to the values they are changing the parameters to and which URL is being requested (this will help identify what the user is trying to attack).

Response Processors: CSRF Processor: Incident - CSRF Remote Script Inclusion

Complexity: Informational (0.0)

Default Response: None.

Cause: WebApp Secure protects against CSRF attacks by using a special interception technique. When a request comes in to WebApp Secure, the referer is checked. In the event that there is a third party referer (the user was following a link from another site), the interception mechanism kicks in. This involves returning a special page to the user that validates that the user is intentionally requesting the resource. If the validation is successful, the user is transparently redirected to the original resource they requested. If the validation fails, the user is then instructed to manually confirm their intentions, or return to the page they came from (to prevent the CSRF attack from working). In most cases, a valid CSRF attack would function in such a way as to hide this manual confirmation step, so the user would probably never see it (for example if the URL was loaded using an image HTML tag, then the resulting HTML confirmation step would not render, because its HTML, not an image). This incident is triggered when a user accesses a page on a third party website which contains a Javascript tag that loads content from the protected site. This would normally represent a victim of a CSRF attack, but because CSRF attacks are blocked, an attacker is unlikely to execute such an attack. Therefore, it is more probable that the attacker is testing a possible vector to see if it will work and encountering this incident.

Behavior: CSRF attacks are generally two-phase. The first phase involves the attacker establishing a functional CSRF attack. This could take quite a while and involves the attacker making requests to the protected site, trying all different types of CSRF techniques. The second phase is when the attacker injects the successful CSRF vector into a public website. In the second phase, legitimate users are visiting the public website and unknowingly executing the CSRF attack in the background. It is not useful to flag the victims of the CSRF attack as hackers, because they might not even know what is going on. However it is useful to flag the original attack vector establishment, because it can shed light on who created the "CSRF" attack. While this incident would potentially be fired for any victims of a CSRF attack, CSRF attacks are blocked by this processor, so it is unlikely that an attacker would ever actually try to use the vector against legitimate users. As such, it is far more likely that the attacker is still in the first phase and trying to uncover a successful CSRF vector. Because of this, careful attention should be paid to the URL that is being requested (this will help identify what the user is trying to exploit).

Response Processors: CSRF Processor: Incident - HTTP Referers Disabled

Complexity: Suspicious (1.0)

Default Response: None.

Cause: The HTTP protocol provides support for a special header called the "referer" (misspelled on purpose). This header tells the webserver where the user just came from. So if the user visits google and follows a link from google to get to another page, the request for that second page will contain a "referer" of "http://www.google.com". Some browsers provide the option to turn off automatic transmission of the "referer" header. This would make it impossible for websites to identify the page the user came from. This incident is triggered whenever a user accesses the website with referers disabled. This is not necessarily a malicious act, as it could be the result of an excessively paranoid

legitimate user, but it is also somewhat unusual and is often a technique employed by malicious users.

Behavior: Hackers will often disable the referer header to make it more difficult to monitor and analyze an attack through the traditional HTTP log files. Many web servers will record the URL the user is accessing, as well as the referer that was submitted. As such, by disabling referers, the hacker is able to eliminate a large percentage of the information collected about the attack.

Response Processors: Header Injection Processor

This processor provides the header injection counter response. It allows extra a custom header to be defined that is injected into a suspected hackers requests to allow custom handling.



NOTE: There are no actual triggers for this processor; it is a form of response.

Table 33: Header Injection Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Default Header Name	String	Random	The default header name to use if one is not specified in the response configuration.
Default Header Value	String	True	The default header value to use if one is not specified in the response configuration.

Response Processors: Force Logout Processor

This processor provides the force logout counter response. It strips out and invalidates the users session tokens logging them out of the site.



NOTE: There are no actual triggers for this processor - it is a form of response.

Table 34: Force Logout Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.

Table 34: Force Logout Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Application Session Cookie	Collection	Collection	A collection of names to use for the Application session cookie.
Advanced			
Clear Session Cookies	Boolean	False	Whether to clear any terminated session cookies from the malicious users browser. This can help the user identify why they are getting logged off, so unless the application has code on the client that reads the session cookie value, or the cookie is used in traffic not protected by the WebApp Secure system, this option should be turned off.

Response Processors: Strip Inputs Processor

This processor is used to transparently remove all user input from requests being issued to the server. This response will make the web application, or the client accessing it, to appear broken from the users perspective. The website will also take on a much smaller attack surface should the client be a vulnerability scanner.



NOTE: There are no actual triggers for this processor; it is a form of response.

Table 35: Strip Inputs Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.

Response Processors: Slow Connection Processor

The slow connection processor is designed to introduce large delays in requests issued by malicious traffic without impacting the performance of legitimate users. There are no actual triggers for this processor; it is a form of response.



NOTE: If default minimum and maximum delay times for the Slow Connection Processor are set to a value greater than the Backend Response Timeout (**Configuration > Proxy/Backends > Connection Timeout**), the connection can timeout, resulting in a 403 error.

Table 36: Slow Connection Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Default Maximum Delay	Integer	5 Seconds	The default maximum number of milliseconds to delay malicious requests.
Default Minimum Delay	Integer	500 Milliseconds	The default minimum number of milliseconds to delay malicious requests.

Response Processors: Warning Processor

The warning processor is designed to allow a warning message to be presented to a user without completely blocking site access. The warning processor only enables the ability to respond to a user with a "warning", which would allow them to continue browsing the page and the site. The warning would be created and activated for a user by the auto response system, or manually from the console. The existing processor overlays semi-transparent HTML elements on top of the entire webpage, which temporarily disables any mouse or keystrokes on the page and, therefore, creating a "modal dialog" effect. This processor isn't designed to completely stop an attacker from using the website; it is there to warn them. Given the browser debugging tools available today, an attacker might be able to dismiss the warning by means of such tools. Any tampering with the warning's default dismissal behavior (waiting 5 seconds until dismissal button is automatically enabled and clicking on dismiss button) will be considered an incident and will be tracked.

Table 37: Warning Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Default Warning Message	String	"Your connection has been detected performing suspicious activity. Your traffic is now being monitored."	The default message to use in the warning dialog. This can be defined on a session by session basis, but if no explicit value is assigned to the warning, this value will be used.
Default Warning Title	String	Security Warning	The default title to use in the warning dialog. This can be defined on a session by session basis, but if no explicit value is assigned to the warning, this value will be used.

Table 37: Warning Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Dismissal Delay	Integer	10 Seconds	The amount of time in seconds that must elapse before the warning can be dismissed. This is a soft limit, as an experienced user might be able to get around enforcement measures.
Dismissal Resource	Configurable	Random	The information needed to define the URL and response used to dismiss a warning.
Warning Directory	String	Random	The name of the directory where the warning Javascript and css code will be served from. For example: warningcode.
Incident: Warning Code Tampering	Boolean	True	The user has attempted to dismiss the warning without waiting the delay and using the provided mechanism. This is probably an attack on the warning system.

Response Processors: Warning Processor: Incident - Warning Code Tampering

Complexity: Medium (3.0)

Default Response: 1x = Logout User, 2x = 5 Day Clear Inputs.

Cause: WebApp Secure is capable of issuing non blocking warning messages to potentially malicious users. These warning messages are designed to force the user to wait for a period of time, before they can dismiss the warning and continue using the site. If the user attempts to exploit or bypass this delay mechanism in order to dismiss the warning early, this incident will be triggered.

Behavior: Once a hacker has been warned, they are then aware that a security system is monitoring their activity. This can cause some hackers to investigate what might be protecting the site. This could involve additional scanning, or it could involve attacking the warning mechanism directly. This type of behavior generally indicates a hacker with moderate to advanced skill levels. Depending on what they modify the warning code input to be, this could represent a simple exploratory test, or the user could be trying to launch a more complex attack against the warning code handler itself, such as "Buffer Overflow", "XSS", "Denial of Service", "Fingerprinting", "Format String", "HTTP Response Splitting", "Integer Overflow", and "SQL injection" among many others.

Response Processors: Application Vulnerability Processor

The application vulnerability processor is designed to block known attack vectors for select third party applications. By default this processor does nothing. If you host a third party application such as WordPress, you should enable the configuration parameters that represent the third party software you are using. This will enable protection for that software component.

Table 38: Application Vulnerability Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Joomla Vulnerability Protection Enabled	Boolean	False	Whether traffic should be analyzed for Joomla vulnerabilities
PHPBB Vulnerability Protection Enabled	Boolean	False	Whether traffic should be analyzed for PHPBB vulnerabilities
Wordpress Vulnerability Protection Enabled	Boolean	False	Whether traffic should be analyzed for Wordpress vulnerabilities
Advanced			
Mode of Operation	Integer	1	Whether to block a request on a positive signature, or just create an incident
Block Response	HTTP Response	404 Error	The default message to use in the warning dialog. This can be defined on a session by session basis, but if no explicit value is assigned to the warning, this value will be used.

Response Processors: Application Vulnerability Processor: Incident - App Vulnerability Detected

Complexity: Low (2.0)

Default Response: 1x = Slow Connection 2-6 seconds, 3x = Slow Connection 2-6 seconds and Clear Inputs for 1 day

Cause: The application vulnerability processor is designed to identify known attack vectors issued to third party applications such as WordPress. This incident indicates that one of those known attack vectors has been issued by the associated user. The exact nature of the vector that was identified should be described in the incident details.

Behavior: One of the easiest ways to compromise a website is to look for third party web applications such as WordPress. If one is found, the attacker can then look up any known vulnerabilities in that software and the version of it that is running on the website. If they find vulnerabilities, they can then launch them and potentially compromise the site with a few minutes with minimal effort.

Response Processors: Support Processor

When a user is blocked or otherwise responded to using one of the countermeasures, this processor provides a way to identify which profile is associated with a user, and to then allow those responses to be deactivated at the discretion of the IT administrator. For example, if a user were to get a 404 error when asking for a PDF document linked from the main site, and they then try to find the file by trying a bunch of different file

names, they can eventually get blocked for performing a directory enumeration attack. When this happens, the blocked user can contact support for assistance getting access to the site again.

This processor works by exposing a special administrative URL (defined in configuration) which the support team can access. When a support request comes in from a blocked users, the support representative can access this administrative URL which will provide another URL. The support representative should then provide this second URL to the affected user. The affected user can then visit that URL and get a special code. This code can be used to search for the profile and deactivate responses in the Web UI (profile list).

If the affected user gets a code of "00000000000000000000" (all zeros), this means that the user is not identified as an attacker and therefore is not being blocked or responded to with a counter response from WebApp Secure. As such, other causes of the user's inability to access the site should be investigated.

DO NOT GIVE OUT THE ADMINISTRATIVE URL. It is only used to get a fresh URL that is safe to provide to the affected user. If the administrative URL is leaked to the public, it should be changed immediately.

The overall workflow is as follows:

1. User is blocked or otherwise responded to with a countermeasure.
2. User calls support for assistance.
3. Support accesses the administrative URL.
4. Support copies the newly created URL in the response and provides to the affected user.
5. The affected user accesses the newly created URL and provides the resulting code to support.
6. Support or an Admin then logs into the Web UI, clicks on the profile graph to get a list of profiles, and then searches for the code.
7. Support or Admin reviews user's list of incidents to verify the user was responded to in error. If so, the Support or Admin disables the responses.



NOTE: Note that the "block" response is by default, configured to return the code. So if a user has been blocked, steps 3-5 can be omitted, and the user can simply provide the code specified in the block message to support. For all other responses, the full workflow needs to be followed, because there is no other way to obtain the code.

Table 39: Support Processor Configuration Parameters

Parameter	Type	Default Value	Description
-----------	------	---------------	-------------

Basic

Table 39: Support Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor.
Advanced			
Private Support URL	String	Random	The URL a support representative would access to get additional details about how to provide support to users who are having issues that can be WebApp Secure related. If the value is "ABC", then the private URL would be <code>http://www.example.com/ABC</code> . It is absolutely imperative that this URL not be leaked to non-internal users. If it is leaked, it must be changed immediately.
Public Support URL Salt	String	Random	A random value used to ensure that support URLs are not predictable. This can be any random string 30 characters in length.
Public URL Expiration	Integer	3	The number of days a public support URL remains valid for. After this many days, the URL will no longer provide support information. This is to prevent any issues from a public support URL being leaked.

Response Processors: Cloppy Processor

The Cloppy processor is a joke response built for demonstration purposes. It creates an animated paperclip in the lower right corner of the website, which belittles and taunts the attacker. This should never be used on a legitimate threat and is not the default counter response for any type of behavior. It is provided to demonstrate the diversity of counter responses WebApp Secure is capable of. You should never activate this response unless you have a good relationship with the user you are activating it on, and they have a good sense of humor.

You can configure the message and options cloppy presents both in configuration (the default messages), or in the response specific config (the XML you define when you manually activate a response or when you write a rule that activates a response). The oldest cloppy response will be the one for which the messages are loaded, so if you create multiple cloppy responses, you can create a dialog of several messages. For example, try activating cloppy three times with the following config values (create them in the following order):

1. Activate Cloppy: `<config message="This is the first message"><option label="First op" url="" /><option label="Second op" url="" /></config>`
2. Activate Cloppy: `<config message="This is the second message"><option label="First op" url="" /><option label="Second op" url="" /></config>`
3. Activate Cloppy: `<config message="This is the third message"><option label="First op" url="" /><option label="Second op" url="" /></config>`

Once you activate the above 3 cloppy responses, you should see that cloppy will present the "This is the first message" dialog first. Once you click on an option in that dialog, the

next page you load will display "This is the second message", and finally, after clicking on one of those options, you should get "This is the third message".

Once you click an option in the cloppy's dialog, it will dismiss that specific cloppy response. That's why you are able to stack the responses and get a dialog going.

Table 40: Cloppy Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor. Note that just because traffic is passing through the processor, does not mean any users will actually have a Cloppy response activated on them. As such, simply enabling this processor will not result in cloppy being activated for any users. You would still need to manually activate the Cloppy response in the Web UI (or define an auto response rule that activates it, but that is highly discouraged).
Cloppy Message	String	"It looks like you're an unsophisticated script kiddie attempting to hack this website"	What do you want cloppy to say when offering help?
Cloppy Options	Collection	Collection	The list of ways cloppy can help with associated URLs.
Advanced			
Cloppy Directory	String	cloppybin	The name of the directory where the binary resources needed to load cloppy are served from. For example: cloppyfiles. The name should be selected not to conflict with a real directory at the top level of the website.
Cloppy Dismiss Directory	String	Random	The name of the directory used to dismiss cloppy. This URL should be random and not conflict with existing directory names on the site.

Response Processors: Login Processor

The login processor is designed to add additional protection to the login dialogs throughout the protected site. By default, it will not provide any additional protection, and must be configured to protect specific login forms. Once a login form has been configured, the processor will begin to monitor the login attempts and start checking for abusive patterns.

This processor is capable of detecting a wide variety of abuse patterns on a login dialog, as well as stopping these abusive activities. One key protection mechanism is to require a captcha if a user attempts to login to an account which has experienced more than 3 failed login attempts since the last successful login attempt. This ensures that a malicious user cannot brute force a specific username, because after 3 failed attempts, the brute force tool will be stopped by a captcha. This does not represent a counter response, but instead is built in functionality that applies to all users on the system. So if user "A"

submits 3 bad passwords, and then user "B" submits a password for the same username, user "B" will get a captcha, as well as user "A" for any additional login attempts they try. As soon as a user successfully logs into the account, it will take another 3 failed login attempts before the next captcha is required.

In addition to protecting against a single username being attacked with a brute force script, the processor also detects "User sharing", "User pooling", "Username scans", "Multi-User brute force scans". See the incident descriptions for more information on what these incidents represent and what counter responses will be activated as a result. In order to configure the Login Processor to protect a login form, edit the "Protected Login Pages" configuration parameter. Add a new row and provide the following information. It will be useful to look at the HTML source code of the login form as it will have critical information you will need to configure protection:

- **Name:** The name of the login page (this is just for your reference, it can be anything)
- **URL Pattern:** The Regular Expression used to identify a username/password submission. This pattern should match the "action" attribute of the HTML <FORM> tag wrapping the login dialog.
- **Username Field Type:** The type of inputs used to submit a username. Normally this will be "POST Parameter", however other options are provided for more specialized login mechanisms.
- **Username Field Name Pattern:** A regular expression used to match the name of the input the username is submitted with. Normally this is "username", but could be other variations such as "usr", "user", and so on. You can simply enter the name of the input in this field if a regular expression is not required.
- **Username Field Value Pattern:** A regular expression used to extract the username from the input value. Normally this should just be "^.*\$", but if the username is wrapped in JSON for example, you might need to create a more complex expression. The username is considered the first matching parenthesis group in the pattern.
- **Username Field Encoding:** The type of data encoding used on the username. Normally this will be "Ascii", however if any client side encoding is performed, other encoding options are available.
- **Password Field Type:** The type of inputs used to submit a password. Normally this will be "POST Parameter", however other options are provided for more specialized login mechanisms.
- **Password Field Name Pattern:** A regular expression used to match the name of the input the password is submitted with. Normally this is "password", but could be other variations such as "pwd", "pass", and so on. You can simply enter the name of the input in this field if a regular expression is not required.
- **Password Field Value Pattern:** A regular expression used to extract the password from the input value. Normally this should just be "^.*\$", but if the password is wrapped in JSON for example, you might need to create a more complex expression. The password is considered the first matching parenthesis group in the pattern.

- **Password Field Encoding:** The type of data encoding used on the password. Normally this will be "Ascii", however if any client side encoding is performed, other encoding options are available.
- **Failure Pattern Target:** In order to identify a failed login attempt, the processor will search for a specific pattern in the response. This attribute specifies where to search for that pattern. Normally this would be "Body" to search the HTML body of the response.
- **Failure Pattern:** The regular expression to search for to check and see if the login attempt was unsuccessful. Assuming the Failure Pattern Target is "Body", this would be something like "you have provided an invalid username and password". However the exact text will need to be set to whatever the site actually returns. View the source of the response after a failed login and search for the error text, so that you get the most accurate version possible. Simply copying the text from the rendered page can exclude embedded HTML tags which will cause the pattern to never match.
- **Failure Pattern Condition:** Specifies whether finding the failure pattern means the login was unsuccessful, or whether not finding the pattern means the login was unsuccessful.
- **Success Pattern Target:** In order to identify a successful login attempt, the processor will search for a specific pattern in the response. This attribute specifies where to search for that pattern. Normally this would be "Body" to search the HTML body of the response.
- **Success Pattern:** The regular expression to search for to check and see if the login attempt was successful. Assuming the Success Pattern Target is "Body", this would be something like "you have successfully logged in". However the exact text will need to be set to whatever the site actually returns. View the source of the response after a successful login and search for something that only gets displayed on a successful login, so that you get the most accurate version possible. Simply copying the text from the rendered page can exclude embedded HTML tags which will cause the pattern to never match.
- **Success Pattern Condition:** Specifies whether finding the success pattern means the login was successful, or whether not finding the pattern means the login was successful.
- **Require Captcha After:** Specifies how many failed login attempts on the same username before requiring all future login attempts on that username to solve a captcha. Entering "0" will allow infinite attempts.

Keep in mind that some website implementations allow login information to be posted to many different URLs. If that is the case, make sure the URL pattern is defined generically enough to match any URL the user might submit a login request to. Only submissions that match the URL pattern will be protected.

Once a login form has been configured, it can be tested by attempting to login to the same username 4 or more times. You should be presented with a captcha. Next, solve the captcha and log in with the correct password. Then logout and attempt to login to the same username again. If you do not get a captcha, then the login form is configured correctly.

Table 41: Login Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	True	Whether traffic should be passed through this processor. Note that just because this processor is enabled, does not mean that any login forms are being protected. Login forms will not be protected until they are configured in the "Protected Login Pages" parameter.
Protected Login Pages	Collection	None	The list of pages that should be protected from login and account abuse. These pages should reflect the URL's that accept username's and passwords and allow login, not necessarily the pages that contain login forms. For example, if every page on the site had a login form, but they all submitted to login.php, then only login.php needs to be configured in this processor.
Advanced			
Bad Request Block Response	HTTP Response	400 Error	The response to return if the user issues a request that either is too large, or uses multipart and multi-part is disabled.
Blocked Replay Response	HTTP Response	400 Error	The response to return if the user attempts to submit the validated request multiple times using the same captcha answer, and that behavior is not allowed.
Cancel URL	String	(empty)	The URL to redirect the user to if they cancel the captcha. This should not be to the same domain, because the domain is being blocked using a captcha, and therefore, canceling would only redirect to a new captcha. An empty value will hide the cancel button,

Table 41: Login Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Captcha Binary Directory	String	Random	The name of the directory where captcha images and audio files will be served from. This should not conflict with any actual directories on the site.
Captcha Characters	String	abcdefghijklmnopqrstuvwxyz ABCEFGHJKLMNPQRTWXYZ 234678	The characters to use when generating a random captcha value. Avoid using characters that can be easily mixed up. This set of characters is case sensitive.
Captcha Expiration	Integer	120	The maximum number of seconds the user has to solve the captcha before the request is no longer possible.
Captcha State Cookie	String	Random	The name of the cookie to use to track the active captchas that have not yet been solved. The cookie is only served to the captcha binary directory.
Captcha Template	File	Default Template	The HTML template used to ask the user to complete a captcha. This template must contain specific key words in order to integrate properly. Please refer to the manual for more information.
Captcha Validation Input Name	String	Random	The name of the form input used to transmit the captcha validation key. This should be obscure so that users who have not been required to enter a captcha cannot supply bad values to this input to profile the system.
Expired Captcha Response	HTTP Response	400 Error	The response to return if the user submits a validated request after the captcha has expired. This can happen if the user refreshes the results of the captcha long after they have solved it.

Table 41: Login Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Maximum Active Captchas	Integer	7	The maximum number of captchas any given user can be solving at any given time. This limit can be overcome, but the majority of users will not be able to. This is primarily for performance, as the more active captchas that are allowed, the larger the state cookie becomes.
Maximum Request Size	Integer	524288 (512KB)	The maximum number of bytes in a request before it is considered not acceptable for captcha validation, and will be blocked.
Support Audio Version	Boolean	True	Whether an audio version of the captcha is provided to the user. This can be a requirement for accessibility, as vision impaired users would otherwise be unable to solve the captcha.
Watermark	String	%DOMAIN	The text to watermark the captcha with. This can be used to prevent the captcha from being used in a phishing attack. For example, an abuser would not be able to simply display the captcha on a different site and ask a user to solve it. The watermark would tip the user off that the captcha was not intended for the site they are visiting. Use %DOMAIN to use the domain name as the watermark.
Incident: Site Invalid Login	Boolean	True	The user has submitted an invalid username or password. This is just an informational incident and is used to identify more complex attacks. It is highly recommended that this incident not be disabled, as it can cause other incidents to no longer register.

Table 41: Login Processor Configuration Parameters (*continued*)

Parameter	Type	Default Value	Description
Incident: Site Login Multiple IP	Boolean	True	The user has submitted a valid username and password or an invalid username and password for an account that has recently been used by a different IP. This is just an informational incident and is used to identify more complex attacks. It is highly recommended that this incident not be disabled, as it can cause other incidents to no longer register.
Incident: Site Login Multiple Usernames	Boolean	True	The user has submitted a valid username and password for more than one account recently. This is just an informational incident and is used to identify more complex attacks. It is highly recommended that this incident not be disabled, as it can cause other incidents to no longer register.

Response Processors: Login Processor: Incident - Site Login Invalid

Complexity: Suspicious (1.0)

Default Response: 16x (3 or more bad passwords per username) = Site Login Brute Force, 16x (less than 3 bad passwords per username) = Site Login Username Scan, 9x (bad passwords for same username) = Site Login User Brute Force.

Cause: The login processor is designed to protect the login dialog of the website. It works by monitoring all login attempts and identifying suspicious and malicious events. This specific incident is triggered when a user attempts to login with an invalid username and password. This incident alone is not considered malicious, but is used to perform additional analysis and can be promoted to a malicious incident if an abusive pattern is identified (such as many invalid logins representing a brute force attack).

Behavior: This incident simply reflects the case where a user has entered bad login information. By itself, this cannot be considered malicious as it is extremely common for a legitimate user to accidentally type their information incorrectly, or to forget their password. As such, it is only an indication of possible abuse and requires additional analysis and data before it can be confirmed as malicious or acceptable.

Response Processors: Login Processor: Incident - Site Login Multiple IP

Complexity: Informational (0.0)

Default Response: 3x = Site Login User Sharing

Cause: The login processor is designed to protect the login dialog of the website. It works by monitoring all login attempts and identifying suspicious and malicious events. This specific incident is triggered when multiple clients successfully log into the same account. Depending on the nature of the protected site, this might be perfectly acceptable behavior, however on some sites this type of behavior can indicate abuse. This incident alone is not considered malicious, but is used to perform additional analysis and potentially promote the event as a malicious incident if an abusive pattern is identified. Note that invalid login attempts from different subnets can also trigger this incident.

Behavior: Many websites provide a way for users to authenticate so that their experience and data can be customized specifically for them. In the case of this incident, credentials for one of those accounts have been distributed to multiple clients and two or more of those clients are logging into the account. Unless the website expects users to share credentials, this would generally indicate a situation where the credentials for an account have been compromised and the account has been hijacked. Additional follow up might be required to recover the account (such as changing the password or locking the account until the actual owner contacts the administrators to resolve the issue).

Response Processors: Login Processor: Incident - Site Login Multiple Usernames

Complexity: Suspicious (1.0)

Default Response: 3x = Site Login User Pooling

Cause: The login processor is designed to protect the login dialog of the website. It works by monitoring all login attempts and identifying suspicious and malicious events. This specific incident is triggered when a single client successfully authenticates with multiple distinct usernames. This incident alone is not considered malicious, but is used to perform additional analysis and potentially promote the event as a malicious incident if an abusive pattern is identified.

Behavior: There are two possibilities for this incident. Firstly, a single user might have signed up for multiple accounts on the protected site, and they are simply using those accounts. On some sites, this alone would be considered malicious, while on other sites, this is considered perfectly acceptable. For example, an online e-mail provider can allow its users to sign up for multiple e-mail accounts. On the other hand, a billing website for your home utility provider would probably not expect a single household to have multiple accounts. The other possibility is that a single user has hijacked several other accounts. This can be more obvious if there is also a "Site Login User Sharing" incident for the username as well. This would indicate that not only is the malicious user logging into multiple accounts, but other users are also logging into those accounts. Generally, an account should be used by a single user unless the website has specific rules about allowing users to share account details.

Response Processors: Login Processor: Incident - Site Login User Sharing

Complexity: Low (2.0)

Default Response: None.

Cause: The login processor is designed to protect the login dialog of the website. It works by monitoring all login attempts and identifying suspicious and malicious events. This specific incident is triggered when multiple clients successfully log into the same account. Depending on the nature of the protected site, this can be perfectly acceptable behavior, however on some sites this type of behavior can indicate abuse.

Behavior: Many websites provide a way for users to authenticate so that their experience and data can be customized specifically for them. In the case of this incident, credentials for one of those accounts have been distributed to multiple clients and two or more of those clients are logging into the account. Unless the website expects users to share credentials, this would generally indicate a situation where the credentials for an account have been compromised and the account has been hijacked. Additional follow up can be required to recover the account (such as changing the password or locking the account until the actual owner contacts the administrators to resolve the issue).

Response Processors: Login Processor: Incident - Site Login User Pooling

Complexity: Low (2.0)

Default Response: None.

Cause: The login processor is designed to protect the login dialog of the website. It works by monitoring all login attempts and identifying suspicious and malicious events. This specific incident is triggered when a single client successfully logs into several different accounts. Depending on the nature of the protected site, this can be perfectly acceptable behavior, however on some sites this type of behavior can be harmful.

Behavior: There are two possibilities for this incident. Firstly, a single user might have signed up for multiple accounts on the protected site, and they are simply using those accounts. On some sites, this alone would be considered malicious, while on other sites, this is considered perfectly acceptable. For example, an online e-mail provider can allow its users to sign up for multiple e-mail accounts. On the other hand, a billing website for your home utility provider would probably not expect a single household to have multiple accounts. The other possibility is that a single user has hijacked several other accounts. This can be more obvious if there is also a "Site Login User Sharing" incident for the username as well. This would indicate that not only is the malicious user logging into multiple accounts, but other users are also logging into those accounts. Generally, an account should be used by a single user unless the website has specific rules about allowing users to share account details.

Response Processors: Login Processor: Incident - Site Login User Brute Force

Complexity: Medium (3.0)

Default Response: 1x = Break Authentication for 1 hour, 2x = Break Authentication for 6 hours, 3x = Clear Inputs for 1 day

Cause: The login processor is designed to protect the login dialog of the website. It works by monitoring all login attempts and identifying suspicious and malicious events. This specific incident is triggered when a user attempts to login with the same username 9 or more times with invalid passwords.

Behavior: In this case, the user is probably attempting to brute force the account indicated in the incident details. Brute force against authentication works by enumerating over a list of common passwords and testing all of them against the target username. The hope is that the target user selected a weak password and that password is in the "dictionary" list of passwords to try. In some cases, a custom brute force tool can be employed, which enumerates over a list of passwords that were carefully constructed using the targets personal information (birthdays, anniversaries, names, ages, phone numbers, and so on.)

Response Processors: Login Processor: Incident - Site Login Brute Force

Complexity: Medium (3.0)

Default Response: 1x = Slow Connection for 6 hours, 3x = Slow Connection & Break Authentication for 6 hours

Cause: The login processor is designed to protect the login dialog of the website. It works by monitoring all login attempts and identifying suspicious and malicious events. This specific incident is triggered when a user completes 16 or more login attempts, each with a different username.

Behavior: A common authentication attack is Brute Force. This attack involves submitting a large number of username and password combinations in an effort to identify users who have chosen weak passwords. This type of attack is extremely noisy and requires thousands of requests to execute.

Response Processors: Login Processor: Incident - Site Login Username Scan

Complexity: Medium (3.0)

Default Response: 1x = Captcha and Slow Connection for 6 hours, 3x = Clear Inputs and Slow Connection for 1 day

Cause: The login processor is designed to protect the login dialog of the website. It works by monitoring all login attempts and identifying suspicious and malicious events. This specific incident is triggered when a user attempts to login against 16 or more different usernames with a small number of passwords for each.

Behavior: One flaw present in a lot of authentication implementations is that the results that are returned when submitting an invalid username and password are different then the results returned when the username is valid but the password is not. By enumerating over a large number of possible usernames and supplying bad passwords, the attacker is able to identify which usernames are actually valid in the system. This is one of the first steps to a large scale brute force attack. Once the user has a list of valid usernames, they can then launch the brute force attack against just those usernames to make the attack quicker and harder to identify. A best practice when developing authentication systems is to ensure that the results that are returned from an invalid username, are the same results returned when providing a valid username and invalid password. For example, the error should read "The username and password you supplied could not be found in our database", instead of "The username you provided does not exist".

Response Processors: Google Map Processor

The Google Map Processor provides a counter response called the “Google Map Response”. When this response is activated, the user will be shown an overlay dialog with a google map of their geo location (as resolved from their IP address using MaxMind Geo IP). It will then recommend 4 google search results on a configured term (default is ‘Criminal Attorney’). The intention is to scare the individual into believing that we know where they live and plan to attempt prosecution.

The google map response requires several things in order to work. First, you must obtain a google map API key and set it in configuration. Until you do this, you will not be able to enable the processor. Once enabled, if you activate the processor on a user, they will only see the response if WebApp Secure can resolve their geo location from MaxMind GeoIP. If a geo location cannot be resolved, the map will not be displayed. Additionally, the google map response is not a default response for any activity, so unless you manually activate it, or create a custom auto response rule to activate it, it will never be used.

Keep in mind that by activating this response, you are effectively broadcasting your public google map API key to the attacker. If the attacker decides to exploit this fact, they can easily drain your google map request and search result quotas. As such, it is important to get an API key for a junk google development account, so that your quota's are not shared with legitimate site functions. You should also not sign up for paid quota extensions on that particular account, as that could allow the attacker to run up your bill. Just use the free quotas.

Table 42: Google Map Processor Configuration Parameters

Parameter	Type	Default Value	Description
Basic			
Processor Enabled	Boolean	False	Whether traffic should be passed through this processor.
Google API Key	String	[Not Set]	The API key issued by Google to authorize the map API to be used on the domain being protected by WebApp Secure. This API key should be enabled for both Google Map API v3, and the Places Search API.
Advanced			
Default Search Term	String	"Criminal Attorney"	The default term to search for localized locations on.
Dismissal Resource	Map Dismissal Resource	mapdata	The information needed to define the URL and response used to dismiss a map.
Map Directory	String	mapdata	The name of the directory where the map Javascript and css code will be served from. For example: mapdata.

CHAPTER 7

Response Rule Configuration

- [Response Overview on page 229](#)
- [Using the Editor on page 232](#)
- [List Of Incident Methods on page 233](#)
















Response Overview

To view existing response rules, and to add your own counter response, navigate to **Configuration > Responses** in the Web UI.

A counter response is composed of a set of rules which define the conditions under which a response should be automatically created and activated for a specific session or profile. To turn on the counter response service, navigate to **Configuration > Services > Counter Response Service** and enable the appropriate responses.

It is possible to have as many rules as needed to protect the system. However, the more rules, the longer it will take to determine if a new incident matches an event condition. In addition, the more conditions in the rule, the longer the rule will take to evaluate if the event condition matches a new malicious incident.

Figure 61: Responses

Name	Status	Description	Actions
Session Management	 	This autoreponse rule handles the core session management incidents generated by the security engine. read more	
Processor: ETag Beacon	 	This autoreponse rule handles incidents generated by the etag beacon processor. These incidents are generally triggered when a user attempts to exploit the tracking mechanism used by the application to re-identify users. This rule is designed to first slow the user's connection down, and, if the behavior continues, slow it down even further. read more	
Processor: Application Vulnerability Processor	 	This autoreponse rule handles incidents generated by the application vulnerability processor. These incidents are generally triggered when a user attempts to exploit a known vulnerability in a 3rd party application. This rule is designed to first slow the user's connection down, and, if the behavior continues, break the application with the clear input response. read more	
Processor: Access Policy	 	This autoreponse rule handles incidents related to the access policy processor. These incidents are generally triggered by malicious spiders and the rule is designed to block those spiders for a prolonged period of time. read more	
Processor: Basic Authentication	 	This autoreponse rule handles the various incidents triggered by the basic authentication honeypot processor. These incidents are triggered by users	



NOTE: You can view the default responses for each rule by clicking [read more](#) at the bottom of the entry's description in the Web UI.



NOTE: To create your own response, click the **Add Autoresponse** button at the top of the window. Creating responses is an advanced task. Refer to the product's API documentation for configuration details.

Table 43: Response Descriptions

Default Response	Description
Session Management	This response rule triggers if the user attempts to manipulate the WebApp Secure session tracking cookie.
Application Vulnerability Processor	If your web-application uses supported third party applications (like Joomla, Wordpress, and so on.), this processor will analyze and act on malicious traffic that intends to exploit them. For more information on which third party tools are supported, refer to the Response documentation in Web UI.
Login Processor	This rule triggers on incidents that are generally triggered by abusive and suspicious activity targeted at the websites authentication system.
Access Policy Processor	This response rule triggers if the user attempts to exploit the fake service exposed by this processor.
ETag Beacon Processor	This response rule triggers if a user attempts to manipulate the WebApp Secure cached based tracking token.
Basic Authentication Processor	This response rule triggers when the user attempts to exploit the fake .htaccess file exposed by this processor.
Robots Processor	This response triggers when the user or malicious spider uses the information in the robots.txt file for illegitimate purposes.
Hidden Input Form Processor	This response rule triggers when the user modifies a hidden form input parameter.
Cookie Processor	This response rule triggers when the user attempts to manipulate the value of a cookie.
AJAX Processor	This response rule triggers when the user interacts with a fake AJAX function injected into the web application. If the user reverse engineers the code and manually invokes its behavior, such as would happen with an automated script or spider, the rule will fire. If the user actually invokes the Javascript function, the rule will fire.
Header Processor	This response rule triggers when the user has unusual headers or header data which a normal browser or well developed spider would not supply. If the user excludes required headers such as Host and UserAgent, manipulates their user agent header, overflows headers beyond RFC standards will cause this rule to activate.
Hidden Link Processor	This response rule triggers when a spider or malicious user attempts to identify unreferenced resources in a fake directory.
Query Parameter Processor	This response rule triggers when a user manipulates the fake query parameter injected by the system more than 3 times.

Table 43: Response Descriptions (*continued*)

Default Response	Description
Method Processor	This response rule triggers when a user or spider sends a request with a malicious HTTP method such as TRACE.
Error Processor	This response rule triggers when a user attempts to find unreferenced resources by guessing file names.
File Processor	This response rule triggers when a user attempts to find sensitive files by guessing file names or changing parts of valid file names.
Warning Processor	This response rule triggers when a user attempts to automate the dismissal of the warning response.
Cookie Protection Processor	This response rule triggers when a user attempts to modify the web application session cookie.
Captcha Processor	This response rule triggers when a user attempts to find a way to bypass the captcha response without solving the captcha.
CSRF Processor	This response rule triggers if a user attempts to manipulate the CSRF protection introduced by the system, potentially to find a filter evasion vulnerability.
Custom Authentication Processor	This response rule triggers if a user attempts to exploit the authentication mechanism offered by the system.
Client Beacon Processor	This response rule triggers when the user attempts to tamper with the client side tracking logic.
New and Modified Profiles	This response rule sends out an alert any time a new profile is created, or a profile elevates its threat level. The severity of the alert will equal the threat of the new or elevated profile that triggered the alert.
Returning Profile	This response rule sends out an alert any time a profile returns on a subsequent day. For example, a new hacker is observed on Monday, if the hacker is only active for 1 hour on Monday, but returns on Tuesday to continue, this rule will issue an alert. The severity of the alert will equal the threat level of the profile.
New Incident	This response rule sends out an alert any time a malicious incident is observed. The severity of the alert will equal the complexity of the incident. Note that non-malicious incidents are excluded by default. Setting your alert contact level to "informational" or "suspicious" will only have an effect when dealing with custom response rules where you manually set these severity levels.
New Response	This response rule sends out an alert any time a new counter response is activated. The severity of the alert will always equal 1.

Related Documentation • [Using the Editor on page 232](#)

Using the Editor

To create a response, open the configuration Web UI and select the **ADD New Rule** button. This will launch the editor which can be used to create and edit a response.

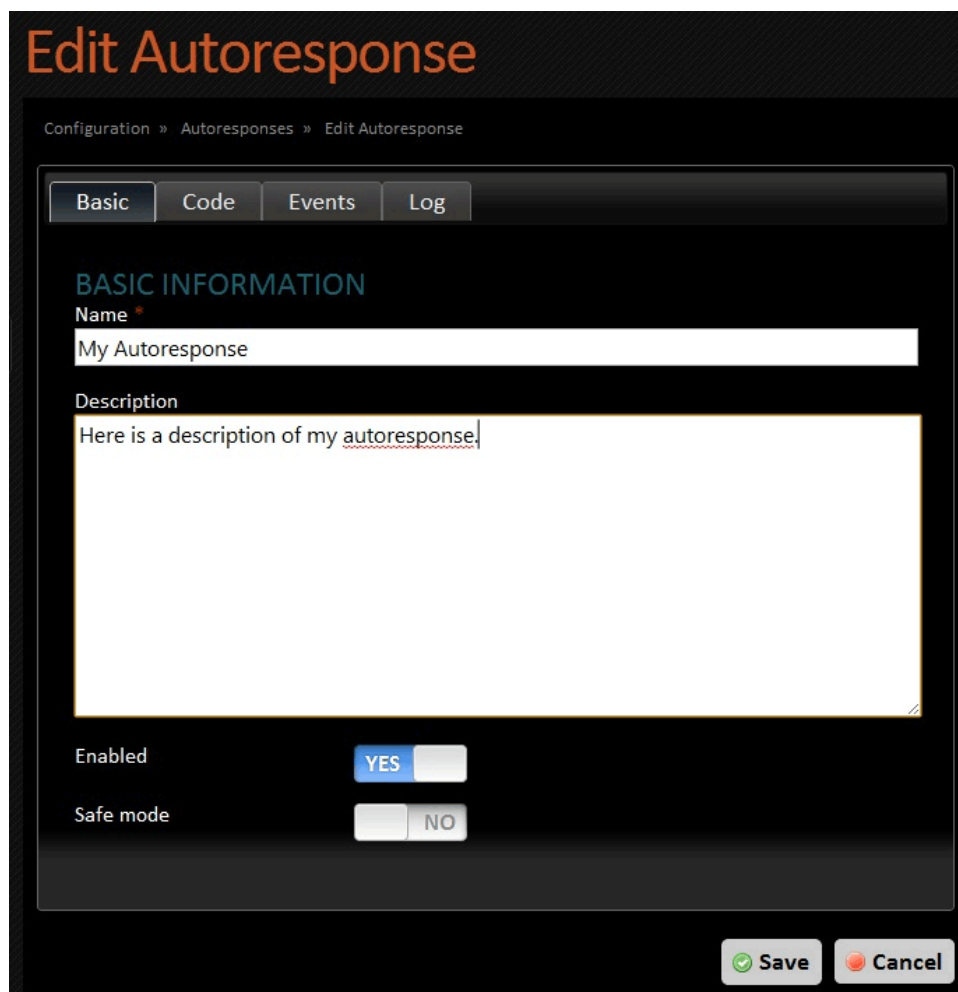


NOTE: Creating responses is an advanced task. Refer to the product's API documentation for configuration details.



NOTE: Once you create a custom rule and save it, the Log tab appears in the Editor UI. Note that the information in the Log tab must be manually refreshed.

Figure 62: Edit Response



Edit Autoresponse

Configuration » Autoresponses » Edit Autoresponse

Basic Code Events Log

BASIC INFORMATION

Name *
My Autoresponse

Description
Here is a description of my autoresponse

Enabled YES

Safe mode NO

Save Cancel

Table 44: Response Editor Fields

Field	Description
Name	The name of the response.
Description	Description of the response and its triggers.
Enabled	Sets a response to be active.
Safe Mode	Allows the response to activate, but does not actually respond. This setting is for testing and debugging responses.
Code	The actual code that defines the response.
Events	The events that will trigger the response.
Log	A table which consists of any log statements printed during response execution. Use the JavaScript console object to output to an response's log.
API Reference	A link to the Autoresponse API documentation.

List Of Incident Methods



NOTE: Parameters wrapped in [] are optional.

Table 45: Incident Methods

Name and Description	Parameters
isIncidentType Check the incident type by either its code or its name.	incident:string
isIncidentDate Check to see if an incident occurred on the given month, day and year. The month, day and year arguments can be left empty to match any value. Note that Jan = 1, and years are in the format YYYY.	[month:int] [day:int] [year:int]
isIncidentDateRange Check to see if an incident occurred between two dates. All values must be defined. Note that Jan = 1, and years are in the format YYYY.	start_month:int start_day:int start_year:int end_month:int end_day:int end_year:int

Table 45: Incident Methods (*continued*)

Name and Description	Parameters
isIncidentTime Check to see if an incident occurred at a given time. The hour, minute and second arguments can be left empty to match any value.	[hour:int] [minute:int] [second:int]
isIncidentTimeRange Check to see if an incident occurred between a given time range. All values must be specified.	start_hour:int start_minute:int start_second:int end_hour:int end_minute:int end_second:int
isIncidentCount Check the number of times an incident has occurred against an integer operation and specified value. Supported operations include (>, <, ==, !=). The results are: (count [operator] value)	operator:string value:int
isIncidentCountRange Check to see if the number of times an incident has occurred is within a given range.	min:int max:int
isIncidentContextSubString Check to see if the context XML associated with the incident contains the provided substring. The search is case sensitive by default, unless the second parameter is "false".	search:string [[caseSensitive]:Boolean]
isIncidentContextPattern Check to see if the context XML associated with an incident contains a simple pattern. Supported pattern wild cards include +, ? and *. Pattern matches are performed case sensitive unless the second parameter to this method is "false".	pattern:string [[caseSensitive]:Boolean]
isIncidentIP Check to see if an incident came from a given IP address. Each parameter specifies the required value for the specific block of the address. Any of the parameters can be left empty to match any value.	[a_block:int] [b_block:int] [c_block:int] [d_block:int]
isIncidentIPRange Check to see if an incident came from a given IP address range. Each parameter specifies a range of accepted values for the specific address block. Ranges are specified in the format: min-max. For example: 10-22, or 0-255	[a_block_range:string] [b_block_range:string] [c_block_range:string] [d_block_range:string]
isIncidentBrowser Check to see if the incident occurred from a given browser. The parameter expects the canonical name of the browser.	name:string

Table 45: Incident Methods (*continued*)

Name and Description	Parameters
isIncidentOperatingSystem Check to see if the incident occurred from a given operation system. The parameter expects the canonical name of the operating system.	name:string
isIncidentBrowserVersion Check to see if the incident occurred from a specified version of the browser. The check is case sensitive by default, unless the second parameter is "false". The version could contain any character and should be considered as an arbitrary user supplied string value.	version:string [[caseSensitive]:Boolean]
isIncidentBrowserVersionPattern Check to see if the incident occurred from a browser with a version that matches a given simple pattern. Pattern wild cards >include ?, * and +. The match is done case sensitive unless the second parameter is "false". The version could contain any character and should be considered as an arbitrary user supplied string value.	pattern:string [[caseSensitive]:Boolean]
isIncidentBrowserVersionSubString Check to see if the incident occurred from a browser with a version that contains the given sub string. The match is done case sensitive unless the second parameter is "false". The version could contain any character and should be considered as an arbitrary user supplied string value.	Search:string [[caseSensitive]:Boolean]
isIncidentCountry Check to see if the incident originated from a given country. The parameter expects a valid 2 character country code, or the canonical name of the country.	country:string
isIncidentLatitude Check to see if the incident originated from a specified geographical latitude. The parameter is expected to be a decimal number between -90.0 and +90.0.	latitude:float
isIncidentLatitudeRange Check to see if the incident originated between a specified geographical latitude range. The parameters are expected to be decimal numbers between -90.0 and +90.0.	min:float max:float
isIncidentLongitude Check to see if the incident originated from a specified geographical longitude. The parameter is expected to be a decimal number between -90.0 and +90.0.	longitude:float
isIncidentLongitudeRange Check to see if the incident originated between a specified geographical longitude. The parameters are expected to be decimal numbers between -90.0 and +90.0.	min:float max:float
isIncidentCity Check to see if the incident originated in a specified city. The parameter is expected to be the city name and is case sensitive unless the second parameter is "false".	city:string [caseSensitive]:Boolean
isIncidentCityPattern Check to see if the incident originated from a city that matches a specified pattern. The supported wild cards are *, ?, and +. The pattern is case sensitive unless the second parameter is "false".	pattern:string [caseSensitive]:Boolean
isIncidentCitySubString Check to see if the incident originated from a city that contains a specified sub string. The substring search is done case sensitive unless the second parameter is "false".	search:string [caseSensitive]:Boolean
isIncidentHost Check to see if the incident originated in a specified host. The parameter is expected to be the host name and is case sensitive unless the second parameter is "false".	host:string [caseSensitive]:Boolean

Table 45: Incident Methods (*continued*)

Name and Description	Parameters
isIncidentHostPattern Check to see if the incident originated from a host name that matches a specified pattern. The supported wild cards are *, ?, and +. The pattern is case sensitive unless the second parameter is "false".	pattern:string [caseSensitive]:Boolean
isIncidentHostSubString Check to see if the incident originated from a host name that contains a specified sub string. The substring search is done case sensitive unless the second parameter is "false".	search:string [caseSensitive]:Boolean
isIncidentRegion Check to see if the incident originated in a specified region. The parameter is expected to be the region name and is case sensitive unless the second parameter is "false".	region:string [caseSensitive]:Boolean
isIncidentRegionPattern Check to see if the incident originated from a region that matches a specified pattern. The supported wild cards are *, ?, and +. The pattern is case sensitive unless the second parameter is "false".	pattern:string [caseSensitive]:Boolean
isIncidentRegionSubString Check to see if the incident originated from a region that contains a specified sub string. The substring search is done case sensitive unless the second parameter is "false".	search:string [caseSensitive]:Boolean
isIncidentZip Check to see if the incident originated in a specified zip code. The parameter is expected to be the zip code and is case sensitive unless the second parameter is "false". While zip codes should generally be numeric, there is the possibility of foreign zip codes containing strange characters.	zip:string [caseSensitive]:Boolean
isIncidentZipPattern Check to see if the incident originated from a zip code that matches a specified pattern. The supported wild cards are *, ?, and +. The pattern is case sensitive unless the second parameter is "false".	pattern:string [caseSensitive]:Boolean
isIncidentZipSubString Check to see if the incident originated from a zip code that contains a specified sub string. The substring search is done case sensitive unless the second parameter is "false".	search:string [caseSensitive]:Boolean

CHAPTER 8

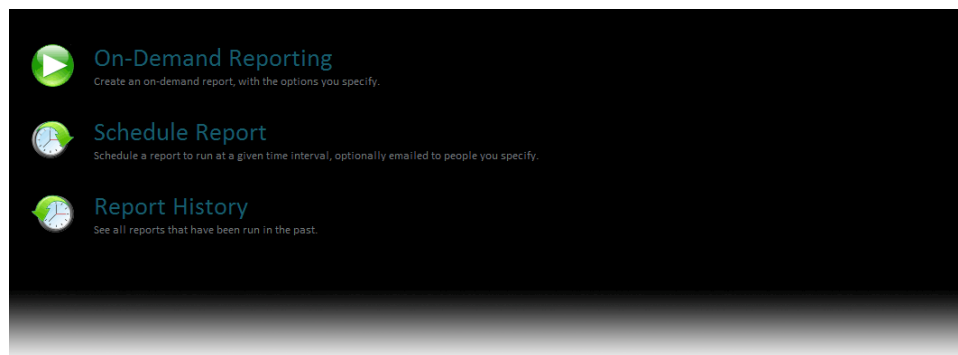
Reporting

- [Reporting Overview on page 237](#)
- [Information for Report Types on page 238](#)
- [Scheduling a Report Overview on page 240](#)
- [Schedule a Report on page 241](#)
- [Report History on page 243](#)
- [Report Details on page 244](#)
- [Report Types on page 246](#)

Reporting Overview

WebApp Secure has a built-in reporting interface that can be accessed through the Web UI, by navigating to the **Reports** menu item on the left-hand side. Administrators can run one of many pre-defined reports or schedule a report or access report history. Most reports can be exported to both PDF and CSV (comma separated value) formats. Reports that are composed of several disparate visual elements (like the Scorecard) are only available in PDF format.

Figure 63: Reporting Interface



Related Documentation

- [Information for Report Types on page 238](#)
- [Scheduling a Report Overview on page 240](#)
- [Schedule a Report on page 241](#)

- [Report History on page 243](#)
- [Report Details on page 244](#)
- [Report Types on page 246](#)

Information for Report Types

This page in the Web UI provides access to reports run on demand. Available reports include:

- **Country Comparison Over Time**—Click this report type and enter the following information:
 - Start Date
 - End Date
 - Enter the number of countries to show.
 - Countries to include: Select **Check All** or select the check boxes beside the countries you want included in the report.
 - Select a Time Zone from the provided list
 - Click **Generate Report**
- **Incident List**—Click this report type and enter the following information:
 - Select the file type: PDF or CSV
 - Start Date
 - End Date
 - Profiles to include: To include all profiles, leave this field blank. Otherwise, start typing a name. Once you have entered at least 3 characters, the Web UI will search for the matching name.
 - Incident types to include: Select **Check All** or select the check boxes beside the incident types you want included in the report.
 - Countries to include: Select **Check All** or select the check boxes beside the countries you want included in the report.
 - Restriction to an application: Select a configured application from the list.
 - The number of incidents to show.
 - Select a Time Zone from the provided list.
 - Click **Generate Report**
- **Incidents With Requests and Responses by IP**—Click this report type and enter the following information:
 - Select the file type: PDF or CSV
 - Start Date

- End Date
- IP address of location. Enter the IP address here.
- Maximum number of records. Enter the number of records here. The default is 500.
- Select a Time Zone from the provided list.
- Click **Generate Report**
- **Incidents with Requests and Responses by Profile**—Click this report type and enter the following information:
 - Select the file type: PDF or CSV
 - Start Date
 - End Date
 - Hacker profile name. Enter the profile name here.
 - Maximum number of records. Enter the number of records here. The default is 500.
 - Select a Time Zone from the provided list.
 - Click **Generate Report**
- **Incidents by Type**—Click this report type and enter the following information:
 - Select the file type: PDF or CSV
 - Start Date
 - End Date
 - Select a Time Zone from the provided list.
 - Click **Generate Report**
- **Incidents by Type for IP**—Click this report type and enter the following information:
 - Select the file type: PDF or CSV
 - Start Date
 - End Date
 - IP address of hacker: Enter the address here.
 - Select a Time Zone from the provided list.
 - Click **Generate Report**
- **Scorecard**—Click this report type and enter the following information:
 - Select a Time Zone from the provided list.
 - Click **Generate Report**
- **Top IP Addresses**—Click this report type and enter the following information:
 - Start Date
 - End Date

- The number of IP addresses to show. The default for this field is 10. The maximum is 50.
- Select a Time Zone from the provided list.
- Click **Generate Report**
- **Top Incident Types**—Click this report type and enter the following information:
 - Start Date
 - End Date
 - The number of incidents to show.
 - Incident types to include: Select **Check All** or select the check boxes beside the incident types you want included in the report.
 - Restriction to an application: Select a configured application from the list.
 - Select a Time Zone from the provided list.
 - Click **Generate Report**
- **Top Locations**—Click this report type and enter the following information:
 - Start Date
 - End Date
 - The number of locations to show.
 - Restriction to an application: Select a configured application from the list.
 - Select a Time Zone from the provided list.
 - Click **Generate Report**

**Related
Documentation**

- [Schedule a Report on page 241](#)
- [Report History on page 243](#)
- [Report Details on page 244](#)
- [Report Types on page 246](#)

Scheduling a Report Overview

The Scheduled Reports screens lets you view all of the reports currently scheduled to run on the system, add a new report to the list, edit an existing report schedule, edit an existing report options, or enable/disable an existing report scheduled to run. You can configure the reporting interface to generate a report on a custom schedule which will be automatically e-mailed to any e-mail address specified.

Figure 64: Scheduled Reports

Schedule Name	Report	Run	Period	Format	Options	To	Actions
My Scheduled Scorecard	Scorecard	Weekly	(not applicable)		time_zone=UTC	executive@example.com	
My Country Report	Country Comparison Over Time	Monthly	1 Month		top=10&time_zone=UTC&countrycodes=AO,AR,AU	threats@example.com	

Related Documentation

- [Schedule a Report on page 241](#)
- [Report History on page 243](#)
- [Report Details on page 244](#)
- [Report Types on page 246](#)

Schedule a Report

To schedule a report, do the following:

1. Select the **Schedule Report** left navigation link.
2. Click the **Add Scheduled Report** button at the top right of the Scheduled Reports page. This brings up a list of reports to run. Choose the report that you want to run on a repeated basis by following its link.
3. On the subsequent page, enter all of the schedule details and report options and then select **Generate Report Schedule**.
4. **Save** the changes.

Most reports share the following items:

- **File type:** The file format that will be used to generate the report. Options usually include PDF or CSV. Certain reports are only available in PDF.
- **Schedule Name:** The name of the report schedule that will appear in the reporting interface.
- **Run:** The time schedule in hours, weeks, months, or years that the report should run on.
- **Period:** The period of time that the report should be run on.
- **Send to:** The e-mail address that this report should be sent to.
- **Enabled:** Sets this report schedule to active (YES) or inactive (NO). Inactive reports will not be run on a scheduled basis.

Figure 65: Schedule Report - Scorecard

Reports » Schedule Report » Scorecard

Schedule Report: Scorecard

File type pdf ▾


Timezone * UTC ▾
Timezone to be used as selection range and display for the dates/times in the report. You may adjust your default timezone in [User Preferences](#). Please note that if you are planning on having this report emailed to people in different timezones, it may make more sense to create two separate scheduled reports.

Schedule name * My Scheduled Scorecard
Pick a name for the scheduled report.

Run * Weekly ▾
How often would you like this report to run?

Send to executive@example.com
Separate email addresses with a comma.

Enabled ☒

 **Add Report Schedule**



NOTE: Individual reports can have various additional options that are specific to that report. For instance the Country Comparison Over Time report contains a field for the number of countries to show and a list of specific countries to include.

Figure 66: Schedule Report - Country Comparison Over Time

Schedule Report: Country Comparison Over Time

File type: pdf

The number of countries to show (top n): 10

Countries to include: Argentina, Australia, Belgium

Timezone *: ☐ Check all ☒ Uncheck all

Schedule name *:

Run *: ☐ Angola ☒ Argentina ☒ Australia ☐ Austria ☒ Belgium ☐ Bosnia and Herzegovina

Period *: 1 Hour
What period would you like each report to cover?

Send to:
Separate email addresses with a comma.

Enabled: ☐

Add Report Schedule

- Related Documentation**
- [Report History on page 243](#)
 - [Report Details on page 244](#)
 - [Report Types on page 246](#)

Report History

An archive option, which allows administrators to view all of the historical scheduled reports that have been run on the system. Previously run reports can be downloaded by clicking the icon with the green arrow pointing down. Reports can also be deleted, to save on disk space, by clicking the icon that looks like a trash can.

Figure 67: Report History

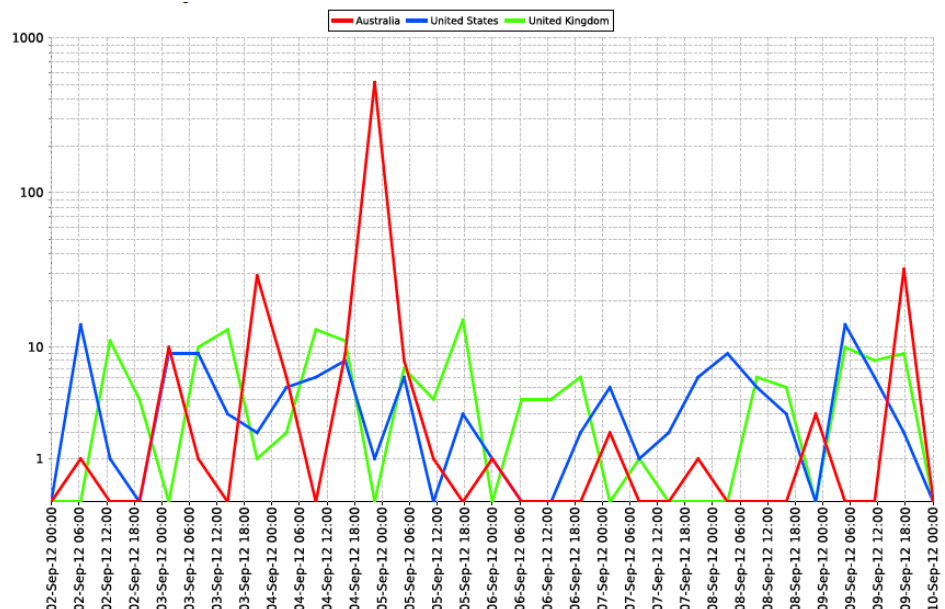
Report History						
Schedule Name	Report	Run	Period	Format	Date	Actions
Scorecard Report	Scorecard	Weekly	1 Week	(52.5 KB)	Thu Apr 26 19:31:05 UTC 2012	
Scorecard Report	Scorecard	Weekly	1 Week	(54.8 KB)	Thu Apr 19 19:30:58 UTC 2012	
Scorecard Report	Scorecard	Weekly	1 Week	(48.8 KB)	Thu Apr 12 19:30:44 UTC 2012	
Scorecard Report	Scorecard	Weekly	1 Week	(48.8 KB)	Mon Apr 09 20:05:42 UTC 2012	
						Delete All

Report Details

There are ten different reports that are available either immediately, through the On-demand Report page, or on a repeated and scheduled basis, through the Scheduled Report page. Each retrieves different sets of information.

For example, the Country Comparison Over Time report displays a graph showing the number of incidents per country for the top N number of countries over a specified date range. Administrators can specify the number of countries to include and the specific countries to include as well. Dates are displayed along the horizontal axis. Incident counts are displayed along the vertical axis scaled logarithmically. This report is only available in PDF format.

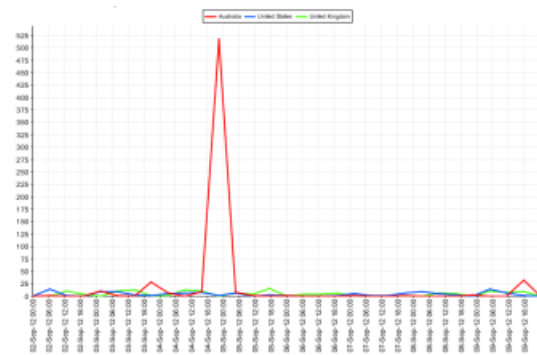
Figure 68: Report Details





NOTE: Please note the logarithmic scale on the vertical axis. The distance between hash marks on the graph vary. For instance the hash marks between 1 and 10 increase by 1 each time (1, 2, 3, and so on). The hash marks between 10 and 100 increase by 10 (10, 20, 30, and so on). The hash marks between 100 and 1000 increase by 100 (100, 200, 300, and so on). It is used in the graph above, and in others described below, because the counts can vary widely and displaying them with a logarithmic scale will allow them to be compressed into a more readable format. For instance if the graph above was displayed without a such a scale most of the detail between the 1 and 10 hash marks would be compressed to the point where it would not be readable, as in the graph below.

Figure 69: Report Details, View



Related Documentation

- [Schedule a Report on page 241](#)
- [Report Details on page 244](#)
- [Report Types on page 246](#)

Report Types

- **Incident List:** This report displays a list of every incident that occurred between the two given dates. Details for each incident include the type, complexity, count of occurrences, name of the hacker profile associated with the incident, location of the hacker and the first and last date of occurrence of that incident. The report can be narrowed to include only selected profiles, incident types, countries, a single application, and/or a specified number of incidents by altering the specific options for this report.

Figure 70: Incident List

Incidents from 05-Aug-12 00:00 to 15-Sep-12 00:00						Mykonos System Report	
Application: All						http://www.mykonossoftware.com/	
Displaying top 100 out of 107 rows							
Incident Name	Complexity	Count	Location	Profile	First	Last	
Missing User Agent Header	Low	39	Macedonia - Unknown	Inez 8633	23-Aug-12 15:32	23-Aug-12 15:32	
Missing User Agent Header	Low	39	China - Shanxi	Beatriz 9780	29-Aug-12 07:35	29-Aug-12 07:36	
Missing User Agent Header	Low	7	Canada - Ontario	Mike 6474	29-Aug-12 16:21	29-Aug-12 17:48	
Missing User Agent Header	Low	5	United States - Oklahoma	Josie 2020	02-Sep-12 00:00	02-Sep-12 01:55	
Missing User Agent Header	Low	5	Canada - Ontario	Rosemarie	30-Aug-12 18:39	30-Aug-12 18:40	
Missing User Agent Header	Low	4	United States - California	Joshua 7944	08-Aug-12 13:29	08-Aug-12 13:29	
Missing User Agent Header	Low	3	United States - Washington	Darren 1899	20-Aug-12 13:28	20-Aug-12 13:28	
Apache Configuration Requested	Low	3	United States - New York	Mercedes 5932	04-Sep-12 11:27	04-Sep-12 11:39	
Missing User Agent Header	Low	2	China - Beijing	Lula 5206	30-Aug-12 14:21	30-Aug-12 14:44	

- **Incidents with Requests and Responses by IP:** This report lists the incidents for a given IP and date range. Details for each incident include: date of the first occurrence, the user agent string, the request content, the response content, the incident type, and the count of occurrences.
- **Incidents with Requests and Responses by Profile:** This report lists the incidents for a given profile name and date range. Details for each incident include: date of the first occurrence, the user agent string, the request content, the response content, the incident type, and the count of occurrences.
- **Incidents by Type:** This report lists the incidents that have occurred within a given date range. Details displayed include: the type of each incident that has occurred and the count for that particular type.
- **Incident by Type for IP:** This report displays a list of incidents created between the given dates for a given IP Address. Details on the report include: the name of the incident type and the count of the number of incident occurrences of that type.
- **Scorecard:** The scorecard report displays a summary of activity on the protected site. The executive summary at the top of the page displays the total number of attackers detected, the number of attackers that have been blocked, and the number of incidents detected for three time periods. These time periods are: from the beginning of the appliance to the current date, the last month from the first of the month through the last of the month, and the last complete week starting from Sunday through Saturday. Below the executive summary section are four graphs that break out the top five incident types, the top five hackers by volume, the top five countries by volume, and the activity of the previous week broken out by day of the week.

Figure 71: Executive Summary

Executive Summary	Attackers Detected	Attackers Blocked	Incidents detected
Since Deployment (15-Nov-11 to 30-Apr-12)	1988	34	9100
Last Month (01-Mar-12 to 31-Mar-12)	444	4	875
Last Week (22-Apr-12 to 28-Apr-12)	39	0	47

Figure 72: Incident Types

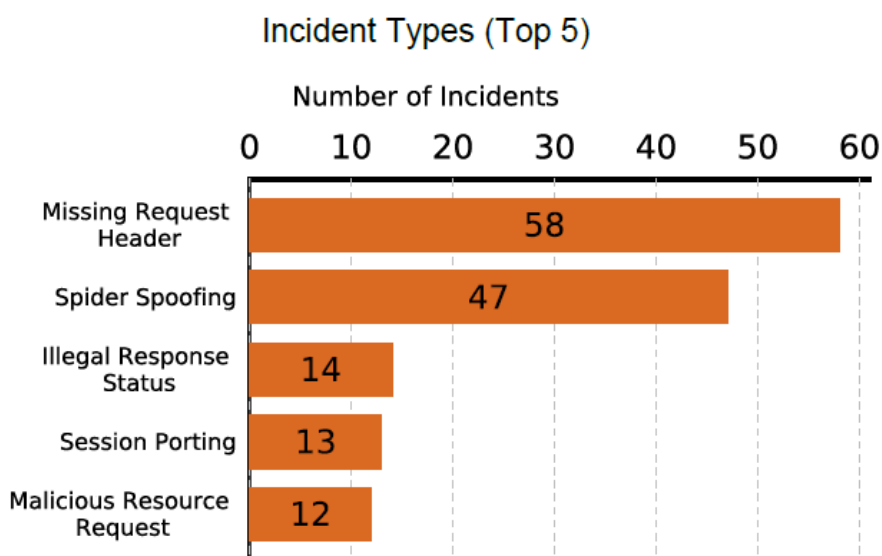


Figure 73: Incident Volume by Hacker

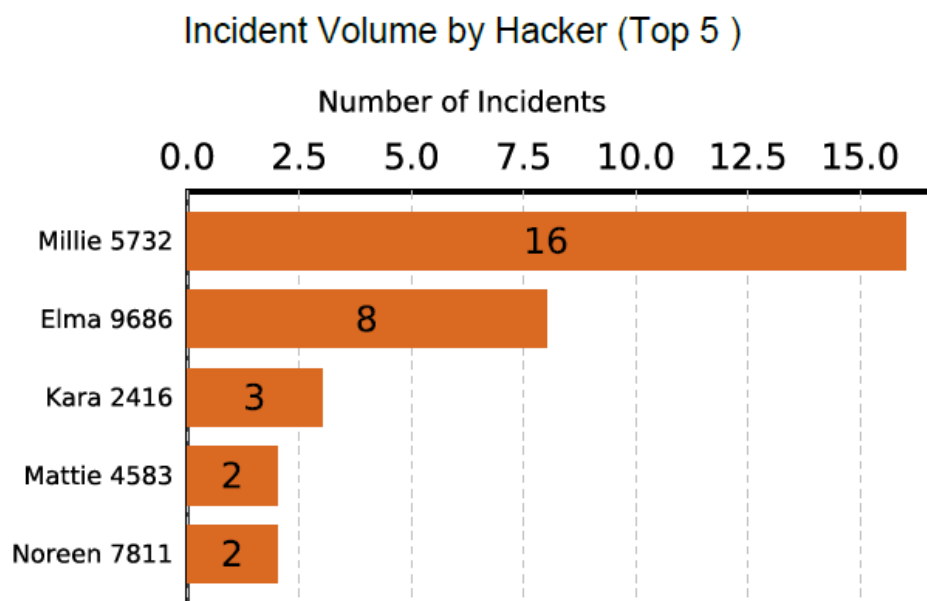


Figure 74: Incident Source Countries

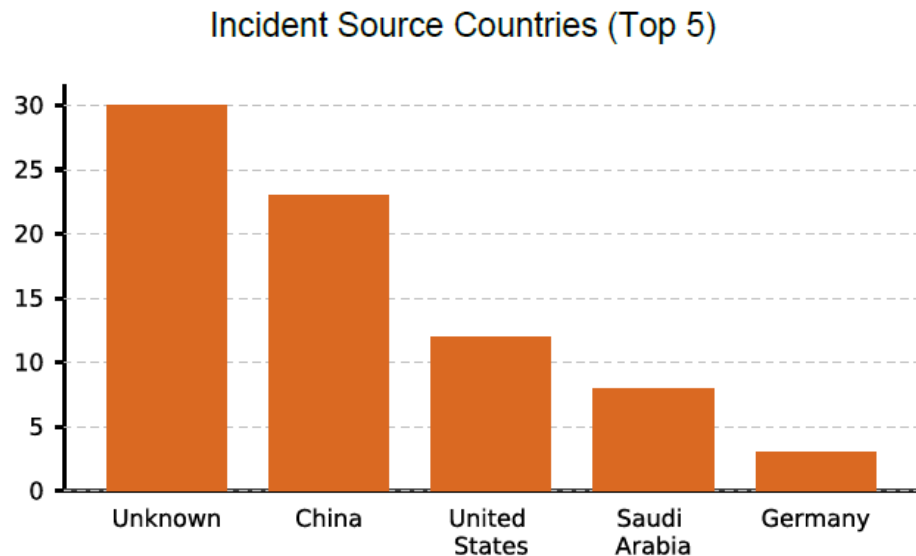
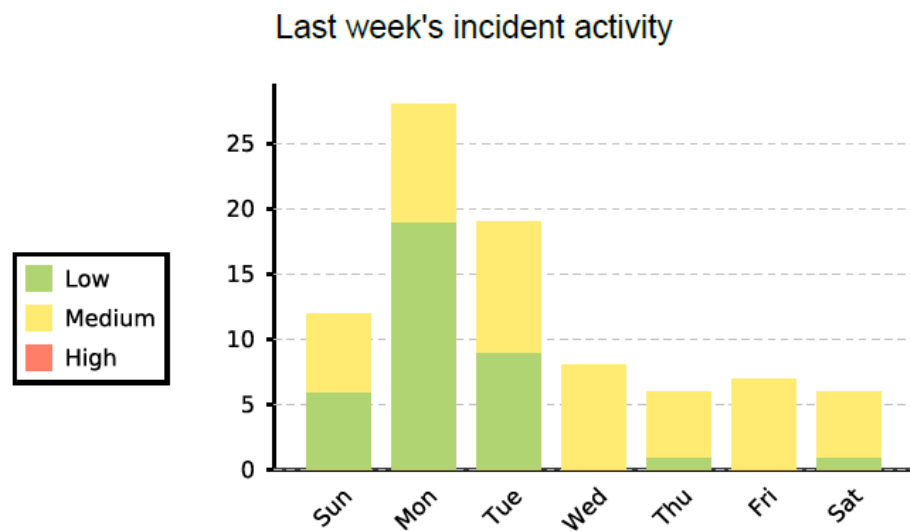


Figure 75: Last Week's Incident Activity



Below the four graphs is the weekly report section, which lists the counts of incidents broken out by threat level and totaled. It also includes counts of the number of hackers who were blocked, the number who were countered with a non blocking response (such as a slowed connection or a warning), the number of hackers that were not responded to (because they were not deemed a high enough threat), and the total number of hackers. This report is only available in PDF format.

Figure 76: Weekly Report**Weekly Report**

Threat level of Attackers	Low	Medium	High	Total
Number of Attackers	11	49	0	60
Responses Deployed	Blocking	Non-blocking	None	Total
Number of Responses	0	15	47	62

- **Top IP Addresses:** The Top IP Addresses report contains up to five graphs, one for each complexity level, that break down the IP addresses that have caused the most incidents. If there were no incidents of a given complexity then there will not be a graph for that complexity. This report is only available in PDF format.
- **Top Incident Types:** The Top Incident Types report contains a list of the top N incident types over the specified time period, ordered by number of occurrences. Included on the list is supplementary detail such as the number of countries, profiles, and IP addresses related to the type of incident.

Figure 77: Top Incident Types

Top 13 incident types from 01-Jan-12 00:00 to 15-Sep-12 00:00

Application: All

Mykonos System Report
<http://www.mykonossoftware.com/>

Missing User Agent Header	Occurrence 8699 Profiles: 1231	Countries 45 IP addresses: 599	
Missing Host Header	Occurrence 6284 Profiles: 321	Countries 31 IP addresses: 169	
Malicious Script Introspection	Occurrence 169 Profiles: 169	Countries 16 IP addresses: 114	
Spider Configuration Requested	Occurrence 150 Profiles: 131	Countries 10 IP addresses: 65	
Query Parameter Manipulation	Occurrence 114 Profiles: 11	Countries 8 IP addresses: 11	

Following the list is a set of graphs each on their own page. Each graph is specific to one type of incident on the list and shows the distribution of those incident occurrences over the selected time period. The time period is shown on the horizontal axis. The count of occurrences of each type of incident are shown on the vertical axis scaled logarithmically.

The report can be narrowed to include a specified number of types of incidents or only a selected set of incidents. It can also be narrowed to only contain data from a specific application. This report is only available in PDF format.

- **Top Locations:** This report contains a list of the top N locations ordered by the number of incidents that originated from each location and timezone during the specified time interval. Included on the list is supplementary information including the number of High, Medium, Low, and Indicator level incidents from each location.

Figure 78: Top Locations

Top 10 locations between 19-Aug-12 00:00 and 19-Sep-12 00:00

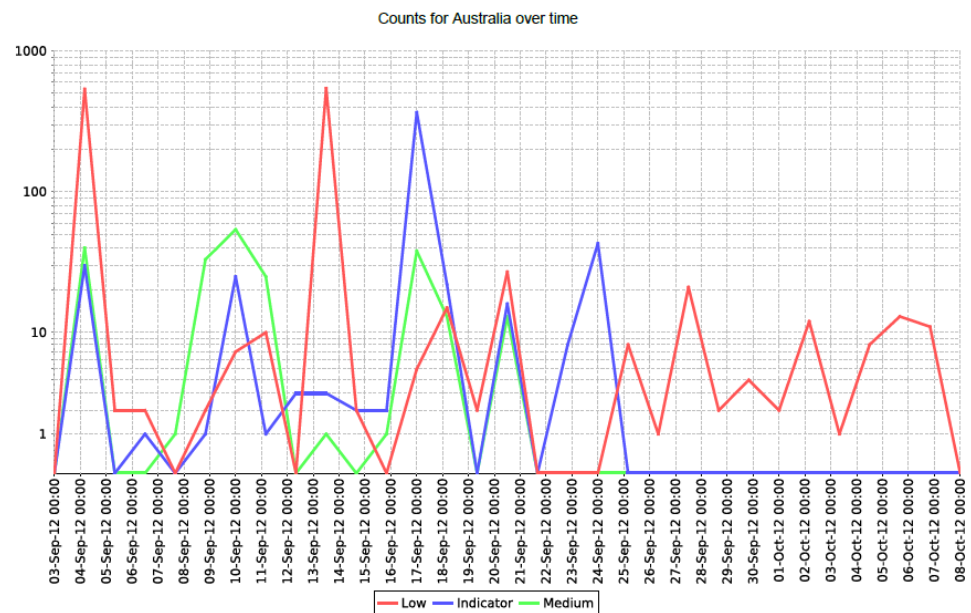
<http://www.mykonossoftware.com/>

Application: All

Australia	Indicators: 6	Mediums: 906	
	Lows: 670	Highs: 0	1,582
United States	Indicators: 10	Mediums: 1068	
	Lows: 412	Highs: 0	1,490
United Kingdom	Indicators: 0	Mediums: 1044	
	Lows: 14	Highs: 0	1,058
China	Indicators: 0	Mediums: 0	
	Lows: 530	Highs: 0	530

Following the list is a set of graphs each on their own page. Each graph is specific to one country on the list and shows the distribution of each incident level over the selected time period. The horizontal axis shows the time period. The count of occurrences of incidents from a specific country are shown on the vertical axis scaled logarithmically. This report is only available in PDF format.

Figure 79: Country Counts Over Time



Related Documentation

- [Schedule a Report on page 241](#)
- [Report History on page 243](#)
- [Report Details on page 244](#)

CHAPTER 9

Using the Web UI

- [Web UI Overview on page 251](#)
- [The Dashboard on page 252](#)
- [Attackers on page 257](#)
- [Attacker Profile Page on page 258](#)
- [Incidents on page 262](#)
- [Incident Details on page 263](#)
- [Counter Responses on page 264](#)
- [Sessions on page 265](#)
- [Session Details on page 265](#)
- [Search on page 266](#)
- [Reports on page 268](#)
- [Configuration on page 269](#)
- [System Status on page 269](#)
- [Updates on page 271](#)

Web UI Overview

The Web UI is really two UIs in one. There's a monitoring portion, which shows you real-time security-related information about the sites WebApp Secure is protecting, and a configuration portion, which allows you to configure most aspects of the system without the need to use the CLI.

In addition to monitoring and configuring WebApp Secure, the Web UI also provides several widgets you can use to filter, find, and view information.

Those widgets are as follows:

- The date filter widget—Available on many of the monitoring screens, the date filter widget allows you to filter lists of data throughout the Web UI by date, as well as by configured application. The currently-selected values of each drop-down field are "sticky" and will persist in a cookie as you move from screen to screen.

Figure 80: Date filter widget



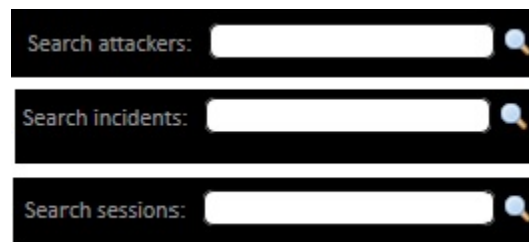
- The user widget—Available throughout the Web UI, the user widget identifies the currently-logged-in user, and provides links to logout and to edit your user preferences.

Figure 81: User widget



- The search widget—The search widget is available in the upper right corner of the Attackers, Sessions, and Incidents screens. This widget is context-aware; that is, you can search only within the scope mentioned in the box's label.

Figure 82: Search widget

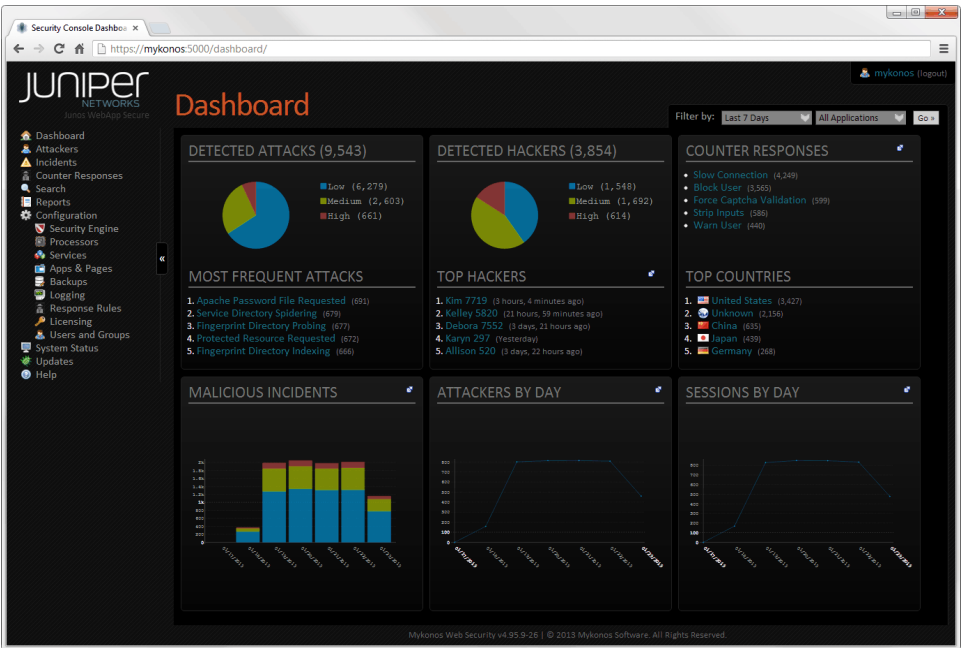


- Related Documentation
- [Edit Web UI User Preferences on page 44](#)
 - [Search on page 266](#)

The Dashboard

The Dashboard is the page used to display currently monitored incidents, sessions, and responses. It can be reached by navigating to **Dashboard** in the left-hand menu, or simply by clicking the logo in the top left of the window.

Figure 83: Web UI Dashboard



The dashboard contains graphs and charts depicting the activity on protected web applications. By default, the dashboard will display the information gathered from all configured applications in the past week (7 days), but you can focus on specific applications or change the date range by using the **Filter By:** tab near the top right of the window.

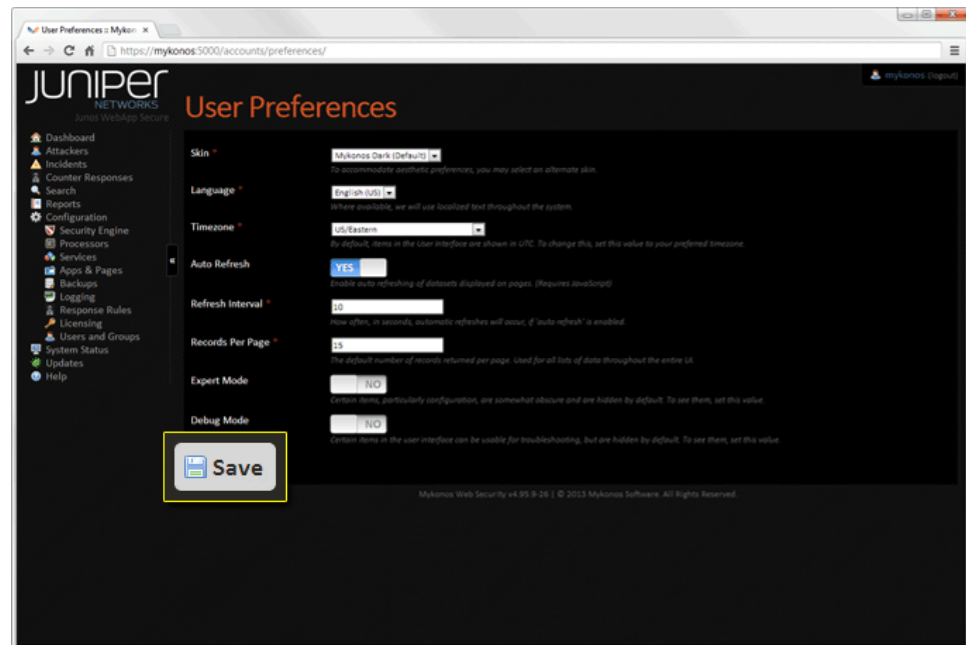
Figure 84: Dashboard - Filter By tab



There are other customization options that can be configured per account by clicking your username in the very top right of the window. Options include setting your timezone, enabling auto-refresh of data contained in the dashboard, and configuring the number

of records returned per page. Change these preferences to your liking and click **Save** at the bottom of the page.

Figure 85: User Preferences



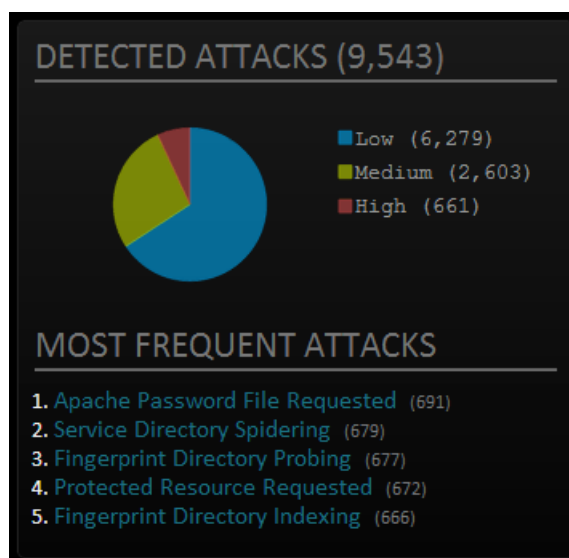
The main portion of the dashboard consists of various panes, each containing information gathered by the Security Engine.



NOTE: The Web UI should be compliant with current browsers. While older browsers might work, we recommend updating to the latest versions for best functionality. Likewise, we recommend JavaScript be enabled in the browser. JavaScript is used in the Web UI to enhance functionality and usability, and while browsing without JavaScript is possible, it is not recommended. The Web UI targets screen resolutions of 1366x768 or higher for normal operations, and targets 1440x900 when debug mode is enabled.

Table 46: Web UI Dashboard Panes

Detected Attacks This pane displays a chart that contains all incidents created (within the filter parameters) segregated by complexity of the attacks. It is a good way to visualize how active your website's attackers are. You can hover over each portion of the pie chart to display the actual number of attacks for each complexity. This pane also contains a short list of the most frequent attacks. Clicking on any attack in the list will open the Incident Type page for that particular incident.



Detected Hackers The Detected Hackers pane displays information on actual hacker profiles created within the specified time frame. Each hacker gets a skill level which segments the pie chart. As in any other chart, you can hover over the pie chart to view specific counts. Additionally, the most active hackers are displayed in a list below the chart. Clicking on any of these hackers will open the Hacker Profile page for that particular profile.

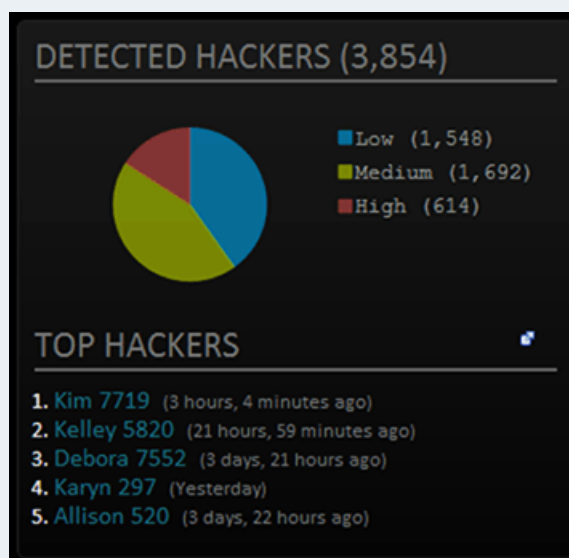


Table 46: Web UI Dashboard Panes (*continued*)

Counter Responses The Counter Responses pane lists the top responses that have been recently triggered by WebApp Secure. The lower portion of the pane lists countries in descending order by incident count. You can click on any of the counter responses to open the Counter Response Type page for that response (explained later), or click on the specific country to find other information WebApp Secure has gathered on that country.



Malicious Incidents The Malicious Incidents dashboard pane consists of a chart depicting the number of incidents over time. It also stacks these incidents by complexity.

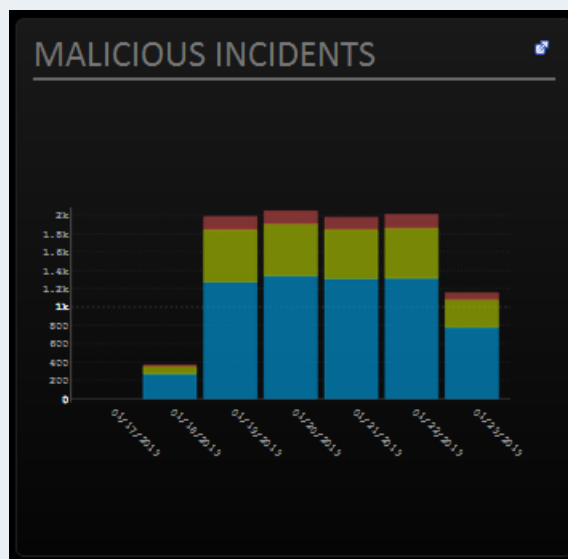
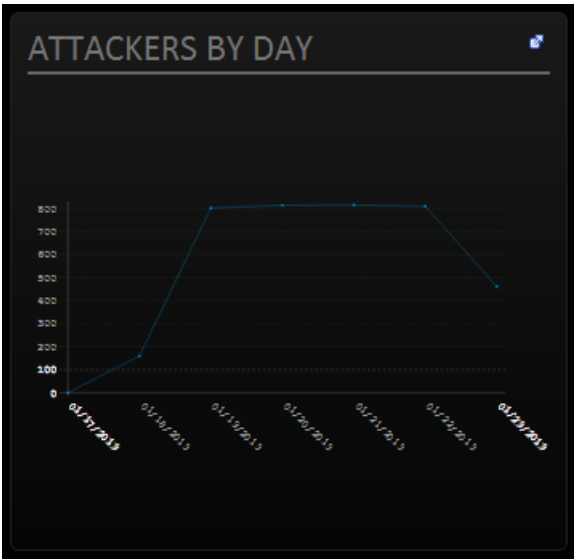
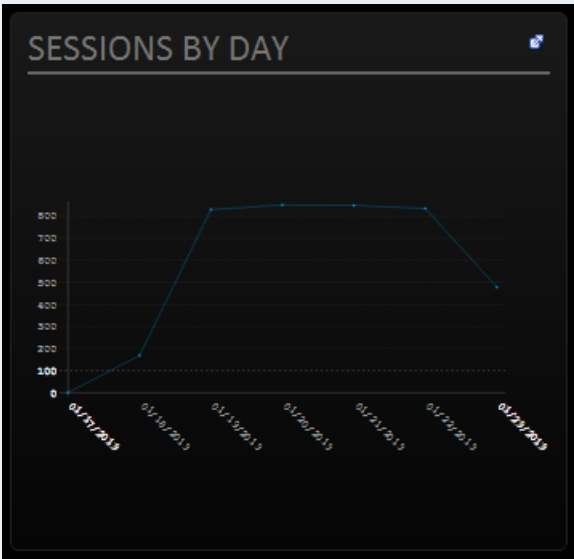


Table 46: Web UI Dashboard Panes (*continued*)

Attackers By Day (Attackers By Hour)
This pane contains a graphical representation of how many detected hackers were active on the protected site, separated by day (or separated by hour if the filter "Last Day" is currently set).



Sessions By Day (Sessions By Hour)
Similar to the Attackers By Day pane, the Sessions By Day pane shows the number of sessions active on the protected site each day (or each hour if the filter "Last Day" is set).

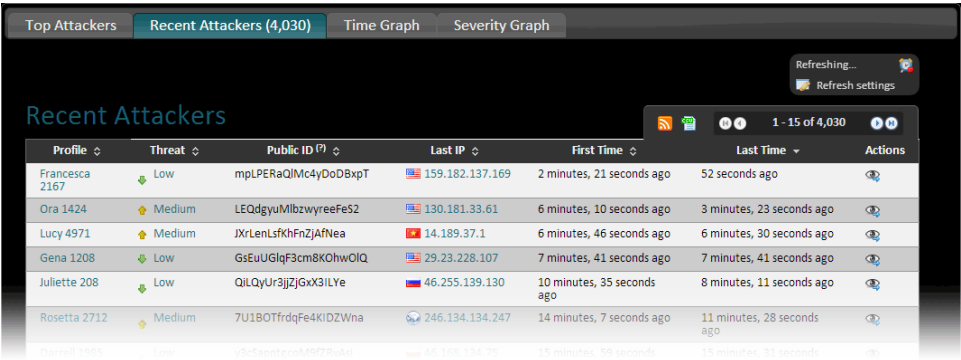


NOTE: If you would like more horizontal space on any page, you can collapse the navigation menu by clicking on the double arrow (<<) button to the right of the menu.

Attackers

The Attackers page contains any information on profiled attackers, and can be accessed by clicking **Attackers** in the left navigation menu.

Figure 86: Recent Attackers



Profile	Threat	Public ID	Last IP	First Time	Last Time	Actions
Francesca 2167	Low	mplPERaQIMc4yDoDBxpT	159.182.137.169	2 minutes, 21 seconds ago	52 seconds ago	
Ora 1424	Medium	LEQdgyuMlbzwyreeFeS2	130.181.33.61	6 minutes, 10 seconds ago	3 minutes, 23 seconds ago	
Lucy 4971	Medium	JXrLenLsfKhFnZjAfNea	14.189.37.1	6 minutes, 46 seconds ago	6 minutes, 30 seconds ago	
Gena 1208	Low	GsEuUGlqF3cm8KOhwOIQ	29.23.228.107	7 minutes, 41 seconds ago	7 minutes, 41 seconds ago	
Juliette 208	Low	QilQyUr3jZjGx3ILYe	46.255.139.130	10 minutes, 35 seconds ago	8 minutes, 11 seconds ago	
Rosetta 2712	Medium	7U1BOTfrdQFe4KIDZWna	246.134.134.247	14 minutes, 7 seconds ago	11 minutes, 28 seconds ago	
Darrell 1895	Low	y3CfapetpnaM9TpuKd	46.108.134.75	15 minutes, 50 seconds	15 minutes, 51 seconds	

There are various data views you can navigate through through the tabs near the top of the page. You can also search for attackers by using the search field in the upper right side of the page, under the **Filter** widget.

- **Top Attackers** This tab contains an ordered list of the most active attackers, calculated based on a weighting algorithm that takes into account the number of incidents and their corresponding complexities.
- **Recent Attackers** This tab displays a table of the most recent profiles active on the protected system. Each row consists of the Profile name, Threat level, their Public ID (for use with the Support Processor), the Last IP they used on the system, the First Time and Last Time they attacked the system, and available actions for that profile. Clicking on the "eye" icon or the profile name will lead you to the page for that particular profile. You can also click on a threat level to view other attackers with similar threat, and you can click on a Last IP to navigate to the Location page for that IP. To keep this data fresh, the monitor will periodically refresh the page (if Auto-refresh is enabled in the User Preferences). To stop this from happening, click the alarm clock icon in the top right corner of the tab to stop refresh.
- **Time Graph** The Time Graph is a larger version of the same line graph displayed on the Dashboard.
- **Severity Graph** This graph is a larger version of the same pie graph displayed on the Dashboard.

Related Documentation

- [Attacker Profile Page on page 258](#)

Attacker Profile Page

You can click on an attacker's given name to navigate to that Attacker's Profile page.

Figure 87: Attacker Profile



The Attacker Profile page displays any information that pertains to a particular attacker. At the top of the page you will see the Attacker Card, which contains a short overview of the profile. This card contains the attacker's assigned name, last IP used, the first and last date the attacker was active, and the Public ID of the attacker, for use with the Support Processor in unblocking that profile. On the right side of the card there is a threat gauge that indicates the current threat of that attacker, where green, yellow, and red indicate low, medium, and high threat, respectively. The severity icons are displayed as follows:

- (n/a): 0.0 - None
- : 1.0 - Suspicious
- : 2.0 - Low
- : 3.0 - Medium
- : 4.0 - High

Available on the right side of the Attacker Profile page is a quick **Actions** box, where you can rapidly perform various profile-related functions such as blocking the attacker, warning the user, editing the profile, and deleting the profile.



NOTE: Deleting the profile will essentially erase all information gathered on that attacker, and will effectively remove all blocks or other responses on that profile.

Underneath the attacker card and quick actions box is a series of tabs, where all of the attacker's specific activity information resides. The Incidents tab contains a list of all incidents triggered by that attacker. The Incident name, complexity, count, first and last time triggered are all available for each item in the list. Additionally you can click the Details icon (the eye) to view more information about any particular incident.

Responses tab– The Responses tab contains information relating to all of the active and inactive responses issued to that attacker. Each entry contains the actual name of the response issued, the configuration (if any) used when issuing the response, the time the response was created, the delay set (if any), the time the response expires (if at all), the time the response was finally deactivated (if it has been deactivated).

If the response is active, you can click the Deactivate Response icon (the stop sign under Actions) to deactivate the response instantly. Alternatively, you can click the **Deactivate Selected** button or to deactivate all responses, click the **Deactivate All** button

Figure 88: Responses tab - Deactivate

Response	Config	Created	Delayed	Expires	Deactivated	Actions
<input type="checkbox"/> Google Map		Just now			Never	
Slow Connection	max=6000, min=2500	4d, 22h ago		3d, 22h ago	3d, 22h ago	
Force Captcha Validation		4d, 22h ago			4d, 22h ago	

Deactivate Selected Deactivate All

It is in this tab that you can manually activate Counter Responses on the current attacker. The available counter responses are:

- **Block User** To block the user from accessing the protected application completely, you can activate the Block User counter response. The next time the attacker tries to visit any page on the application, they will see a configurable message indicating they have been blocked from accessing the content. If the Support Processor is enabled, they are also given their Public ID (also shown on the Attacker Profile page for that profile) that they can give to support if they feel the block was in error.
- **Filter on SRX series** For more information on what this counter response does, see: SRX Series Integration. In jest, it feeds a message to an SRX series device that can handle traffic at the network level.



NOTE: This counter response can be activated without configuring an external network device, but it will not do anything. WebApp Secure requires a properly configured external device for this counter response to function properly.

- **Break Authentication** Hashes any incoming passwords when attempting to login, effectively thwarting brute-force attacks that have correct credentials. Even with the correct password, the login will be unsuccessful.
- **Cloppy** Activating this counter response will activate an animated paper clip that intimidates the user with configurable messages. For information on how to customize this response, see the Cloppy Processor in Processor Reference section.
- **Force Captcha Validation** The user will be prompted with a Captcha that has to be solved to continue using the website.
- **Google Map** The user will be shown a map of lawyers near their determined location. The search term fed into Google Maps can be configured, see the Google Map Processor in the Processor Reference section.
- **Inject Header** The suspected hackers requests will have a custom header injected into them, useful for tracking.
- **Logout User** Terminates any current user sessions for this profile on a site.
- **Slow Connection** The user's requests to the site will be delayed by a configurable window of milliseconds. This can frustrate the attacker and cause them to abandon their future attacks. This response can take a `<config/>` node with 'min' and 'max' parameters, for example; `<config min=1000 max=5000 />` will slow the attackers requests by 1 to 5 seconds.
- **Strip Inputs** If you suspect the attacker's inputs shouldn't be trusted (such as those inputs submitted in forms on the site), you can choose to activate this response which will strip them from all incoming requests. This will also strip any query parameters from the request URL as well.
- **Warn User** The next request sent by the attacker will respond with a pop-up warning message that lets the attacker know he/she is being watched. The warning message can be configured, see the Warning Processor in the Processor Reference section.

Consecutive requests might be grouped together and are viewable through the Sessions tab. Each entry in this tab contains the Remote Address used during the session (the IP), the Browser and Operating System used during the session, the number of Requests made and Pages returned during that session, the number of Errors generated by the server in response to requests in that session, as well as the First and Last Active times. You can also click on the **Details** icon (the eye) to view more information about any particular session.

Locations tab—The Locations tab contains a list of all locations used by the attacker. For each location, you are able to see the Remote Address (IP) associated with that location, the City, Region, and Country associated with the location (if they can be found), and the First and Last Active times for the location. Depending on the location, you might

also be able to load a map showing that location (if it can be determined) by clicking on the **Map** icon. You can also click on the **Details** icon (the eye) to view more information on any particular location, including all other attackers that were found to be using the same location, and other Incidents, Sessions, or Environments used in conjunction with that location. If WebApp Secure can determine the attacker was using a specific Browser and Operating System combination, an entry in the Environments tab will be added. Each entry contains the Browser and Operating System used, along with the full User Agent string and First and Last active dates. If you want to find other attackers that used the same Environment, click on the magnifying glass icon. This will bring you to a page where you can see other Attackers that used this Environment, Incidents produced with this Environment, Sessions found that were using this Environment, and Locations that used this Environment.

Incidents

The Incidents page contains any information on specific incidents that have been triggered, and offers additional information on all of the incidents that can be detected by WebApp Secure.

Figure 89: Incidents Table



Incident	Attacker	Complexity	Count	First Time	Last Time	Actions
Service Directory Spidering	Karl 3462	Medium	1	1 minute, 35 seconds ago	1 minute, 35 seconds ago	
Apache Password File Requested	Karl 3462	Low	1	2 minutes, 28 seconds ago	2 minutes, 28 seconds ago	
Captcha Directory Indexing	Lynnette 1107	Low	1	3 minutes, 28 seconds ago	3 minutes, 28 seconds ago	
Apache Password File Requested	Lynnette 1107	Low	1	4 minutes, 31 seconds ago	4 minutes, 31 seconds ago	
Service Directory Indexing	Pam 4739	Medium	1	5 minutes, 29 seconds ago	5 minutes, 29 seconds ago	



NOTE: Incident tables include a column labeled Count which indicates the number of times a particular attacker performed a specific incident (in the case where the attacker actually triggered an incident multiple times in quick succession). However, if enough time has passed, the incident will be given a new row in the Incident table with separate counts.

There are various data views you can navigate through through the tabs near the top of the page. You can also search within Incidents by using the search field in the upper right side of the page, under the **Filter** widget.

- **Most Common** This tab displays a list of the most frequently triggered incidents in descending order. Count of triggered incidents of that type is displayed to the right of each item in the list, and a graphic depicting the complexity of that incident is visible to the left. Clicking on a particular incident in this list will bring you to a page with additional information on that incident. By default, WebApp Secure only displays malicious incidents (those that might be of direct interest to WebApp Secure users). If you want to show all incidents triggered, you can click on the **Show all incidents** link above the list.
- **Most Recent** This tab displays a table of the most recent incidents triggered. The incident name is displayed along with the profile that triggered the incident, the complexity of that incident, the Count indicating the number of times that incident was triggered at one time (using the same data), the first and last times the profile activated that particular incident, and any actions available to the WebApp Secure user regarding that incident. You can navigate to other pages by using the tab above the table. Here you can jump to the next page, previous page, first page, and last page by using the corresponding buttons. You can also jump to a specific page or change the number of rows returned per page by clicking on the label between the navigation buttons. By default, only malicious incidents are displayed. To display all malicious and non-malicious incidents, click the **Show all incidents** link above the title. To keep this data fresh, the monitor will periodically refresh the page (if Auto-refresh is enabled in the User Preferences). To stop this from happening, click the alarm clock icon in the top right corner of the tab to stop refresh.
- **Browse by Complexity** For informational purposes, this tab allows you to browse the list of detectable incidents, grouped by complexity. Clicking on an incident will bring you to an informational page that contains a description of that incident, and allows you to search for triggered incidents of that type.
- **Time Graph** The Time Graph is a larger version of the same bar graph displayed on the Dashboard.
- **Severity Graph** This graph is a larger version of the same pie graph displayed on the Dashboard.

**Related
Documentation**

- [Incident Details on page 263](#)


Incident Details




Clicking on a particular incident's name will bring you to the **Incident Details** page for that incident. On this page all information about that particular incident is shown.

Incident Details

Incidents » Incident Detail

Apache Configuration Requested



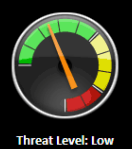
Attacker: Lesa 9502
Location:  Columbus Ohio, United States
Environment:  Internet Explorer 9.0,  Windows
Session: 130.90.162.56
Last Time: 16 days, 19 hours ago **First Time:** 16 days, 19 hours ago

Description Details Request Response

CAUSE:

Apache is a very common web server. As a result, hackers will often look for vulnerabilities specific to Apache, since there is a good chance that any given website is running Apache. One such vulnerability involves the use of an `.htaccess` file to provide directory-level configuration (password-protected resources, directory indexing options, etc...), while not sufficiently protecting the `.htaccess` file itself. By convention, configuration files should not be exposed to the public — so if a user requests `.htaccess` or a related resource, they should get either a "404 Not Found" or "403 Forbidden" error. Unfortunately, an improperly-configured installation of Apache may not block requests for these resources. In such a scenario, a hacker could gain valuable knowledge of the way the server is configured.

Junos WebApp Secure will automatically block any requests for the `.htaccess` resource, and instead return a fake version of the file, which contains the directives necessary to password-protect a fake resource. It is safe to assume the request is malicious because no legitimate user should ever be requesting this resource.



Threat Level: Low

Near the top of the page there is an incident infobox that contains a summary of the incident, including the Attacker that caused the incident, the Location and Environment that attacker was using, the Session (IP) used when triggering the incident, and the First and Last times that particular incident occurred. Underneath the infobox there is a series of tabs that display the Description of the Incident type, Details for the incident (differs from incident to incident), and the raw Request and Response objects.

Related Documentation

- [Incidents on page 262](#)

Counter Responses









The Counter Responses window contains any information on the various responses WebApp Secure can issue to potential threats. It contains the following tabs:

Browse by Type **Active Responses (5,492)** Inactive Responses (4,439)

Refresh in... 2 Refresh settings

ACTIVE RESPONSES

1 - 15 of 5,492

Attacker	Response	Config	Created	Delayed	Expires	Actions
Tammie 6840	Block User		44 seconds ago		In 4 days, 23 hours	
Karl 3462	Block User		2 minutes, 48 seconds ago		In 4 days, 23 hours	
Karl 3462	Slow Connection	max=6000, min=2500	3 minutes, 47 seconds ago		In 23 hours, 56 minutes	
Lynette 1107	Block User		4 minutes, 40 seconds ago		In 23 hours, 55 minutes	
Lynette 1107	Slow Connection	max=6000, min=2500	4 minutes, 40 seconds ago		In 23 hours, 55 minutes	
Lynette 1107	Slow Connection	max=6000, min=2500	5 minutes, 48 seconds ago		In 23 hours, 54 minutes	
Mara 8293	Slow Connection	max=6000, min=2500	6 minutes, 27 seconds ago		In 23 hours, 53 minutes	
Tom 4709	Block User		6 minutes, 47 seconds ago		In 4 days, 23 hours	

- **Browse by Type** In this tab, you can view information on any of the counter responses WebApp Secure can issue. Clicking on a specific response will take you to a page that

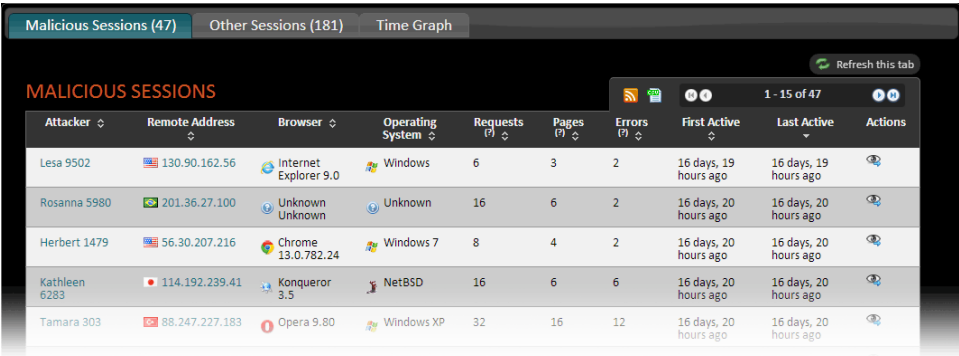
explains that response, and allows you to search for profiles that were issued that response.

- **Active Responses** In the Active Responses tab, a table displays the most recently activated responses, along with the profile that the response was issued on, the specific response issued, any configuration used in that response (blank if there wasn't any), the time the response was issued, how long to delay the response.
- **Inactive Responses** The Inactive Responses tab is formatted like the Active Responses tab, but shows all responses which have been deactivated, either manually or due to response expiration.

Sessions

When users browse the protected site, similar or back-to-back requests can be grouped together in a Session. The Sessions page allows you to view each of these browsing sessions.

Figure 90: Sessions



The screenshot shows the 'MALICIOUS SESSIONS' tab selected, displaying a table with 5 sessions. The table has columns for Attacker, Remote Address, Browser, Operating System, Requests, Pages, Errors, First Active, Last Active, and Actions. The data is as follows:

Attacker	Remote Address	Browser	Operating System	Requests	Pages	Errors	First Active	Last Active	Actions
Lesa 9502	130.90.162.56	Internet Explorer 9.0	Windows	6	3	2	16 days, 19 hours ago	16 days, 19 hours ago	[Details icon]
Rosanna 5980	201.36.27.100	Unknown	Unknown	16	6	2	16 days, 20 hours ago	16 days, 20 hours ago	[Details icon]
Herbert 1479	56.30.207.216	Chrome 13.0.782.24	Windows 7	8	4	2	16 days, 20 hours ago	16 days, 20 hours ago	[Details icon]
Kathleen 6283	114.192.239.41	Konqueror 3.5	NetBSD	16	6	6	16 days, 20 hours ago	16 days, 20 hours ago	[Details icon]
Tamara 303	88.247.227.183	Opera 9.80	Windows XP	32	16	12	16 days, 20 hours ago	16 days, 20 hours ago	[Details icon]

The tabs available in the Sessions page show Malicious Sessions, Other (non-malicious) sessions, and a graph of sessions over time. Each Session entry contains information including the Attacker the session belongs to (if it was a session with malicious intent), the Remote Address used during the session (the IP), the Browser and Operating System used during the session, the number of Requests made and Pages returned during that session, the number of Errors generated by the server in response to requests in that session, as well as the First and Last Active times. You can also click on the Details icon (the eye) to view more information about any particular session.

Related Documentation

- [Session Details on page 265](#)

Session Details

Clicking on the **Details** icon will bring you to the **Session Details** page for that session. On this page all information about that particular session is shown.

Session Detail

Sessions » Session Details

Attacker: Vera 105
Last Remote Address: 10.38.61.25
Last Location: Local Network
Last Environment: Opera 9.80 Linux
Requests: 9 **Pages:** 2 **Errors:** 2
Last Active: 16 days, 20 hours ago **First Active:** 16 days, 20 hours ago

Incidents (1) Locations (1) Environments (2)

Refresh this tab

INCIDENTS

Showing malicious incidents only. [Show all incidents](#)

Incident	Attacker	Complexity	Count	First Time	Last Time	Actions
Apache Configuration Requested	Vera 105	Low	1	16 days, 20 hours ago	16 days, 20 hours ago	

Near the top of the page there is a session infobox which contains a summary of the session, including the Attacker associated with the session, the Last known address (IP) used in conjunction with the session, the Last Location and Environment used during the session, and information regarding the number of Requests issued, Pages returned, and Errors generated by the server as a result of a request. Underneath the infobox is a series of tabs that display other Incidents, Locations, and Environments used during this browsing session.

Related
Documentation

- [Sessions on page 265](#)

Search

You can find a particular attacker, incident, or session by using the search functionality in the Web UI. To search, type the keyword in the Query form field, and optionally modify the desired date-range (last 7 days by default), applications (all applications by default), and the scope of your search. The scope can include Attackers, Incidents and/or Sessions.

Depending on the complexity of your search parameters, it might take a couple seconds to complete. Once finished, the results will be displayed.

Search

Query *

What do you want to search for?

Date Range *

How far back to you want to search?

Application *

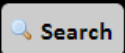
Which application do you want to filter by?

Scope *

Where do you want to search for it?

Max Results/Scope *

The maximum number of records to return, per scope.



Search Results (3)

ATTACKERS (1)

Profile Name	Threat Level	Public ID (?)	Last IP	First Time	Last Time	Actions
Ella 2360	Low	HC3YION6RztdIIY2Dky	10.10.10.113	2 days, 21 hours ago	2 days, 21 hours ago	

INCIDENTS (1)

Incident	Attacker	Complexity	Count	First Time	Last Time	Actions
Apache Configuration Requested	Ella 2360	Low	2	2 days, 21 hours ago	2 days, 21 hours ago	

SESSIONS (1)

Attacker	Remote Address	Browser	Operating System	Location	Requests	Errors	First Active	Last Active	Actions
Ella 2360	10.10.10.113	Chrome 26.0.1410.43	Windows 7	Local Network	2	0	2 days, 21 hours ago	2 days, 21 hours ago	

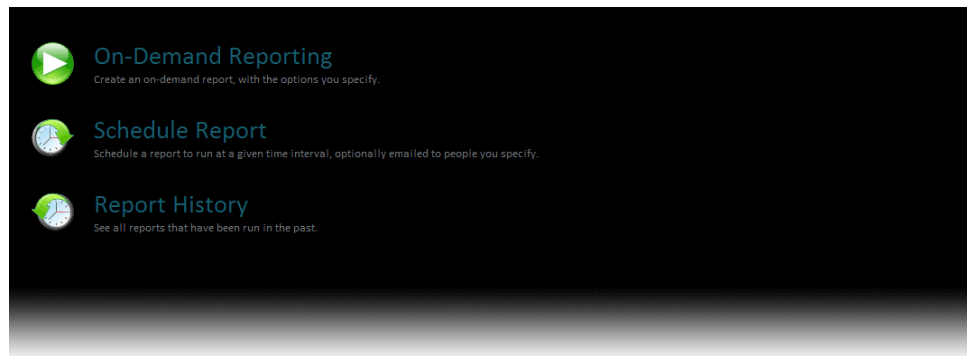
The following items are indexed in the search (meaning if the string matches any items in these categories, it is displayed.):

- User Agent
- Browser Name
- Browser Version
- Incident Name
- IP Address
- Host
- Geographic Region

- Geographic City
- Geographic ZIP
- Country Name
- Country Code
- Profile Name
- Profile Description
- Profile Public ID
- Incident Request Content
- Incident Response Content

Reports

The Reports page is responsible for producing graphical and textual representations of the activity passed through WebApp Secure.

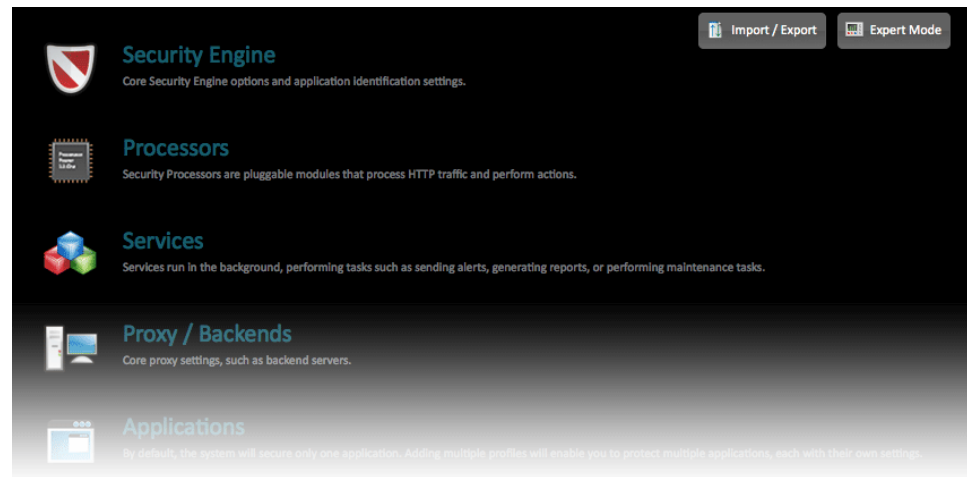


Related Documentation

- [Information for Report Types on page 238](#)
- [Scheduling a Report Overview on page 240](#)
- [Report History on page 243](#)
- [Report Details on page 244](#)

Configuration

The Configuration section of the Web UI allows you to change numerous aspects of the software. See Related Topics below.



Related Documentation

- [Basic Configuration Mode on page 46](#)
- [Processors Overview on page 122](#)
- [Configure Support for Akamai Dynamic Site Accelerator on page 49](#)
- [Security Engine Incident Monitoring on page 50](#)
- [Security Engine Whitelist Settings on page 53](#)
- [Proxy/Backends on page 54](#)
- [Applications Overview on page 55](#)
- [Backup and Recovery Overview on page 105](#)
- [Log File Destination on page 104](#)
- [Response Overview on page 229](#)
- [Configuring Role-Based Access Control on page 70](#)

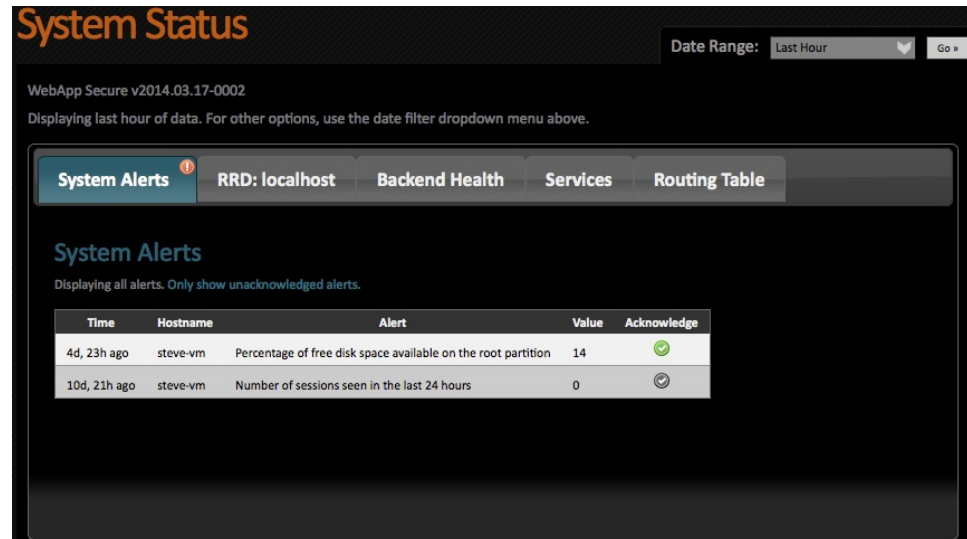
System Status

When the System Status icon in the navigation bar on the left side of the Web UI indicates there is an Alert, click the icon to access the System Alerts tab in the System Status window. From the System Alerts tab, you can view and acknowledge alerts. When viewing an alert, the following information is displayed:

- The time of the incident that triggered the alert.
- The host system on which the incident occurred.

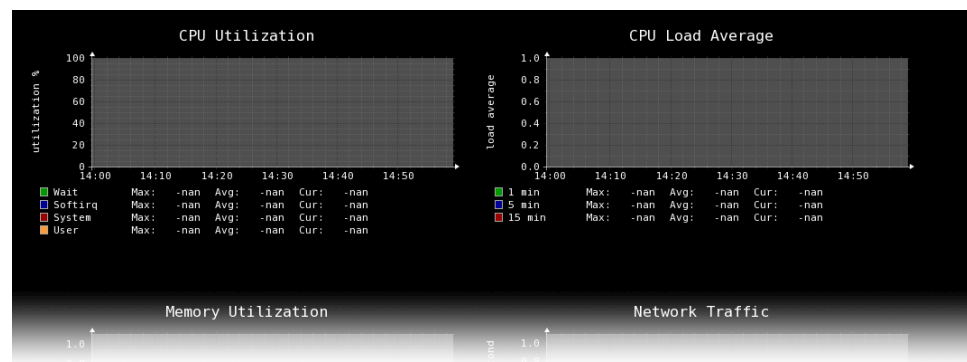
- A description of the alert.
- The value of the alert. This value represents something different for each alert type. The description of the alert tells you what the value means in each case.
- A check mark icon you can click to acknowledge the alert. Once you acknowledge it, it is no longer displayed by default. To display all alerts, including acknowledged alerts, you can click the **Show all alerts** link in the System Alerts tab.

Figure 91: System Status, System Alerts Tab



The RRD: localhost tab lets you to view performance metrics of your installation. This includes information on system health, running services, and the routing table.

Figure 92: System Status, RRD: localhost



The Backend Health tab displays the status of the security engine and any backend servers. While the Services tab lists the status of the running services on the system. You can select the Routing Table tab to view the Kernel IP routing table.

On hardware systems, a RAID Status tab appears in the System Status window. The following are examples of status information you could see for RAID.

Figure 93: Raid Status

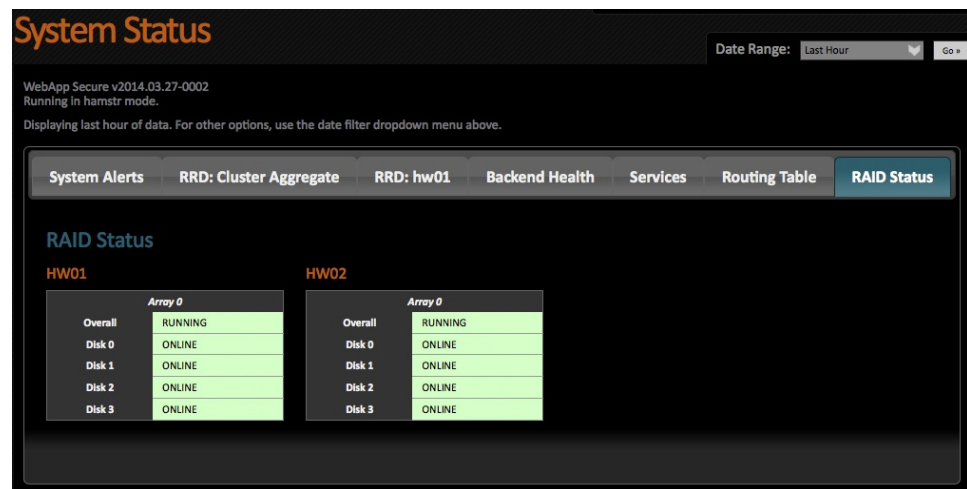
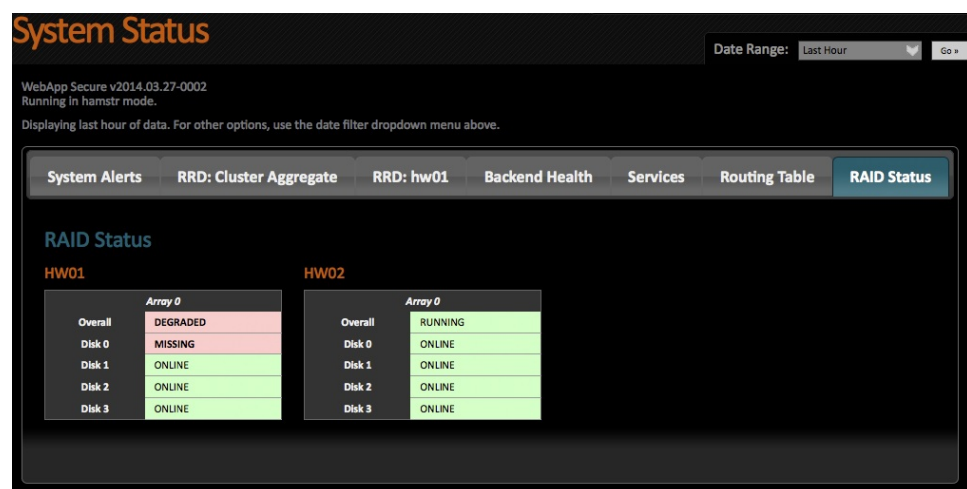


Figure 94: RAID Status-Missing





Updates

When updates are available, the Dashboard window alerts you to that fact. Perform updates to the installation by navigating to the **Updates** window in the Web UI. From the Updates window, you can set the following:

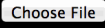
- Online Updates—**Enable** or **Disable** automatic update downloads.
- Offline Updates—When an update is available, you can save the update file for uploading and updating manually at a later time. Browse to the file and click the **Upload File** button to upload the update to the system. When it is finished uploading, it will appear as an Online Update

Figure 95: Updates


ONLINE UPDATES
Automatic updates download is enabled.  Disable  Manually Check for Updates

There are no updates ready for installation at this time.

OFFLINE UPDATES
If you would like to apply an update manually, you may use this form to upload the file you have received.

File *  No file chosen

Start after upload? ☐ After uploading is complete, whether or not to start the update process.

 Upload File

PART 2

Appendices

- [CAPTCHA Template on page 275](#)
- [Log Formats on page 279](#)
- [RBAC Groups and Roles on page 289](#)

APPENDIX A

CAPTCHA Template

- [CAPTCHA Template on page 275](#)

CAPTCHA Template

There are several processors that utilize captchas to prevent automation. These processors include:

- Request Captcha Processor

This processor allows you to attach a captcha to any page on the web application. It is also responsible for enforcing the "Force Captcha Validation" counter response

- Login Processor

This processor utilizes captchas to prevent brute force attacks on login dialogs. Once there have been more than three (3) failed login attempts on a single username (from any users), any future attempts to login as that user will require a captcha.

When a captcha must be presented, the format in which it is displayed is defined as a Captcha Template. By default, there is a captcha template defined for both processors that will work on all websites. In the event that you would like to customize the way the captcha looks when it is presented (such as wrapping it with the standard template of the website being protected), the captcha template can be modified. This is done by accessing the advanced configuration parameters for the two aforementioned processors and editing the "Captcha Template" parameter.

In order to edit the parameter, we recommend that you first download a copy of the existing default template. If you have already made modifications to the template, you can get the original by selecting the suggestion "Default Unbranded Template", and then downloading the associated file.

Once you have a copy of the default template, open it in a text editor. You can make any modifications to the HTML as required, but be sure not to modify the existing JavaScript or remove any of the existing HTML. To prevent introducing changes that might prevent the captcha from functioning, we recommend that modifications be limited to stylistic changes (do not alter the content of the SCRIPT tags, and do not alter the contents of the FORM tag). After your modifications, you can upload the new file into the parameter to update the captcha HTML served by WebApp Secure. It is recommended that you keep a copy of the modified template to make future modifications easier.

You will also notice that there are a few special HTML tags in the template. These tags are replaced by WebApp Secure before the template is served to the end-user. These tags reside either in a SCRIPT tag or in a FORM tag, so as long as those elements are not modified, these tags should continue to function correctly. These special tags include:

- **<%captchaDir>** The directory name that all captcha images and audio files are served from.
- **<%signature>** The file name for the captcha image or audio resource to load.
- **<%includeAudio>...<%includeAudio>** Displays the content between the open and closing tags only if audio captchas are enabled.
- **<%cancel>** The URL to redirect the user to if they cancel the captcha operation.
- **<%delay>** The number of seconds the user has to complete the captcha before it expires.
- **<%multiPart>...<%multiPart>** Displays the content between the opening and closing tag only if the original request that is being protected by a captcha was a multipart form submission (vs. a URL encoded form post [by default, forms are URL encoded]).
- **<%datasignature>** The signature of the data that was originally posted to the page protected by the captcha. This is used to ensure that the data is not modified after submission, but before the captcha is solved.
- **<%data>** The encrypted data submitted to the original page that required a captcha. This is used so that once the captcha is solved, the original request can be reconstructed and submitted to the backend servers.
- **<%inputname>** The name of the input used to identify when a user submits a captcha. The value for this input name is configurable and should not conflict with any existing inputs the site uses. A random string of 5 or more characters should be sufficient (but must be set in configuration so that it can be injected in place of the custom tag when serving a captcha).

After the new template has been uploaded and saved in configuration, you can test your changes by triggering the applicable captcha.

- **Request Captcha Processor** Access the protected page and request `http://www.domain.com/.htaccess` which will generate a profile for your session. Find the new profile in the Web UI and manually activate the "Force Captcha Validation" response. Then go back to the protected site and make a few more requests until the captcha shows up.
- **Login Processor** If the login processor is configured to protect a login dialog on the site, then simply provide 3 or more invalid passwords for the same username. On the 4th attempt, you should be presented with the login processor captcha.



.....

NOTE: Note: Changes to the captcha template are made to the live deployment. So if you break the captcha template during modifications, it can cause the captcha to stop working for some of the users on the site until the template is repaired. Creating a new "Page" in configuration for a fictitious URL and making the changes on that page first would allow you to test the modifications without impacting every use on the site.

.....

APPENDIX B

Log Formats

- [Access Log Format on page 279](#)
- [Security Log Format on page 281](#)
- [Audit Log Format on page 283](#)
- [Firewall Log Format on page 284](#)
- [Postgres Log Format on page 285](#)
- [mws Log Format on page 286](#)

Access Log Format

WebApp Secure lets you configure logging for all traffic coming to and from the box. All of this communication between the clients and WebApp Secure will be sent to access.log. Depending on the type of access logging enabled in the WebApp Secure configuration, there are some different formats the log entries can take.

- Basic

```
<date_utc> <hostname> [<log_level>] [mws-access][<thread>]  
key:<unique_request_key>, PHASE_<"REQUEST" or "RESPONSE">_<"PRE or  
"POST">_PROCESS, <proxy_client_ip>, <url>
```

- Basic with Headers

```
<date_utc> <hostname> [<log_level>] [mws-access][<thread>]  
key:<unique_request_key>, PHASE_<"REQUEST" or "RESPONSE">_<"PRE or  
"POST">_PROCESS, <proxy_client_ip>, <url>
```

```
<header_name>: <header_value>
```

```
<header_name>: <header_value>
```

```
<header_name>: <header_value>
```

- Basic with Headers and Body

```
<date_utc> <hostname> [<log_level>] [mws-access][<thread>]  
key:<unique_request_key>, PHASE_<"REQUEST" or "RESPONSE">_<"PRE or  
"POST">_PROCESS, <proxy_client_ip>, <url>
```

```
<header_name>: <header_value>
```

```
<header_name>: <header_value>
```

<header_name>: <header_value>

(blank line)

<body_content>

Field definitions:

- <date_utc>—The date of the log entry, in UTC.
- <hostname>—The hostname of the appliance.
- <log_level>—The importance level of a log entry. Can be TRACE, DEBUG, INFO, WARN, or ERROR.
- <thread>—The specific thread that is handling the request or response. It might take the form of [se-request-#], where # is the thread number, or [pool-#-thread-#], where # represents the pool and thread number, respectively.
- <unique_request_key>—This is a key used to uniquely identify requests. It can be useful when searching for a specific request in a large file.
- <"REQUEST" or "RESPONSE">—Whether the HTTP packet is a client request, or a server response.
- <"PRE" or "POST">—Whether the HTTP packet is being logged before or after Security Engine processes it (and potentially manipulates it).
- <proxy_client_ip>—The incoming client IP. Since WebApp Secure works around a Nginx proxy, the client IP will most-likely be "127.0.0.1".
- <url>—The full request or response URL.
- <header_name>—The name of a header sent in a request or response. There can be multiple headers in an HTTP packet.
- <header_value>—The value of a header sent in a request or response. There can be multiple headers in an HTTP packet.
- <body_content>—The full content of the body in requests that contain a body. In the case of GET, for example, there will most likely not be a body.

Examples:

Basic

```
Mar 19 21:11:47 webappsecure [INFO][mws-access][se-request-6]
key:12521298-13f1-4019-8e21-c6046cf2dac7/PHASE_REQUEST_PRE_PROCESS,127.0.0.1,http://10.20.0.53:80/
```

Basic with Headers

```
Mar 19 19:48:14 webappsecure [INFO][mws-access][se-request-25]
key:12521298-13f1-4019-8e21-c6046cf2dac7/PHASE_REQUEST_POST_PROCESS,127.0.0.1,http://10.20.0.53:80/
GET / HTTP/1.1 host: 10.20.0.53 x-forwarded-for: 213.85.244.190, 10.20.1.23
x-myk-request-info: http, HTTP/1.1, 80 x-myk-appid: default x-myk-port: 80 x-myk-use-ssl:
false x-myk-ssl: false connection: close user-agent: Mozilla/5.0 (X11 U Linux x86_64 en-us)
```


AppleWebKit/532+ (KHTML, like Gecko) Safari/419.3 Midori/0.1.8 x-myk-access-log-id:
12521298-13f1-4019-8e21-c6046cf2dac7

Basic with Headers and Body

Mar 19 19:48:14 webappsecure [INFO][mws-access][se-request-13]
key:cfde0089-2b93-4bad-a8f5-555ac29ef4b6 PHASE_REQUEST_POST_PROCESS:127.0.0.1 http://10.20.0.53:80/
POST / HTTP/1.1 host: 10.20.0.53 x-forwarded-for: 213.85.244.190, 10.20.1.23
x-myk-request-info: http, HTTP/1.1, 80 x-myk-appid: default x-myk-port: 80 x-myk-use-ssl:
false x-myk-ssl: false connection: close user-agent: tsung content-type:
application/x-www-form-urlencoded content-length: 3009 x-myk-access-log-id:
cfde0089-2b93-4bad-a8f5-555ac29ef4b6
data=bWltZXR5cGVzOjI2MzowLHBsdWdpbnM6MTA5NjowLHRpbWV6b25lOjM6MCxb25zb2xlOjQlQj

Related Documentation

- [Security Log Format on page 281](#)
- [Audit Log Format on page 283](#)
- [Firewall Log Format on page 284](#)
- [Postgres Log Format on page 285](#)
- [mws Log Format on page 286](#)

Security Log Format

Webapp Secure is configured to log security incidents to **mws-security.log**. All security alerts should be sent to security.log (previously named security-alert.log). There are different types of security incidents that will be a part of this log: new profiles, security incidents, new counter responses. The following section explains the format of these security log messages.

- New profile


```
<date_utc> <hostname> [<log_level>][mws-security-alert][<service>]
MKS_Category="New Profile" MKS_ProfileId="<profile_id>"
MKS_ProfileName="<profile_name>" MKS_PubKey="<pubkey>"
```
- Security incidents


```
<date_utc> <hostname> [<log_level>][mws-security-alert][<service>]
MKS_Category="Security Incident" MKS_Type="<incident>" MKS_Severity="<severity>"
MKS_ProfileName="<profile_name>" MKS_SrcIP="<source_ip>"
MKS_PubKey="<pubkey>" MKS_useragent="<user_agent>" MKS_url="<url>"
MKS_count="<count>" MKS_fakeresponse="<fake_response>"
```
- New counter responses


```
<date_utc> <hostname> [<log_level>][mws-security-alert][<service>]
MKS_Category="New Counter Response" MKS_ResponseCode="<response_code>"
MKS_ResponseName="<response_name>" MKS_ProfileId="<profile_id>"
MKS_ProfileName="<profile_name>" MKS_ResponseCreated="<created_date>"
MKS_ResponseDelayed="<delay_date>" MKS_ResponseExpires="<expiration_date>"
MKS_ResponseConfig="<response_config>"
```

Field definitions:

- `<date_utc>`--The date of the log entry, in UTC.
- `<hostname>`--The hostname of the appliance.
- `<log_level>`--The importance level of a log entry. Can be TRACE, DEBUG, INFO, WARN, or ERROR.
- `<service>`--The WebApp Secure service that triggered the security log entry. Possible services include:
 - `[auto-response]`--The auto response service will most likely generate New Counter Response log entries.
 - `[traffic-info]`-- The traffic information service will usually generate New Incident and New Profile log entries in security.log.
- `<profile_id>`--The numerical ID assigned to the Profile that caused the security alert, or the profile ID that received a Response.
- `<profile_name>`--The friendly name assigned to the Profile that caused the security alert, or the Profile that received a Response. For example, "Bob 1234".
- `<pubkey>`-- The Public ID that can be used in conjunction with the Support_Processor to unblock Profiles. For example, "tTtHvXuby4gxNVmPleIE".
- `<incident>`--The name of the incident that triggered this security alert.
- `<severity>`--The numerical severity of the incident that triggered this security alert. This can be a number from 0 to 4, inclusive.
- `<source_ip>`--The IP the request that generated this alert originated from.
- `<user_agent>`--The client's user agent string that generated this alert.
- `<url>`--The client's user agent string that generated this alert.
- `<count>`--The number of times the profile triggered this incident. This is used for certain incidents to decide whether or not to elevate the profile or increase the responses on the profile.
- `<fake_response>`--Whether or not (true or false) the response sent back to the client was a fake one created by WebApp Secure.
- `<response_code>`--The numerical code for the response issued. For example, "13007".
- `<response_name>`--The friendly name for the response issued on the profile indicated in the alert.
- `<created_date>`--The date and time the response was created.
- `<delay_date>`--The date and time the response is set to be delayed until.
- `<expiration_date>`--The date and time the response is set to expire.
- `<response_config>`--The configuration used in this response. Displayed as an XML-like node.

Logfile Example.

```

Mar 19 18:20:04 my-vm [INFO][mws-security-alert][traffic-info] MKS_Category="New
Profile" MKS_ProfileId="197382" MKS_ProfileName="Sandy 5021"
MKS_PubKey="c0tcXdDev0XMwwOu30uD" Mar 19 18:20:04 my-vm
[INFO][mws-security-alert][auto-response] MKS_Category="New Counter Response"
MKS_ResponseCode="SL" MKS_ResponseName="Slow Connection"
MKS_ProfileId="197180" MKS_ProfileName="Rhoda 4027"
MKS_ResponseCreated="2014-03-19 18:20:00.583" MKS_ResponseDelayed="2014-03-19
18:20:00.583" MKS_ResponseExpires="2014-03-20 18:20:00.583"
MKS_ResponseConfig="<config ix0ix4002='1' min='2500' max='6000' />" Mar 19 18:20:05
my-vm [INFO][mws-security-alert][traffic-info] MKS_Category="Security Incident"
MKS_Type="Apache Configuration Requested" MKS_Severity="2"
MKS_ProfileName="Janelle 3524" MKS_SrcIP="10.20.1.23"
MKS_pubkey="ami4U5RExf4d4NO59xxT" MKS_useragent="Mozilla/5.0 (X11 U Linux
x86_64 pl-PL rv:1.9.2.13) Gecko/20101206 Ubuntu/10.04 (lucid) Firefox/3.6.13"
MKS_url="http://10.20.0.53:80/htaccess" MKS_count="1" MKS_fakeresponse="true"

```

Related Documentation

- [Access Log Format on page 279](#)
- [Audit Log Format on page 283](#)
- [Firewall Log Format on page 284](#)
- [Postgres Log Format on page 285](#)
- [mws Log Format on page 286](#)

Audit Log Format

The audit log contains log entries that indicate non-idempotent (state changing) actions performed on WebApp Secure. For example:

- Configuration additions, changes, deletions, insertions
- Manual Response deactivations
- Log in attempts and notices
- Applying license keys
- User permission violations (attempted actions by users that are not allowed to perform such actions) will all be shown in audit.log. This log is a good candidate for regular auditing (hence the name), as it will allow administrators to see various changes or other activity that took place on the appliance, along with identifiers that indicate who took the action.

The format of audit log messages is as follows:

```
<date_utc> <hostname> [mws-audit][<log_level>] [<api_key>] <message>
```

Field definitions:

- **<api_key>** – The key used to perform the action described in the **<message>**.

- `<message>`—The message. Can indicate any of the previously mentioned actions. In the case of logins, an additional field shows the user the person logged in as, as well as the IP they were connecting from.

Logfile Example:

```
Jan 22 16:14:23 my-jwas [mws-audit][INFO] [mykonos] [10.10.0.117] Logged in successfully
Jan 23 19:16:22 my-jwas [mws-audit][INFO] [ea77722a8516b0d1135abb19b1982852]
Deactivate response 1832840420318015488 Feb 7 20:29:51 my-jwas [mws-audit][INFO]
[mykonos] [10.10.0.113] Login failed. Attempt: 1 Feb 14 19:02:54 my-jwas
[mws-audit][INFO][mykonos] Changed configuration parameters:
services.spotlight.enabled, services.spotlight.server_address
```

Related Documentation

- [Access Log Format on page 279](#)
- [Security Log Format on page 281](#)
- [Firewall Log Format on page 284](#)
- [Postgres Log Format on page 285](#)
- [mws Log Format on page 286](#)

Firewall Log Format

This log stores information about dropped packets from the iptables firewall. For various reasons (intentional and/or unintentional) iptables might drop a particular packet. If this happens, the event's information is logged to `firewall.log`. From each log entry, you can find the incoming interface, the outgoing interface, the source address, and other information related to the packet that was dropped. The format of firewall log files is as follows:

```
<date_utc> <hostname> kernel: IPTABLES <event>: <message>
```

Field definitions:

- `<date_utc>`—The date and time in UTC.
- `<hostname>`—The hostname of the machine.
- `<event>`— The event being logged. Currently only exhibits "Dropped" packets, although there is no restriction on the event that can be logged.
- `<message>`— The message, in standard iptables log format.

Example:

```
Mar 19 18:49:32 myjwas kernel: IPTABLES Dropped: IN=eth0 OUT=
MAC=00:0c:29:cf:4d:c8:2c:21:72:c6:99:08:08:00 SRC=10.10.0.117 DST=10.20.0.53 LEN=40
TOS=0x00 PREC=0x00 TTL=63 ID=51749 DF PROTO=TCP SPT=51093 DPT=5000
WINDOW=0 RES=0x00 RST URGP=0 Mar 19 20:56:59 myjwas kernel: IPTABLES Dropped:
IN=eth0 OUT= MAC=ff:ff:ff:ff:ff:00:0c:29:0f:48:ec:08:00 SRC=0.0.0.0
DST=255.255.255.255 LEN=337 TOS=0x10 PREC=0x00 TTL=128 ID=0 PROTO=UDP
SPT=68 DPT=67 LEN=317 Mar 20 11:03:24 myjwas kernel: IPTABLES Dropped: IN=eth0
```

```
OUT= MAC=00:0c:29:cf:4d:c8:2c:21:72:c6:99:08:08:00 SRC=10.10.0.17 DST=10.20.0.53
LEN=52 TOS=0x00 PREC=0x00 TTL=126 ID=18544 DF PROTO=TCP SPT=53543 DPT=443
WINDOW=8192 RES=0x00 SYN URGP=0 Mar 20 11:03:25 myjwas kernel: IPTABLES
Dropped: IN=eth0 OUT= MAC=00:0c:29:cf:4d:c8:2c:21:72:c6:99:08:08:00 SRC=10.10.0.17
DST=10.20.0.53 LEN=52 TOS=0x00 PREC=0x00 TTL=126 ID=18545 DF PROTO=TCP
SPT=53544 DPT=443 WINDOW=8192 RES=0x00 SYN URGP=0 Mar 20 11:03:27 myjwas
kernel: IPTABLES Dropped: IN=eth0 OUT= MAC=00:0c:29:cf:4d:c8:2c:21:72:c6:99:08:08:00
SRC=10.10.0.17 DST=10.20.0.53 LEN=52 TOS=0x00 PREC=0x00 TTL=126 ID=18561 DF
PROTO=TCP SPT=53543 DPT=443 WINDOW=8192 RES=0x00 SYN URGP=0
```

- Related Documentation**
- [Access Log Format on page 279](#)
 - [Security Log Format on page 281](#)
 - [Audit Log Format on page 283](#)
 - [Postgres Log Format on page 285](#)
 - [mws Log Format on page 286](#)

Postgres Log Format

WebApp Secure uses postgres for its permanent data store. postgres.log contains logs of manipulations on the schema of the database, as well as any errors that occurred during database operations. For more information on postgres, go to <http://www.postgresql.org/docs/9.0/static/runtime-config-logging.html>. The format of the postgres log file is as follows:

```
<date_utc> <hostname> postgres[<pid>]: [<group_id>] <sql_error_code> <session_id>
<message_type>: <message>
```

Field definitions:

- <date_utc>—The date and time in UTC.
- <hostname>—The hostname of the machine.
- <pid>—The process ID of the postgres instance.
- <group_id>— A set of two numbers, a major number and a minor number, respectively, that represent a piece of a log. Since postgres logs the statements that generated errors, these statements are issued as separate log lines, and in order to distinguish multiple lines of logs with each other, we look at the time, major group number, and minor group number. For example, if a major group number is 22 and a minor group number is 2, that log entry is the second line of output from a postgres log. To find all other lines that go with this line, look for other lines that were output at the same time with the same major group number. For example, [22-1], [22-3], [22-4] would all be log lines that belong to the line that has [22-2] (provided they happened at the same <date_utc>).
- <sql_error_code>—The SQL error code. Used with the documented table at <http://www.postgresql.org/docs/9.0/static/errcodes-appendix.html> which explains each error code and the meaning.

- <session_id>—A somewhat unique session identifier that can be used to search for specific lines in the log.
- <message_type>—The type of the message. Can be LOG, WARNING, ERROR, or STATEMENT.
- <message>—The message.

Example:

```
Feb 24 14:58:08 webappsecure postgres[7694]: [10-1] 42701 530b5e00.1e0e ERROR:
column "requests" of relation "sessiongroup" already exists Feb 24 14:58:08 webappsecure
postgres[7694]: [10-2] 42701 530b5e00.1e0e STATEMENT: ALTER TABLE sessiongroup
ADD COLUMN requests bigint DEFAULT 0; Feb 24 16:52:41 webappsecure postgres[12501]:
[48-1] 42703 530b7873.30d5 ERROR: column it.description does not exist at character 35
Feb 24 16:52:41 webappsecure postgres[12501]: [48-2] 42703 530b7873.30d5 STATEMENT:
SELECT it.itid, it.code, it.name, it.description FROM incidenttype AS it Mar 4 16:18:00
webappsecure postgres[13931]: [743-1] 01000 5315d18a.366b WARNING: skipping
"pg_tablespace" --- only superuser can vacuum it Mar 5 13:12:40 webappsecure
postgres[21640]: [20-1] 08P01 531647dc.5488 LOG: unexpected EOF on client connection
```

Related Documentation

- [Access Log Format on page 279](#)
- [Security Log Format on page 281](#)
- [Audit Log Format on page 283](#)
- [Firewall Log Format on page 284](#)
- [mws Log Format on page 286](#)

mws Log Format

mws.log is the main log file for most WebApp Secure logging needs. All messages that don't have a specific log location are sent, by default, to mws.log. The format of the mws.log is as follows:

```
<utc_date><hostname>[<log_level>][<service_name>][<service_component>]<log_message>
```

Field definitions:

- <utc_date>—The date of the log entry, in UTC.
- <log_level>—The importance level of a log entry. Can be TRACE, DEBUG, INFO, WARN, or ERROR.
- <service_name>—The WebApp Secure service that generated the log entry. Possible service names include:

- `mws-cluster-services` -- Various smaller services that don't warrant a completely separate service are logged within this context. The term 'Cluster Services' is used as a way to reference certain services that should only have one instance in the case of a clustered WebApp Secure configuration. This is different from 'Local Services', described below.
- `mws-services` -- All local services are logged through this context. A 'Local Service' (service for short) is any service that must be running on each instance of WebApp Secure in a clustered configuration.
- `mws-security-engine` -- Messages that don't belong to a specific service, but rather deal with the core engine of WebApp Secure itself.
- `mws-ui` -- All messages that are sent from the UI. Things like spawning UI HTTP worker threads and handling UI functionality are parts of this service.
- `mws-updates` -- All messages that are sent during an upgrade. This service facilitates migrating from an older version of WebApp Secure to a newer version.
- `mws-backups` -- Messages that are sent from the backup service. This service creates backups automatically.
- `mws-reports-api` -- Messages that are sent from the reporting service. This service is responsible for running both on-demand and scheduled reports.
- `mws-pyro` -- Pyro is used for interprocess communication.
- `<service_component>` -- The specific component that is issuing the log message. There are many components, but a few of the major services and their components are listed here:
 - `[mws-cluster-services][db-cleanup]` -- The DB cleanup service deletes traffic information stored in the database after they reach their configurable expiration date (specified in WebApp Secure Configuration). The information available to this service includes statistics information, malicious traffic, and non-malicious traffic. Each type of information has their own separate configuration.
 - `[mws-cluster-services][auto-response]` -- The auto response service is the service responsible for delivering automated responses to profiles that have activated a set of incidents. The rules in which the auto response service are activated against can be turned on and off through the Web UI under Configuration > Response Rules.
 - `[mws-cluster-services][traffic-info]` -- The traffic information service is used to control the requests and responses within the WebApp Secure system. Incoming requests are put into a processing queue and pulled off for processing in chunks by the traffic info service.
- `<log_message>` -- The message. This can be anything, but usually contains information to help you narrow down problems or confirm certain events have occurred as they should.



NOTE: Some log entries might not have an applicable service or component, like core security engine log messages. In this case, the fields are not displayed.



NOTE: Due to complications, currently all log entries with [mws-ui] do not have a <log_level>.

Example:

Mar 19 18:42:38 my-jwas-instance [INFO][mws-cluster-services][db-cleanup] Database cleanup completed. Removed record count: 0 Mar 19 19:42:16 my-jwas-instance [mws-ui]: spawned uWSGI worker 1 (pid: 11209, cores: 1) Mar 19 20:18:26 my-jwas-instance [INFO][mws-security-engine] Server startup in 3080 ms

**Related
Documentation**

- [Access Log Format on page 279](#)
- [Security Log Format on page 281](#)
- [Audit Log Format on page 283](#)
- [Firewall Log Format on page 284](#)

APPENDIX C

RBAC Groups and Roles

- [RBAC Groups and Roles on page 289](#)

RBAC Groups and Roles

This is a list of all WebApp Secure roles, and their corresponding permissions.

Table 47: RBAC Groups and Roles.

	Super Administrator	Security Administrator	Security Support Staff	RBAC Administrator	Web UI Administrator	Device Administrator	Security User
Can Manage Processors	Yes	Yes	No	No	Yes	No	No
Can Manage Response Rules	Yes	Yes	No	No	Yes	No	No
Can View System Status	Yes	Yes	Yes	No	Yes	Yes	Yes
Can Edit Profiles	Yes	Yes	No	No	Yes	No	Yes
Can Use Expert Mode	Yes	No	No	No	Yes	No	No
Can Delete Profiles	Yes	Yes	No	No	Yes	No	Yes
Can Manage Logical Services	Yes	Yes	No	No	Yes	Yes	No
Can View Security Data	Yes	Yes	Yes	No	Yes	No	Yes
Can Manage Licensing	Yes	No	No	No	Yes	Yes	No

Table 47: RBAC Groups and Roles. *(continued)*

	Super Administrator	Security Administrator	Security Support Staff	RBAC Administrator	Web UI Administrator	Device Administrator	Security User
Can Manage Authentication	Yes	No	No	No	Yes	Yes	No
Can Manage Applications	Yes	No	No	No	Yes	No	No
Can Import Configuration	Yes	No	No	No	Yes	No	No
Can Initialize Appliance	Yes	No	No	No	No	Yes	No
Can Log Into Web UI	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Can Manage Backups	Yes	No	No	No	Yes	Yes	No
Can Activate Responses	Yes	Yes	No	No	Yes	No	Yes
Can Export Configuration	Yes	No	No	No	Yes	No	No
Can Manage Physical Services	Yes	No	No	No	No	Yes	No
Can Manage Logging	Yes	No	No	No	Yes	Yes	No
Can Manage Spotlight	Yes	Yes	No	No	Yes	No	No
Can Deactivate Responses	Yes	Yes	No	No	Yes	No	Yes
Can Log Into Console	Yes	No	No	No	No	Yes	No
Can Schedule Reports	Yes	Yes	No	No	Yes	No	Yes
Can Update Appliance	Yes	No	No	No	Yes	Yes	No

Table 47: RBAC Groups and Roles. *(continued)*

	Super Administrator	Security Administrator	Security Support Staff	RBAC Administrator	Web UI Administrator	Device Administrator	Security User
Can Manage High Availability	Yes	No	No	No	Yes	Yes	No
Can Configure Updates	Yes	No	No	No	No	Yes	No
Can Manage Security Engine	Yes	Yes	No	No	Yes	No	No
Can Run Reports	Yes	Yes	Yes	No	Yes	No	Yes
Can Manage Authorization	Yes	No	No	Yes	Yes	No	No
Can Manage SRX series Settings	Yes	No	No	No	No	Yes	No
Can Restart Appliance	Yes	No	No	No	No	Yes	No

PART 3

Index

- [Index on page 295](#)

Index

Symbols

#, comments in configuration statements.....	xiii
(), in syntax descriptions.....	xiii
< >, in syntax descriptions.....	xiii
[], in configuration statements.....	xiii
{ }, in configuration statements.....	xiii
(pipe), in syntax descriptions.....	xiii

A

activity processors.....	149
application cookie manipulation.....	154
auth cookie tampering.....	151
auth input parameter tampering.....	150
auth invalid login.....	152
auth query parameter tampering.....	151
authentication brute force.....	152
common directory enumeration.....	162
cookie protection processor.....	153
duplicate request header.....	165
duplicate response header.....	166
error processor.....	154
header processor.....	164
illegal method requested.....	171
illegal request header.....	166
illegal response header.....	167
illegal response status.....	159
method processor.....	170
missing all headers.....	167
missing host header.....	168
missing http protocol.....	173
missing request header.....	168
missing response header.....	169
missing user agent header.....	169
request header overflow.....	169
resource enumeration.....	163
suspicious response status.....	160
unexpected method requested.....	172
unexpected request header.....	170
unexpected response status.....	160
unknown common directory requested.....	161
unknown http protocol.....	173

unknown user directory requested.....	161
user directory enumeration.....	162
Akamai Dynamic Site Accelerator	
configure support.....	49
alert service.....	63
alerts.....	269
appliance	
initial configuration.....	24
terminology.....	23
appliance deployment.....	11
applications	
edit.....	58
new.....	56
patterns.....	58
architecture and key components	
services.....	6
assigning the instance and IP	
CLI.....	19
verify.....	20
web interface.....	19
attacker profile page.....	258
Attackers screen.....	257

B

backend servers	
define.....	60
backup	
restore.....	107
backup and recovery	
overview.....	105
basic configuration mode	
available sections.....	46
braces, in configuration statements.....	xiii
brackets	
angle, in syntax descriptions.....	xiii
square, in configuration statements.....	xiii

C

captcha template.....	275
CLI	
config context.....	86
configuration level commands.....	81
general and base commands.....	77
import/export.....	88
initialize configuration.....	88
navigating.....	74
overview.....	74
proxy exclusion.....	89
set command.....	75

set config parameter.....	87
system level commands.....	84
cluster	
configure.....	34
clustering overview.....	20
comments, in configuration statements.....	xiii
configuration	
DNS.....	27
first time.....	24
hostname.....	27
initialization.....	27
network interface.....	26
web interface.....	44
Configuration	
Web UI.....	269
configuration wizard.....	35
using.....	36
conventions	
text and syntax.....	xii
counter response	
overview.....	229
Counter Responses window.....	264
curly braces, in configuration statements.....	xiii
customer support.....	xiv
contacting JTAC.....	xiv
D	
Dashboard.....	252
dedicated management interface.....	26
Developers Guide.....	45
DNS settings.....	27
documentation	
comments on.....	xiii
E	
EC2 deployment	
CLI.....	14
overview.....	14
web interface.....	15
Editor	
overview.....	232
expert configuration mode	
details.....	47
F	
features and benefits.....	4
first time configuration.....	24
font conventions.....	xii
H	
health check URL.....	97
high availability	
network failure detection.....	95
overview.....	21
settings.....	32
update.....	30
high availability modes.....	27
honeypot processors	
access policy processor.....	125
ajax processor.....	128
apache configuration requested.....	132
apache password file requested.....	133
basic authentication brute force.....	136
basic authentication processor.....	130
cookie parameter manipulation.....	138
cookie processor.....	137
file processor.....	139
hidden input form processor.....	141
hidden link processor.....	144
hidden parameter manipulation.....	142
invalid credentials.....	133
link directory indexing.....	145
link directory spidering.....	145
malicious resource request.....	146
malicious script execution.....	129
malicious script introspection.....	130
malicious service call.....	126
malicious spider activity.....	148
parameter type manipulation.....	143
password cracked.....	135
protected resource requested.....	134
query parameter manipulation.....	147
query string processor.....	146
robots processor.....	148
service directory indexing.....	126
service directory spider.....	127
suspicious file exposed.....	140
suspicious filename.....	139
suspicious resource enumeration.....	141
hostname spoofing attempt.....	124
hostname, setting.....	27
HTTP request/response diagram.....	8
I	
import/export.....	48
Incident details.....	263
incident methods	
list.....	233

Incidents.....	262
initialize system.....	27

L

LDAP	
settings.....	70
license	
add.....	29
limitations.....	10
load-balanced environments	
options.....	13
log files	
destinations.....	104
managing and viewing.....	103
log format	
access.....	279
firewall.....	284
incident.....	283
mws.....	286
postgres.....	285
security.....	281
login ban	
unblock.....	97

M

manuals	
comments on.....	xiii
methodology.....	4
multiple web servers	
securing.....	55

N

network interface configuration	
management interface.....	26
network placement.....	11
NTP service.....	62

O

online help	
question mark.....	45

P

pages.....	62
parentheses, in syntax descriptions.....	xiii
password	
change.....	25
reset.....	25

processors	
complexity ratings.....	122
overview.....	122
product documentation.....	45
product overview.....	3
proxy exclusion.....	89
proxy-backends	
configure.....	54

R

RADIUS	
settings.....	70
RBAC	
list of groups and roles.....	289
report	
details.....	244
history.....	243
schedule	241
schedule overview.....	240
report types.....	246
reporting	
on-demand.....	238
overview.....	237
Reports.....	268
response processors.....	184
app vulnerability detected.....	215
application vulnerability processor.....	214
bad captcha answer.....	192
block processor.....	185
captcha answer automation.....	189
captcha cookie manipulation.....	196
captcha directory indexing.....	199
captcha directory probing.....	200
captcha disallowed multipart.....	198
captcha image probing.....	197
captcha parameter manipulation.....	201
captcha request replay attack.....	202
captcha request size limit exceeded.....	198
captcha request tampering.....	194
captcha signature spoofing.....	196
captcha signature tampering.....	195
clippy processor.....	217
csrf parameter tampering.....	208
csrf processor.....	205
csrf remote script inclusion.....	209
expired captcha request.....	193
force logout processor.....	211
Google map processor.....	228
header injection processor.....	211

http referers disabled.....	210	sessions.....	265
login processor.....	218	Spotlight Connector	
mismatched captcha session.....	193	about.....	109
multiple captcha parameter		cookies and location.....	110
manipulation.....	204	Spotlight Secure	
multiple captcha disallow multipart.....	204	about.....	112
multiple captcha replays.....	203	enable.....	113
multiple captcha request overflow.....	190	SRX series integration	
multiple csrf parameter tampering.....	209	configure.....	66
no captcha answer provided.....	190	create filter and terms.....	65
request captcha processor.....	186	filters and terms overview.....	64
site invalid login.....	224	overview.....	63
site login brute force.....	227	test.....	68
site login multiple ip.....	224	SSL to client	
site login multiple usernames.....	225	enable.....	60
site login user brute force.....	226	SSL traffic considerations.....	13
site login user pooling.....	226	statistics.....	92
site login user sharing.....	225	support, technical See technical support	
site login username scan.....	227	syntax conventions.....	xii
slow connection processor.....	212	System Status screen.....	269
strips input processor.....	212	system updates.....	90
support processor.....	215		
unsupported audio captcha requested.....	191	T	
warning code tampering.....	214	technical support	
warning processor.....	213	contacting JTAC.....	xiv
Responses tab.....	258	third-party load balancer.....	41
role-based access control.....	69	tracking processors	
configure.....	70	beacon parameter tampering.....	176
		beacon session tampering.....	177
S		client beacon processor.....	175
Search window.....	266	client classification processor.....	182
secure cluster		client fingerprint processor.....	177
update.....	35	etag beacon processor.....	174
security engine		fingerprint directory indexing.....	180
Content Delivery Network (CDN).....	48	fingerprint directory probing.....	181
incident monitoring.....	50	fingerprint manipulation.....	181
server identity and cloaking.....	52	session etag spoofing.....	174
traffic.....	52	traffic.....	48
whitelist settings.....	53		
Security Intelligence		U	
about.....	107	updates	
security processors		initial.....	35
overview.....	124	Updates screen.....	271
self-healing.....	98	user preferences.....	44
self-monitoring.....	98		
configuration variables.....	98	V	
server identity and cloaking.....	48	verify connectivity.....	28
session cookie tampering.....	123	verify installation.....	42
session details.....	265	verify instance is running.....	20

W

web interface configuration.....	44
widgets	
data filter.....	251
search.....	251
user.....	251

