TEST METHODOLOGY

Web Application Firewall (WAF)

v2.1

Table of Contents

1 Introduction

1.1 The Need for Web Application Firewalls

NSS Labs defines web application firewalls (WAFs) as stand-alone or virtual appliances, or as self-contained software designed specifically to secure web-based traffic. WAFs employ a wide range of functions to work in conjunction with perimeter firewalls and intrusion prevention system (IPS) technologies and provide protections specifically for web applications. WAFs should include HTTP/HTTPS protocol enforcement and native signature detection along with other protection mechanisms, such as URL normalization and scanning; positive or negative security enforcement model functionality (or both) that enforces proper application operation and page logic flow; and adaptive learning modules for automated policy updates. WAFs block attacks masked by HTTPS encryption by inspecting SSL sessions using a web server's private key; they also detect policy violations and reset offending connections. These sessions are either passively decrypted and inspected or actively terminated and re-encrypted. WAFs should be able to identify and police the use of specific web application elements and functions, such as web objects, form fields, and, most importantly, application session logic.

This methodology describes how NSS will evaluate WAF products to provide an objective and fair assessment of the technology. Individual tests have been developed that represent real-world use cases of the WAF to protect web applications and other mission-critical applications sitting at the edge of an enterprise network.

1.2 About This Test Methodology

NSS Test Reports are designed to address the challenges faced by IT professionals in selecting and managing security products. The scope of this particular methodology includes:

- Security effectiveness
- Performance
- Stability and reliability
- Total cost of ownership (TCO)

As organizations come to rely on WAF technology to protect critical assets, the stability and reliability of WAF solutions is imperative. Therefore, regardless of any new deep inspection capabilities, a significant requirement of any WAF technology is that it must be as stable, as reliable, as fast, and as flexible as the network it is protecting.

As part of its WAF test, NSS subjects each product to a battery of tests that verify the stability and performance of each device tested, determine the accuracy of its security coverage, and ensure that the device will not block legitimate traffic. To assess the complex matrix of WAF performance and security requirements, NSS has developed a specialized lab environment that is designed to properly exercise a WAF product. The test suite contains a large variety of individual tests that evaluate the performance, reliability, security effectiveness, and usability of WAF products, providing the most complete evaluation of WAF products available anywhere today.

Based on NSS' research, most enterprises have a broad range of applications, some of which are typically expected to be protected by a WAF, including but not limited to:

- HTTP support is fundamental to most applications that are hosted by enterprises today. For this reason, the WAF product is expected to support the following versions of HTTP:
  - HTTP/0.9
  - HTTP/1.0
  - HTTP/1.1
- Java Script Object Notation (JSON)
- Asynchronous Javascript and XML (AJAX)
- Extensible Markup Language (XML)
- Javascript (JS)

WAF products offer various deployment options that may impact their ability to provide adequate security effectiveness:

- **Inline deployment options**
  - **Transparent bridge** – The WAF does not take an active role in intercepting the traffic. The WAF inspects traffic and defeats attacks by blocking the malicious connection. In this scenario, the WAF cannot actively modify the connection between the client and the web application.
  - **Transparent reverse proxy** – The WAF takes an active role in intercepting all communications between the client and web application by inspecting both the requests and responses prior to passing the HTTP communications to the protected web application or the client. In this scenario, the client and the protected web application are communicating directly with each other, and the WAF is not visible to either the client or the web application. The reverse proxy WAF can actively modify the communications between the client and the protected web application, allowing for changes such as sanitization of potentially sensitive information from being disclosed (i.e., credit card numbers).
  - **Reverse proxy** – As with the transparent reverse proxy, the reverse proxy WAF takes an active role in intercepting all communications. The main difference between this scenario and transparent reverse proxy is that the WAF is no longer transparent. The client sends its HTTP transactions directly to the WAF and the WAF communicates directly with the web application, and vice versa. In this scenario, the WAF can still actively modify communications between the client and protected web application as well as offer other advantages, such as masking the true IP address of the protected web application.
- **Out-of-band deployment options**
  - **Passive** – The WAF does not take an active role in the traffic flow between the client and the protected web application. Instead, the WAF is deployed on a SPAN port that is configured to mirror the traffic to be inspected. Some WAF products that offer this mode may also offer the ability to transmit a TCP RST as an attempt to block malicious attacks. However, in this model, race conditions can occur (with logic dependent on sequencing and thus protection), and not all attacks will be successfully mitigated.

1.3 Inclusion Criteria

NSS invites all vendors claiming WAF capabilities to submit products at no cost. Vendors with major market share, as well as challengers with new technology, will be included.

1.4 Deployment

WAF products should be implemented as a transparent bridge or a reverse proxy. Products should be supplied as a single appliance where possible, with the appropriate number of physical interfaces capable of achieving the required level of connectivity and performance (minimum of one inline port pair per gigabit of throughput with a maximum of 10 gigabits per second).

Once installed in the test environment, the product will be configured for the use case appropriate to the target deployment (for example, an e-commerce data center). As such, the WAF should be configured to block all traffic when resources are exhausted or when traffic cannot be analyzed for any reason.

2 Product Guidance

NSS issues summary product guidance based on evaluation criteria that is important to information security professionals. The evaluation criteria are weighted as follows:

- **Security effectiveness –** The purpose of a WAF is to separate internal trusted networks from external untrusted networks through policy and routing, and to identify and block attacks against assets, while allowing select controlled traffic to flow between trusted and untrusted networks.
- **Resistance to evasion –** Failure in any evasion class permits attackers to circumvent protection.
- **Stability –** Long-term stability is particularly important for an in-line device, where failure can produce network outages.
- **Performance –** Correctly sizing a WAF is essential.
- **Value –** Customers should seek low TCO and high effectiveness and performance rankings.

Products are listed in rank order according to their guidance rating.

2.1 Recommended

A *Recommended* rating from NSS indicates that a product has performed well and deserves strong consideration. Only the top technical products earn a *Recommended* rating from NSS, regardless of market share, company size, or brand recognition.

2.2 Neutral

A *Neutral* rating from NSS indicates that a product has performed reasonably well and should continue to be used if it is the incumbent within an organization. Products that earn a *Neutral* rating from NSS deserve consideration during the purchasing process.

2.3 Caution

A *Caution* rating from NSS indicates that a product has performed poorly. Organizations using one of these products should review their security posture and other threat mitigation factors, including possible alternative configurations and replacement. Products that earn a *Caution* rating from NSS should not be short-listed or renewed.

3 Security Effectiveness

The aim of this section is to verify that the device under test (DUT) is capable of detecting, preventing, and logging attack attempts accurately, while remaining resistant to false positives.

The DUT will be configured on site by the vendor to protect the target websites, either by "training" the DUT—walking through the e-commerce sites (automatically, or manually)—or by manually creating rule sets and a security policy. NSS considers it unacceptable for a product of this nature to be sold without some standard approach and/or recommended settings, or without consultancy included to create a policy specific to the target environment. The product version tested must be available to the general public at the time of testing.

3.1 Attack Types

NSS testing demonstrates the effectiveness of the DUT in protecting vulnerable web application servers against targeted exploitation. This asset/target and threat-based approach forms the basis on which the security effectiveness of the DUT is measured.

The *NSS Exploit Library* for *WAF* contains publically available exploits (including multiple variants of each exploit) and a number of complex web applications that have been constructed to include known vulnerabilities and coding errors. Each exploit has been validated to impact the target vulnerable host(s) by compromising either the underlying OS, the web server, or the web application itself. A compromise may include executing a denial-of-service (DoS); providing adminstrator/root access to the host server; allowing malicious users to amend system parameters or application data before submission; giving the attacker the ability to browse and/or retrieve files stored on the host server; escalating user privileges.

3.2 OWASP Top 10[1]

The OWASP Top 10 represents a broad industry consensus about the most critical web application security flaws. The following OWASP tests will be conducted:

3.2.1 OWASP Category – Injection

3.2.1.1 SQL Injection

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

3.2.1.2 LDAP Injection

An LDAP injection attack exploits web-based applications that construct LDAP statements based on user input. This could result in the execution of arbitrary commands such as granting permissions to unauthorized queries, and content modification inside the LDAP tree.

[1] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

3.2.1.3 XML Injection

An XML injection attack attempts to inject an XML document into an application. If the XML parser fails to contextually validate data, then the test will yield a positive result.

3.2.1.4 SSI Injection

In an SSI injection attack, data is injected into the application and then interpreted by SSI mechanisms. A successful exploitation of this vulnerability allows an attacker to inject code into HTML pages or even perform remote code execution.

3.2.1.5 XPATH Injection

Similar to SQL injection, an XPath injection attack occurs when a web site uses user-supplied information to construct an XPath query for XML data. By sending intentionally malformed information to the web site, attackers can find out how the XML data is structured, or access data that they may not typically have access to.

3.2.1.6 SMTP Injection

This threat affects all applications that communicate with mail servers (IMAP/SMTP), typically webmail applications. The aim of this test is to verify whether IMAP/SMTP commands can be injected into mail servers if data has not been properly sanitized.

3.2.1.7 Code Injection

A code injection attack differs from a command injection attack in that an attacker is limited only by the functionality of the injected language itself. If an attacker is able to inject PHP code into an application and have it executed, he is limited only by what PHP is capable of.

3.2.1.8 Command Injection

Command injection attacks are possible when an application passes unsafe user-supplied data (for example, forms, cookies, HTTP headers) to a system shell. In this attack, attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application.

3.2.1.9 Buffer Overflow:

By sending specific input to a web application, an attacker can cause the web application to execute arbitrary code and possibly take over the machine.

3.2.2 OWASP Category – Weak Authentication and Session Management

3.2.2.1 Account Enumeration and Guessable User Account

This test verifies if it is possible to collect valid usernames by interacting with the authentication mechanism of an application. This test will be useful for brute force testing, in which NSS verifies whether or not it is possible to find out the corresponding password for a valid username.

3.2.2.2 Credentials Transported Over HTTP

This test verifies whether a user's authentication data is being transferred via an unencrypted channel to avoid being intercepted by malicious users.

3.2.2.3 Privilege Escalation

Privilege escalation occurs when users acquire access to more resources or functionality than they are normally allowed, and such elevation or changes should have been prevented by the application.

3.2.2.4 Session Fixation

A session fixation attack permits an attacker to hijack a valid user session. The attack exploits a limitation in the way the web application manages the session ID, more specifically the vulnerable web application.

3.2.2.5 Session Timeout

Session timeout occurs when a user does not perform any actions on a web site for an interval of time (defined by the web server). On the server side, the status of the user session is changed to “invalid” (i.e., "not used anymore”) and the web server is instructed to destroy the session (all data is deleted).

3.2.3 OWASP Category – Cross-Site Scripting

3.2.3.1 Cross-Site Scripting

Cross-site scripting (XSS) attacks occur when an attacker uses a web application to send malicious code (typically in the form of a browser side script) to a different end user.

3.2.3.2 HTML Injection

The HTML injection attack is closely related to cross-site scripting (XSS). The difference is not in the vulnerability, but in the type of attack that leverages the vulnerability. While XSS uses script tags to run JavaScript, HTML injection simply uses HTML to modify the page for malicious reasons.

3.2.4 OWASP Category – Insecure Direct Object Reference

3.2.4.1 Insecure Direct Object Reference

Insecure direct object references occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability, attackers can bypass authorization and access resources in the system directly, for example, database records or files.

3.2.4.2 Local and Remote File Inclusion

Local and remote file inclusion vulnerabilities allow an attacker to include a file, usually by exploiting the “dynamic file inclusion” mechanisms implemented in the target application. The vulnerability occurs when user-supplied input is used without proper validation.

3.2.5 OWASP Category – Security Misconfiguration

3.2.5.1 Fingerprint Web Server

Information about the web server can be derived by sending it specific commands and then analyzing the output, as each version of a web server’s software may respond differently to these commands.

3.2.5.2 Fingerprint Web Application Framework:

3.2.5.3 Fingerprint Web Application:

3.2.5.4 HTTP Methods

Many HTTP methods are designed to aid developers in deploying and testing HTTP applications. These methods can be used for nefarious purposes if the web server is misconfigured. Additionally, cross-site tracing (XST), a form of cross site scripting using the server's HTTP TRACE method, is examined.

3.2.5.5 Server-Side Request Forgery

A server-side request forgery occurs when an attacker influences a network connection made by the application server. The network connection will originate from the application server internal IP, and the attacker uses this connection to bypass network controls and scan or attack internal resources that are not otherwise exposed.

3.2.6 OWASP Category – Sensitive Data Exposure

3.2.6.1 Insufficient TLS

Even if high-grade ciphers are in use, a server misconfiguration can be used to force the use of a weak cipher, or even worse, force the transmission of data with no encryption, allowing an attacker to gain access to what has been assumed to be a secure communication channel.

3.2.6.2 Heartbleed

Heartbleed (CVE-2014-0160), is a vulnerability in OpenSSL that results from improper input limits and validation of those inputs when a bounds check is missing in the implementation of the TLS/DTLS heartbeat extension. Exploitation of this vulnerability allows an attacker to retrieve 64 KB of memory at a time from an application, which for web applications often includes credentials and other sensitive information used in the recent past.

3.2.7 OWASP Category – Missing Function Level Access Control

3.2.7.1 Directory Traversal/File Include

In web servers and web applications, this vulnerability enables attackers to read directories or files that are not intended for public access. This includes access to data outside the web document root or scripts and files from external websites.

3.2.7.2 Bypassing Authorization Schema

This test verifies how the authorization schema has been implemented in order for each role or privilege to gain access to reserved functions and resources.

3.2.8 OWASP Category – Cross-Site Request Forgery

A cross-site request forgery (CSRF) attack forces end users to execute unwanted actions on web applications in which they are currently authenticated.

3.2.9 OWASP Category – Using Components with Known Vulnerabilities

3.2.9.1 Denial of Service

The denial-of-service (DoS) attack makes a resource (site, application, server) unavailable for the purpose for which it was designed.

3.2.9.2 Shellshock

Shellshock is a bash vulnerability that is found in most Linux, Unix, and OSX operating systems and that can allow an attacker to remotely execute commands without authentication, which could lead to the takeover of remote operating systems and confidential data.

3.2.9.3 PHP CGI Remote Code Execution

A vulnerability has been discovered in the PHP that could allow an attacker to remotely disclose source code and potentially execute arbitrary code.

3.2.10 OWASP Category – Unvalidated Redirects and Forwards

3.2.10.1 Client-Side URL Redirect

This vulnerability occurs when a web application accepts untrusted input containing a URL value that has not been sanitized. This URL could cause the application to redirect the user to another page, for example, a malicious page that is being controlled by the attacker.

3.3 Virtual Patching

Virtual patching should be consistent and repeatable. The following workflow mimics the industry-accepted practice for conducting IT incident response: preparation, identification, analysis, virtual patch creation, implementation/testing, and recovery/follow-up.

This test will provide a CVE, which the DUT will be expected to protect against.

3.4 Evasion

There will not be a separate scoring metric for evasions. Instead, all evasions will be embedded within the tests listed in section 3.2 according to their attack method.

3.5 SSL Encryption

SSL encryption is often used to encrypt the communications between a browser and applications. In order to inspect and protect an application, a WAF must be able to access the unencrypted data stream.

- **SSL access via termination**: The WAF decrypts the encrypted traffic to gain access to the HTTP data. The communication between the WAF and the web server can be in plain text, or it can be SSL-encrypted using ciphers such as but not limited to:
  - AES128-SHA
  - AES256-SHA
- **Passive decryption**: Configured with a copy of the web server's SSL private key, the WAF decrypts the SSL traffic. The original data stream travels unaffected to the web servers, where it is separately decrypted and processed.

3.6 False Positive Testing

The ability of the DUT to identify and allow legitimate traffic while maintaining protection against attacks and exploits is as important as its ability to protect against malicious content. This test will include a varied sample of legitimate application traffic that should properly be identified and allowed.

4 Performance

This section measures the performance of the DUT using various traffic conditions that provide metrics for real-world performance. Individual implementations will vary based on usage;however, these quantitative metrics provide a gauge as to whether a particular DUT is appropriate for a given environment.

4.1 Maximum Capacity

The use of traffic generation appliances allows NSS engineers to create true "real-world" traffic at multi-gigabit speeds as a background load for the tests.

The goal of these tests is to stress the inspection engine and determine how it handles high volumes of TCP connections per second, application layer transactions per second, and concurrent open connections. All packets contain valid payload and address data, and these tests provide an excellent representation of a live network at various connection/transaction rates.

Note that in all tests, the following critical "breaking points"—where the final measurements are taken—are used:

- **Excessive concurrent TCP connections** – Latency within the WAF is causing unacceptable increase in open connections
- **Excessive concurrent HTTP connections** – Latency within the WAF is causing excessive delays and increased response time
- **Unsuccessful HTTP transactions** – Normally, there should be zero unsuccessful transactions. Once these appear, it is an indication that excessive latency within the WAF is causing connections to time out

4.1.1 Maximum HTTP Connections per Second

This test is designed to determine the maximum TCP connection rate of the DUT with a 1-byte HTTP response size. The response size defines the number of bytes contained in the body, excluding any bytes associated with the HTTP header. A 1-byte response size is designed to provide a theoretical maximum HTTP connections per second rate.

Client and server are using HTTP 1.0 without keep-alive, and the client will open a TCP connection, send one HTTP request, and close the connection. This ensures that all TCP connections are closed immediately upon the request being satisfied; thus, any concurrent TCP connections will be caused purely as a result of latency the DUT introduces on the network. Load is increased until one or more of the breaking points defined earlier is reached.

4.1.2 Maximum HTTP Transactions per Second

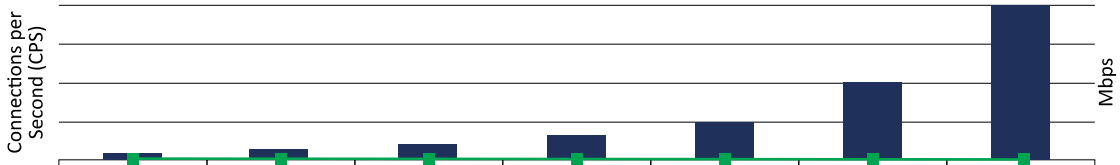
This test is designed to determine the maximum HTTP transaction rate of the DUT with a 1-byte HTTP response size. The object size defines the number of bytes contained in the body, excluding any bytes associated with the HTTP header. A 1-byte response size is designed to provide a theoretical maximum connections per second rate.

Client and server are using HTTP 1.1 with persistence, and the client will open a TCP connection, send ten HTTP requests, and close the connection. This ensures that TCP connections remain open until all ten HTTP transactions are complete, thus eliminating the maximum connection per second rate as a bottleneck (one TCP connection = 10 HTTP transactions). Load is increased until one or more of the breaking points defined earlier is reached.

4.2 HTTP Capacity without Caching and without Transaction Delays

The goal of these tests is to stress the HTTP detection engine and determine how the DUT copes with network loads of varying average packet size and varying connections per second. By creating genuine session-based traffic with varying session lengths, the DUT is forced to track valid TCP sessions, thus ensuring a higher workload than for simple packet-based background traffic. This provides a test environment that is as close to "real world" as it is possible to achieve in a lab environment, while ensuring absolute accuracy and repeatability.

Each transaction consists of a single HTTP GET request ,and there are no transaction delays (i.e., the web server responds immediately to all requests). All packets contain valid payload (a mix of binary and ASCII objects) and address data, and this test provides an excellent representation of a live network (albeit one biased towards HTTP traffic) at various network loads.



	1,000 KB response	500 KB response	44 KB response	21 KB response	10 KB response	4.5 KB response	1.7 KB response
CPS	115	235	2,500	5,000	10,000	20,000	40,000
Mbps	1,000	1,000	1,000	1,000	1,000	1,000	1,000

4.2.1 1,000 KB HTTP Response Size – 115 Connections per Second

Maximum 115 new connections per second per Gigabit of traffic with a 1,000 KB HTTP response size—maximum 59,500 packets per second per Gigabit of traffic. With relatively low connection rates and large packet sizes, all hosts should be capable of performing well throughout this test.

4.2.2 500 KB HTTP Response Size – 235 Connections per Second

Maximum 235 new connections per second per Gigabit of traffic with a 500 KB HTTP response size—maximum 60,000 packets per second per Gigabit of traffic. With relatively low connection rates and large packet sizes, all hosts should be capable of performing well throughout this test.

4.2.3 44 KB HTTP Response Size – 2,500 Connections per Second

Maximum 2,500 new connections per second per Gigabit of traffic with a 44 KB HTTP response size—maximum 140,000 packets per second per Gigabit of traffic. With relatively low connection rates and large packet sizes, all hosts should be capable of performing well throughout this test.

4.2.4 21 KB HTTP Response Size – 5,000 Connections per Second

Maximum 5,000 new connections per second per Gigabit of traffic with a 21 KB HTTP response size – maximum 185,000 packets per second per Gigabit of traffic. With average connection rates and average packet sizes, this is a good approximation of a real-world production network, and all hosts should be capable of performing well throughout this test.

4.2.5 10 KB HTTP Response Size – 10,000 Connections per Second

Maximum 10,000 new connections per second per Gigabit of traffic with a 10 KB HTTP response size—maximum 225,000 packets per second per Gigabit of traffic. With smaller packet sizes coupled with high connection rates, this represents a very heavily used production network.

4.2.6 4.5 KB HTTP Response Size – 20,000 Connections per Second

Maximum 20,000 new connections per second per Gigabit of traffic with a 4.5 KB HTTP response size—maximum 300,000 packets per second per Gigabit of traffic. With small packet sizes and extremely high connection rates, this is an extreme test for any host.

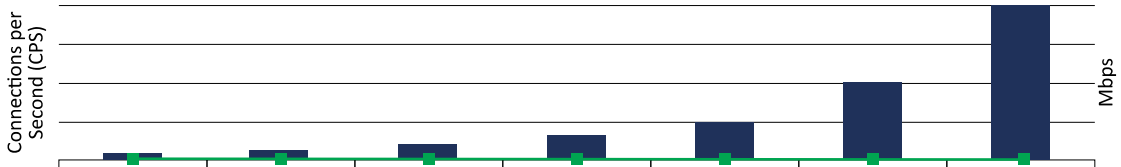
4.2.7 1.7 KB HTTP Response Size – 40,000 Connections per Second

Maximum 40,000 new connections per second per Gigabit of traffic with a 1.7 KB HTTP response size—maximum 445,000 packets per second per Gigabit of traffic. With small packet sizes and extremely high connection rates, this is an extreme test for any host.

4.3 HTTP Capacity with Caching and without Transaction Delays

The goal of these tests is to stress the HTTP detection engine and determine how the DUT copes with network loads of varying average packet size and varying connections per second. By creating genuine session-based traffic with varying session lengths, the DUT is forced to track valid TCP sessions, thus ensuring a higher workload than for simple packet-based background traffic. This provides a test environment that is as close to "real world" as it is possible to achieve in a lab environment, while ensuring absolute accuracy and repeatability.

Each transaction consists of a single HTTP GET request ,and there are no transaction delays (i.e., the web server responds immediately to all requests). All packets contain valid payload (a mix of binary and ASCII objects) and address data, and this test provides an excellent representation of a live network (albeit one biased towards HTTP traffic) at various network loads.



	1,000 KB response	500 KB response	44 KB response	21 KB response	10 KB response	4.5 KB response	1.7 KB response
CPS	115	235	2,500	5,000	10,000	20,000	40,000
Mbps	1,000	1,000	1,000	1,000	1,000	1,000	1,000

4.3.1 1,000 KB HTTP Response Size – 115 Connections per Second

Maximum 115 new connections per second per Gigabit of traffic with a 1,000 KB HTTP response size—maximum 59,500 packets per second per Gigabit of traffic. With relatively low connection rates and large packet sizes, all DUT should be capable of performing well throughout this test.

4.3.2 500 KB HTTP Response Size – 235 Connections per Second

Maximum 235 new connections per second per Gigabit of traffic with a 500 KB HTTP response size—maximum 60,000 packets per second per Gigabit of traffic. With relatively low connection rates and large packet sizes, all DUTs should be capable of performing well throughout this test.

4.3.3 44 KB HTTP Response Size – 2,500 Connections per Second

Maximum 2,500 new connections per second per Gigabit of traffic with a 44 KB HTTP response size—maximum 140,000 packets per second per Gigabit of traffic. With relatively low connection rates and large packet sizes, all DUT should be capable of performing well throughout this test.

4.3.4 21 KB HTTP Response Size – 5,000 Connections per Second

Maximum 5,000 new connections per second per Gigabit of traffic with a 21 KB HTTP response size—maximum 185,000 packets per second per Gigabit of traffic. With average connection rates and average packet sizes, this is a good approximation of a real-world production network, and all DUTs should be capable of performing well throughout this test.

4.3.5 10 KB HTTP Response Size – 10,000 Connections per Second

Maximum 10,000 new connections per second per Gigabit of traffic with a 10 KB HTTP response size—maximum 225,000 packets per second per Gigabit of traffic. With smaller packet sizes coupled with high connection rates, this represents a very heavily used production network.

4.3.6 4.5 KB HTTP Response Size – 20,000 Connections per Second

Maximum 20,000 new connections per second per Gigabit of traffic with a 4.5 KB HTTP response size—maximum 300,000 packets per second per Gigabit of traffic. With small packet sizes and extremely high connection rates, this is an extreme test for any DUT.

4.3.7 1.7 KB HTTP Response Size – 40,000 Connections per Second

Maximum 40,000 new connections per second per Gigabit of traffic with a 1.7 KB HTTP response size—maximum 445,000 packets per second per Gigabit of traffic. With small packet sizes and extremely high connection rates, this is an extreme test for any DUT.

4.4 HTTP Capacity without Caching and with Transaction Delays

The following tests are identical to the test in section 4.3 except that these include a 5-second delay in the server response, simulating server load time. This has the effect of maintaining a high number of open connections throughout the test, thus forcing the sensor to utilize additional resources to track those connections.

4.4.1 21 KB HTTP Response Size with Delay

Maximum 5,000 new connections per second per Gigabit of traffic with a 21 KB HTTP response size—maximum 185,000 packets per second per Gigabit of traffic. There is a 5-second transaction delay resulting in an additional 50,000 open connections per Gigabit over the test described in section 4.2.4. With average connection rates and average packet sizes, this is a good approximation of a real-world production network, and all DUTs should be capable of performing well throughout this test.

4.4.2 10 KB HTTP Response Size with Delay

Maximum 10,000 new connections per second per gigabit of traffic with a 10 KB HTTP response size—maximum 225,000 packets per second per Gigabit of traffic. Repeated with background traffic loads of 25%, 50%, 75%, and 100% of maximum throughput of the DUT. There is a 5-second transaction delay resulting in an additional 100,000 open connections over the test described in section 4.2.5. With large average packet sizes coupled with very high connection rates, this represents a very heavily used production network, and is a strenuous test for any sensor.

4.5 HTTP Capacity with Caching and with Transaction Delays

The following tests are identical to the tests in section 4.4 except that these include a 5-second delay in the server response, simulating server load time. This has the effect of maintaining a high number of open connections throughout the test, thus forcing the sensor to utilize additional resources to track those connections.

4.5.1 21 KB HTTP Response Size with Delay

Maximum 5,000 new connections per second per gigabit of traffic with a 21 KB HTTP response size—maximum 185,000 packets per second per Gigabit of traffic. There is a 5-second transaction delay resulting in an additional 50,000 open connections per Gigabit over the test described in section 4.2.4. With average connection rates and average packet sizes, this is a good approximation of a real-world production network, and all sensors should be capable of performing well throughout this test.

4.5.2 10 KB HTTP Response Size with Delay

Maximum 10,000 new connections per second per Gigabit of traffic with a 10 KB HTTP response size—maximum 225,000 packets per second per Gigabit of traffic. Repeated with background traffic loads of 25%, 50%, 75%, and 100% of maximum throughput of the DUT. There is a 5-second transaction delay resulting in an additional 100,000 open connections over the test described in section 4.2.5. With large average packet sizes coupled with very high connection rates, this represents a very heavily used production network, and is a strenuous test for any sensor.

5 "Real-World" Traffic

Where previous tests provide a pure HTTP environment with varying connection rates and average packet sizes, the goal of this test is to simulate a real-world environment by introducing additional protocols and real content, while still maintaining a precisely repeatable and consistent background traffic load. The result is a background traffic load that is closer to what may be found on a heavily-utilized "normal" production network.

5.1 "Real-World" Protocol Mix (Data Center – Web-based Applications)

Traffic is generated across the DUT comprising a protocol mix typical of that seen in a web hosting data center.

6 Stability and Reliability

Long-term stability is particularly important for an inline device, where failure can produce network outages. These tests verify the stability of the DUT along with its ability to maintain security effectiveness while under normal load and while passing malicious traffic. Products that are not able to sustain legitimate traffic, or that crash, while under hostile attack will not pass.

The DUT is required to remain operational and stable throughout these tests, and to block 100% of previously blocked traffic, raising an alert for each. If any non-allowed traffic passes successfully, caused by either the volume of traffic or by the DUT failing for any reason, this will result in a FAIL.

6.1 Blocking Under Extended Attack

The DUT is exposed to a constant stream of security policy violations over an extended period of time. The DUT is configured to block and alert, and thus this test provides an indication of the effectiveness of both the blocking and alert handling mechanisms.

A continuous stream of security policy violations mixed with legitimate traffic is transmitted through the DUT at a maximum of 100 Mbps for a minimum of 8 hours with no additional background traffic. This is not intended as a stress test in terms of traffic load (covered in the previous section); it is merely a reliability test in terms of consistency of blocking performance.

The DUT is expected to remain operational and stable throughout this test, and to block 100% of recognizable violations, raising an alert for each. If any recognizable policy violations are passed, caused by either the volume of traffic or by the DUT failing open for any reason, this will result in a FAIL.

6.2 Passing Legitimate Traffic Under Extended Attack

This test is identical to section 6.1, where the DUT is exposed to a constant stream of security policy violations over an extended period of time.

The DUT is expected to remain operational and stable throughout this test, and to pass most/all of the legitimate traffic. If an excessive amount of legitimate traffic is blocked throughout this test, caused by either the volume of traffic or the DUT failing for any reason, this will result in a FAIL.

6.3 Protocol Fuzzing and Mutation

This test stresses the protocol stacks of the DUT by exposing it to traffic from various protocol randomizer and mutation tools. Several of the tools in this category are based on the ISIC test suite and other well-known test tools/suites.

Traffic load is a maximum of 350 Mbps and 60,000 packets per second (average packet size is 690 bytes). Results are presented as a simple PASS/FAIL—the DUT is expected to remain operational and capable of detecting and logging exploits throughout the test.

6.3.1 Protocol Fuzzing and Mutation – Detection Ports

The DUT is exposed to protocol fuzzing and mutation traffic across its inspection ports.

6.4 Power Fail

Power to the DUT is cut whilst passing a mixture of legitimate and malicious traffic. On restoring power, the DUT should continue to log and block malicious traffic with no intervention required by the administrator.

6.5 Redundancy

Does the DUT include multiple redundant critical components?
(Yes/No/Option)

6.6 Persistence Of Data

The DUT should retain all configuration data, policy data and locally logged data once restored to operation following power failure.

7 Management and Configuration Features

The aim of this section is to fully evaluate each product in terms of ease of use, management and configuration, and alerting and reporting capabilities. Clearly it is not possible to "benchmark" a product in these areas, and so this section comprises an in-depth technical evaluation covering all of the main features and advantages of the product.

The following criteria will be evaluated:

- Ease of installation – Appliance and management/logging server (if applicable)
- Ease of management – Once installed, DUTs should be manageable from a centralized management server.
- Authentication and encryption – Between DUT and management server
- Policy definition – Ease of use and flexibility
- Policy deployment – Can policies be defined centrally and rolled out to multiple DUTs (globally, or in logical groups)?
- Policy changes – Can the management console determine which DUTs are using which policies and deploy automatically following amendments?
- Ability to define custom attack signatures – If available, how flexible is this process?
- Attack signtures – How are they obtained/deployed, and how frequently are they updated?
- Attack notifications and alerts – How easy is it to filter and extract individual events?
- Forensic analysis – Ability to capture individual packets, a range of packets, or an entire session where required (globally, or on a rule-by-rule basis)
- Alerts – How accurate and readable are the alerts ?
- Log file maintenance – Automatic rotation, archiving, reportings from archived logs, etc.
- Management reporting – Range of reports/custom reports/the ease with which detail can be extracted and reported
- Reporting – Can reports consolidate output from every DUT?
- Documentation – What documentation is included? Online or hard copy? Supplemental information available on-line?

8 Total Cost Of Ownership

Implementation of security solutions can be complex, with several factors affecting the overall cost of deployment, maintenance, and upkeep. All of these should be considered over the course of the useful life of the solution.

- **Product Purchase** – The cost of acquisition
- **Product Maintenance** – The fees paid to the vendor (including software and hardware support, maintenance and other updates)
- **Installation** – The time required to take the device out of the box, configure it, put it into the network, apply updates and patches, and set up desired logging and reporting
- **Upkeep** – The time required to apply periodic updates and patches from vendors, including hardware, software, and other updates

Appendix A: Change Log

Version 2.0

- Defined device deployment options
- Removed HTML Obfuscation section
- Removed HTTPS Capacity Testing section
- Edited verbiage discussing exploit harness
- Edited messaging in “About this Methodology” section
- Removed duplicate references to Heartbleed

Version 2.1

- Added tests with increased payload sizes in performance sections 4.2 and 4.3
- Evasion testing in section 3.4 incorporated into security section 3.2

Contact Information

NSS Labs, Inc.
206 Wild Basin Road
Building A, Suite 200
Austin, TX 78746 USA
info@nsslabs.com
www.nsslabs.com

This and other related documents available at: **www.nsslabs.com**. To receive a licensed copy or report misuse, please contact NSS Labs.