

Web Application Firewall Testing Procedure (V1.0)

The aim of this procedure is to provide a thorough test of all the main components of a Web Application Firewall device in a controlled and repeatable manner and in the most “real world” environment that can be simulated in a test lab.

The Test Environment

The network is 100/1000Mbit Ethernet with CAT 5e cabling and multiple Cisco Catalyst 6500-series switches (these have a mix of fibre and copper Gigabit interfaces). All devices are expected to be provided as appliances - if software-only, the supplier pre-installs the software on the recommended hardware platform. The sensor is configured as an in-line device during testing (if possible). There is no firewall protecting the target subnet.

Traffic generation equipment - such as the machines generating exploits, Spirent Avalanche and Spirent Smartbits *transmit* port - is connected to the “external” network, whilst the “receiving” equipment - such as the “target” hosts for the exploits, Spirent Reflector and Spirent Smartbits *receive* port - is connected to the internal network. The device under test is connected between two “gateway” switches - one at the edge of the external network, and one at the edge of the internal network.

All “normal” network traffic, background load traffic and exploit traffic will therefore be transmitted **through** the device under test, from external to internal. The same traffic is mirrored to a single SPAN port of the external gateway switch, to which an Adtech network monitoring device is connected. The Adtech AX/4000 monitors the same mirrored traffic to ensure that the total amount of traffic never exceeds 1Gbps (which would invalidate the test run).

The management interface (where available) is used to connect the appliance to the management console on a private subnet. This ensures that the sensor and console can communicate even when the target subnet is subjected to heavy loads, in addition to preventing attacks on the console itself.

Section 1 – Detection Engine

The aim of this section is to verify that the sensor is capable of detecting and blocking a wide range of common attacks accurately, whilst remaining resistant to false positives. All tests in this section are completed with **no background network load**.

Test 1.1 - Attack Recognition

Whilst it is not possible to validate completely the entire range of attacks which can be detected by any sensor, this test attempts to demonstrate how accurately the sensor detects and blocks a wide range of common attacks as detailed in the *Open Web Application Security Project (OWASP) Top Ten Most Critical Web Application Security Vulnerabilities* (www.owasp.org). All attacks are run with no load on the network and are not subjected to any evasion techniques.

To demonstrate these vulnerabilities and provide a vulnerable application for the device to protect, we use a mixture of applications developed in-house specifically for this test, and the WebGoat application. WebGoat is a full J2EE web application designed to teach web application security lessons, and is available from the OWASP Web site. Most of the tests we devised for this methodology can be simulated or approximated using WebGoat.

Our attack suite contains over 30 basic attacks (plus variants) covering the following areas:

Test ID	Attack Category	Test Description
1.1.1	Buffer Overflows	Attempt to overflow input field on form
1.1.2	Hidden Field Tampering	Change price in hidden field after SUBMIT
1.1.3	Cross Site Scripting (GET)	Attempt to display document cookie via script in URL
1.1.4	Cross Site Scripting (POST)	Enter script in input field on e-mail submission form
1.1.5	Parameter tampering	Alter target e-mail address on e-mail submission form after SUBMIT
1.1.6	Buffer Overflows	Restrict size of message returned in e-mail submission form
1.1.7	Input Validation	Change validated values after SUBMIT (product should enforce same field validation)
1.1.8	Injection Flaws	Enter ..\..\.\<filename> when selecting from list of files on-screen
1.1.9	Broken Authentication	Remove parameter field after SUBMIT
1.1.10	Broken Authentication	Change USER or ROLE after submit
1.1.11	Cookie Poisoning	Add AuthCookie=***** to Cookie: Header
1.1.12	Cookie Poisoning	Modify Cookie after SUBMIT
1.1.13	Cookie Poisoning	Modify Cookie file after it has been stored on local hard drive
1.1.14	Cross Site Scripting (POST)	Enter script in message field on guestbook form
1.1.15	Cross Site Scripting (POST)	Enter script in USER field on guestbook form
1.1.16	Injection Flaws	Attempt local directory listing (concatenate OS commands when accessing local OS)
1.1.17	SQL Injection	Attempt to list all entries of database - enter SQL commands in URL
1.1.18	SQL Injection	Attempt to list all entries of database - enter SQL commands in input field
1.1.19	Broken Authentication	Remove PASSWORD field after SUBMIT (fail-open authentication)
1.1.20	Invalid request	Attempt a non-HTTP connection or chunked request
1.1.21	Invalid Request	Attempt a disallowed method (HEAD instead of GET)
1.1.22	Invalid Request	Enter invalid server ID in HTTP/1.1 HOST header field
1.1.23	Invalid Request	Submit request containing shell code
1.1.24	Forceful Browsing	Attempt to access disallowed Web page directly
1.1.25	Forceful Browsing	Attempt to access "sample" Web site directly
1.1.26	Forceful Browsing	Attempt to subvert application flow - change STEP number directly in URL
1.1.27	Forceful Browsing	Attempt to subvert application flow - change STEP number after SUBMIT
1.1.28	Parameter Tampering	Change submitted parameter directly in URL
1.1.29	Information Disclosure	Filter/replace server banners
1.1.30	Information Disclosure	Strip comments from HTML/Java code
1.1.31	Common Exploits	Attempt common HTTP exploits (test-cgi, PHF, etc.)

Each of these attacks are proven to be effective against the back-end applications, and we expect the device under test to prevent all of them and raise appropriate alerts. Tuning of the application policy and/or signatures is allowed in order to ensure that the device is able to protect the application completely.

Section 2 – Evasion

The aim of this section is to verify that the sensor is capable of detecting and blocking basic HTTP attacks when subjected to varying common evasion techniques.

Test 2.1 - URL Obfuscation

A number of common HTTP exploits are launched whilst applying various URL obfuscation techniques made popular by the Whisker Web server vulnerability scanner, including:

Test ID	Test Description
2.1.1	<i>URL encoding</i>
2.1.2	<i>../ directory insertion</i>
2.1.3	<i>Premature URL ending</i>
2.1.4	<i>Long URL</i>
2.1.5	<i>Fake parameter</i>
2.1.6	<i>TAB separation</i>
2.1.7	<i>Case sensitivity</i>
2.1.8	<i>Windows \ delimiter</i>
2.1.9	<i>Session splicing</i>

For each of the evasion techniques, we note if (i) the attempted attack is blocked successfully, (ii) the attempted attack is detected and an alert raised in **any** form, and (iii) if the exploit is successfully “decoded” to provide an accurate alert relating to the original exploit, rather than alerting purely on anomalous traffic detected as a result of the evasion technique itself.

Section 3 – Performance Under Load (No Security Policies Applied)

The aim of this section is to verify the maximum raw processing performance of the sensor. This is achieved by measuring packet processing performance and HTTP performance with no security policies applied.

Test 3.1 - UDP Traffic To Random Valid Ports

This test uses UDP packets of varying sizes generated by a **SmartBits SMB6000** with LAN-3301A 10/100/1000Mbps **TeraMetrics** cards installed.

A constant stream of the appropriate mix of packets - with variable source IP addresses and ports transmitting to a single fixed IP address/port - is transmitted through the sensor (bi-directionally, maximum of 1Gbps). Each packet contains dummy data, and is targeted at a valid port (not port 80) on a valid IP address on the target subnet. The percentage load and packets per second (pps) figures are verified by the Adtech Gigabit network monitoring tool before each test begins. Multiple tests are run and averages taken where necessary.

This traffic does not attempt to simulate any form of “real world” network condition.

The aim of this test is purely to determine the raw packet processing capability of the sensor, and its effectiveness at passing “useless” packets quickly in order to pass potential attack packets to the detection engine.

Test ID	Test Description
---------	------------------

3.1.1	256 byte packets - maximum 452,000 packets per second
-------	--

This test measures packet processing performance under the most extreme conditions for a Web Application Firewall - it is unlikely that any real-life HTTP server will ever be expected to handle network loads of over 450,000 256-byte packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic

3.1.2	512 byte packets - maximum 235,000 packets per second
-------	--

This test has been included to provide a comparison with our “real world” packet mixes, since the average packet size is similar. No sessions are created during this test, however, and there should be nothing for the detection engine to do in the way of protocol analysis. This test provides a reasonable indication of the ability of a device to process packets from the wire on an “average” network. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic

3.1.3	1024 byte packets - maximum 120,000 packets per second
-------	---

*This test demonstrates the ability of the device to process large packets. This provides the easiest environment for the device under test - beware of test results that **only** quote performance figures using similar (or larger) packet sizes. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic*

Test 3.2 - Maximum Capacity HTTP Traffic

The use of multiple Spirent Communications **Avalanche 2500** and **Reflector 2500** devices allows us to create true “real world” traffic at speeds of up to 4.2 Gbps as a background load for our tests. Our Avalanche configuration is capable of simulating over 5 million users, with over 5 million concurrent sessions, and over 200,000 HTTP requests per second.

By creating genuine session-based traffic with varying session lengths, this provides a test environment that is as close to “real world” as it is possible to achieve in a lab environment, whilst ensuring absolute accuracy and repeatability. The aim of this test is to stress the HTTP detection engine to determine the maximum capacity (connections per second and Mbits per second) at varying packet sizes and using varying sizes of HTTP response.

Each transaction consists of a single HTTP GET request (thus *connections per second* = *transactions per second* in all these tests) 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, though there are no embedded links to other pages or elements which need to be defined within a security context. This test provides an excellent representation of a live network (albeit one biased towards “sterile” HTTP traffic) at various network loads.

Test ID	Test Description
---------	------------------

3.2.1	1000 byte packets - 44KByte response (31 packets per transaction)
-------	--

3.2.2	550 byte packets - 22KBytes response (37 packets per transaction)
-------	--

3.2.3	440 byte packets - 11KBytes response (19 packets per transaction)
-------	--

3.2.4	360 byte packets - 5KBytes response (9 packets per transaction)
-------	--

3.2.5	285 byte packets - 2KBytes response (4 packets per transaction)
-------	--

Section 4 – Performance Under Load (Security Policies Applied)

The aim of this section is to verify the ability of the sensor to handle varying loads of HTTP traffic whilst being expected to inspect the contents of that traffic according to the security policies applied.

Test 4.1 - Standard Spirent Avalanche Traffic

The default traffic generated by the Spirent Avalanche/Reflector devices is very realistic and provides for very repeatable tests. However, it is also very “sterile” inasmuch as it does not contain a variety of elements which must be inspected and acted upon by the device under test.

These tests, therefore, are expected to provide the best results under all test conditions.

Test ID	Test Description
4.1.1	1000 byte packets - 44KBytes response <i>Although the packet size is large for this test, so is the response size. This represents a worst-case scenario for the device under test due to the amount of data that is returned for each transaction, all of which needs to be inspected</i>
4.1.2	550 byte packets - 22KBytes response <i>With smaller packets sizes than the previous test, the response size is still large enough to cause problems for the device under test.</i>
4.1.3	440 byte packets - 11KBytes response <i>This test demonstrates a good average packet size and response size</i>
4.1.4	360 byte packets - 5KBytes response <i>This test demonstrates a good average packet size and response size</i>
4.1.5	285 byte packets - 2KBytes response <i>With very small packets and the smallest of the response sizes, this test is a best-case scenario for the device under test</i>

Test 4.2 - NSS Home Page Only - No Images

In order to introduce a realistic workload for the device under test, we switch out the Spirent Reflector and replace with a high-specification Web server running IIS and containing a copy of the NSS Group Web site. The dual-processor Dell PowerEdge 1750 server is capable of 1Gbps throughput so as not to act as a bottleneck for the device under test.

Test ID	Test Description
4.2.1	440 byte packets - 11KBytes response <i>As with the tests in Section 4.1, each transaction consists of a single page. In this case the page is the home page of the NSS Group Web site. This page is 11Kbytes in size, and contains a total of 24 embedded objects and links to other pages which may be of interest to any Web Application Firewall's inspection process. This test represents a worst-case "real world" scenario for the device under test.</i>

Test 4.3 - NSS Home Page + Ten Associated Images

The aim of this test is the same as for Test 4.2.1, but this time we add a number of images to the transaction.

This is intended to provide a more realistic - and easier - test scenario for the device under test since it does not have to inspect or process the image data, and thus although the transaction response size is greater, the processing overhead is lower than the previous test.

It is not often that the device under test will be faced with a high load of transactions which consist *purely* of HTML or application code which all needs to be inspected and processed - most Web pages contain a mixture of data types. Thus, this test is intended to replicate a typical “real world” situation.

Test ID	Test Description
---------	------------------

4.3.1	360 byte packets - 42KBytes response
-------	---

In this test each transaction consists of a single HTML page along with ten associated GIF/JPG images. As with Test 4.2.1, the page is the home page of the NSS Group Web site. This page is 11Kbytes in size, and contains a total of 24 embedded objects and links to other pages which may be of interest to any Web Application Firewall's inspection process. The transaction also contains a total of 31KB of images which do not need to be inspected or processed by the device under test.

Test 4.4 - Maximum Open Connections

It is important that the device under test cannot only support a sufficiently high rate of connection set-up and tear-down, but that it can also support a sufficiently large number of simultaneous connections. This test determines its maximum capacity.

Test ID	Test Description
---------	------------------

4.4.1	Maximum Open Connections
-------	---------------------------------

In this test the Spirent Avalanche is set to continually open HTTP connections with the Reflector. The Reflector is set to implement a delay on each transaction, thus ensuring that transactions remain open for a significant period of time (typical of a real world situation). The Avalanche is monitored and the point where transactions begin to fail is noted. Once the exact point of failure has been determined, the test is run for several hours with that number of transactions held open throughout.

Section 5 – Latency & User Response Times

The aim of this section is to determine the effect the sensor has on the traffic passing through it under various load conditions.

Should a device impose an unacceptably high degree of latency on the packets passing through it, a network or security administrator would need to think carefully before installing such a device in front of a high capacity Web server or server farm.

Test 5.1 - Latency

We use Spirent SmartFlow software and The SmartBits SMB6000 with Gigabit TeraMetrics cards to create multiple traffic flows through the appliance and measure the basic throughput, packet loss, and latency through the sensor. This test - whilst not indicative of real-life network traffic - provides an indication of how much the sensor affects the traffic flow through it. This data is particularly useful for network administrators who need to gauge the effect of any form of in-line device which is likely to be placed at critical points within the corporate network.

SmartFlow runs through several iterations of the test varying the traffic load from 250Mbps to 1Gbps bi-directionally (or up to the maximum rated throughput of the device should this be less than 1Gbps) in steps of 250Mbps. This is repeated for a range of packet sizes (256 bytes, 512 bytes and 1024 bytes) of UDP traffic with variable IP addresses and ports. At each iteration of the test, SmartFlow records the number of packets dropped, together with average and maximum latency.

Test ID	Test Description
---------	------------------

5.1.1	Device latency with no background traffic
-------	--

SmartFlow traffic is passed across the infrastructure switches and through the device (the latency of the basic infrastructure is known and is constant throughout the tests). The packet loss and average latency are recorded at each packet size and each load level from 250Mbps to 1Gbps (in 250Mbps steps)

Section 6 – Stability & Reliability

These tests attempt to verify the stability of the device under test under various extreme conditions. Long term stability is particularly important for an in-line protection device, where failure can produce network outages.

Test ID	Test Description
---------	------------------

6.1.1	Blocking Under Extended Attack
-------	---------------------------------------

For this test, we expose the external interface of the device to a constant stream of alerts over an extended period of time - this is intended to provide an indication of the effectiveness of both the blocking and alert handling mechanisms. A continuous stream of exploits mixed with some legitimate sessions is transmitted through the device for 8 hours with no additional background traffic. This is not intended as a stress test in terms of traffic load - merely a reliability test in terms of consistency of blocking performance.

The device is expected to remain operational and stable throughout this test, and to block 100 per cent of recognisable attacks, raising an alert for each. Results are presented as a percentage of attacks blocked. If any recognisable attacks are passed - caused by either the volume of traffic or the sensor failing open for any reason - this will result in a FAIL

6.1.2	Passing Legitimate Traffic Under Extended Attack
-------	---

This test is identical to 6.1.1, where we expose the external interface of the device to a constant stream of alerts over an extended period of time. The device is expected to remain operational and stable throughout this test, and to pass 100 per cent of legitimate traffic. Results are presented as a percentage of legitimate sessions allowed to pass. If any legitimate traffic is blocked - caused by either the volume of traffic or the sensor failing closed for any reason - this will result in a FAIL.

6.1.3	ISIC/ESIC/TCPSIC/UDPSIC/ICMPSIC
-------	--

This test attempts to stress the protocol stack of the device under test by exposing it to traffic from the ISIC test tool. The ISIC test tool host is connected to the external network, and the ISIC target is on the internal network. ISIC traffic is transmitted through the sensor and the effects noted.

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

Section 7 – Management and Configuration

The aim of this section is to determine the features of the management system, together with the ability of the management port on the device under test to resist attack.

Test 7.1 - Management Port

Clearly the ability to manage the alert data collected by the sensor is a critical part of any firewall system. For this reason, an attacker could decide that it is more effective to attack the management interface of the device than the detection interface.

Given access to the management network, this interface is often more visible and more easily subverted than the detection interface, and with the management interface disabled, the administrator has no means of knowing his network is under attack.

Test ID	Test Description
---------	------------------

7.1.1	Open ports
-------	-------------------

We will scan the open ports and active services on the management interface and report on known vulnerabilities.

7.1.2	ISIC/ESIC/TCPSIC/UDPSIC/ICMPSIC
-------	--

This test attempts to stress the protocol stack of the management interface of the device under test by exposing it to traffic from the ISIC test tool. The ISIC test tool host is connected directly to the management interface of the IPS sensor, and that interface is also the target. ISIC traffic is transmitted to the management interface of the IPS device and the effects noted.

Traffic load is a maximum of 350Mbps and 60,000 packets per second (average packet size is 690 bytes). Results are presented as a simple PASS/FAIL - the device is expected to remain (a) operational and capable of detecting and blocking exploits, and (b) capable of communicating in both directions with the management server/console throughout the test to attain a PASS.