

## Network IPS Testing Procedure (V4.0)

---

The aim of this procedure is to provide a thorough test of all the main components of an in-line Network Intrusion Prevention System (IPS) device in a controlled and repeatable manner and in the most “real world” environment that can be simulated in a test lab.

The IPS test is currently designed to test devices to a maximum of 1Gbps using a single in-line 1Gbps port pair (a separate methodology exists for multi-Gigabit devices). The DUT should be supplied as a single appliance, with the smallest number of ports capable of achieving the required level of connectivity and performance.

There will be four levels of **NSS Approved** award for this category:

- **IPS (Basic)** - For devices which meet the minimum level of requirements, but where performance/port density is not sufficient to be placed in the enterprise, or where management capabilities are lacking for enterprise deployments. This is equivalent to the previous (pre-April 2006) NSS Approved award, although the more rigorous new test methodology makes it harder to obtain.
- **IPS + Branch Office** - Devices designed for branch office deployment may offer restricted port density and/or bandwidth, and will be designed specifically for perimeter operation. However, management capabilities should be scalable and suitable for centralised administration of multiple devices as specified in the methodology document. Where high port density and High Availability functions are not required, some of these devices may also be suitable for low-end enterprise deployments.
- **IPS + Enterprise** - Devices designed for the enterprise should demonstrate sufficient levels of port density and performance (high bandwidth and low latency) to allow them to be deployed in the enterprise core (slightly higher latency is permitted for deployment at the enterprise perimeter), in addition to good, scalable centralised management capabilities which meet minimum requirements as specified in the methodology document
- **IPS + ISP/MSP** - These devices should demonstrate exceptional levels of performance and very low latency, in addition to well-developed virtualisation capabilities and role-based management (as specified in the methodology document) to allow them to be deployed securely in a managed services environment

## 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 internal device during testing (i.e. as if installed inside the main Internet gateway/firewall) and thus low latency is important. 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* (DUT) is connected

between two “gateway” switches - one at the edge of the external network, and one at the edge of the external network.

All “normal” network traffic, background load traffic and exploit traffic will therefore be transmitted **through** the DUT, 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 per in-line port pair never exceeds 1Gbps. This test will verify performance up to a **maximum of 1Gbps**, using **one in-line port pair**.

The management interface 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 exploits accurately, whilst remaining resistant to false positives. All tests in this section are performed initially with no background network load. The tests are then repeated under varying levels and mixes of background traffic to ensure that the results do not vary when handling normal network traffic.

The latest signature pack is acquired from the vendor, and sensors are deployed with the **DEFAULT SECURITY POLICY** or **RECOMMENDED SETTINGS** only, since this is how most users will first deploy the device.

When the device is deployed in blocking mode, therefore, this should result in a proportion of the signatures which will block and log, and a portion which will log only. The device will also be configured to block all SYN Flood attempts against all internal servers (including Avalanche/Reflector “virtual” servers) by the most secure means available.

NSS considers it unacceptable for a device of this nature to be sold without a default policy and/or recommended settings. Therefore, should a vendor provide a product in this state, it will be considered an automatic fail. Note that the default policy/recommended settings cannot be created purely for this test - it should be offered as a standard configuration of the DUT as provided for general sale.

### Test 1.1 - Attack Recognition

Whilst it is not possible to validate completely the entire signature set of any sensor, this test attempts to demonstrate how accurately the sensor detects and blocks a range of recent common exploits, port scans, and Denial of Service attempts. All tests are repeated with both zero load **and** background network traffic. No custom signatures are permitted in this test - all signatures used must be available to the general public at the time of testing.

Our new attack suite currently contains thousands of basic exploits (including multiple variants of each base exploit) from which we will carefully select groups of **recent** exploits to test the following:

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

1.1.1	<b>High Severity attacks</b>
-------	------------------------------

*These are recent attacks which are deemed to be serious and which can result in remote server compromise of systems which are widely deployed and*

generally Internet-facing (i.e. Microsoft OS, IIS, Apache, etc.). Also included are some older remote compromise attacks which are included in readily-available attack tools such as Metasploit, Core Impact, etc. It is possible to detect these attacks with a high degree of confidence, leading to low incidence of false positives. These attacks should therefore be enabled in block mode by default.

#### **1.1.2 Medium Severity attacks**

These are recent attacks which are deemed to be of lower severity than those in Test 1.1.1, but which nevertheless could have serious consequences - for example: remote server compromise of systems which are not generally Internet-facing, or DOS conditions which are easily recoverable and do not lead to server compromise. Because it is not always possible to detect these attacks with 100 per cent confidence, or because they are not considered critical, these attacks may be rated differently by different vendors, and may be enabled in either block or log mode.

#### **1.1.3 Low Severity attacks**

These are recent attacks which are deemed to be of lower severity than those in Test 1.1.2 by virtue of the fact that the systems are not generally Internet-facing or are not widely deployed. These attacks may be rated differently by different vendors, and may be enabled in either block or log mode.

#### **1.1.4 Audit only**

These are recent and older attacks which are deemed to be of low severity and will have little or no impact on target hosts. Some older attacks are used here to determine when vendors are "shedding" older signatures for performance reasons - these will typically be those types of attacks which are used most often in widely available vulnerability scanner software, and for which we believe alerts should still be raised as an indication that reconnaissance is being performed against protected systems. Alerts are raised for information purposes only. Policy tuning is allowed to ensure all attacks in this section are covered.

#### **1.1.5 Reconnaissance**

These are system probes and enumeration attempts which are of no serious threat, and whose sole purpose is to elicit information from target hosts. Alerts are raised for information purposes only. Policy tuning is allowed to ensure all attacks in this section are covered.

#### **1.1.6 DOS/DDOS**

These are serious Denial of Service attempts using SYN Flood techniques (up to 100Mbps on a single in-line port pair). Two types of SYN Flood attack are used - one with a single source IP address (DOS), and one with multiple source IP addresses (DDOS). During the attacks, latency will be measured and attempts made to create legitimate sessions to servers through the DUT. The effect on the server(s) under attack will also be measured.

#### **1.1.7 P2P (optional)**

This is peer-to-peer file-sharing traffic from common P2P programs. Not all products will offer the ability to filter this traffic (this is a policy rather than pure security issue), and therefore failure to detect and block will not be seen as cause for failure in the test. However, because of potential for false positives and the possible impact on legitimate traffic (especially if P2P traffic is permitted), this traffic should not be blocked using the default policy or recommended settings.

#### **1.1.8 Spyware (optional)**

This is traffic from common Spyware/Adware programs which are rated as "critical" or "high" severity (i.e. potential to cause remote compromise), and test cases will be created to include initial download/infection and subsequent "phone home" traffic. Not all IPS products will offer the ability to filter this traffic (some vendors may consider this more of an Anti Virus or dedicated Anti Spyware issue), and therefore failure to detect and block will not be seen as cause for failure in the test, and detection requirements are lower (for the moment) than other types of exploits.

#### **1.1.9 Server-to-Client Exploits (optional)**

Most HTTP exploits occur in client requests to servers (attempted buffer overflows, malformed requests, and so on). Given the amount of data that is returned by the server, many devices do not monitor this side of the traffic flow for performance reasons. Some vendors do not include such signatures because they also offer host-based IPS solutions which are often better suited to preventing this type of exploit. For the moment, this section remains optional, though given the increasing use of such exploits to compromise client systems,

*this section will become mandatory in future tests. This section is composed of test cases where the exploit traffic is “reversed”, from server to client. Where the DUT does provide server-to-client signatures, they will be tested here - given the lack of coverage of this type of signature in the IPS industry at present, however, there is no minimum detection requirement. The vendor will also be given the option to disable these signatures for the main performance tests, but this will be noted in the report. For this test, we leave it to the reader or potential purchaser to decide if this feature is essential to them, and, if included, whether the DUT should be capable of detecting such exploits at full performance.*

A wide range of vulnerable target operating systems and applications are used, and the majority of the high-severity attacks are successful, gaining root shell or administrator privileges on the target machine. In addition to packet captures of live exploits, we also use commonly available exploit tools - such as Metasploit and CORE IMPACT - against vulnerable live servers.

We expect all the attacks to be reported in as straightforward and clear a manner as possible (i.e. an “RDS MDAC attack” should be reported as such, rather than a “Generic IIS Attack”). Wherever possible, attacks should be identified by their assigned CVE reference. It will also be noted when a response to an exploit is considered too “noisy”, generating multiple similar or identical alerts for the same attack. Finally, we will note whether the device blocks the attack packet only or the entire “suspicious” TCP session.

## Test 1.2 - Resistance To False Positives

The aim of this test is to demonstrate how likely it is that a sensor raises a false positive alert - particularly critical for IPS devices.

The product is marked as “SUCCESSFUL” for each test case if it does **not** raise an alert and does **not** block the traffic. Raising an alert on any of these test cases is considered “UNSUCCESSFUL”, since none of the “exploits” used in this test represents a genuine threat. A high number of “UNSUCCESSFUL” results would thus indicate the chance that the sensor could block legitimate traffic inadvertently.

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

1.2.1	False Positives
-------	-----------------

*Trace files of normal traffic with “suspicious” content are sent across the DUT, together with several “neutered” exploits which have been rendered completely ineffective. In addition, for every malicious test case used in Test 1.1, normal traffic is sent using the same mechanism. If a signature has been coded for a specific piece of exploit code rather than the underlying vulnerability, or if it relies purely on pattern matching, or it is too generic in its detection methods, some of these false alarms could be alerted upon*

## Test 1.3 - Resistance To False Negatives

The aim of this test is to demonstrate the susceptibility of a sensor to false negative cases. This ensures that vendors have written signatures to handle the underlying vulnerability, instead of - or as well as - handling specific exploits.

For this test, NSS has taken a number of exploits and modified them in various ways so that, whilst still accomplishing their intended task (i.e. to gain a root shell) their “signature” looks quite different. This is achieved by changing obvious banners or shell code patterns which are used by many vendors to create simplistic, exploit specific, pattern-matching signatures.

The DUT is subjected both to the original exploit (as a baseline), followed by the modified exploit. The product is marked as “SUCCESSFUL” for each test

case if it raises an alert (blocking requirement depends on the specific test case) for both the original AND the modified exploits.

Failure to raise an alert on any of the modified test cases is considered an indication that the signature has been written for the exploit rather than the underlying vulnerability, and that test case is marked as "UNSUCCESSFUL".

Should the DUT not have a signature for the original exploit, that test case is marked as "N/A".

<b>Test ID</b>	<b>Test Description</b>
----------------	-------------------------

<b>1.3.1</b>	<b><i>False Negatives</i></b>
--------------	-------------------------------

*A number of common exploits are modified such that, whilst still accomplishing their intended task (i.e. to gain a root shell) their "signature" looks quite different. If a signature has been coded for a specific piece of exploit code rather than the underlying vulnerability, or if it relies purely on pattern matching, some of these false negative cases could evade the IPS and succeed in their purpose*

## Section 2 – Evasion

---

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

### Test 2.1 - Baselines

The aim of this test is to establish that the sensor is capable of detecting and blocking a number of common basic attacks (our baseline suite) in their normal state, with no evasion techniques applied.

<b>Test ID</b>	<b>Test Description</b>
----------------	-------------------------

<b>2.1.1</b>	<b><i>Baseline Attack Replay</i></b>
--------------	--------------------------------------

*A number of common exploits are replayed across the DUT to ensure that they are detected in their unmodified state. These will be chosen from a suite of older/common basic exploits for which NSS is certain that all vendors will have signatures. None of the exploits used in Section 1 will be used as evasion baselines, to ensure that vendors are not provided with any information on the content of the Section 1 test suite.*

### Test 2.2 - Packet Fragmentation

These tests determine the effectiveness of the fragment reassembly mechanism of the DUT. The baseline HTTP attacks are repeated, running them through fragroute using IP fragmentation evasion techniques, including:

<b>Test ID</b>	<b>Test Description</b>
----------------	-------------------------

<b>2.2.1</b>	<i>Ordered 8 byte fragments</i>
--------------	---------------------------------

<b>2.2.2</b>	<i>Ordered 24 byte fragments</i>
--------------	----------------------------------

<b>2.2.3</b>	<i>Out of order 8 byte fragments</i>
--------------	--------------------------------------

<b>2.2.4</b>	<i>Ordered 8 byte fragments, duplicate last packet</i>
--------------	--

<b>2.2.5</b>	<i>Out of order 8 byte fragments, duplicate last packet</i>
--------------	---

<b>2.2.6</b>	<i>Ordered 8 byte fragments, reorder fragments in reverse</i>
--------------	---

<b>2.2.7</b>	<i>Ordered 16 byte fragments, fragment overlap (favour new)</i>
--------------	---

<b>2.2.8</b>	<i>Ordered 16 byte fragments, fragment overlap (favour old)</i>
--------------	---

<b>2.2.9</b>	<i>Out of order 8 byte fragments, interleaved duplicate packets scheduled for later delivery</i>
--------------	--

For each of the evasion techniques, we note if (i) the attempted attack is blocked successfully (the primary aim of any IPS device), (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.

It is a requirement of the test that devices submitted should have all IP fragmentation reassembly options enabled by default in the shipping product.

### Test 2.3 - Stream Segmentation

These tests determine the effectiveness of the stream reassembly mechanism of the DUT. The baseline HTTP attacks are repeated, running them through fragroute using TCP segmentation evasion techniques, including:

Test ID	Test Description
2.3.1	<i>Ordered 1 byte segments, interleaved duplicate segments with invalid TCP checksums</i>
2.3.2	<i>Ordered 1 byte segments, interleaved duplicate segments with null TCP control flags</i>
2.3.3	<i>Ordered 1 byte segments, interleaved duplicate segments with requests to resync sequence numbers mid-stream</i>
2.3.4	<i>Ordered 1 byte segments, duplicate last packet</i>
2.3.5	<i>Ordered 2 byte segments, segment overlap (favour new)</i>
2.3.6	<i>Ordered 1 byte segments, interleaved duplicate segments with out-of-window sequence numbers</i>
2.3.7	<i>Out of order 1 byte segments</i>
2.3.8	<i>Out of order 1 byte segments, interleaved duplicate segments with faked retransmits</i>
2.3.9	<i>Ordered 1 byte segments, segment overlap (favour new)</i>
2.3.10	<i>Out of order 1 byte segments, PAWS elimination (interleaved dup segs with older TCP timestamp options)</i>
2.3.11	<i>Ordered 16 byte segments, segment overlap (favour new (Unix))</i>

For each of the evasion techniques, we note if (i) the attempted attack is blocked successfully (the primary aim of any IPS device), (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.

It is a requirement of the test that devices submitted should have all TCP stream reassembly options enabled by default in the shipping product.

### Test 2.4 - RPC Fragmentation

Both Sun/ONC RPC and MS-RPC allow the sending application to fragment requests, and all MS-RPC services have a built-in fragmentation reassembly mechanism.

An attacker can transmit the BIND followed by a single request fragmented over a hundred actual requests with small fragments of the malicious payload. Alternatively, the attacker could transmit both the BIND and request fragments in one large TCP segment, thus foiling any signatures which use a simple size check. Immunitysec's CANVAS Reference Implementation combines large

writes with many tiny MS-RPC fragments, and provides up to ten levels of fragmentation.

The baseline RPC attacks are repeated, this time applying various evasion techniques, including:

Test ID	Test Description
2.4.1	<i>One-byte fragmentation (ONC)</i>
2.4.2	<i>Two-byte fragmentation (ONC)</i>
2.4.3	<i>All fragments, including Last Fragment (LF) will be sent in one TCP segment (ONC)</i>
2.4.4	<i>All fragments except Last Fragment (LF) will be sent in one TCP segment. LF will be sent in separate TCP segment (ONC)</i>
2.4.5	<i>One RPC fragment will be sent per TCP segment (ONC)</i>
2.4.6	<i>One LF split over more than one TCP segment. In this case no RPC fragmentation is performed (ONC)</i>
2.4.7	<i>Canvas Reference Implementation Level 1 (MS)</i>
2.4.8	<i>Canvas Reference Implementation Level 2 (MS)</i>
2.4.9	<i>Canvas Reference Implementation Level 3 (MS)</i>
2.4.10	<i>Canvas Reference Implementation Level 4 (MS)</i>
2.4.11	<i>Canvas Reference Implementation Level 5 (MS)</i>
2.4.12	<i>Canvas Reference Implementation Level 6 (MS)</i>
2.4.13	<i>Canvas Reference Implementation Level 7 (MS)</i>
2.4.14	<i>Canvas Reference Implementation Level 8 (MS)</i>
2.4.15	<i>Canvas Reference Implementation Level 9 (MS)</i>
2.4.16	<i>Canvas Reference Implementation Level 10 (MS)</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.

## Test 2.5 - URL Obfuscation

The baseline HTTP attacks are repeated, this time applying various URL obfuscation techniques. Some of these were made popular by the LibWhisker Web server testing library, but in-house tools have taken some of the older LibWhisker techniques much further.

Random URL encoding techniques are employed to transform simple URLs which are often used in pattern-matching signatures to apparently meaningless strings of escape sequences and expanded patch characters using a combination of the following techniques:

- *Escape encoding (% encoding)*
- *Microsoft %u encoding*
- *Path character transformations and expansions (./, //, \)*

These techniques are combined in various ways for each URL tested, ranging from minimal transformation, to extreme (every character transformed). All transformed URLs are verified to ensure they still function as expected after transformation.

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

- 2.5.1 *URL encoding - Level 1 (minimal)*
- 2.5.2 *URL encoding - Level 2*
- 2.5.3 *URL encoding - Level 3*
- 2.5.4 *URL encoding - Level 4*
- 2.5.5 *URL encoding - Level 5*
- 2.5.6 *URL encoding - Level 6*
- 2.5.7 *URL encoding - Level 7*
- 2.5.8 *URL encoding - Level 8 (extreme)*
- 2.5.9 *Premature URL ending*
- 2.5.10 *Long URL*
- 2.5.11 *Fake parameter*
- 2.5.12 *TAB separation*
- 2.5.13 *Case sensitivity*
- 2.5.14 *Windows \ delimiter*
- 2.5.15 *Session splicing*

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.

## Test 2.6 - HTML Obfuscation

Recognising malicious HTML documents is becoming increasingly important when protecting the enterprise.

Malicious HTML documents exploit flaws in common web browsers to install spyware, facilitate phishing attacks, or even to gain complete control of the client system. Therefore, it is becoming increasingly important that security products charged with protecting end systems must correctly interpret HTML documents. Many security products use simple pattern matching systems with very little semantic or syntactic understanding of the data they are analysing. This leaves them vulnerable to evasion through use of redundant, but equivalent, alternative representations of malicious documents.

This test suite uses a number of malicious HTML documents which are transferred from server to client through the DUT. Each malicious HTML document is served with a different form of obfuscation, as follows:

<b>Test ID</b>	<b>Test Description</b>
<b>2.6.1</b>	<b><i>UTF-16 character set encoding (big-endian)</i></b>  <i>The UTF-16 character set specifies a 2-byte sequence for most characters and a 4-byte sequence for the others (a small percentage). Recoding an HTML document in UTF-16 changes its appearance significantly. A document that contains just the ASCII subset of characters will appear to have a null byte between every one of the original characters. There are also two different forms of the UTF-16 encoding depending on whether the null high byte comes first (big-endian) or second (little-endian) - this test uses big-endian byte ordering</i>
<b>2.6.2</b>	<b><i>UTF-16 character set encoding (little-endian)</i></b>  <i>As for Test 2.6.1, but using little-endian byte ordering</i>
<b>2.6.3</b>	<b><i>UTF-32 character set encoding (big-endian)</i></b>  <i>This UTF-32 character set specifies a 4-byte sequence. Like the UTF-16 character set encoding there are two variations - big-endian and little-endian -</i>

*and this test case uses big-endian byte ordering*

**2.6.4 UTF-32 character set encoding (little-endian)**

*As for Test 2.6.3, but using little-endian byte ordering*

**2.6.5 UTF-7 character set encoding**

*The UTF-7 character set encodes most ASCII characters as themselves. However, in addition to recoding non-English characters as other encodings do, it also recodes many punctuation symbols, including many of the symbols that are important to the HTML specification. Therefore, recoding an HTML document in UTF-7 significantly changes its appearance*

**2.6.6 Chunked encoding (random chunk size)**

*Chunked encoding allows the server to break a document into smaller chunks and transmit them individually. The server needs only to specify the size of each chunk before it is transmitted and then indicate when the last chunk has been transmitted. Since chunked encoding intersperses arbitrary numbers (chunk sizes) with the elements of the original document, it can be used to greatly change the appearance of the original document as observed "on the wire". In addition, the server can choose to break the document into chunks at arbitrary points. This makes it difficult for simple pattern matching systems to reliably identify the original HTML document from the raw data on the network*

**2.6.7 Chunked encoding (fixed chunk size)**

*As for Test 2.6.6 but with fixed 8 byte chunk sizes*

**2.6.8 Chunked encoding (chaffing)**

*As for Test 2.6.6 but with "chaffing" (arbitrary numbers inserted between chunks)*

**2.6.9 Compression (Deflate)**

*Per RFC 2616, the HTTP protocol allows the client to request and the server to use several compression methods. These compression methods not only improve performance in many circumstances, they completely change the characteristic size and appearance of HTML documents. Furthermore, small changes in the original document, can greatly change the final appearance of the compressed document. This property of these algorithms could be used to obfuscate hostile content for the purpose of evading detection. The deflate compression method is a Lempel-Ziv coding (LZ77), specified in RFC 1951*

**2.6.10 Compression (Gzip)**

*As Test 2.6.9, but using gzip compression as specified in RFC 1952*

**2.6.11 Combination**

*This test applies a combination of UTF-7 encoding (Test 2.6.5), Gzip compression (Test 2.6.10) and chunked encoding with random chunk sizes (Test 2.6.6) to the malicious Web page.*

For each of the above, it is verified that a standard Web browser (such as Internet Explorer 6.0) is capable of rendering the results of the evasion.

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.

As with Test 1.1.9, those devices which do not support server-to-client signatures will be unable to complete this test. For this reason, failure to complete this particular test will not be cause for overall failure, and we leave it to the reader or potential purchaser to decide if this anti-evasion feature is essential to them.

## Test 2.7 - FTP Evasion

When attempting FTP exploits, it is possible to evade some IDS/IPS products by inserting additional spaces and telnet control sequences in FTP commands.

These tests insert a range of valid telnet control sequences that can be parsed and handled by IIS FTP server and wu-ftpd, and which also conform to Section 2.3 of RFC 959. Control opcodes are inserted at random, ranging from minimal insertion (only one pair of opcodes), to extreme (opcodes between every character in the FTP command):

Test ID	Test Description
2.7.1	<i>Inserting spaces in FTP command lines</i>
2.7.2	<i>Inserting non-text Telnet opcodes - Level 1 (minimal)</i>
2.7.3	<i>Inserting non-text Telnet opcodes - Level 2</i>
2.7.4	<i>Inserting non-text Telnet opcodes - Level 3</i>
2.7.5	<i>Inserting non-text Telnet opcodes - Level 4</i>
2.7.6	<i>Inserting non-text Telnet opcodes - Level 5</i>
2.7.7	<i>Inserting non-text Telnet opcodes - Level 6</i>
2.7.8	<i>Inserting non-text Telnet opcodes - Level 7</i>
2.7.9	<i>Inserting non-text Telnet opcodes - Level 8 (extreme)</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.

## Test 2.8 - Miscellaneous Evasion Techniques

Certain baseline attacks are repeated, and are subjected to various protocol- or exploit-specific evasion techniques, including:

Test ID	Test Description
2.8.1	<i>Polymorphic mutation (ADMmutate)</i>
2.8.2	<i>Altering protocol and RPC PROC numbers</i>
2.8.3	<i>HTTP exploits to non-standard ports</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.

It is permitted to configure the device to define additional ports for HTTP services.

## Section 3 – Stateful Operation

---

The aim of this section is to be able to determine whether the sensor is capable of monitoring stateful sessions established through the device at various traffic loads without either losing state or incorrectly inferring state. NB: the DUT should be configured to block all traffic when resources are exhausted or when traffic cannot be analysed for any reason. Any device which passes malicious traffic under the above conditions will fail.

### Test 3.1 - Maximum Simultaneous Open Connections

This test determines the maximum number of concurrent TCP connections which can be maintained by the device with various response sizes.

It is permitted to tune the device to allow the maximum number of simultaneous open connections, but this will be noted in the report, and these settings must be retained for **ALL** performance tests.

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 DUT is causing unacceptable increase in open connections on the server-side
- **Excessive response time for HTTP transactions/SMTP sessions** - latency within the DUT is causing excessive delays and increased response time to client
- **Unsuccessful HTTP transactions/SMTP sessions** - normally there should be zero unsuccessful transactions. Once these appear, it is an indication that excessive latency within the DUT is causing connections to time out

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

3.1.1	<b>Theoretical Maximum Concurrent TCP Connections</b>
-------	---

*This test is designed to determine the maximum concurrent TCP connections of the DUT with a 8 byte object size. This response size would not typically be found on a normal network, but it provides the means to determine the maximum possible concurrent connections figure.*

*Client and server are using HTTP 1.0 with keep-alive, and the client will open a TCP connection, send 10 HTTP requests, and close the connection. This ensures that TCP connections remain open until all ten HTTP transactions are complete, and a “user think time” of 30 seconds between every HTTP Transaction ensures a high number of concurrent connections. Load is increased until one or more of the defined breaking points is reached (the concurrent TCP connections breaking point does not apply to this test). Avalanche load specification is **Connections Per Second**.*

3.1.2	<b>Maximum Concurrent TCP Connections with 1K Response</b>
-------	--

*This test is designed to determine the maximum concurrent TCP connections of the DUT with a 1Kbyte object size.*

3.1.3	<b>Maximum Concurrent TCP Connections with 5K Response</b>
-------	--

*This test is designed to determine the maximum concurrent TCP connections of the DUT with a 5Kbyte object size. This is more typical of a “normal” network.*

## Test 3.2 - Behaviour Of The State Engine Under Load

This test determines whether the DUT is capable of preserving state across a large number of open connections over an extended time period.

At various points throughout the test (including after the maximum has been reached), it is confirmed that the DUT is still capable of detecting and blocking freshly-launched exploits, as well as confirming that the device does not block legitimate traffic (perhaps as a result of state tables filling up).

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

3.2.1	<b>Attack Detection - Light Load</b>
-------	--------------------------------------

*This test determines if the sensor is able to detect new exploits as the number of open sessions reaches 10 per cent of the maximum determined in Test 3.1.1*

3.2.2	<b>Attack Blocking - Light Load</b>
-------	-------------------------------------

*This test ensures that the sensor continues to block new exploits as the number of open sessions reaches 10 per cent of the maximum determined in Test 3.1.1*

### **3.2.3 State Preservation - Light Load**

*This test determines if the sensor maintains the state of pre-existing sessions as the number of open sessions reaches 10 per cent of the maximum determined in Test 3.1.1.*

*A legitimate HTTP session is opened and the first packet of a two-packet exploit is transmitted. As the number of open connections approached the maximum, the initial HTTP session is then completed with the second half of the exploit and the session is closed. If the sensor is still maintaining state on the original session, the exploit will be recorded. If the state tables have been exhausted, the exploit string will be seen as a non-stateful attack, and will thus be ignored.*

*Both halves of the exploit are required to trigger an alert - a product will fail the test if it fails to generate an alert after the second packet is transmitted, or if it raises an alert on either half of the exploit on its own*

### **3.2.4 Pass Legitimate Traffic - Light Load**

*This test ensures that the sensor continues to pass legitimate traffic as the number of open sessions reaches 10 per cent of the maximum determined in Test 3.1.1*

### **3.2.5 Attack Detection - Heavy Load**

*This test determines if the sensor is able to detect new exploits as the number of open sessions approaches the maximum determined in Test 3.1.1*

### **3.2.6 Attack Blocking - Heavy Load**

*This test ensures that the sensor continues to block new exploits as the number of open sessions approaches the maximum determined in Test 3.1.1*

### **3.2.7 State Preservation - Heavy Load**

*This test determines if the sensor maintains the state of pre-existing sessions as the number of open sessions approaches the maximum determined in Test 3.1.1. Method of execution is identical to Test 3.2.3*

### **3.2.8 Pass Legitimate Traffic - Heavy Load**

*This test ensures that the sensor continues to pass legitimate traffic as the number of open sessions approaches the maximum determined in Test 3.1.1*

### **3.2.9 Attack Detection - Maximum Exceeded**

*This test determines if the sensor is able to detect new exploits as the number of open sessions exceed the maximum determined in Test 3.1.1*

### **3.2.10 Attack Blocking - Maximum Exceeded**

*This test ensures that the sensor continues to block new exploits as the number of open sessions exceed the maximum determined in Test 3.1.1*

### **3.2.11 State Preservation - Maximum Exceeded**

*This test determines if the sensor maintains the state of pre-existing sessions as the number of open sessions exceed the maximum determined in Test 3.1.1. Method of execution is identical to Test 3.2.3*

### **3.2.12 Pass Legitimate Traffic - Maximum Exceeded**

*This test ensures that the sensor continues to pass legitimate traffic as the number of open sessions exceed the maximum determined in Test 3.1.1*

## **Test 3.3 - Stateless Attack Replay (Mid-Flows)**

This test determines whether the sensor is resistant to stateless attack flooding tools such as *Stick* and *Snot* - these utilities are used to generate large numbers of false alerts on the protected subnet using valid source and destination addresses and a range of protocols.

The main characteristic of tools such as *Stick* and *Snot* is the fact that they generate single packets containing "trigger" patterns without first attempting to establish a connection with the target server. Whilst this can be effective in raising alerts with some stateless protocols such as UDP and ICMP, they should never be capable of raising an alert for exploits based on stateful protocols such as FTP and HTTP.

In this test, we transmit a number of packets taken from capture files of valid exploits, but without first establishing a valid session with the target server. We also remove the session tear down and acknowledgement packets so that the sensor can not “infer” that a valid connection was made.

In order to receive a “PASS” in this test, no alerts should be raised for any of the actual exploits. However, each packet should be blocked if possible since it represents a “broken” or “incomplete” session.

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

3.3.1	<b>Stateless Attack Replay - Trigger Packet Only</b>
-------	--

*Packets containing valid exploit “triggers” are transmitted via the DUT, but without first establishing a valid session with the target server. The session tear down and acknowledgement packets are also removed so that the sensor can not “infer” that a valid connection was made. This is typical of the traffic generated by tools such as Stick and Snot.*

3.3.2	<b>Stateless Attack Replay - Trigger Packet Plus Session Tear-Down</b>
-------	--

*Packets containing valid exploit “triggers” plus session tear down and acknowledgement packets are transmitted via the DUT, but without first establishing a valid session with the target server.*

It is permitted to configure the device to ignore mid-flows should the default be to raise alerts.

## **Section 4 – Detection/Blocking Performance Under Load**

---

The aim of this section is to verify that the DUT is capable of detecting and blocking exploits when subjected to increasing loads of background traffic up to the maximum bandwidth supported, as claimed by the vendor. NB: the DUT should be configured to block all traffic when resources are exhausted or when traffic cannot be analysed for any reason. Any device which passes malicious traffic under the above conditions will fail.

The latest signature pack is acquired from the vendor, and sensors are deployed with the default/recommended settings applied as used for Test 1.1 (some audit/informational signatures may be disabled). Each sensor is configured to **detect and block** suspicious traffic.

The “attacker” host launches a fixed number of exploits at a target host on the subnet being protected by the DUT. The Adtech network monitor is configured to monitor the switch SPAN port consisting of normal, exploit and background traffic, and is capable of reporting the total number of exploit packets seen on the wire as verification.

A fixed number of exploits are launched with zero background traffic to ensure the DUT is capable of detecting the baseline attacks. Once that has been established, increasing levels of varying types of background traffic are generated **through** the DUT in order to determine the point at which the DUT begins to miss attacks.

All tests are repeated with 250Mbps, 500Mbps, 750Mbps and 1Gbps of background traffic (or up to the maximum rated throughput of the device should this be less than 1Gbps) running across a single in-line port pair.

At all stages, the Adtech network monitor verifies both the overall traffic loading and the total number of exploits seen on the target subnet. An additional confirmation is provided by the target host which reports the number of exploits which actually made it through.

The *Attack Blocking Rate* (ABR) at each background load is expressed as a percentage of the number of exploits blocked by the sensor (when in blocking mode) against the number verified by the Adtech network monitor and target host. The *Attack Detection Rate* (ADR) at each background load is expressed as a percentage of the number of exploits detected by the sensor (with blocking mode disabled) against the number verified by the Adtech network monitor and target host.

For each type of background traffic, we also determine the maximum load the sensor can sustain before it begins to drop packets/miss alerts. It is worth noting that devices which demonstrate 100 per cent ABR (blocking) but less than 100 per cent ADR (detection) in these tests will be prone to blocking **legitimate** traffic under similar loads.

Any device which demonstrates less than 100 per cent ABR under all load conditions (i.e. malicious traffic is permitted to pass through the DUT) will fail the overall test immediately.

## Test 4.1 - Raw Packet Processing Performance (UDP Traffic)

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 packet size - with variable source IP addresses and ports transmitting to a single fixed IP address/port - is transmitted through a single in-line port pair in the DUT (bi-directionally, maximum of 1Gbps total in both directions).

Each packet contains dummy data, and is targeted at a valid port 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. No TCP sessions are created during this test, and there is very little for the detection engine to do in the way of protocol analysis (although each vendor will be required to write a signature to detect the test packets to ensure that they are being passed through the detection engine and not “fast-tracked” from the inbound to outbound port).

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

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

4.1.0	<b>64 Byte Packets</b>
-------	------------------------

*Maximum 1,480,000 Packets Per Second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic.*

*Note that the device is tested with 64 byte packets to ensure it remains stable, but the figures are not reported.*

4.1.1	<b>128 Byte Packets</b>
-------	-------------------------

*Maximum 842,000 Packets Per Second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic.*

4.1.2	<b>256 Byte Packets</b>
-------	-------------------------

*Maximum 452,000 Packets Per Second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic.*

#### 4.1.3 512 Byte Packets

*Maximum 235,000 Packets Per Second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic.*

*This test provides a reasonable indication of the ability of a device to process packets from the wire on an “average” network.*

#### 4.1.4 1024 Byte Packets

*Maximum 120,000 Packets Per Second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic.*

#### 4.1.5 1514 Byte Packets

*Maximum 82,000 Packets Per Second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic.*

*This test has been included mainly to demonstrate how easy it is to achieve good results using large packets – beware of test results that **only** quote performance figures using similar packet sizes.*

## Test 4.2 - HTTP Maximum Capacity

HTTP is the most widely used protocol in most normal networks, as well as being one of the most widely exploited. The number of potential HTTP exploits for the protocol makes a pure HTTP network something of a torture test for the average sensor.

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 8 Gbps as a background load for our tests. Our Avalanche configuration is capable of simulating over 8 million users, with over 8 million concurrent sessions, and over 400,000 HTTP requests per second. The IPS test is currently designed to test devices to a maximum of 1Gbps using a **single** port pair connected in-line between two switches.

The aim of this test is to stress the HTTP detection engine and determine how the sensor copes with large numbers of TCP connections per second and HTTP transactions per second. All packets contain valid payload and address data, and this test provides an excellent representation of a live network (albeit one biased towards HTTP traffic) at various connection/transaction rates.

### Test ID Test Description

#### 4.2.1 Maximum TCP Connections Per Second (8 byte Response)

*This test is designed to determine the maximum TCP connection rate of the DUT with an 8 byte HTTP response size. The response size defines the number of bytes contained in the body, excluding any bytes associated with the HTTP header. An 8 byte response size is designed to provide a theoretical maximum 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 the request is satisfied, thus any concurrent TCP connections will be caused purely as a result of latency within the DUT. Load is increased until one or more of the breaking points defined in Test 3.1 is reached. Avalanche load specification is **Connections Per Second**.*

#### 4.2.2 **Maximum TCP Connections Per Second (1Kbyte Response)**

*This test is designed to determine the maximum TCP connection rate of the DUT with a 1000 byte HTTP response size. The object size defines the number of bytes contained in the body, excluding any bytes associated with the HTTP header.*

*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 the request is satisfied, thus any concurrent TCP connections will be caused purely as a result of latency within the DUT. Load is increased until one or more of the breaking points defined in Test 3.1 is reached. Avalanche load specification is **Connections Per Second**.*

#### 4.2.3 **Maximum TCP Connections Per Second (5Kbyte Response)**

*This test is designed to determine the maximum TCP connection rate of the DUT with a 4700 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 5Kbyte response size is designed to provide an indication of connections per second rate on a typical network.*

*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 the request is satisfied, thus any concurrent TCP connections will be caused purely as a result of latency within the DUT. Load is increased until one or more of the breaking points defined in Test 3.1 is reached. Avalanche load specification is **Connections Per Second**.*

#### 4.2.4 **Maximum HTTP Transactions Per Second (8 Byte Response)**

*This test is designed to determine the maximum HTTP transaction rate of the DUT with an 8 byte HTTP response size. The object size defines the number of bytes contained in the body, excluding any bytes associated with the HTTP header. An 8 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 in Test 3.1 is reached. Avalanche load specification is **Transactions Per Second**.*

#### 4.2.5 **Maximum HTTP Transactions Per Second (1Kbyte Response)**

*This test is designed to determine the maximum HTTP transaction rate of the DUT with a 1000 byte HTTP response size. The object size defines the number of bytes contained in the body, excluding any bytes associated with the HTTP header.*

*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 in Test 3.1 is reached. Avalanche load specification is **Transactions Per Second**.*

#### 4.2.6 **Maximum HTTP Transactions Per Second (5Kbyte Response)**

*This test is designed to determine the maximum HTTP transaction rate of the DUT with a 4700 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 5Kbyte response size is designed to provide an indication of connections per second rate on a typical network.*

*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 in Test 3.1 is reached. Avalanche load specification is **Transactions Per Second**.*

## Test 4.3 - HTTP Capacity With No Transaction Delays

The aim of these tests is to stress the HTTP detection engine and determine how the sensor copes with detecting and blocking exploits under network loads of varying average packet size and varying connections per second. By creating genuine session-based traffic with varying session lengths, the sensor 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, whilst 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.

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

<b>4.3.1</b>	<b>44Kbyte Response</b>
--------------	-------------------------

*Max 2,500 new connections per second - average packet size 1000 bytes - maximum 120,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1Gbps of background traffic. With relatively low connection rates and large packet sizes, we expect all sensors to perform well throughout this test.*

<b>4.3.2</b>	<b>21Kbyte Response</b>
--------------	-------------------------

*Max 5,000 new connections per second - average packet size 540 bytes - maximum 225,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1Gbps of background traffic. With average connection rates and average packet sizes, this is a good approximation of a real-world production network, and we expect all sensors to perform well throughout this test.*

<b>4.3.3</b>	<b>11Kbyte Response</b>
--------------	-------------------------

*Max 10,000 new connections per second - average packet size 440 bytes - maximum 275,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1Gbps of background traffic. With average packet sizes coupled with very high connection rates, this is a strenuous test for any sensor, and represents a very heavily used production network.*

<b>4.3.4</b>	<b>5Kbyte Response</b>
--------------	------------------------

*Max 20,000 new connections per second - average packet size 360 bytes - maximum 320,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1Gbps of background traffic. With small packet sizes and extremely high connection rates this is an extreme test for any sensor. Not many sensors will perform well at all levels of this test.*

## Test 4.4 - HTTP Capacity With Transaction Delays

This test is identical to Test 4.3 except that we introduce a 10 second delay in the server response for each transaction. This has the effect of maintaining a high number of open connections throughout the test, thus forcing the sensor to utilise additional resources to track those connections.

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

<b>4.4.1</b>	<b>21Kbyte Response With Delay</b>
--------------	------------------------------------

*Max 5,000 new connections per second - average packet size 540 bytes - maximum 225,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1Gbps of background traffic. 10 second transaction delay resulting in a maximum of 50,000 open connections during the test. With average connection rates and average packet sizes, this is a good approximation of a real-world production network, and we expect all sensors to perform well throughout this test.*

<b>4.4.2</b>	<b>11Kbyte Response With Delay</b>
--------------	------------------------------------

*Max 10,000 new connections per second - average packet size 440 bytes -*

*maximum 275,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1Gbps of background traffic. 10 second transaction delay resulting in a maximum of 100,000 open connections during the test. With average packet sizes coupled with very high connection rates, this is a strenuous test for any sensor, and represents a very heavily used production network.*

## Test 4.5 - “Real World” Protocol Mix Traffic

Whereas Tests 4.3 and 4.4 provide a pure HTTP environment with varying connection rates and average packet sizes, the aim of this test is to simulate more of a “real world” environment by introducing additional protocols whilst still maintaining a precisely repeatable and consistent background traffic load (something rarely seen in a real world environment).

The result is a background traffic load that, whilst less stressful than previous tests, is closer to what may be found on a heavily-utilised “normal” production network.

### Test ID    Test Description

#### 4.5.1    “Real World” Protocol Mix

*Traffic is played across the DUT comprising the following protocol mix:*

*60% HTTP  
20% SMTP  
10% POP3  
5% FTP  
5% DNS*

*Max 5,000 new connections per second - average packet size 540 bytes - maximum 200,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1Gbps of background traffic. Maximum of 1,000 open connections during the test.*

*With lower connection rates, average packets sizes, and a common protocol mix comprising protocols which all require inspection by the IPS engine, this is a good approximation of a heavily-used production network. We expect all sensors to perform well throughout this test.*

## 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 a high degree of latency on the packets passing through it, a network or security administrator would need to think carefully about how many devices could be installed in a single data path before user response times became unacceptable or the combination of devices caused excessive timeouts. This is particularly critical for a Gigabit-capable device, since it is more likely to be installed in the core of a high-speed network.

These tests also determine the effect of high levels of normal HTTP traffic and a basic DOS attack on the average latency and user response times.

### Test 5.1 - Latency

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

The Spirent SmartFlow software runs through several iterations of the test, varying the traffic load through a single in-line port pair from 250Mbps to 1Gbps bi-directionally (or up to the maximum rated throughput of the DUT should this be less than 1Gbps) in steps of 250Mbps.

This is repeated for a range of packet sizes (128, 256, 512, 1024 and 1514 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. Baseline latency figures of the test rig are then subtracted to provide actual latency figures for the DUT. It will be verified that each port pair demonstrates the same latency characteristics.

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.

<b>Test ID</b>	<b>Test Description</b>
----------------	-------------------------

<b>5.1.1</b>	<b><i>Latency With No Background Traffic</i></b>
--------------	--

*SmartFlow traffic is passed across the infrastructure switches and through one in-line port pair of the DUT (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 (128, 256, 512, 1024 and 1514 bytes) and each load level from 250Mbps to 1Gbps (in 250Mbps steps).*

<b>5.1.2</b>	<b><i>Latency With Background Traffic Load</i></b>
--------------	--

*The Avalanche devices are configured to generate HTTP traffic through each in-line port pair of the DUT up to 50 per cent of the maximum rated bandwidth of the DUT - maximum 2Gbps (500Mbps per port pair) - maximum 10,000 new connections per second - average packet size 540 bytes - maximum 450,000 packets per second.*

*A trickle (less than 1Mbps) of bi-directional SmartFlow traffic at various packet sizes (128, 256, 512, 1024 and 1514 bytes) is then passed across the infrastructure switches and through the DUT, and the packet loss and average latency are recorded.*

<b>5.1.3</b>	<b><i>Latency Under DDOS Attack</i></b>
--------------	---

*The Spirent WebSuite software is used to generate a fixed load of DOS/DDOS traffic of 10 per cent of the maximum rated bandwidth (maximum 100Mbps) on any one of the in-line port pairs of the DUT. A trickle (less than 1Mbps) of bi-directional SmartFlow traffic at various packet (128, 256, 512, 1024 and 1514 bytes) is then passed across the infrastructure switches and through the DUT, and the packet loss and average latency are recorded. The DUT should be configured to detect/block/mitigate the DOS attack by the most efficient method available.*

## **Test 5.2 - User Response Times**

Avalanche and Reflector devices are used to generate HTTP sessions through the device in order to gauge how any increases in latency will impact the user experience in terms of failed connections and increased Web response times.

Baseline HTTP response figures of the test rig are subtracted to provide actual response figures for the DUT.

<b>Test ID</b>	<b>Test Description</b>
----------------	-------------------------

<b>5.2.1</b>	<b><i>Web Response With No Background Traffic (21Kbyte Response)</i></b>
--------------	--

*The Avalanche devices are configured to generate HTTP traffic through the DUT up to 50 per cent of the maximum rated bandwidth of the DUT - maximum 500Mbps - maximum 2,500 new connections per second - average packet size 540 bytes - maximum 112,500 packets per second.*

*The minimum, maximum and average page response times and number of failed connections for each port pair (plus overall DUT performance) are recorded by Avalanche to provide an indication of the expected response times under normal traffic conditions.*

<b>5.2.2</b>	<b><i>Web Response Under DDOS Attack (21Kbyte Response)</i></b>
--------------	---

*The Avalanche devices are configured to generate HTTP traffic through the DUT*

as for Test 5.2.1. The Spirent WebSuite software is then used to generate DOS/DDOS traffic up to 10 per cent of the maximum rated bandwidth (maximum 100Mbps) of the DUT.

The minimum, maximum and average page response times and number of failed connections are recorded by Avalanche to provide an indication of the expected response times when the device is under attack.

## 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 IPS device, where failure can produce network outages.

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

<b>6.1.1</b>	<b>Blocking Under Extended Attack</b>
--------------	---------------------------------------

*For this test, the external interface of the device is exposed to a constant stream of exploits over an extended period of time. The device is configured to block and alert, and thus this test provides an indication 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 at a maximum of 100Mbps (max 50,000 packets per second, average packet sizes in the range of 120-350 bytes) 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 exploits, raising an alert for each. If any recognisable exploits are passed - caused by either the volume of traffic or the sensor failing open for any reason - this will result in a FAIL.*

<b>6.1.2</b>	<b>Passing Legitimate Traffic Under Extended Attack:</b> 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 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 sensor failing closed for any reason - this will result in a FAIL.*

<b>6.1.3</b>	<b>ISIC/ESIC/TCPSIC/UDPSIC/ICMPSIC</b>
--------------	--

*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 switch on the external interfaces of the DUT, and the ISIC target to the switch on the internal interfaces. ISIC traffic is transmitted through a single in-line port pair of the DUT 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.*

<b>6.1.4</b>	<b>Policy Push</b>
--------------	--------------------

*The Avalanche devices are configured to generate HTTP traffic through each in-line port pair of the DUT up to 50 per cent of the maximum rated bandwidth of the DUT - maximum 500Mbps - maximum 2,500 new connections per second - average packet size 540 bytes - maximum 112,500 packets per second.*

*A new policy is pushed to the DUT during the test, and the minimum, maximum and average page response times and number of failed connections are recorded by Avalanche. These results can be compared with Test 5.2.1 to provide an indication of the effect of pushing policies on legitimate traffic.*

<b>6.1.5</b>	<b>Power Fail</b>
--------------	-------------------

*The Avalanche devices are configured to generate HTTP traffic through each in-line port pair of the DUT up to 50 per cent of the maximum rated bandwidth of the DUT - maximum 500Mbps - maximum 2,500 new connections per second - average packet size 540 bytes - maximum 112,500 packets per second.*

*Power to the DUT is cut during the test, and the minimum, maximum and average page response times and number of failed connections are recorded by*

*Avalanche. If the device is configured to fail open, there should be minimal loss of legitimate sessions throughout the test (over and above the baseline loss expected through switch renegotiation). If the device is configured to fail closed, no traffic should be passed once power has been cut.*

**6.1.6 Redundancy**

*Does the DUT include multiple redundant critical components (fans, power supplies, hard drive, etc.) (YES/NO/OPTION)*

**6.1.7 Fail Open (Power Fail)**

*Does the DUT provide the ability to fail open with minimal/zero loss of legitimate traffic (either via built-in, or optional hardware bypass) following power failure (YES/NO/OPTION).*

**6.1.8 Fail Open (Resource Issues)**

*Does the DUT provide the ability to pass all traffic when resources are exhausted or it is no longer possible to analyse traffic for any reason (i.e. packet rate exceeds device capabilities)*

**6.1.9 Fail Closed (Power Fail)**

*Does the DUT provide the ability to fail closed following power failure (YES/NO/OPTION)*

**6.1.10 Fail Closed (Resource Issues)**

*Does the DUT provide the ability to block all traffic when resources are exhausted or it is no longer possible to analyse traffic for any reason (i.e. packet rate exceeds device capabilities)*

**6.1.11 High Availability (HA) Option (Stateful)**

*Is an HA option available for this device, providing fully stateful active-active or active-passive failover between devices (YES/NO)*

**6.1.12 High Availability (HA) Option (Non-stateful)**

*Is an HA option available for this device, providing any form of failover between devices where existing connections may be lost during failover (YES/NO)*

**6.1.13 Persistence Of Data**

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

**6.1.14 IPV6**

*The DUT should be capable of detecting exploits over both IPV6 and IPV4.*

## 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 IDS/IPS 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 Required**

*The vendor will list the open ports and active services on the management*

interface along with their use.

**7.1.2 Open Ports Detected**

The management port will be scanned to determine ports/services visible on the management interface. If any ports additional to those listed in Test 7.1.1 are discovered, this will result in an automatic FAIL.

**7.1.3 ISIC/ESIC/TCPSIC/UDPSIC/ICMPSIC:** This test attempts to stress the protocol stack of the management interface of the DUT by exposing it to traffic from the ISIC test tool. The ISIC test tool host is connected to the switch on the management interface of the DUT, and that interface is also the target. ISIC traffic is transmitted to the management interface of the DUT 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

**7.1.4 ISIC/ESIC/TCPSIC/UDPSIC/ICMPSIC Detection**

Are ISIC attacks detected by the DUT even though targeted at the management port (YES/NO).

## Test 7.2 - Management & Configuration - General

In addition to the specific tests noted below, NSS will execute an in-depth technical evaluation covering all the main features and benefits of the DUT. The accompanying report will fully evaluate each product in terms of ease of use, management and configuration, and alerting and reporting capabilities.

**Test ID Test Description**

**7.2.1 Transparent Mode**

The DUT must be capable of running in transparent bridge mode, with no IP address assigned to detection ports. Detection ports should ignore all direct connection attempts.

**7.2.2 Management Port**

The DUT should feature a dedicated management port, separate from detection ports. Although this is the preferred configuration, lack of a management port (requiring DUT to be managed via one of the detection ports) will not be cause for failure providing management connection and communication is securely encrypted.

**7.2.3 Management Protocol**

Connection from management console to DUT should be protected by a minimum of a user name/password combination or multi-factor authentication system, and all communications between should be securely encrypted.

Where a three-tier management architecture is employed, all communication between console and management server(s), and between management server(s) and sensor(s) should be securely encrypted.

**7.2.4 Authentication**

Access to management console should be protected by a granular user authentication system which allows for separation of read only and read-write access, preventing users who require reporting access only from modifying device parameters, etc. No access to administrative functions should be permitted (using either direct or centralised administration capabilities) without proper authentication.

**7.2.5 Enterprise Authentication**

Access to management console should be protected by a granular user authentication system which allows for restriction of individual users to specific devices, ports, reports, alerts and security policies. Authenticated users should be unable to access devices/ports/policies/alerts/reports/etc. restricted to other users of the system.

**7.2.6 Direct Device Management (Optional)**

Direct access to the DUT should be provided (either via command line or Web

interface) for single-device management.

**7.2.7 Centralised Device Management**

A centralised management system should be provided to manage one or more sensors from a single point, including centralised device configuration, policy definition, alert handling and reporting for all sensors under the control of the management system. This should be scalable to large numbers of sensors.

**7.2.8 Pass-Through Mode (Optional)**

It should be possible to place the DUT into a mode whereby all traffic is allowed to pass through the device, but data will be logged according to the policy in place at the time (thus, the DUT will log alerts and state whether the packets would have been dropped, session terminated, etc., but without enforcing those actions on the traffic processed). This should be via a single system-wide operation via the management console or DUT command line (i.e. it is not permitted to achieve this by requiring that all BLOCK signatures be amended to LOG ONLY, or by switching policies - it must be achieved without affecting the current policy in force).

**7.2.9 Signature Update**

The vendor should demonstrate access to a vulnerability research capability (either in-house or via a recognised third-party) which is able to provide timely and accurate signature updates at regular intervals.

**7.2.10 Secure Sensor Registration**

Initial registration of sensor to central management console should be in a fully secure manner (it is permitted to offer a less secure/rapid option, but this should not be the default).

**7.2.11 Documentation**

Adequate documentation should be provided for both installation, and day-to-day management.

## Test 7.3 - Management & Configuration - Policy

**Test ID Test Description**

**7.3.1 Sensor Configuration**

The management system should provide the means to configure one or more sensors from a central location, assigning signatures, sensor settings, etc.

**7.3.2 Policy Definition**

The management system should provide the means to define and save multiple security policies, consisting of:

- General sensor configuration
- System-wide parameters
- Signatures enabled/disabled
- Actions to take when malicious traffic discovered

**7.3.3 Recommended Settings**

The vendor should provide a default policy or suite of recommended settings which comprises the optimum configuration for a typical network (including which signatures are enabled/disabled, which are enabled in blocking mode, required actions, etc.)

**7.3.4 Custom Attack Signatures**

It should be possible for the administrator to be able to define custom attack signatures for use in standard policies

**7.3.5 Bulk Operations**

It should be possible to search quickly and easily for individual signatures or groups/classes of signatures, and subsequently to apply one or more operations to an entire group in a single operation (for example, to enable or disable a group of signatures, or to switch a group from block mode to log mode, etc.)

**7.3.6 Granularity**

The DUT should be capable of blocking or creating exceptions based on IP address, application, protocol, VLAN tag, etc. (i.e. never block HTTP traffic between two specific IP addresses, always block FTP traffic to one specific IP

address, etc.).

**7.3.7 Policy Association**

Once policies have been defined, it should be possible to associate them with specific devices or groups of devices.

**7.3.8 Inheritance**

It should be possible to create groups and sub-groups of devices such that sub-groups can inherit certain aspects of configuration and policy definition from parent groups.

**7.3.9 Virtualisation**

Once policies have been defined, it should be possible to associate them with specific "virtual" devices or groups of devices, comprising an entire DUT, individual ports, port groups, IP address range, subnet or VLAN.

**7.3.10 Policy Deployment**

Once policies have been defined, it should be possible to distribute them to the appropriate device(s), virtual device(s), or groups of devices in a single operation.

**7.3.11 Policy Auditing**

All changes to policies should be logged centrally. Log data should include at a minimum the date/time the changes were made, and the identity of the user who made them. If possible (OPTIONAL) the system should record the actual changes.

**7.3.12 Policy Version Control**

All changes to policies should be recorded by saving a version of the policy before each change. It should be possible to roll-back to a previous version of any policy via a single operation.

## Test 7.4 - Management & Configuration - Alert Handling

**Test ID Test Description**

**7.4.1 Required Log Events**

The DUT should record log entries for the following events:

- Detection of malicious traffic
- Termination of a session
- Successful authentication by administrator
- Unsuccessful authentication by administrator
- Policy changed
- Policy deployed
- Hardware failure
- Power cycle

**7.4.2 Log Location (Optional)**

The log events should be logged on the DUT initially, in a secure manner, and subsequently transmitted to a central repository for permanent storage.

**7.4.3 Communication Interruption**

Where communications between sensor and console/management server are interrupted, storage capacity on the DUT should be sufficient to hold one week's worth of log data on a typical network. If it is not possible to restore communication in a timely manner, once the local logs are full, the DUT should either (1) continue passing traffic and overwrite oldest log entries, or (2) stop passing traffic. This option should be configurable by the administrator.

**7.4.4 Log Flooding**

Mechanisms should be in place (aggregation) to prevent the DUT from flooding the management server/console with too many events of the same type in a short interval. However, it should be possible to disable aggregation/flood protection completely for testing purposes to ensure NSS can see every individual alert.

**7.4.5 Alerts**

The DUT should record log entries each time it detects malicious traffic. At a minimum (depending on protocol), these log entries should contain:

Unique event ID  
Date and time of event  
Device ID (includes sensor ID, port ID, etc.)  
Direction of traffic (physical/logical source and destination interfaces)  
Detection engine which raised the alert (OPTIONAL)  
Source IP address  
Source port/service (where applicable)  
Destination IP address  
Destination port/service (where applicable)  
ICMP message type and code (where applicable)  
Protocol  
Unique signature ID  
Human-readable description of the event/exploit  
CVE reference, Bugtraq ID, or other non-vendor-specific identifier  
Action taken by the DUT (block, log, etc.)

#### **7.4.6 Alert Accuracy**

The DUT should record log entries which are accurate and human readable without having to use additional reference material. The DUT should attempt to minimise the number of alerts raised for a single event wherever possible.

#### **7.4.7 Centralised Alerts**

No matter how many sensors are installed, all alerts should be delivered to, and handled by, a single, central, management console. From that console, it should be possible to view all alerts globally, or select alerts from individual devices (logical or physical).

#### **7.4.8 Alert Delivery Mechanism**

At a minimum, the DUT should be able to deliver alerts in a timely manner to a central database for permanent storage, central console for a real-time display, and SMTP server for e-mail alerts.

#### **7.4.9 Alert Actions (Mandatory)**

On detecting malicious traffic, the DUT should be able to perform the following actions at a minimum:

Ignore  
Log only  
Drop packet (no reset)  
Drop session (no reset)  
E-mail administrator

#### **7.4.10 Alert Actions (Optional)**

On detecting malicious traffic, the DUT may optionally be able to perform the following actions:

Send TCP reset (or ICMP redirect) to source only  
Send TCP reset (or ICMP redirect) to destination only  
Send TCP reset (or ICMP redirect) to both source and destination  
Reconfigure external firewall  
Reconfigure switch to isolate/quarantine offending port  
Page administrator

#### **7.4.11 Forensic Analysis**

The DUT should provide the ability to capture individual packets, a range of packets, or an entire session where required (globally, or on a rule-by-rule basis)

#### **7.4.12 Summarise Alerts**

The central console should provide the ability to select a particular piece of data from an alert and summarise on that data field (i.e. select a source IP address and view all alerts for that source IP). Alternatively, it should be possible to construct data filters manually in a search form and summarise on the specified search criteria. The preferred scenario is to offer both of these options.

#### **7.4.13 View Alert Detail**

The central console should provide the ability to select an individual alert and view the following information at a minimum:

Detailed alert data (including all data mentioned in Test 7.4.5)  
Detailed exploit data (description of the exploit research)  
Signature/rule  
Remediation data/preventative action

#### **7.4.14 View Policy**

*Having selected an alert, the system should provide the ability to access directly the policy and rule which triggered the event in order to view and/or modify the policy for further fine tuning.*

**7.4.15 View Packet Contents (Optional)**

*The central console should provide the ability to select an individual alert and view the contents of the trigger packet or context data for the exploit.*

**7.4.16 Alert Suppression**

*The central console should provide the ability to create exception filters based on alert data to eliminate further alerts which match the specified criteria (i.e. same alert ID from same source IP). This does not disable detection, logging or blocking, but merely excludes alerts from the console display.*

**7.4.17 Correlation (Automatic)**

*The system should provide the means to infer connections between multiple alerts and group them together as incidents automatically.*

**7.4.18 Correlation (Manual)**

*The system should provide the means for the administrator to infer connections between multiple alerts and group them together as incidents manually.*

**7.4.19 Incident Workflow**

*The system should provide the ability to annotate and track incidents to resolution.*

## **Test 7.5 - Management & Configuration - Reporting**

**Test ID Test Description**

**7.5.1 Centralised Reports**

*No matter how many sensors are installed, the system should be capable of reporting on all alerts from a single, central, management console. From that console, it should be possible to report all alerts globally, or to report on alerts from individual devices (logical or physical).*

**7.5.2 Top Attacks**

*The system should provide a report listing the top N attacks in the previous hour, day, week, month, year, or custom date range.*

**7.5.3 Top Sources**

*The system should provide a report listing the top N source IPs from which attacks have been detected in the previous hour, day, week, month, year, or custom date range.*

**7.5.4 Top Targets**

*The system should provide a report listing the top N target IPs at which attacks have been launched in the previous hour, day, week, month, year, or custom date range.*

**7.5.5 Top Services**

*The system should provide a report listing the top N target ports/services at which attacks have been launched in the previous hour, day, week, month, year, or custom date range.*

**7.5.6 Top Protocols**

*The system should provide a report listing the top N protocols over which attacks have been launched in the previous hour, day, week, month, year, or custom date range.*

**7.5.7 Custom Reports**

*The report generator should provide the ability to construct complex data filters in a search form and summarise alerts on the specified search criteria.*

**7.5.8 Saved Reports**

*Having defined a custom report filter, it should be possible to save it for subsequent recall.*

**7.5.9 Scheduled Reports**

*It should be possible to schedule saved reports for regular unattended runs. The output should be saved as HTML or PDF at a minimum. It should optionally be possible to publish to a central FTP/Web server, and/or e-mail reports to specified recipients.*

**7.5.10 Log File Maintenance**

*The system should provide for automatic rotation of log files, archiving, restoring from archive, and reporting from archived logs.*