NEXT GENERATION INTRUSION PREVENTION SYSTEM (NGIPS) TEST REPORT

Fortinet FortiGate 600D v5.4.5

(Vendor-Provided Settings)

NOVEMBER 7, 2017

Authors – Thomas Williams, Michael Shirley

Overview

NSS Labs performed an independent test of the Fortinet FortiGate 600D v5.4.5. The product was subjected to thorough testing at the NSS facility in Austin, Texas, based on the Next Generation Intrusion Prevention System (NGIPS) Test Methodology v3.1 available at www.nsslabs.com. This test was conducted free of charge and NSS did not receive any compensation in return for Fortinet's participation.

Vendor-Provided Settings

NGIPS products are deployed at the perimeter of a corporate network as well as within the network to protect employee desktops, laptops, and PCs. NSS research has determined that the majority of enterprises do not tune their NGIPS products, but rather rely on a vendor's default/recommended policies and settings. This product was tested using vendor-provided settings, i.e., the signatures/filters/rules that trigger false positives were turned off in order to replicate an enterprise environment.

This Test Report provides results of the product tested as configured and submitted by the vendor.

Product	Exploit Block Rate[1]	NSS-Tested Throughput	3-Year TCO (US$)
Fortinet FortiGate 600D v5.4.5	99.72%	3,707 Mbps	$14,139
	False Positives	Evasions Blocked	Stability and Reliability
	PASS	157/157	PASS

Figure 1 – Overall Test Results

Using the vendor-provided settings, the FortiGate 600D blocked 99.72% of attacks. The device proved to be effective against all evasion techniques tested. The device passed all of the stability and reliability tests.

The FortiGate 600D is rated by NSS at 3,707 Mbps, which is lower than the vendor-claimed performance (Fortinet rates this device at 4 Gbps). NSS-tested throughput is calculated as an average of all of the "real-world" protocol mixes and the 21 KB HTTP response-based capacity tests.

[1] Exploit block rate is defined as the number of live exploits (CAWS) and exploits from the *NSS Exploit Library* blocked under test.

Table of Contents

Table of Figures

Security Effectiveness

While the companion Comparative Reports on security, performance, and total cost of ownership (TCO) provide information about all tested products, this Test Report provides detailed information not available elsewhere.

This section verifies that the device under test is capable of enforcing the security policy effectively.

Vendor-Provided Settings

The policy used in this group test is as follows:

- Firmware Version: 5.4.5
- Build Number: 1138
- IPS Engine Version: 3.00428
- IPS Definitions Version: 11.00204
- AV Engine Version: 5.00247
- AV Definitions Version: 1.00000

False Positive Testing

The FortiGate 600D did not alert on non-malicious test traffic.

CAWS (Live Exploits)

For live testing, NSS employs a unique live test harness, the CAWS Continuous Security Validation Platform, to measure how well security products protect against "drive-by" exploits that target client applications.

CAWS captures thousands of suspicious URLs per day from threat data generated from NSS and its customers as well as data from open-source and commercial threat feeds. This list of URLs is optimized and assigned to victim machines, each of which has a unique combination of operating system (including service pack/patch level), browser, and client application. For further details, please visit www.nsslabs.com.

Figure 2 depicts the threat encounter block rate for the FortiGate 600D.

Product	CAWS (Live Exploits) Threat Encounters	Total Number Blocked	Block Percentage
Fortinet FortiGate 600D v5.4.5	1,491	1,491	100.00%

Figure 2 – Number of Threat Encounters Blocked (%)

Exploit Library

NSS' security effectiveness testing leverages the deep expertise of our engineers who utilize multiple commercial, open-source, and proprietary tools, including NSS' network live stack test environment[2] as appropriate. With 2,348 exploits, this is the industry's most comprehensive test to date. Most notably, all of the exploits and payloads in this test have been validated such that:

[2] See the NSS Cyber Advanced Warning System™ for more details.

- A reverse shell is returned
- A bind shell is opened on the target, allowing the attacker to execute arbitrary commands
- Arbitrary code is executed
- A malicious payload is installed
- A system is rendered unresponsive
- Etc.

Product	Total Number of Exploits Run	Total Number Blocked	Block Percentage
Fortinet FortiGate 600D v5.4.5	2,348	2,335	99.45%

Figure 3 – Number of Exploits Blocked (%)

Coverage by Attack Vector

Because a failure to block attacks could result in significant compromise and could severely impact critical business systems, network intrusion prevention systems should be evaluated against a broad set of exploits. Exploits can be categorized as either *attacker-initiated* or *target-initiated*. Attacker-initiated exploits are threats executed remotely against a vulnerable application and/or operating system by an individual, while target-initiated exploits are initiated by the vulnerable target. Target-initiated exploits are the most common type of attack experienced by the end user, and the attacker has little or no control as to when the threat is executed.

Figure 4 – Coverage by Attack Vector

Coverage by Impact Type

The most serious exploits are those that result in a remote system compromise, providing the attacker with the ability to execute arbitrary system-level commands. Most exploits in this class are “weaponized” and offer the attacker a fully interactive remote shell on the target client or server. Slightly less serious are attacks that result in an individual service compromise, but not arbitrary system-level command execution. Finally, there are attacks that result in a system- or service-level fault that crashes the targeted service or application and requires administrative action to restart the service or reboot the system. Clients can contact NSS for more information about these tests.

Coverage by Date

Figure 5 provides insight into whether or not a vendor is aging out protection signatures aggressively enough to preserve performance levels. It also reveals whether a product lags behind in protection for the most current vulnerabilities. NSS reports exploits by individual years for the past ten years. Exploits older than ten years are grouped together.

Figure 5 – Product Coverage by Date

Coverage by Target Vendor

Exploits within the *NSS Exploit Library* target a wide range of protocols and applications. Figure 6 depicts the coverage offered by the FortiGate 600D for some of the top vendor targets (out of more than 70) represented for this round of testing.

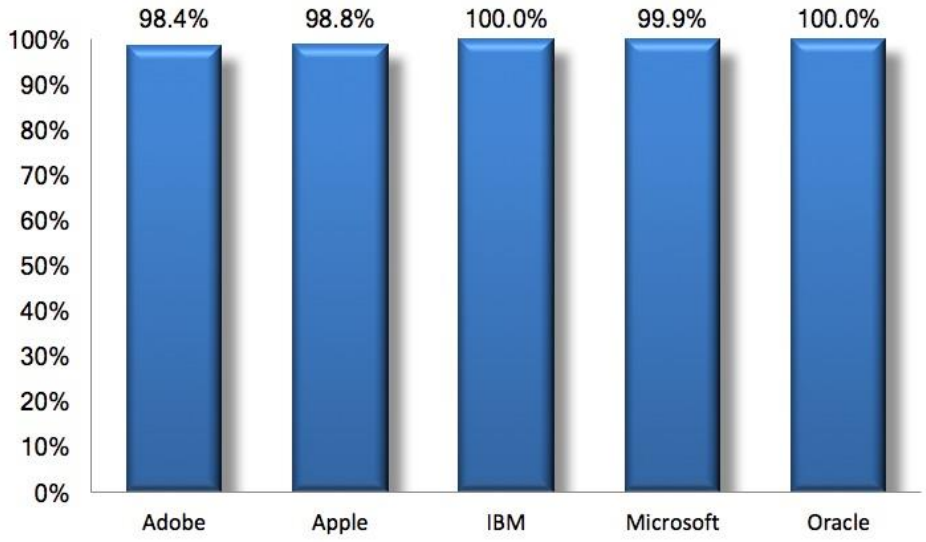


Figure 6 – Product Coverage by Target Vendor

Coverage by Target Type

These tests determine the protection provided against different types of exploits based on the target environment, for example, web server, web browser, database, ActiveX, Java, browser plugins, etc. Further details are available to NSS clients via inquiry call.

Resistance to Evasion Techniques

Evasion techniques are a means of disguising and modifying attacks at the point of delivery to avoid detection and blocking by security products. Failure of a security device to correctly identify a specific type of evasion potentially allows an attacker to use an entire class of exploits for which the device is assumed to have protection. This renders the device virtually useless. Many of the techniques used in this test have been widely known for years and should be considered minimum requirements for the NGIPS product category.

Providing exploit protection results without fully factoring in evasion can be misleading. The more classes of evasion that are missed (such as HTTP evasion, IP packet fragmentation, stream segmentation, RPC fragmentation, URL obfuscation, HTML obfuscation, and FTP evasion), the less effective the device. For example, it is better to miss all techniques in one evasion category, such as FTP evasion, than one technique in each category, which would result in a broader attack surface.

Furthermore, evasions operating at the lower layers of the network stack (IP packet fragmentation or stream segmentation) have a greater impact on security effectiveness than those operating at the upper layers (URL or FTP obfuscation.) Lower-level evasions will potentially impact a wider number of exploits; missing TCP segmentation, for example, is a much more serious issue than missing FTP obfuscation.

Figure 7 provides the results of the evasion tests for the FortiGate 600D. The FortiGate 600D blocked all 157 evasions it was tested against. For further detail, please reference Appendix A.

Test Procedure	Result
IP Packet Fragmentation	PASS
TCP Stream Segmentation	PASS
RPC Fragmentation	PASS
SMB & NetBIOS Evasion	PASS
URL Obfuscation	PASS
HTML Obfuscation	PASS
FTP/Telnet Evasion	PASS
Payload Encoding	PASS
IP Packet Fragmentation + TCP Segmentation	PASS
IP Fragmentation + MSRPC Fragmentation	PASS
IP Fragmentation + SMB Evasions	PASS
TCP Segmentation + SMB/NetBIOS Evasions	PASS
TCP Split Handshake	PASS
HTTP Evasion	PASS

Figure 7 – Resistance to Evasion Results

Performance

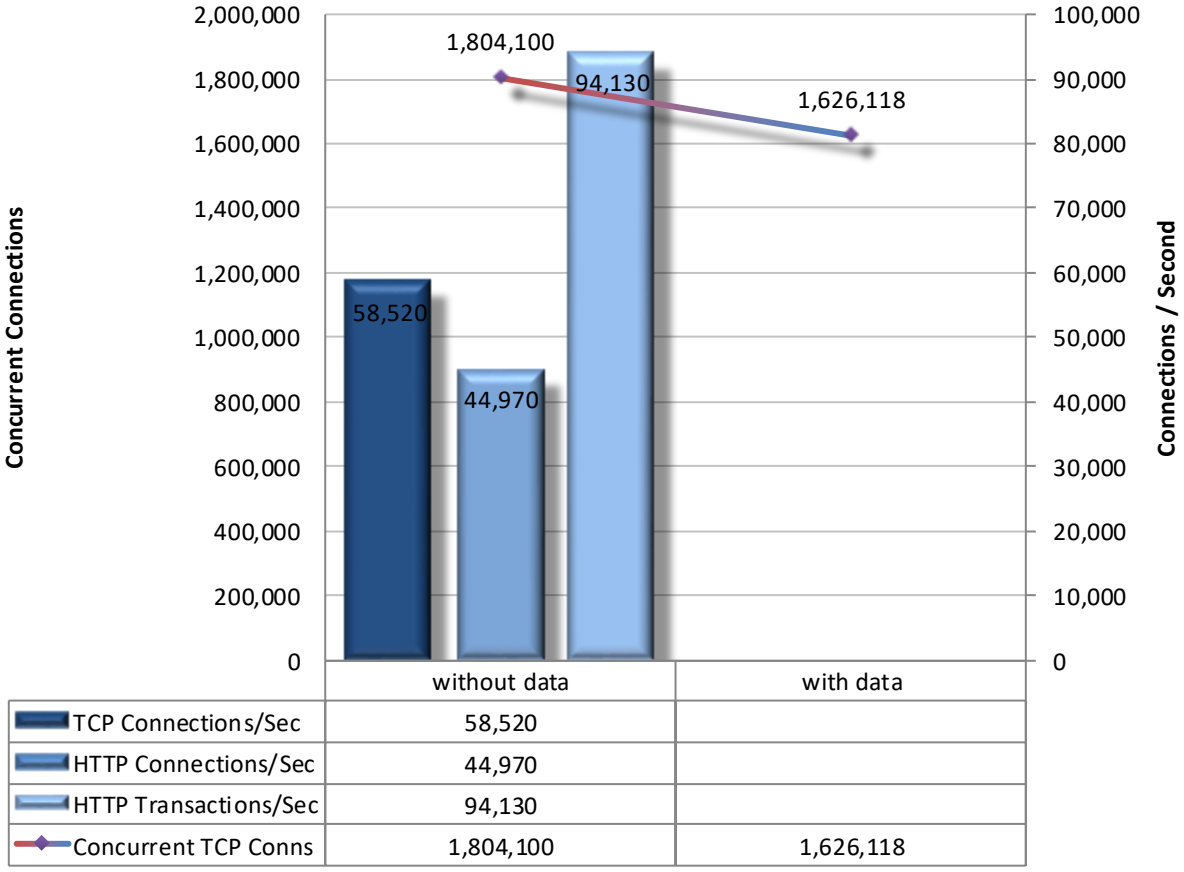
There is frequently a trade-off between security effectiveness and performance. Because of this trade-off, it is important to judge a product's security effectiveness within the context of its performance and vice versa. This ensures that new security protections do not adversely impact performance and that security shortcuts are not taken to maintain or improve performance.

Maximum Capacity

The use of traffic generation appliances allows NSS engineers to create "real-world" traffic at multi-Gigabit speeds as a background load for the tests. The aim of these tests is to stress the inspection engine and determine how it copes with high volumes of TCP connections per second, application layer transactions per second, and concurrent open connections. All packets contain valid payload and address data, and these tests provide an excellent representation of a live network at various connection/transaction rates.

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

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



	without data	with data
TCP Connections/Sec	58,520	
HTTP Connections/Sec	44,970	
HTTP Transactions/Sec	94,130	
Concurrent TCP Conns	1,804,100	1,626,118

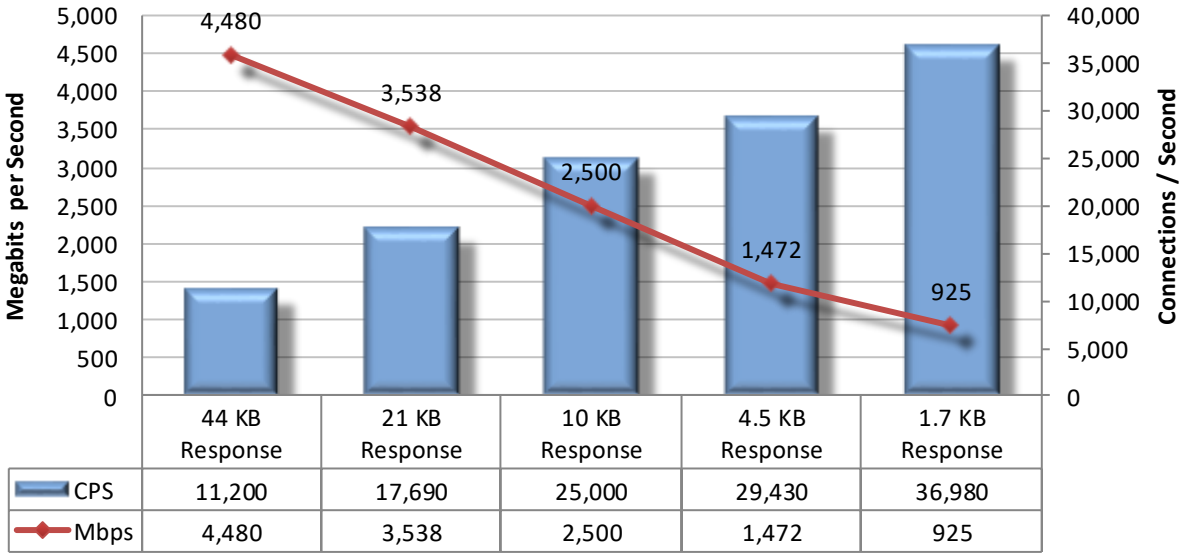
Figure 8 – Concurrency and Connection Rates

HTTP Connections per Second and Capacity

The aim of these tests is to stress the HTTP detection engine and determine how the device copes with network loads of varying average packet size and varying connections per second. By creating genuine session-based traffic with varying session lengths, the device 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 conditions as possible, while ensuring absolute accuracy and repeatability.

HTTP Capacity with No Transaction Delays

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. This test provides an excellent representation of a live network (albeit one biased toward HTTP traffic) at various network loads.



	44 KB Response	21 KB Response	10 KB Response	4.5 KB Response	1.7 KB Response
CPS	11,200	17,690	25,000	29,430	36,980
Mbps	4,480	3,538	2,500	1,472	925

Figure 9 – HTTP Connections per Second and Capacity

HTTP Capacity with Transaction Delays

Typical user behavior introduces delays between requests and responses (for example, "think time") as users read web pages and decide which links to click next. This group of tests is identical to the previous group except that these include a five-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 utilize additional resources to track those connections.

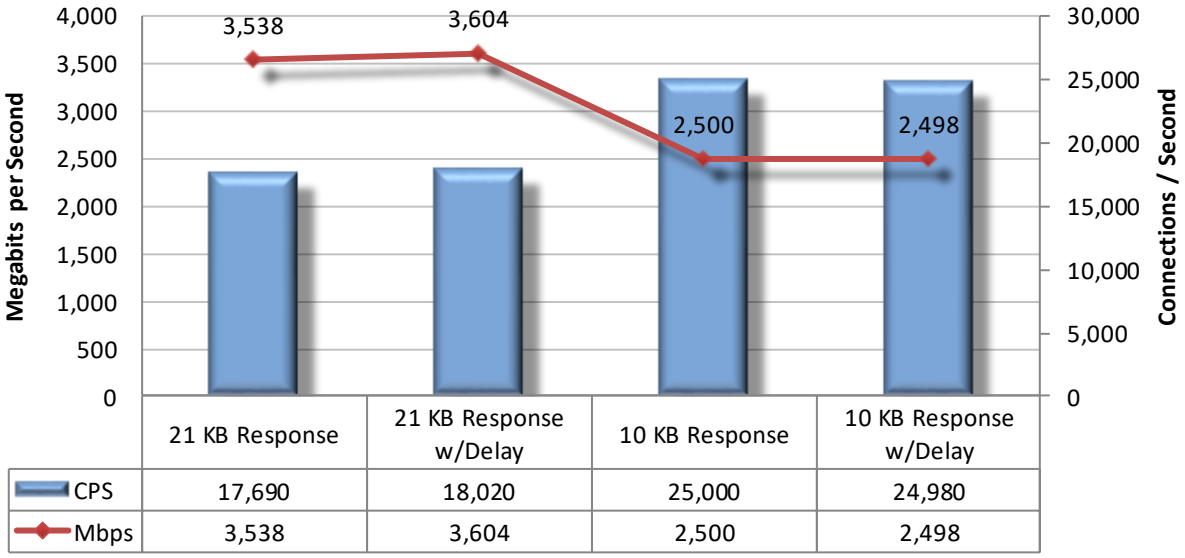


Figure 10 – HTTP Capacity with Transaction Delays

Application Average Response Time – HTTP

Application Average Response Time – HTTP (at 95% Maximum Load)	Milliseconds
44 KB Response	0.11
21 KB Response	0.15
10 KB Response	0.17
4.5 KB Response	0.12
1.7 KB Response	0.17

Figure 11 – Average Application Response Time (Milliseconds)

Real-World Traffic Mixes

This test measures the performance of the device in a "real-world" environment by introducing additional protocols and real content, while still maintaining a precisely repeatable and consistent background traffic load. Different protocol mixes are utilized based on the intended location of the device (network core or perimeter) to reflect real use cases. For details about real-world traffic protocol types and percentages, see the NSS Labs Next Generation Intrusion Prevention System Test Methodology, available at www.nsslabs.com.

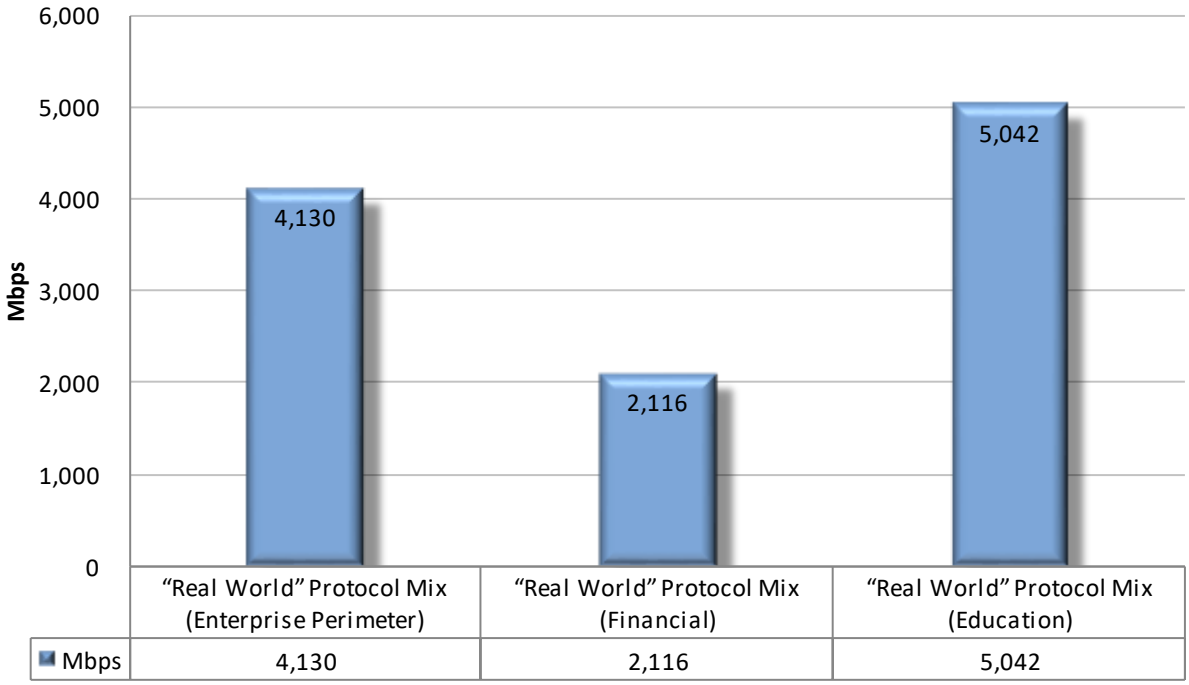


Figure 12 – Real-World Traffic Mixes

The FortiGate 600D was tested by NSS and performed above the throughput claimed by the vendor for all "real-world" traffic mixes except for the Financial mix, where it performed below its vendor-claimed throughput.

Raw Packet Processing Performance (UDP Throughput)

This test uses UDP packets of varying sizes generated by test equipment. A constant stream of the appropriate packet size, with variable source and destination IP addresses transmitting from a fixed source port to a fixed destination port, is transmitted bidirectionally through each port pair of the device.

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 frames per second (fps) figures across each inline port pair are verified by network monitoring tools before each test begins. Multiple tests are run and averages are taken where necessary.

This traffic does not attempt to simulate any form of a "real-world" network condition. No TCP sessions are created during this test, and there is very little for the state engine to do. The aim of this test is to determine the raw packet processing capability of each inline port pair of the device, and to determine the device's effectiveness at forwarding packets quickly, in order to provide the highest level of network performance with the least amount of latency.

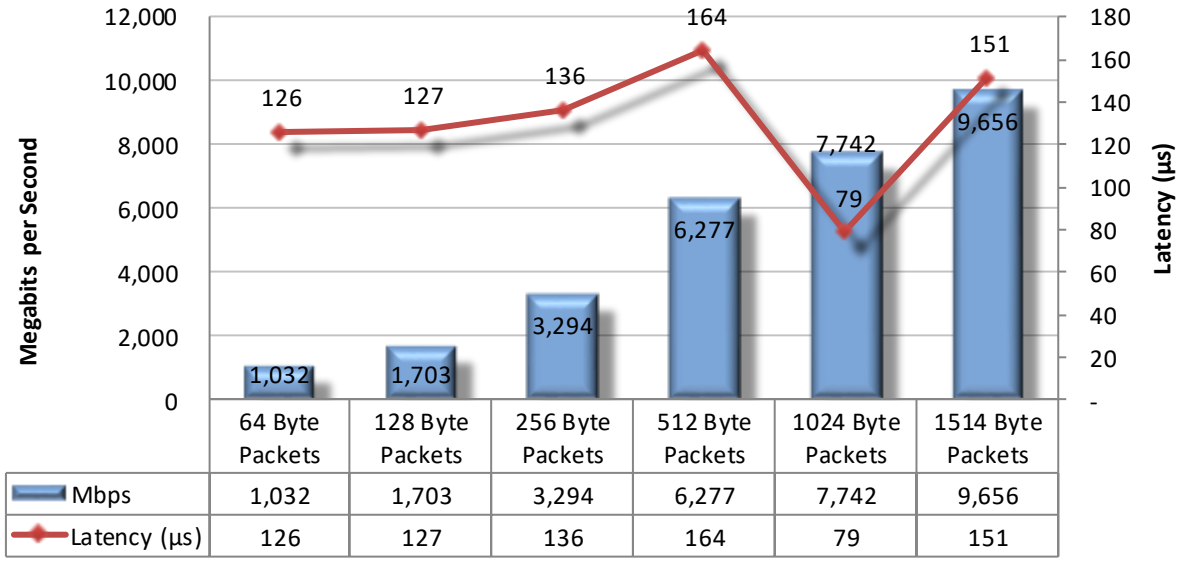


Figure 13 – Raw Packet Processing Performance (UDP Traffic)

Raw Packet Processing Performance (UDP Latency)

NGIPS that introduce high levels of latency lead to unacceptable response times for users, especially where multiple security devices are placed in the data path. Figure 14 depicts UDP latency (in microseconds) as recorded during the UDP throughput tests at 95% of maximum load.

Latency – UDP	Microseconds
64-Byte Packets	126.03
128-Byte Packets	127.01
256-Byte Packets	136.43
512-Byte Packets	164.14
1024-Byte Packets	78.69
1514-Byte Packets	151.10

Figure 14 – UDP Latency in Microseconds

Stability and Reliability

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

The device is required to remain operational and stable throughout these tests, and to block 100% of previously blocked traffic, raising an alert for each. If any non-allowed traffic passes successfully, caused either by the volume of traffic or by the device failing open for any reason, it will fail the test.

Stability and Reliability	Result
Blocking under Extended Attack	PASS
Passing Legitimate Traffic under Extended Attack	PASS
Power Fail	PASS
Redundancy[3]	NO
Persistence of Data	PASS

Figure 15 – Stability and Reliability Results

These tests also determine the behavior of the state engine under load. All NGIPS devices must choose whether to risk denying legitimate traffic or risk allowing malicious traffic once they run low on resources. A product will drop new connections when resources (such as state table memory) are low, or when traffic loads exceed its capacity. In theory, this means the NGIPS will block legitimate traffic but maintain state on existing connections (and prevent attack leakage).

[3] A device is considered to provide redundancy if it includes multiple redundant critical components, such as fans, power supplies, hard drives, etc.

Total Cost of Ownership (TCO)

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

- **Product Purchase** – The cost of acquisition
- **Product Maintenance** – The fees paid to the vendor, including software and hardware support, maintenance, and other updates
- **Installation** – The time required to take the device out of the box, configure it, put it into the network, apply updates and patches, and set up desired logging and reporting
- **Upkeep** – The time required to apply periodic updates and patches from vendors, including hardware, software, and other updates
- **Management** – Day-to-day management tasks, including device configuration, policy updates, policy deployment, alert handling, and so on

For the purposes of this report, capital expenditure (capex) items are included for a single device only (the cost of acquisition and installation).

Installation Hours

This table depicts the number of hours of labor required to install each device using only local device management options. The table accurately reflects the amount of time that NSS engineers, with the help of vendor engineers, needed to install and configure the device to the point where it operated successfully in the test harness, passed legitimate traffic, and blocked and detected prohibited or malicious traffic. This closely mimics a typical enterprise deployment scenario for a single device.

The installation cost is based on the time that an experienced security engineer would require to perform the installation tasks described above. This approach allows NSS to hold constant the talent cost and measure only the difference in time required for installation. Readers should substitute their own costs to obtain accurate TCO figures.

Product	Installation (Hours)
Fortinet FortiGate 600D v5.4.5	8

Figure 16 – Sensor Installation Time (Hours)

Total Cost of Ownership

Calculations are based on vendor-provided pricing information. Where possible, the 24/7 maintenance and support option with 24-hour replacement is utilized, since this is the option typically selected by enterprise customers. Prices are for single device management and maintenance only; costs for central management solutions (CMS) may be extra.

Product	Purchase Price	Maintenance / Year	Year 1 Cost	Year 2 Cost	Year 3 Cost	3-Year TCO
Fortinet FortiGate 600D v5.4.5	$6,000	$2,513	$9,113	$2,513	$2,513	$14,139

Figure 17 –3-Year TCO (US$)

- **Year 1 Cost** is calculated by adding installation costs (US$75 per hour fully loaded labor x installation time) + purchase price + first-year maintenance/support fees.
- **Year 2 Cost** consists only of maintenance/support fees.
- **Year 3 Cost** consists only of maintenance/support fees.

For additional TCO analysis, including for the CMS, refer to the TCO Comparative Report.

Appendix A: Product Scorecard

Security Effectiveness	
Intrusion Prevention Policies	
False Positive Testing	PASS
Coverage by Attack Vector (NSS Exploit Library)	
Attacker-Initiated	99.37%
Target-Initiated	99.52%
Combined Total	99.45%
Coverage by Impact Type	
System Exposure	Contact NSS
Service Exposure	Contact NSS
System or Service Fault	Contact NSS
Coverage by Date	Contact NSS
Coverage by Target Vendor	Contact NSS
Coverage by Result	Contact NSS
Coverage by Target Type	Contact NSS
Exploit Block Rate	**99.72%**
CAWS (Live Exploits) Block Rate	100.0%
NSS Exploit Library Block Rate	99.45%
Evasions and Attack Leakage	
Resistance to Evasion	**PASS**
IP Packet Fragmentation	**PASS**
Ordered 8 byte fragments	PASS
Ordered 16 byte fragments	PASS
Ordered 24 byte fragments	PASS
Ordered 32 byte fragments	PASS
Out of order 8 byte fragments	PASS
Ordered 8 byte fragments, duplicate last packet	PASS
Out of order 8 byte fragments, duplicate last packet	PASS
Ordered 8 byte fragments, reorder fragments in reverse	PASS
Ordered 16 byte fragments, fragment overlap (favor new)	PASS
Ordered 16 byte fragments, fragment overlap (favor old)	PASS
Out of order 8 byte fragments, interleaved duplicate packets scheduled for later delivery	PASS
Ordered 8 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload.	PASS
Ordered 16 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload.	PASS
Ordered 24 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload.	PASS
Ordered 32 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload.	PASS
TCP Stream Segmentation	**PASS**
Ordered 1 byte segments, interleaved duplicate segments with invalid TCP checksums	PASS
Ordered 1 byte segments, interleaved duplicate segments with null TCP control flags	PASS
Ordered 1 byte segments, interleaved duplicate segments with requests to resync sequence numbers mid-stream	PASS
Ordered 1 byte segments, duplicate last packet	PASS
Ordered 2 byte segments, segment overlap (favor new)	PASS
Ordered 1 byte segments, interleaved duplicate segments with out-of-window sequence numbers	PASS
Out of order 1 byte segments	PASS

Out of order 1 byte segments, interleaved duplicate segments with faked retransmits	PASS
Ordered 1 byte segments, segment overlap (favor new)	PASS
Out of order 1 byte segments, PAWS elimination (interleaved duplicate segments with older TCP timestamp options)	PASS
Ordered 16 byte segments, segment overlap (favor new (Unix))	PASS
Ordered 32 byte segments	PASS
Ordered 64 byte segments	PASS
Ordered 128 byte segments	PASS
Ordered 256 byte segments	PASS
Ordered 512 byte segments	PASS
Ordered 1024 byte segments	PASS
Ordered 2048 byte segments (sending MSRPC request with exploit)	PASS
Reverse Ordered 256 byte segments, segment overlap (favor new) with random data	PASS
Reverse Ordered 512 byte segments, segment overlap (favor new) with random data	PASS
Reverse Ordered 1024 byte segments, segment overlap (favor new) with random data	PASS
Reverse Ordered 2048 byte segments, segment overlap (favor new) with random data	PASS
Out of order 1024 byte segments, segment overlap (favor new) with random data, Initial TCP sequence number is set to 0xffffffff - 4294967295	PASS
Out of order 2048 byte segments, segment overlap (favor new) with random data, Initial TCP sequence number is set to 0xffffffff - 4294967295	PASS
RPC Fragmentation	**PASS**
One-byte fragmentation (ONC)	PASS
Two-byte fragmentation (ONC)	PASS
All fragments, including Last Fragment (LF) will be sent in one TCP segment (ONC)	PASS
All frags except Last Fragment (LF) will be sent in one TCP segment. LF will be sent in separate TCP seg (ONC)	PASS
One RPC fragment will be sent per TCP segment (ONC)	PASS
One LF split over more than one TCP segment. In this case no RPC fragmentation is performed (ONC)	PASS
Canvas Reference Implementation Level 1 (MS)	PASS
Canvas Reference Implementation Level 2 (MS)	PASS
Canvas Reference Implementation Level 3 (MS)	PASS
Canvas Reference Implementation Level 4 (MS)	PASS
Canvas Reference Implementation Level 5 (MS)	PASS
Canvas Reference Implementation Level 6 (MS)	PASS
Canvas Reference Implementation Level 7 (MS)	PASS
Canvas Reference Implementation Level 8 (MS)	PASS
Canvas Reference Implementation Level 9 (MS)	PASS
Canvas Reference Implementation Level 10 (MS)	PASS
MSRPC messages are sent in the big endian byte order, 16 MSRPC fragments are sent in the same lower layer message, MSRPC requests are fragmented to contain at most 2048 bytes of payload	PASS
MSRPC messages are sent in the big endian byte order, 32 MSRPC fragments are sent in the same lower layer message, MSRPC requests are fragmented to contain at most 2048 bytes of payload	PASS
MSRPC messages are sent in the big endian byte order, 64 MSRPC fragments are sent in the same lower layer message, MSRPC requests are fragmented to contain at most 2048 bytes of payload	PASS
MSRPC messages are sent in the big endian byte order, 128 MSRPC fragments are sent in the same lower layer message, MSRPC requests are fragmented to contain at most 2048 bytes of payload	PASS
MSRPC messages are sent in the big endian byte order, 256 MSRPC fragments are sent in the same lower layer message, MSRPC requests are fragmented to contain at most 2048 bytes of payload	PASS
MSRPC messages are sent in the big endian byte order, 512 MSRPC fragments are sent in the same lower layer message, MSRPC requests are fragmented to contain at most 2048 bytes of payload	PASS
MSRPC messages are sent in the big endian byte order, 1024 MSRPC fragments are sent in the same lower layer message, MSRPC requests are fragmented to contain at most 2048 bytes of payload	PASS
SMB & NetBIOS Evasions	**PASS**

A chaffed NetBIOS message is sent before the first actual NetBIOS message. The chaff message is an unspecified NetBIOS message with HTTP GET request like payload	PASS
A chaffed NetBIOS message is sent before the first actual NetBIOS message. The chaff message is an unspecified NetBIOS message with HTTP POST request like payload	PASS
A chaffed NetBIOS message is sent before the first actual NetBIOS message. The chaff message is an unspecified NetBIOS message with MSRPC request like payload	PASS
URL Obfuscation	**PASS**
URL encoding – Level 1 (minimal)	PASS
URL encoding – Level 2	PASS
URL encoding – Level 3	PASS
URL encoding – Level 4	PASS
URL encoding – Level 5	PASS
URL encoding – Level 6	PASS
URL encoding – Level 7	PASS
URL encoding – Level 8 (extreme)	PASS
Directory Insertion	PASS
Premature URL ending	PASS
Long URL	PASS
Fake parameter	PASS
TAB separation	PASS
Case sensitivity	PASS
Windows \ delimiter	PASS
Session splicing	PASS
HTML Obfuscation	**PASS**
UTF-16 character set encoding (little-endian)	PASS
UTF-32 character set encoding (little-endian)	PASS
Chunked encoding (random chunk size)	PASS
Chunked encoding (fixed chunk size)	PASS
Chunked encoding (chaffing)	PASS
Compression (Deflate)	PASS
Compression (Gzip)	PASS
Base-64 Encoding	PASS
Base-64 Encoding (shifting 1 bit)	PASS
Base-64 Encoding (shifting 2 bits)	PASS
Base-64 Encoding (chaffing)	PASS
FTP/Telnet Evasion	**PASS**
Inserting spaces in FTP command lines	PASS
Inserting non-text Telnet opcodes – Level 1 (minimal)	PASS
Inserting non-text Telnet opcodes – Level 2	PASS
Inserting non-text Telnet opcodes – Level 3	PASS
Inserting non-text Telnet opcodes – Level 4	PASS
Inserting non-text Telnet opcodes – Level 5	PASS
Inserting non-text Telnet opcodes – Level 6	PASS
Inserting non-text Telnet opcodes – Level 7	PASS
Inserting non-text Telnet opcodes – Level 8 (extreme)	PASS
Payload Encoding	**PASS**
x86/call4_dword_xor	PASS
x86/fnstenv_mov	PASS
x86/jmp_call_additive	PASS
Layered Evasions	
IP Fragmentation + TCP Segmentation	**PASS**

Ordered 8-byte fragments + Ordered TCP segments except that the last segment comes first	PASS
Ordered 24-byte fragments + Ordered TCP segments except that the last segment comes first	PASS
Ordered 32-byte fragments + Ordered TCP segments except that the last segment comes first	PASS
Ordered 8-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Reverse order TCP segments, segment overlap (favor new), Overlapping data is set to zero bytes	PASS
Ordered 16-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Out of order TCP segments, segment overlap (favor new), Overlapping data is set to zero bytes	PASS
Ordered 24-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Out of order TCP segments, segment overlap (favor new), Overlapping data is set to zero bytes	PASS
Ordered 32-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Out of order TCP segments, segment overlap (favor new), Overlapping data is set to zero bytes	PASS
Ordered 8-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Out of order TCP segments, segment overlap (favor new), Overlapping data is set to random alphanumeric	PASS
Ordered 16-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Out of order TCP segments, segment overlap (favor new), Overlapping data is set to random alphanumeric	PASS
Ordered 32-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Out of order TCP segments, segment overlap (favor new), Overlapping data is set to random alphanumeric	PASS
Ordered 8-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Out of order TCP segments, segment overlap (favor new), Overlapping data is set to random bytes	PASS
Ordered 16-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Out of order TCP segments, segment overlap (favor new), Overlapping data is set to random bytes	PASS
Ordered 24-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Out of order TCP segments, segment overlap (favor new), Overlapping data is set to random bytes	PASS
Ordered 32-byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has random payload + Out of order TCP segments, segment overlap (favor new), Overlapping data is set to random bytes	PASS
IP Fragmentation + MSRPC Fragmentation	**PASS**
Ordered 8 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a shuffled payload + MSRPC messages are sent in the big endian byte order with 8 MSRPC fragments sent in the same lower layer message. MSRPC requests are fragmented to contain at most 2048 bytes of payload.	PASS
Ordered 16 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a shuffled payload + MSRPC messages are sent in the big endian byte order with 16 MSRPC fragments sent in the same lower layer message. MSRPC requests are fragmented to contain at most 2048 bytes of payload.	PASS
Ordered 32 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a shuffled payload + MSRPC messages are sent in the big endian byte order with 32 MSRPC fragments sent in the same lower layer message. MSRPC requests are fragmented to contain at most 64 bytes of payload.	PASS
Ordered 64 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a shuffled payload + MSRPC messages are sent in the big endian byte order with 64 MSRPC fragments sent in the same lower layer message. MSRPC requests are fragmented to contain at most 64 bytes of payload.	PASS
Ordered 128 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a random payload + MSRPC messages are sent in the big endian byte order with 1024 MSRPC fragments sent in the same lower layer message. MSRPC requests are fragmented to contain at most 128 bytes of payload.	PASS

Ordered 256 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a random payload + MSRPC messages are sent in the big endian byte order with 1024 MSRPC fragments sent in the same lower layer message. MSRPC requests are fragmented to contain at most 256 bytes of payload.	PASS
Ordered 512 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a random payload + MSRPC messages are sent in the big endian byte order with 1024 MSRPC fragments sent in the same lower layer message. MSRPC requests are fragmented to contain at most 512 bytes of payload.	PASS
Ordered 1024 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a random payload + MSRPC messages are sent in the big endian byte order with 1024 MSRPC fragments sent in the same lower layer message. MSRPC requests are fragmented to contain at most 1024 bytes of payload.	PASS
IP Fragmentation + SMB Evasions	**PASS**
Ordered 1024 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a random payload + SMB chaff message before real messages. The chaff is a WriteAndX message with a broken write mode flag, and has random MSRPC request-like payload	PASS
Ordered 8 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a random payload + A chaffed NetBIOS message is sent before the first actual NetBIOS message. The chaff message is an unspecified NetBIOS message with MSRPC request like payload	PASS
Ordered 8 byte fragments, duplicate packet with an incrementing DWORD in the options field. The duplicate packet has a random payload + A chaffed NetBIOS message is sent before the first actual NetBIOS message. The chaff message is an unspecified NetBIOS message with HTTP GET request like payload	PASS
TCP Segmentation + SMB / NETBIOS Evasions	**PASS**
Reverse Ordered 2048 byte TCP segments, segment overlap (favor new) with random data + A chaffed NetBIOS message is sent before the first actual NetBIOS message. The chaff message is an unspecified NetBIOS message with MSRPC request like payload	PASS
TCP Split Handshake	**PASS**
HTTP Evasion	**PASS**
HTTP/0.9 response (no response headers)	PASS
Declared HTTP/0.9 response, but includes response headers; space (hex '20') after server header	PASS
HTTP/1.1 response with content-encoding header for gzip, followed by content-encoding header for deflate; no space between ':' and declaration of encoding types; served with no compression	PASS
HTTP/1.1 chunked response with chunk sizes followed by a space (hex '20'); space (hex '20') after server header	PASS
HTTP/1.1 chunked response with chunk sizes followed by a tab (hex '09'); space (hex '20') after server header	PASS
HTTP/1.1 chunked response with chunk sizes followed by an 'x' (hex '78'); space (hex '20') after server header	PASS
HTTP/1.1 chunked response with chunk sizes followed by a comma (hex '2c'); space (hex '20') after server header	PASS
HTTP/1.1 chunked response with chunk sizes followed by null character (hex '00'); space (hex '20') after server header	PASS
HTTP/1.1 chunked response with chunk sizes followed by a vertical tab (hex '0b'); space (hex '20') after server header	PASS
HTTP/1.1 chunked response with chunk sizes followed by form feed (hex '0c'); space (hex '20') after server header	PASS
HTTP/1.1 chunked response with final chunk size of '00' (hex '20 20' rather than hex '20'); space (hex '20') after server header	PASS
HTTP/1.1 chunked response with final chunk size of '0000000000000000000' (rather than '0'); space (hex '20') after server header	PASS
HTTP/1.1 response with content-encoding declaration of "gzip x"; served uncompressed; space (hex '20') after server header	PASS
HTTP/1.1 chunked response with chunk sizes followed by a space (hex '20') then an 'x' (hex '78'); space (hex '20') after server header	PASS
HTTP/1.1 response with line folded transfer-encoding header declaring chunking ('Transfer-Encoding: ' followed by CRLF (hex '0d 0a') followed by space (hex '20') followed by 'chunked' followed by CRLF (hex '0d 0a')); served without chunking; space (hex '20') after server header	PASS
HTTP/1.1 response with transfer-encoding header declaring chunking with lots of whitespace ('Transfer-Encoding: ' followed by 500 spaces (hex '20' * 500) followed by 'chunked' followed by CRLF (hex '0d 0a')); served chunked; space (hex '20') after server header	PASS

HTTP/1.0 response with status code 100 followed by message-body; no content-length header; space (hex '20') after server header	PASS
HTTP/1.0 response with status code 206 followed by message-body; no content-length header; space (hex '20') after server header	PASS
HTTP/1.0 response with status code 304 followed by message-body; no content-length header; space (hex '20') after server header	PASS
HTTP/1.0 response with status code 404 followed by message-body; no content-length header; space (hex '20') after server header	PASS
HTTP/1.0 response with status code 500 followed by message-body; no content-length header; space (hex '20') after server header	PASS
HTTP/1.0 response with status code 600 followed by message-body; no content-length header; space (hex '20') after server header	PASS
HTTP/1.0 response with status code 900 followed by message-body; no content-length header; space (hex '20') after server header	PASS
HTTP/1.0 response declaring chunking; served without chunking; space (hex '20') after server header	PASS
HTTP/1.0 response declaring chunking with content-length header; served without chunking; space (hex '20') after server header	PASS
HTTP/1.1 response with content-length header size declaration followed by space and letter A (hex '20 41'); space (hex '20') after server header; message-body followed by junk (e.g. '</html>HBGIBFJ236MJXICVNGRXKRADDPXAMVOLLCCK3KXWGBOP0TKBNKQEGS7MM0EOEHDTDZIY553OGE7WFSG8ISOYGB1B033W2S3FHX4VCL9FZ3ETCGD8LYD1A3680')	PASS
Performance	
Raw Packet Processing Performance (UDP Traffic)	**Mbps**
64-Byte Packets	1,032
128-Byte Packets	1,703
256-Byte Packets	3,294
512-Byte Packets	6,277
1024-Byte Packets	7,742
1514-Byte Packets	9,656
Latency – UDP	**Microseconds**
64-Byte Packets	126.03
128-Byte Packets	127.01
256-Byte Packets	136.43
512-Byte Packets	164.14
1024-Byte Packets	78.69
1514-Byte Packets	151.10
Maximum Capacity	
Theoretical Max. Concurrent TCP Connections	1,804,100
Theoretical Max. Concurrent TCP Connections w/Data	1,626,118
Maximum TCP Connections per Second	58,520
Maximum HTTP Connections per Second	44,970
Maximum HTTP Transactions per Second	94,130
HTTP Capacity with No Transaction Delays	**CPS**
2,500 Connections per Second – 44 KB Response	11,200
5,000 Connections per Second – 21 KB Response	17,690
10,000 Connections per Second – 10 KB Response	25,000
20,000 Connections per Second – 4.5 KB Response	29,430
40,000 Connections per Second – 1.7 KB Response	36,980
Application Average Response Time – HTTP (at 95% Max Load)	**Milliseconds**
2.500 Connections per Second – 44 KB Response	0.113
5,000 Connections per Second – 21 KB Response	0.154
10,000 Connections per Second – 10 KB Response	0.172
20,000 Connections per Second – 4.5 KB Response	0.117

40,000 Connections per Second – 1.7 KB Response	0.168
HTTP CPS & Capacity with Transaction Delays	**CPS**
21 KB Response with Delay	18,020
10 KB Response with Delay	24,980
"Real-World" Traffic	**Mbps**
"Real-World" Protocol Mix (Enterprise Perimeter)	4,130
"Real-World" Protocol Mix (Financial)	2,116
"Real-World" Protocol Mix (Education)	5,042
Stability and Reliability	
Blocking Under Extended Attack	PASS
Passing Legitimate Traffic Under Extended Attack	PASS
Power Fail	PASS
Redundancy	NO
Persistence of Data	PASS
Total Cost of Ownership	
Ease of Use	
Initial Setup (Hours)	8
Time Required for Upkeep (Hours per Year)	Contact NSS
Time Required to Tune (Hours per Year)	Contact NSS
Expected Costs	
Initial Purchase (hardware as tested)	$6,000
Installation Labor Cost (@$75/hr)	$600
Annual Cost of Maintenance & Support (hardware/software)	$1,313
Annual Cost of Updates (IPS/AV/etc.)	$1,200
Initial Purchase (centralized management system)	Contact NSS
Annual Cost of Maintenance & Support (centralized management system)	Contact NSS
Management Labor Cost (per Year @$75/hr)	Contact NSS
Tuning Labor Cost (per Year @$75/hr)	Contact NSS
Total Cost of Ownership	
Year 1	$9,113
Year 2	$2,513
Year 3	$2,513
3-Year Total Cost of Ownership	$14,139

Figure 18 – Detailed Scorecard

Test Methodology

Next Generation Intrusion Prevention System Test Methodology v3.1

A copy of the test methodology is available on the NSS Labs website at www.nsslabs.com.

Contact Information

NSS Labs, Inc.
3711 South MoPac Expressway
Building 1, Suite 400
Austin, TX 78746-8022
USA
info@nsslabs.com
www.nsslabs.com