# BREACH PREVENTION SYSTEMS TEST REPORT

**Fortinet FortiGate 500E** v6.0.3 **+ FortiClient** v6.0.3.6219 **+ FortiSandbox** v3.0.2

## AUGUST 7, 2019


## Authors – Jessica Williams, Ryan Turner, Tim Otto


## Test Methodologies

NSS Labs Breach Prevention Systems (BPS) Test Methodology v2.0

NSS Labs Evasions Test Methodology v1.2

# Fortinet FortiGate 500E v6.0.3 + FortiClient v6.0.3.6219 + FortiSandbox v3.0.2 (AWS BYOL)

**Summary**

In Q1 2019 NSS Labs performed an independent test of the **Fortinet FortiGate 500E** v6.0.3 + **FortiClient** v6.0.3.6219 + **FortiSandbox** v3.0.2 (AWS BYOL). As part of the initial test setup, we tuned devices as they would be by a customer. Every effort was made to eliminate false positives while achieving optimal security effectiveness and performance. This executive summary highlights how the product performed in key areas of testing.

**Security**

NSS security effectiveness tests verify that a solution is capable of blocking and logging threats accurately while remaining resistant to false positives.

| Block Rates | |
|---|---|
| Drive-By Exploits | 100.0% |
| Social Exploits | 100.0% |
| Malware delivered over HTTP | 100.0% |
| Malware delivered over Email (IMAP) | 98.9% |
| False Positives | 0.0% |
| **Subsequently Detected** | |
| Drive-By Exploits | 0.0% |
| Social Exploits | 0.0% |
| Malware delivered over HTTP | 0.0% |
| Malware delivered over Email (IMAP) | 0.5% |
| False Positives | NA |
| **Resistance to Evasions** | |
| Network Evasions Blocked | 103/119 |
| Binary Evasions Blocked | 96/114 |
| **Security Effectiveness** | |
| 97.8% | |

**Performance**

There is frequently a trade-off between security effectiveness and performance; a product's security effectiveness should be evaluated within the context of its performance, and vice versa.

| NSS-Tested Throughput | |
|---|---|
| 5,717 Mbps | |
| **Maximum Capacity** | |
| Max Concurrent TCP Connections | 4,275,101 |
| Max TCP Connections per Second | 78,420 |
| Maximum HTTP Connections per Second | 64,000 |
| **HTTP Capacity** | |
| 2,500 Connections per Second – 44 Kbyte HTTP Response | 10,040 CPS \| 4,016 Mbps |
| 5,000 Connections per Second – 21 Kbyte HTTP Response | 13,460 CPS \| 2,692 Mbps |

**TCO**

| Total Cost of Ownership | |
|---|---|
| 3-Year Total Cost of Ownership | $22,602 |

The product was subjected to thorough testing based on the Breach Prevention Systems (BPS) Test Methodology v2.0 (available at www.nsslabs.com). As with any NSS Labs group test, the test described herein was conducted free of charge.

# Table of Contents

Test Methodologies ........ 1
**Test Environment ........ 6**
**Security Effectiveness ........ 7**
Tuning and False Positives ........ 7
Attacks Against Users ........ 8
*Malware Delivered over Email ........ 8*
*Malware Delivered over HTTP ........ 9*
Attacks Against Computers ........ 10
*Drive-By Exploits ........ 10*
Attacks Against Users and Computers (Blended Attacks) ........ 11
*Social Exploits ........ 12*
*Offline Infection ........ 13*
Resistance to Network Evasions ........ 15
*IP Fragmentation ........ 16*
*TCP Segmentation ........ 17*
*HTTP Obfuscation ........ 18*
*HTTP Compression ........ 20*
*HTML Obfuscation ........ 21*
*Protection Resiliency ........ 23*
Resistance to Binary Evasions ........ 28
*Packers 28*
*Compressors ........ 29*
Anti-Discovery ........ 32
*Anti-Sandbox ........ 32*
*Anti-Debugger ........ 33*
*Anti-Monitor ........ 33*
*Data Exfiltration ........ 33*
**Network Device(s) Performance ........ 36**
Maximum Capacity ........ 36
HTTP Capacity ........ 37
Application Average Response Time – HTTP ........ 37
HTTP Capacity with HTTP Persistent Connections ........ 38
Single Application Flows ........ 39
**Total Cost of Ownership (TCO) ........ 40**

Calculating the Total Cost of Ownership (TCO) ........................................ 40
Installation Time ........................................ 40
3-Year Total Cost of Ownership ........................................ 41
**Appendix: Product Scorecard ........................................ 42**
**Test Environment ........................................ 55**
**Test Methodology ........................................ 56**
**Contact Information ........................................ 56**

## Table of Figures

Figure 1 – False Positive Rate ....7
Figure 2 – Malware Delivered over Email....8
Figure 3 – Malware Delivered over HTTP ....9
Figure 4 – Malware Delivered by Drive-by Exploits....10
Figure 5 – Malware Delivered by Social Exploits ....12
Figure 6 – Offline Infection (Employee Use Case) ....13
Figure 7 – Offline Infection (Contractor Use Case)....14
Figure 8 – Evasion Overview Scores ....15
Figure 9 – IP Fragmentation ....16
Figure 10 – TCP Segmentation....17
Figure 11 – HTTP Obfuscation ....19
Figure 12 – HTTP Compression....20
Figure 13 – HTML Obfuscation ....22
Figure 14 – Protection Resiliency....27
Figure 15 – Packer Evasion Results....29
Figure 16 – Compressor Evasion Results ....31
Figure 17 – Detailed Anti-Sandbox Evasion Results....32
Figure 18 – Detailed Anti-Debugger Evasion Results....33
Figure 19 – Detailed Anti-Monitor Evasion Results....33
Figure 20 – Ncat Shell and Ncat Data Exfiltration....34
Figure 21 – HTTP POST Data Exfiltration ....34
Figure 22 – SSH Shell & SSH Tunnel Data Exfiltration ....34
Figure 23 – ICMP Shell & ICMP Tunnel/HTTP POST Data Exfiltration....34
Figure 24 – DNS Tunnel & HTTP POST Data Exfiltration ....34
Figure 25 – USB Keyboard & Ncat Data Exfiltration (Windows 7) ....34
Figure 26 – USB Keyboard/ Storage & Ncat Data Exfiltration (Windows 7) ....35
Figure 27 – USB Keyboard & Ncat Data Exfiltration (Windows 10) ....35
Figure 28 – USB Keyboard/ Storage & Ncat Data Exfiltration (Windows 10) ....35
Figure 29 – Concurrency and Connection Rates....36
Figure 30 – HTTP Capacity ....37
Figure 31 – Average Application Response Time (Milliseconds) ....37
Figure 32 – HTTP Capacity with HTTP Persistent Connections....38
Figure 33 – Single Application Flows ....39
Figure 34 – Number of Users....40
Figure 35 – Installation Time (Hours) ....40
Figure 36 – 3-Year TCO (US$)....41
Figure 37 – Scorecard ....54

# Test Environment

NSS has created a unique testing infrastructure—the NSS Labs live Continuous Test—which incorporates multiple product combinations, or “stacks,” within the attack chain. Each stack consists of either an operating system alone or an operating system with additional applications installed (e.g., a browser, Java, and Adobe Acrobat). This test harness continuously captures suspicious URLs, exploits, and malicious files from threat data generated from NSS and its customers, as well as data from open-source and commercial threat feeds. Captured live attacks are then run again—this time with protection from the system under test enabled.

All live exploits and payloads in this test have been validated in our lab such that:

- a reverse shell is returned
- a bind shell is opened on the target allowing the attacker to execute arbitrary commands
- a malicious payload installed
- a system is rendered unresponsive

All live malware in this test has been validated such that they perform the malicious activity for which they were intended, for example:

- execute arbitrary commands
- call home to a C&C
- encrypt a disk
- install a keylogger
- steal credit card, social security number, or other private information
- etc.

For the purposes of the test, we define *Block Rate* as the percentage of exploits and malware blocked within 15 minutes of introduction. The *Subsequently Detected Rate* is defined as the percentage of exploits and malware detected but not blocked within 15 minutes of introduction.

For additional details on live continuous testing, please refer to the latest Security Stack (Network) Test Methodology, which can be found at www.nsslabs.com.

# Security Effectiveness

The threat landscape is evolving constantly; attackers are refining their strategies and increasing both the volume and complexity of their attacks. Enterprises now are having to defend against everyday cybercriminal attacks as well as targeted attacks and even the rare advanced persistent threats (APTs). As attacks have increased in both volume and sophistication, it has become increasingly complicated for an enterprise to monitor its entire network and endpoints for abnormalities and emerging attack patterns and to take preventative or responsive action.

For this reason, we test several types of attacks ranging from widespread day-to-day attacks and current threat actor campaigns to targeted attacks and advanced (modified, custom, evasions) attacks. In this test, we validated whether or not the breach prevention system (BPS) could protect against a wide range of threats and whether or not these solutions are providing enterprises with the protection they believe they are purchasing.

Our security effectiveness tests verify that a BPS is capable of blocking and logging threats accurately while remaining resistant to false positives. Testing leverages the deep expertise of NSS engineers who utilize multiple commercial, open-source, and proprietary tools to employ attack methods that are currently being used by cybercriminals and other threat actors. All tests in this section are completed with no background network load.

## Tuning and False Positives

We performed false-positive testing on machines running 64-bit Microsoft Windows 10 (version 1607 (Build: 14393.0) with Internet Explorer IE 11 (version 11.0.14393.0 – Update version 11.0.33). We downloaded each sample individually and subsequently executed a subset of the samples to ensure they were not blocked. Enterprise-grade solutions should produce a false positive rate of 0.0%.

| Product | False Positive Rate |
|---|---|
| **Fortinet FortiGate 500E v6.0.3** + **FortiClient** v6.0.3.6219 + **FortiSandbox** v3.0.2 | 0.0% |

**Figure 1 – False Positive Rate**

The **Fortinet FortiGate 500E** v6.0.3 + **FortiClient** v6.0.3.6219 + **FortiSandbox** v3.0.2 did not alert on any of the 555 false positive samples it was tested against.

This test includes a varied sample of legitimate application traffic that may be falsely identified as malicious (also known as false positives). As part of the initial setup, we tuned the BPS as it would be by a customer. Every effort was made to eliminate false positives while achieving optimal security effectiveness and performance, as would be the aim of a typical customer deploying the device in a live network environment. To ensure that the vendor did not deploy unrealistic (overly aggressive) security policies that blocked access to legitimate software and websites, we tested the BPS against 555 false positive samples, including but not limited to the following file formats: .exe, .jar, .xlsm, .css, .pdf, .doc, .docx, .zip, .DLL, .js, xls, .chm, .rar, .lnk, .cur, .xrc.

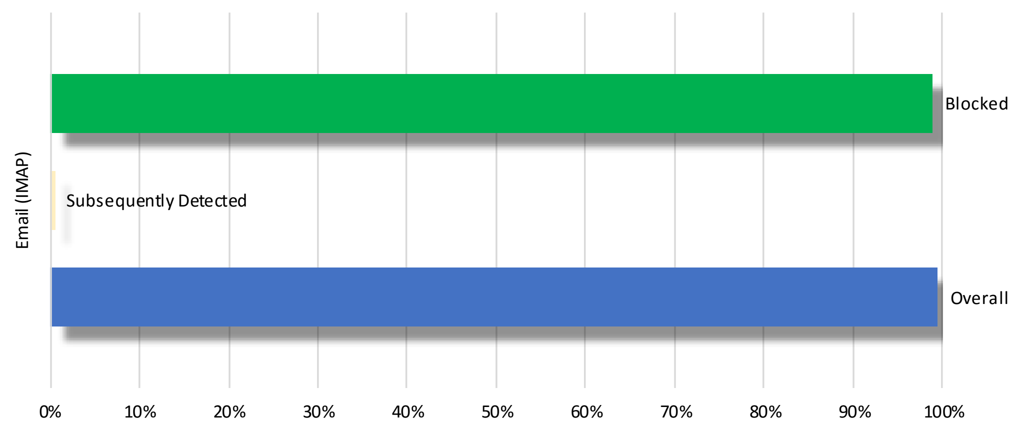
## Attacks Against Users

The "attack against a user" scenario is one in which a user is tricked into unknowingly clicking on a web link, which contains a malicious file that is subsequently downloaded and installed. These attacks are commonly referred to as socially engineered malware (SEM).

- **Via email (IMAP):** An example could be an email with the topic "Sales Commission Calculator" and containing a malicious attachment labeled "commission.exe" that an employee opens and inadvertently installs.
- **From a website/via HTTP:** An example would be where an employee is tricked into downloading and installing a malicious application named "speedy.exe" that claims it will "speed up your PC."

### Malware Delivered over Email

To test how well the BPS is able to protect against this type of attack, email was delivered to the desktop client via IMAP. A CentOS 7 Linux mail store with kernel 3.10.0-957.5.1.el7.x86_64 running Dovecot v2.2.36 for IMAP was deployed. The victim machine was running 32-bit Windows 7 (version 6.1 (Build 7601: SP1). We measured the solution's capability to block or detect malware delivered over IMAP. (The samples were executed if the download was not blocked).




| | Email (IMAP) |
|---|---|
| Blocked | 98.9% |
| Subsequently Detected | 0.5% |
| Overall | 99.5% |

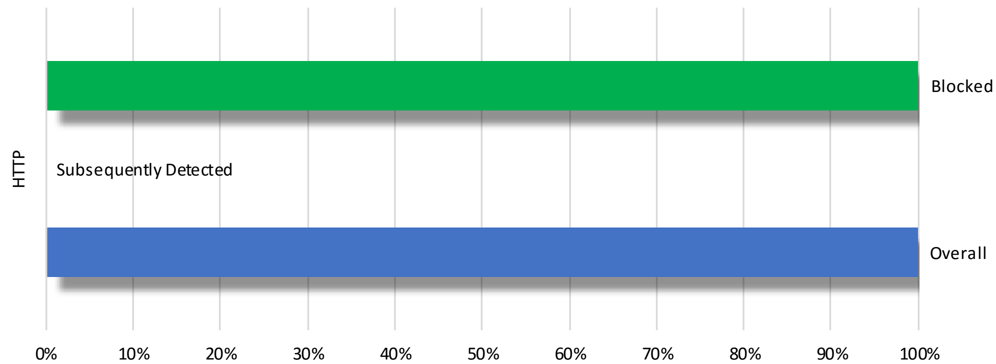
**Figure 2 – Malware Delivered over Email**

> The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v3.0.2 + **FortiClient** v6.0.3.6219 blocked 729 and subsequently detected four of the 737 samples it was tested against.

One of the most common ways in which users are compromised is through malware delivered over email. For several years, the use of social engineering has accounted for the bulk of cyberattacks against consumers and enterprises. Socially engineered malware attacks often use a dynamic combination of social media, hijacked email accounts, false notification of computer problems, and other deceptions to encourage users to download malware. One well-known social engineering attack method is spear phishing. Cybercriminals use hijacked email accounts to take advantage of the implicit trust between contacts and deceive victims into believing that the sender is trustworthy. The victim is tricked into opening the email attachment, which then launches the malicious malware program.

## Malware Delivered over HTTP

We tested the capability of the BPS to protect against malware that was downloaded over HTTP and then executed (if the download was not blocked) using 1,037 malware samples against 1,037 victim machines running 64-bit Windows 7 (version 6.1 (Build 7601: SP1) with Internet Explorer 11 (version 11.0.9600.17843 – Update version 11.0.20). Microsoft Internet Explorer's SmartScreen reputation system was disabled so that the BPS was not inadvertently credited for protection offered by the web browser.




| | HTTP |
|---|---|
| Blocked | 100.0% |
| Subsequently Detected | 0.0% |
| Overall | 100.0% |

**Figure 3 – Malware Delivered over HTTP**

One of the more widespread threats to the enterprise involves attackers using websites to deliver malware. In these web-based attacks, the user is deceived into clicking on a malicious link (on, for example, a web page or a banner advertisement) to download and execute malware. In cases where an attacker is aiming for a large number of victims, the attacker may hijack widely used reputable websites to distribute the malware. However, in cases where an attacker plans to target specific individuals, the attacker typically would use an industry-specific "watering hole" plus one or more social engineering techniques to deceive a user into unknowingly installing malware. Malware delivered via HTTP frequently employs one or more evasion techniques. Please see the Resistance to Network Evasions section for more detail.

The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v3.0.2 + **FortiClient** v6.0.3.6219 blocked all 1,037 of the samples it was tested against.

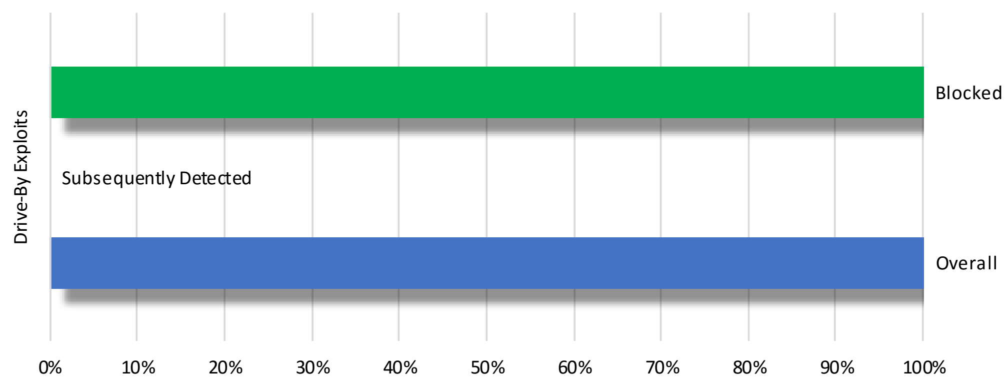
## Attacks Against Computers

While vulnerabilities are patched and defenses against exploits incorporated into new versions of operating systems (i.e., Windows), many organizations cannot easily upgrade due to financial, technical, or other constraints. NSS research has found that as of June 2019, Net Marketshare[1] reports OS market share for Windows 7 (released 10 years ago in 2009) at 38.06% and for Windows 10 (released in 2015) at 40.61%.

Research has shown that oftentimes the most valuable assets have the most stringent change control to avoid business interruption. This creates a challenging dynamic whereby the most valuable assets tend to be the most difficult to defend (e.g., older OS, unpatched, etc.). Therefore, as vulnerabilities are patched and defenses against exploits are incorporated into new versions of operating systems (i.e., Windows)—which makes exploitation of computers more difficult—the value of a BPS is often associated with its ability to protect older, unpatched, and generally more vulnerable systems.

### Drive-By Exploits

To test how well the solution was able to protect against drive-by exploits, we deployed 66 victim machines running 32-bit Windows 7 (version 6.1 (Build 7601: SP1) with Internet Explorer 9 (version 9.0.8112.16421 – Update version 9.0.26). Depending on the victim machine, one or more of the following applications was installed: Adobe Flash Player 18.0.0.160, Adobe Flash Player 18.0.0.160, Java 6 Update 27 and Microsoft Silverlight 5.1.20125, Adobe Flash Player 29.0.0.171, Adobe Reader 9.40, Adobe Reader DC 2017.012.20093, Java 6 Update 27, Java 8 Update 171, Java 8 Update 181, Microsoft Silverlight 5.1.20125. Microsoft Internet Explorer's SmartScreen reputation system was disabled so that the BPS was not inadvertently credited for protection that was offered by the web browser.




| | Drive-By Exploits |
|---|---|
| Blocked | 100.0% |
| Subsequently Detected | 0.0% |
| Overall | 100.0% |

**Figure 4 – Malware Delivered by Drive-by Exploits**

[1] Source: https://netmarketshare.com

The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v3.0.2 + **FortiClient** v6.0.3.6219 blocked all 66 of the samples it was tested against.

In a drive-by exploit, an employee visits a website containing malicious code that exploits the user's computer and installs malware without the knowledge or permission of the user. An example of this would be where an employee visits WSJ.com (Wall Street Journal), which is inadvertently hosting an advertisement that contains an exploit. A single exploit can silently deliver and install millions of malware samples to unsuspecting victim computers.

In a given year, there are hundreds of unique exploits, but there are millions (or hundreds of millions) of malware samples. An enterprise would typically see fewer unique exploits than unique malware since the exploits are reused many times over, while malware tends to be used once and then discarded by attackers.

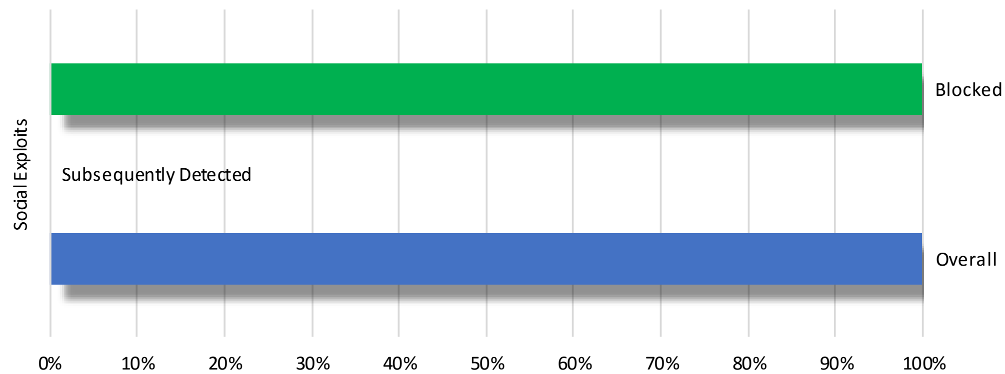
## Attacks Against Users and Computers (Blended Attacks)

These attacks combine tricking a user (social engineering) with exploiting a technical weakness in order to take control of a computer and install malware without the knowledge or permission of the user.

- **Social Exploits** – An employee is tricked into opening an email attachment containing malicious code that exploits the user's computer and installs malware without the knowledge or permission of the user. An example of this would be an email with "Your Bonus" as a subject line and containing a malicious spreadsheet labeled "bonus.xlsx" (which the employee opens).
- **Offline infections** – Computers can become infected while an employee is disconnected from the corporate network and the Internet. What happens once the infected devices are reattached to the corporate network? An example could be where an employee goes on a business trip to China where Internet traffic is tightly controlled. Access to the corporate VPN is blocked and the security software on the employee's laptop cannot receive updates or communicate in general. The employee is (from a security perspective) offline. During this time period, her laptop is infected with malware. What happens when the employee returns to the office?

## Social Exploits

Social exploits combine social engineering (manipulating people into doing what you want them to do) and exploitation (malicious code designed to take advantage of existing deficiencies in hardware or software systems, such as vulnerabilities or bugs). As with drive-by exploits, these attacks are limited to specific operating systems and/or applications. However, the exploits contained within Excel spreadsheets or Word documents may target kernel functions or common functions such as object handling, which provides attackers with a wide attack surface. As such, sending social exploits through mass email (phishing), could yield profit as the number of victims would be large, albeit smaller than in the case of malware since exploits would have technical dependencies.




| | Social Exploits |
|---|---|
| Blocked | 100.0% |
| Subsequently Detected | 0.0% |
| Overall | 100.0% |

**Figure 5 – Malware Delivered by Social Exploits**

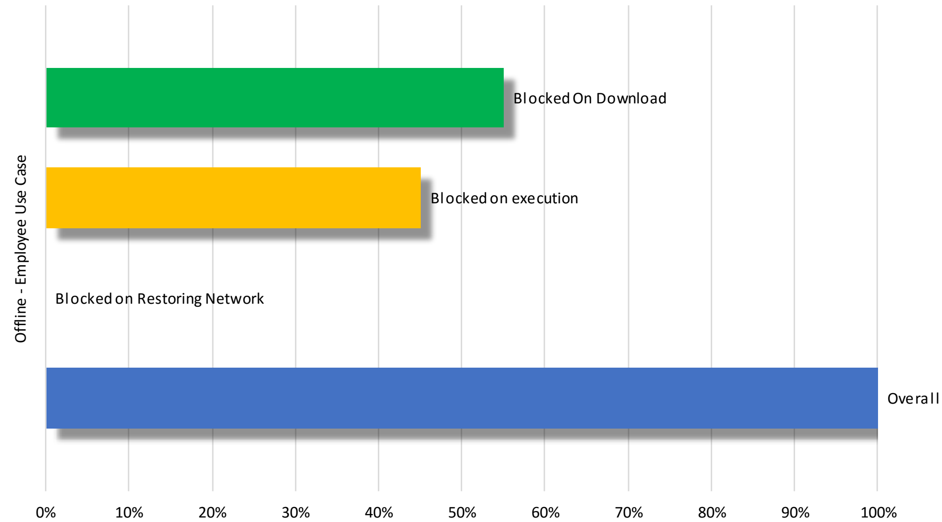
The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v3.0.2 + **FortiClient** v6.0.3.6219 blocked all six of the samples it was tested against.

To test how well the solution was able to protect against social exploits, we deployed six victim machines. Three of the machines ran combinations of 32-bit Windows 7 (version 6.1 (Build 7601: SP1) with Internet Explorer 9 (version 9.0.8112.16421 – Update version 9.0.26), Nitro Pro PDF Reader 11.0.3.173, WinRar 4.20, Adobe Reader 9.3.4. Three of the machines ran combinations of 64-bit Windows 10 (version 1607 (Build: 14393.0), Internet Explorer 11 (version 11.0.14393.0 – Update version 11.0.33) and Microsoft Office 2016 with Microsoft Word (version 1609, Build 7369.2038).

## Offline Infection

In an offline infection, a host (e.g., a laptop) is infected with malware while not connected to any networks. We tested two use cases for offline infections:

- **Employee Use Case:** In this scenario, an employee's laptop is infected with malware while the employee is traveling and outside the corporate network. In this test, the security endpoint was enabled and running but not connected to the Internet. If the endpoint did not discover the malicious sample, it was connected to the Internet. The BPS was then given 15 minutes to detect any suspicious activity and, if necessary, take action.




| | Offline - Employee Use Case |
|---|---|
| Blocked On Download | 55% |
| Blocked on execution | 45% |
| Blocked on Restoring Network | 0% |
| Overall | 100% |

**Figure 6 – Offline Infection (Employee Use Case)**

To test how well the solution was able to protect against offline infections, we deployed 20 victim machines. Six of the machines ran 64-bit Windows 7 (version 6.1 (Build 7601: SP1) and Internet Explorer 11 (version 11.0.9600.17843 – Update version 11.0.20), six of which were running 32-Bit Windows 7 (version 6.1 (Build 7601: SP1) with Internet Explorer 9 (version 9.0.8112.16421 – Update version 9.0.26). Eight of the machines ran 64-bit Windows 10 (version 1607 (Build: 14393.0) with Internet Explorer 11 (version 11.0.14393.0 – Update version 11.0.33).

The **Fortinet FortiGate 500E** v6.0.3 + **FortiClient** v6.0.3.6219 + **FortiSandbox** v6.0.3 blocked 11 samples on download and the remaining nine on execution.

- **Contractor Use Case**: In this scenario, a contractor laptop is infected with malware before the contractor connects to the corporate network. In this test, there was no security endpoint, or if there was, it was not up-to-date or properly maintained, which enabled an attacker to compromise the laptop. The laptop was then connected to the network and we observed the malware executing suspicious actions.

| Test | Result |
|---|---|
| Contractor Use Case | PASS |

**Figure 7 – Offline Infection (Contractor Use Case)**

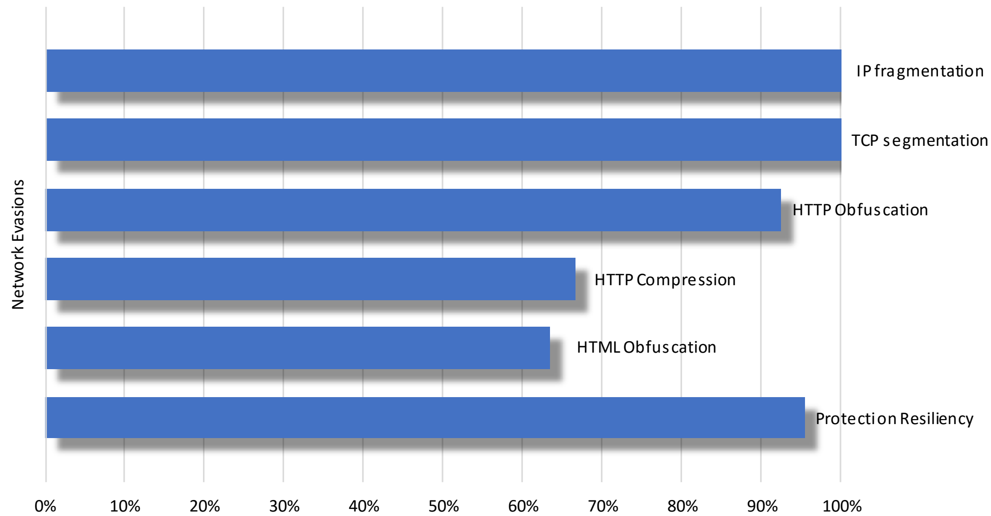
To test how well the solution was able to protect against offline infections, we deployed eight victim machines with 64-bit Windows 7 (version 6.1 (Build 7601: SP1) and Internet Explorer 11 (version 11.0.9600.17843 – Update version 11.0.20).

The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v6.0.3 passed the offline contractor use case.

## Resistance to Network Evasions

We tested the BPS against more than 190 network evasions. Each evasion used active exploits (i.e., no pcaps). If an evasion evaded the victim machine's protections, it popped a shell on the victim machine. Victim machines in the test harness did not have endpoints installed. This is because the goal of a network-based evasion is to bypass the network component of a BPS.

Successful attackers can use such evasions to attack assets such as printers, file shares, and mobile phones that do not have endpoint protection installed. This test was conducted using a custom HTTP server on Kali Linux operating system (version 2017.2 with Kernel 4.12 64-bit). The clients were running a 32-Bit Windows 7 (Enterprise Service Pack 1) and Internet Explorer 11 (version 11.0.96000.17843). During testing, we layered evasions, on occasion combining obfuscation methods. Over 190 different combinations of evasions were used and as many as 15 different layers of obfuscation.




| | Network Evasions |
|---|---|
| IP fragmentation | 100.0% |
| TCP segmentation | 100.0% |
| HTTP Obfuscation | 92.5% |
| HTTP Compression | 66.7% |
| HTML Obfuscation | 63.4% |
| Protection Resiliency | 95.5% |

**Figure 8 – Evasion Overview Scores**

Evasion techniques are a means of disguising and modifying attacks at the point of delivery to avoid detection and blocking by security solutions. 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. Many of the techniques used in this test have been widely known for years and should be considered minimum requirements.

The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v6.0.3 blocked 103 of the 119 evasions it was tested against. Script obfuscations and resiliency are included in the security effectiveness score since these types or attacks are considered "complex evasions" (HTML/JavaScript/VBScript) and require real-time code analysis in order to determine whether a function is legitimate or obfuscating an attack. For details, please see Appendix: Product Scorecard.

## IP Fragmentation

The Internet uses the Internet Protocol (IP) to transmit and route traffic from one computer to another. IP is connectionless, meaning that it transmits data to a remote host without knowing whether or not the host is ready to exchange the data. IP does not have any error detection/correction facility, and it does not guarantee the receipt of the datagrams.

There is always a possibility that a datagram will be lost or corrupted during transmission. The IP datagram is forwarded in "as-is" condition to the Transmission Control Protocol (TCP) layer at the receiving end. The TCP then has to make a request for datagrams that are either missing or contain errors.

| IP Fragmentation | Result |
|---|---|
| Small IP fragments; overlapping duplicate fragments with garbage payloads | PASS |
| Small IP fragments in reverse order | PASS |
| Small IP fragments in random order | PASS |
| Small IP fragments; delay first fragment | PASS |
| Small IP fragments in reverse order; delay last fragment | PASS |
| Small IP fragments; interleave chaff after (invalid IP options) | PASS |
| Small IP fragments in random order; interleave chaff sandwich (invalid IP options) | PASS |
| Small IP fragments in random order; interleave chaff sandwich (invalid IP options); delay random fragment | PASS |
| Small IP fragments; interleave chaff before (invalid IP options); DSCP value 16 | PASS |
| Small IP fragments in random order; interleave chaff after (invalid IP options); delay random fragment; DSCP value 34 | PASS |

**Figure 9 – IP Fragmentation**

The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v6.0.3 blocked all 10 of the samples it was tested against.

Among other capabilities, IP includes support for the fragmentation of larger packets into multiple smaller packets. When one computer uses IP to communicate with another, the instructions for how to put the fragments back together are contained within the IP Header. IP fragmentation is the process of breaking up a single IP packet into multiple packets of smaller size. *This happens all the time on networks and is in itself not an indicator of an attack.* Therefore, inline security solutions conducting deep inspection must reassemble IP fragments before inspection can occur. If the programmers developing the product made a mistake (and developers make mistakes all the time) reassembling IP packets, an attacker may be able to evade detection by fragmenting the IP packets in any number of ways, such as sending them in reverse order, delaying the first fragment, or sending overlapping duplicate fragments with garbage payload.

### TCP Segmentation

| TCP Segmentation | Result |
|---|---|
| Small TCP segments; overlapping duplicate segments with garbage payloads | PASS |
| Small TCP segments in reverse order | PASS |
| Small TCP segments in random order | PASS |
| Small TCP segments; delay first segment | PASS |
| Small TCP segments in reverse order; delay last segment | PASS |
| Small TCP segments; interleave chaff after (invalid TCP checksums); delay first segment | PASS |
| Small TCP segments in random order; interleave chaff before (invalid TCP checksums); delay random segment | PASS |
| Small TCP segments in random order; interleave chaff sandwich (out-of-window sequence numbers); TCP MSS option | PASS |
| Small TCP segments in random order; interleave chaff after (requests to resynch sequence numbers mid-stream); TCP window scale option | PASS |
| Small TCP segments in random order; interleave chaff sandwich (requests to resynch sequence numbers mid-stream); TCP window scale option; delay first segment | PASS |
| Small overlapping TCP segments | PASS |
| Small TCP segments; small IP fragments | PASS |
| Small TCP segments; small IP fragments in reverse order | PASS |
| Small TCP segments in random order; small IP fragments | PASS |
| Small TCP segments; small IP fragments in random order | PASS |
| Small TCP segments in random order; small IP fragments in reverse order | PASS |
| Small TCP segments in random order; interleave chaff sandwich (invalid TCP checksums); small IP fragments in reverse order; interleave chaff after (invalid IP options) | PASS |
| Small TCP segments; interleave chaff after (invalid TCP checksums); delay last segment; small IP fragments; interleave chaff before (invalid IP options) | PASS |
| Small TCP segments; interleave chaff sandwich (invalid TCP checksums); small IP fragments; interleave chaff sandwich (invalid IP options); delay last fragment | PASS |
| Small TCP segments in random order; interleave chaff before (out-of-window sequence numbers); TCP MSS option; small IP fragments in random order; interleave chaff before (invalid IP options); delay random fragment | PASS |
| Small TCP segments in random order; interleave chaff sandwich (requests to resynch sequence numbers mid-stream); TCP window scale option; delay first segment; small IP fragments | PASS |
| Small overlapping TCP segments; small fragments | PASS |
| Small overlapping TCP segments; delay last segment; small fragments; delay last fragment | PASS |

**Figure 10 – TCP Segmentation**

The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v6.0.3 blocked all 23 of the TCP segmentation evasions it was tested against.

TCP is one of the main protocols that run atop of the IP. Where IP is stateless, TCP is stateful, meaning that it tracks what has been sent and received via the TCP/IP. Just as IP can be fragmented, so too can TCP. When one computer uses TCP/IP to communicate with another, the instructions for how to put the TCP segments back together are contained within the TCP Header. This is common within network traffic and is not itself an indicator of an attack. Therefore, inline security solutions conducting deep inspection must reassemble TCP streams before inspection can occur. If the programmers developing the product made a mistake reassembling TCP streams, an attacker may be able to evade detection by segmenting the TCP streams in any number of ways, such as sending them in reverse order, delaying the first segment, or sending overlapping duplicate segments with garbage payload. In addition, an attacker can combine evasion techniques both segmenting TCP and fragmenting IP.

### HTTP Obfuscation

| HTTP Obfuscation | Result |
|---|---|
| Declared HTTP/0.9 response; but includes response headers; chunking declared but served without chunking | PASS |
| HTTP/1.1 chunked response with chunk sizes preceded by multiple zeros (hex '30') | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by backspace (hex '08') | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by end of text (hex '03') | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by escape (hex '1b') | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by null (hex '00') | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by a space (hex '20') then a zero (hex '30') | PASS |
| HTTP/1.1 chunked response with final chunk size of '0000000000000000000000000000000000000000000000000000000000000000000000' (rather than '0') | PASS |
| HTTP/1.1 response with line folded transfer-encoding header declaring chunking ('Transfer-Encoding: ' followed by CRLF (hex '0d 0a') followed by 'chunked' followed by CRLF (hex '0d 0a'); served without chunking | PASS |
| HTTP/1.1 response with transfer-encoding header declaring chunking with lots of whitespace ('Transfer-Encoding:' followed by 8000 spaces (hex '20' * 8000) followed by 'chunked' followed by CRLF (hex '0d 0a'); served chunked | PASS |
| HTTP/1.0 response declaring chunking; served without chunking | PASS |
| HTTP/1.0 response declaring chunking with invalid content-length header; served without chunking | PASS |
| HTTP/1.1 response with "\tTransfer-Encoding: chunked"; served chunked | PASS |
| HTTP/1.1 response with "\tTransfer-Encoding: chonked" after custom header line with "chunked" as value; served without chunking | PASS |
| HTTP/1.1 response with header with no field name and colon+junk string; followed by '\tTransfer-Encoding: chunked' header; followed by custom header; served chunked | PASS |
| HTTP/1.1 response with "\r\rTransfer-Encoding: chunked"; served chunked | PASS |
| HTTP/1.1 response with using single "\n"'s instead of "\r\n"'s; chunked | PASS |
| HTTP/1.1 response with \r\n\r\n before first header; chunked | PASS |
| HTTP/1.1 response with "SIP/2.0 200 OK\r\n" before status header; chunked | PASS |
| HTTP/1.1 response with space+junk string followed by \r\n before first header; chunked | PASS |
| HTTP/1.1 response with junk string before status header; chunked | PASS |
| HTTP/1.1 response with header end \n\014\n\n; chunked | PASS |
| HTTP/1.1 response with header end \r\n\016\r\n\r\n; chunked | PASS |

| | |
|---|---|
| HTTP/1.1 response with header end \n\r\r\n; chunked | PASS |
| HTTP/1.1 response with header end \n\017\018\n\n; chunked | PASS |
| HTTP/1.1 response with header end \n\030\n\019\n\n; chunked | PASS |
| HTTP/1.1 response with status code -203.030; with message-body; chunked | PASS |
| HTTP/1.1 response with status code 402; with message-body; chunked | PASS |
| HTTP/1.1 response with status code 403; with message-body; chunked | PASS |
| HTTP/1.1 response with status code 406; with message-body; chunked | PASS |
| HTTP/1.1 response with status code 505; with message-body; chunked | PASS |
| HTTP/1.1 chunked response with no status indicated | PASS |
| No status line; chunking indicated; served unchunked | PASS |
| HTTP/1.1 response with invalid content-length header size declaration followed by space and null (hex '20 00') | PASS |
| HTTP/1.01 declared; served chunked | PASS |
| HTTP/01.1 declared; served chunked | PASS |
| HTTP/2.B declared; served chunked | PASS |
| HTTP/9.-1 declared; served chunked | PASS |
| Double Transfer-Encoding: first empty; last chunked. Served with invalid content-length; not chunked. | PASS |
| Relevant headers padded by preceding with hundreds of random custom headers | PASS |

**Figure 11 – HTTP Obfuscation**

The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v6.0.3 blocked 37 of the 40 HTTP obfuscation evasions it was tested against.

Web browsers request content from servers over HTTP using the ASCII character-set. HTTP encoding replaces unsafe non-ASCII characters with a “%” followed by two hexadecimal digits. Web servers and clients understand how to decode the request and responses. However, this mechanism can be abused to circumvent protection that is looking to match specific strings of characters. Sample methods include chunked encoding and header folding.

Chunked encoding allows the server to break a document into smaller chunks and transmit the chunks 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 content as observed “on the wire” during transmission. In addition, the server can choose to break the document into chunks at arbitrary points. This makes it difficult to reliably identify the original HTML content from the raw data on the network.

### HTTP Compression

Per RFC 2616, the HTTP protocol allows the server to use several compression methods. These compression methods not only improve performance but, in many circumstances, they completely change the characteristic size and appearance of HTML documents.

| HTTP Compression | Result |
|---|---|
| HTTP/1.1 response compressed with gzip; invalid content-length | PASS |
| HTTP/1.1 response declaring gzip followed by junk string; invalid content-length; served uncompressed | PASS |
| HTTP/1.1 response compressed with deflate; invalid content-length | PASS |
| HTTP/1.1 response declaring deflate followed by junk string; invalid content-length; served uncompressed | PASS |
| HTTP/1.1 response with content-encoding declaration of gzip followed by space+junk string; served uncompressed and chunked | PASS |
| HTTP/1.1 response with content-encoding header for deflate; followed by content-encoding header for gzip; served uncompressed and chunked | PASS |
| HTTP/1.1 chunked response with chunk sizes preceded by multiple zeros (hex '30'); compressed with gzip | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by backspace (hex '08'); compressed with gzip | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by end of text (hex '03'); compressed with gzip | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by escape (hex '1b'); compressed with gzip | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by null (hex '00'); compressed with gzip | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by a space (hex '20') then a zero (hex '30'); compressed with gzip | PASS |
| HTTP/1.1 chunked response with chunk sizes preceded by multiple zeros (hex '30'); compressed with deflate | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by backspace (hex '08'); compressed with deflate | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by end of text (hex '03'); compressed with deflate | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by escape (hex '1b'); compressed with deflate | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by null (hex '00'); compressed with deflate | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by a space (hex '20') then a zero (hex '30'); compressed with deflate | PASS |

**Figure 12 – HTTP Compression**

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. The gzip compression method is specified in RFC 1952.

The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v6.0.3 blocked 12 of the 18 HTTP compression evasions it was tested against.

## HTML Obfuscation

HTML is a file type that a web server transmits via HTTP to a web browser, which the browser then renders for the user. So, whereas HTTP obfuscations evade detection by manipulating the transmission, HTML obfuscations are contained within the content itself.

| HTML Obfuscation[2] | Result |
|---|---|
| js-binary-obfuscation* | FAIL |
| babel-minify* | PASS |
| closure* | PASS |
| code-protect* | FAIL |
| confusion* | FAIL |
| jfogs* | FAIL |
| jfogs-reverse* | FAIL |
| jjencode* | PASS |
| jsbeautifier* | PASS |
| jsmin* | PASS |
| js-obfuscator* | FAIL |
| qzx-obfuscator* | FAIL |
| chunked and gzip compressed js-binary-obfuscation* | FAIL |
| chunked and deflate compressed js-binary-obfuscation* | FAIL |
| UTF-8 encoding | PASS |
| UTF-8 encoding with BOM | PASS |
| UTF-16 encoding with BOM | PASS |
| UTF-8 encoding; no http or html declarations | PASS |
| UTF-8 encoding with BOM; no http or html declarations | PASS |
| UTF-16 encoding with BOM; no http or html declarations | PASS |
| UTF-16-LE encoding without BOM | FAIL |
| UTF-16-BE encoding without BOM | FAIL |
| UTF-16-LE encoding without BOM; no http or html declarations | FAIL |
| UTF-16-BE encoding without BOM; no http or html declarations | FAIL |
| UTF-7 encoding | PASS |
| UTF-8 encoding | PASS |
| UTF-8 encoding | PASS |

[2] Not included in the evasion calculations

| | |
|---|---|
| EICAR string included at top of HTML | PASS |
| Hex encoded script decoded using JavaScript unescape* | PASS |
| Unicode encoded script decoded using JavaScript unescape* | FAIL |
| Hex encoded script as variable decoded using JavaScript unescape* | PASS |
| Unicode encoded script as variable decoded using JavaScript unescape* | FAIL |
| padded with <=5MB | PASS |
| padded with <=25MB | PASS |
| padded with >25MB | PASS |
| padded with <=5MB; chunked and compressed with gzip | PASS |
| padded with <=25MB; chunked and compressed with gzip | PASS |
| padded with >25MB; chunked and compressed with gzip | PASS |
| padded with <=5MB; chunked and compressed with deflate | PASS |
| padded with <=25MB; chunked and compressed with deflate | PASS |
| padded with >25MB; chunked and compressed with deflate | PASS |

**Figure 13 – HTML Obfuscation**

The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v6.0.3 blocked 26 of the 41 HTML obfuscation evasions it was tested against.

It is important that security solutions charged with protecting end systems correctly interpret HTML content and have semantic or syntactic understanding of the data they are analyzing. Otherwise, they could be vulnerable to evasions through the use of redundant, but equivalent, alternative representations of malicious content. For example, an attacker can encode HTML content using different UTF encoding. A security product that does not properly decode the content will miss the attack. This test suite uses malicious HTML content that is transferred from web server to web browser.

## Protection Resiliency

Different variations of an exploit can be used to exploit a vulnerability. And many security vendors claim their solutions provide vulnerability-based protection that will block exploitation of vulnerabilities regardless of the specific exploit. A product that is able to defend against multiple exploit variations provides resilient protection.

| Protection Resiliency[3] | Result |
|---|---|
| res-esc-001 Hex encoded VBScript decoded using JavaScript unescape* | PASS |
| res-esc-002 Hex encoded VBScript as variable decoded using JavaScript unescape* | PASS |
| res-sep-001 External VBScript file loaded from HTML* | PASS |
| res-sep-002 Multiple VBScript files loaded from HTML* | PASS |
| res-sep-003 Multiple VBScript files loaded with external JavaScript file* | PASS |
| res-nb-001 VBScript interspersed randomly with null bytes* | PASS |
| res-pay-001 nishang bind shell obfuscated with Unicorn* | PASS |
| res-pay-002 native Unicorn generated bind shell* | PASS |
| res-pay-003 nishang bind shell obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| res-pay-004 Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| res-pay-005 custom bind shell shellcode obfuscated with Invoke-Obfuscation* | PASS |
| res-pay-006 custom bind shell shellcode with password prompt obfuscated with Invoke-Obfuscation* | PASS |
| res-mth-mrg-ord-pay-spl-chr-wsp-001 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; nishang bind shell obfuscated with Unicorn* | PASS |
| res-mth-mrg-ord-pay-spl-chr-ws-pch-001 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; nishang bind shell obfuscated with Unicorn; chunked* | PASS |
| res-mth-mrg-ord-pay-spl-chr-wsp-002 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; native Unicorn generated bind shell* | PASS |
| res-mth-mrg-ord-pay-spl-chr-wsp-cg-002 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode | PASS |

[3] Not included in the evasion calculations

| | |
|---|---|
| function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; native Unicorn generated bind shell; chunked and gzip compressed* | |
| res-mth-mrg-ord-pay-spl-chr-wsp-003 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; nishang bind shell obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| res-mth-mrg-ord-pay-spl-chr-wsp-cd-003 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H using online vbscript obfuscator; both spaces and linefeeds replaced with multiples of each; nishang bind shell obfuscated with PowerSploit's Out-EncodedCommand; chunked and deflate compressed* | PASS |
| res-mth-mrg-ord-pay-spl-chr-wsp-004 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| res-mth-mrg-ord-pay-spl-chr-wsp-ch-004 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand; chunked* | PASS |
| res-mth-mrg-ord-pay-spl-chr-wsp-005 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; custom bind shell shellcode obfuscated with Invoke-Obfuscation* | PASS |
| res-mth-mrg-ord-pay-spl-chr-wsp-cg-005 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; custom bind shell shellcode obfuscated with Invoke-Obfuscation; chunked and gzip compressed* | PASS |
| res-mth-mrg-ord-pay-splc-hrw-sp-006 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; custom bind shell shellcode with password prompt obfuscated with Invoke-Obfuscation* | PASS |
| res-mth-mrg-ord-pay-spl-chr-wsp-cd-006 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function | PASS |

| | |
|---|---|
| function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; custom bind shell shellcode with password prompt obfuscated with Invoke-Obfuscation; chunked and deflate compressed* | |
| res-ren-chr-wsp-pay-mth-spl-001 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; nishang bind shell obfuscated with Unicorn* | PASS |
| res-ren-chr-wsp-pay-mth-spl-ch-001 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; nishang bind shell obfuscated with Unicorn; chunked* | PASS |
| res-ren-chr-wsp-pay-mth-spl-002 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; native Unicorn generated bind shell* | PASS |
| res-ren-chr-wsp-pay-mth-spl-cg-002 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; native Unicorn generated bind shell; chunked and gzip compressed* | PASS |
| res-ren-chr-wsp-pay-mth-spl-003 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; nishang bind shell obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| res-ren-chr-wsp-pay-mth-spl-cd-003 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; nishang bind shell obfuscated with PowerSploit's Out-EncodedCommand; chunked and deflate compressed* | PASS |
| res-ren-chr-wsp-pay-mth-spl-004 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| res-ren-chr-wsp-pay-mth-spl-ch-004 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand; chunked* | PASS |
| res-ren-chr-wsp-pay-mth-spl-005 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; custom bind shell shellcode obfuscated with Invoke-Obfuscation* | PASS |
| res-ren-chr-wsp-pay-mth-spl-cg-005 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; custom bind shell shellcode obfuscated with Invoke-Obfuscation; chunked and gzip compressed* | PASS |

| | |
|---|---|
| res-ren-chr-wsp-pay-mth-spl-006 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; custom bind shell shellcode with password prompt obfuscated with Invoke-Obfuscation* | PASS |
| res-ren-chr-wsp-pay-mth-spl-cd-006 procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; custom bind shell shellcode with password prompt obfuscated with Invoke-Obfuscation; chunked and deflate compressed* | PASS |
| res-wsp-001 both spaces and linefeeds replaced with multiples of each* | PASS |
| res-ren-001 procedures and variables renamed* | PASS |
| res-mth-001 numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values* | PASS |
| res-chr-001 change all chr() to chrw() and vice versa where possible* | PASS |
| res-chr-002 change chr() and chrw() to chrb()* | PASS |
| res-chr-003 some script commands/strings converted to series of chr()/Clng/&H using online vbscript obfuscator* | PASS |
| res-pay-007 Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| res-pay-008 Use wscript to call original payload (PoshRat method) * | PASS |
| res-pay-009 nishang bind shell obfuscated with Unicorn* | PASS |
| res-ord-001 Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script* | PASS |
| res-spl-001 Some strings split with "+" and "&"; some lines split with "_"* | PASS |
| res-mrg-001 combine 'myarray' instantiation into single line; combine PowerShell command into single line* | PASS |
| res-ren-chr-001 Combination of techniques used in res-ren-001 and res-chr-003* | PASS |
| res-ren-chr-wsp-001 Combination of techniques used in res-ren-001; res-chr-003; and res-wsp-001* | PASS |
| res-ren-chr-wsp-pay-001 Combination of techniques used in res-ren-001; res-chr-003; res-wsp-001; and res-pay-004* | PASS |
| res-ren-pay-001 Combination of techniques used in res-ren-001 and res-pay-007* | PASS |
| res-ren-chr-wsp-pay-mth-001 Combination of techniques used in res-ren-001; res-chr-003; res-wsp-001; res-pay-007; and res-mth-001* | PASS |
| res-mth-mrg-001 Combination of techniques used in res-mth-001 and res-mrg-001* | PASS |
| res-mth-mrg-ord-001 Combination of techniques used in res-mth-001; res-mrg-001; and res-ord-001* | PASS |
| res-mth-mrg-ord-pay-001 Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; and res-pay-008* | PASS |
| res-mth-mrg-ord-pay-spl-001 Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-008; and res-spl-001* | PASS |
| res-mth-mrg-ord-pay-spl-chr-001 Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-008; res-spl-001; and res-chr-003* | PASS |
| res-mth-mrg-ord-pay-spl-chr-002 Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-008; res-spl-001; and res-chr-003; plus removal of all CLng's* | PASS |
| res-mth-mrg-ord-pay-chr-001 Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-008; and res-chr-003* | PASS |

| | |
|---|---|
| res-mth-mrg-ord-pay-spl-chr-wsp-007 Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-008; res-spl-001; res-chr-003; res-wsp-001; plus removal of all CLng's* | PASS |
| res-mth-mrg-ord-pay-spl-chr-wsp-008 Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-009; res-spl-001; res-chr-003; res-wsp-001; res-ren-001; plus removal of all CLng's; replace 'LANGUAGE="VBScript"' with 'type="text/vbScript"'* | PASS |
| combo-001 UTF-8 encoding; HTTP/1.1 chunked response with chunk sizes preceded by multiple zeros (hex '30'); small TCP segments; small IP fragments; padding | PASS |
| combo-002 UTF-8 encoding with BOM; HTTP/1.1 chunked response with chunk sizes followed by backspace (hex '08'); small TCP segments; small IP fragments in reverse order; padding | FAIL |
| combo-003 UTF-16 encoding with BOM; HTTP/1.1 chunked response with chunk sizes followed by end of text (hex '03'); small TCP segments in random order; small IP fragments; padding | FAIL |
| combo-004 UTF-8 encoding; no http or html declarations; HTTP/1.1 chunked response with chunk sizes followed by escape (hex '1b'); small TCP segments; small IP fragments in random order; padding | FAIL |
| combo-005 UTF-8 encoding with BOM; no http or html declarations; HTTP/1.1 chunked response with chunk sizes followed by null (hex '00'); small TCP segments in random order; small IP fragments in reverse order; padding | PASS |

**Figure 14 – Protection Resiliency**

The **Fortinet FortiGate 500E** v6.0.3 + **FortiSandbox** v6.0.3 blocked 64 of the 67 resiliency samples it was tested against.

To confirm the baseline use case, we exploited a known vulnerability leveraging a known exploit. Next, we introduced the security product and attempted to exploit the same vulnerability using the same exploit. The expected behavior was confirmed, namely that the exploit was blocked by the BPS. Finally, the experimental use case was run, during which we attempted to exploit the same vulnerability using previously unseen variations of that exploit. A security product with resilient protection will be able to block different variations of the same exploit.

## Resistance to Binary Evasions

Cybercriminals deploy evasions to disguise and modify attacks at the point of delivery in order to avoid detection by security solutions. Given that BPS employ a number of network and endpoint protection technologies, bypassing any one of its components means that an attacker has likely bypassed all defenses. Therefore, it is imperative that all of the BPS components can correctly handle evasions. Attackers can modify attacks and malicious code in a number of ways order to evade detection.

### Packers

Packers are primarily used to obfuscate and "protect" compiled binaries. Along with the compressed/obfuscated data (the original binary in obfuscated form), they contain a "stub," which, upon execution, de-obfuscates the binary and jumps to its restored entry point. Malware authors typically use packing techniques to obfuscate binaries so they cannot be easily analyzed. We tested the BPS' capability to protect against binary evasions using 26 victim machines running 64-bit Windows 7 (version 6.1 (Build 7601: SP1) with Internet Explorer 11 (version 11.0.9600.17843 – Update version 11.0.20).

| Packers | Sample Name | Action on Download | Action on Execution |
|---|---|---|---|
| Anskaya | DNA_Sample-C.exe | Blocked | |
| Anskaya | DNA_Sample-E.exe | Blocked | |
| Anskaya | DNA_Sample-F.exe | Blocked | |
| Exestealth | Sample-C.exe | Blocked | |
| Krypton | Sample-C.exe | Blocked | |
| Winkypt | Sample-C.exe | Blocked | |
| Winupack | Sample-A.exe | Blocked | |
| excalibur | Sample-C.exe | Blocked | |
| exefog | Sample-C.exe | Blocked | |
| fearzpacker | Sample-A.exe | Blocked | |
| fearzpacker | Sample-E.exe | Blocked | |
| fishPE | Sample-C.exe | Blocked | |
| fishPE | Sample-F.exe | Blocked | |
| hidepx | Sample-B.exe | Blocked | |
| hidepx | Sample-F.exe | Blocked | |
| kkrunchy | Sample-A.exe | Blocked | |
| kkrunchy | Sample-C.exe | Blocked | |
| mew | Sample-F.exe | Blocked | |
| petite24 | Sample-A.exe | Blocked | |
| petite24 | Sample-C.exe | Blocked | |
| telock | Sample-C.exe | Blocked | |
| upx | Sample-A.exe | Blocked | |

| | | | |
|---|---|---|---|
| upx | Sample-B.exe | Blocked | |
| yc | Sample-C.exe | Blocked | |
| yc | Sample-D.exe | Blocked | |
| yc | Sample-F.exe | Blocked | |

**Figure 15 – Packer Evasion Results**

To establish the baseline use case, we downloaded and executed well-known malware to confirm it functioned as expected in the test harness. Next, we introduced, the security product and attempted to attempted to download and execute the known malware samples. The expected behavior was confirmed and recorded, namely that the malware was blocked or detected by the security product. Finally, the experimental use case was run, during which we attempted to download and execute the same known malware samples they had packed using commercially available packers. The results were recorded.

The **Fortinet FortiGate 500E** v6.0.3 + **FortiClient** v6.0.3.6219 + **FortiSandbox** v6.0.3 blocked all 26 of the packers it was tested against.

### Compressors

Compressors are primarily used to reduce the size of a file. They are also used by attackers to obfuscate malware since compressed files do not look the same to pattern-matching engines. As such, a security product must have the matching compression algorithm in order to detect malware that has been compressed. Malware authors typically use compression techniques to obfuscate binaries so they cannot be easily analyzed.

We tested the capability of the BPS to protect against binary evasions using 64 victim machines running 64-bit Windows 7 (version 6.1 (Build 7601: SP1) with Internet Explorer 11 (version 11.0.9600.17843 – Update version 11.0.20).

*Note:* Failure to detect compressed malware is a potential security risk since it enables attackers to move content laterally or exfiltrate data.

| Compressors | Sample Name | Action on Download | Action on Manual Scan |
|---|---|---|---|
| 7zip\7zip\ | Sample-A | Blocked | |
| 7zip\7zip\ | Sample-B | Blocked | |
| 7zip\7zip\ | Sample-C | Blocked | |
| 7zip\7zip\ | Sample-D | Blocked | |
| 7zip\7zip\ | Sample-E | Blocked | |
| 7zip\7zip\ | Sample-F | Blocked | |
| 7zip\bzip2\ | Sample-A | Blocked | |
| 7zip\bzip2\ | Sample-B | Blocked | |
| 7zip\bzip2\ | Sample-C | Blocked | |
| 7zip\bzip2\ | Sample-D | Blocked | |
| 7zip\bzip2\ | Sample-E | Blocked | |
| 7zip\bzip2\ | Sample-F | Blocked | |
| 7zip\gzip\ | Sample-A | Blocked | |

| | | | |
|---|---|---|---|
| 7zip\gzip\ | Sample-B | Blocked | |
| 7zip\gzip\ | Sample-C | Blocked | |
| 7zip\gzip\ | Sample-D | Blocked | |
| 7zip\gzip\ | Sample-E | Blocked | |
| 7zip\gzip\ | Sample-F | Blocked | |
| 7zip\xz\ | Sample-A | Blocked | |
| 7zip\xz\ | Sample-B | Blocked | |
| 7zip\xz\ | Sample-C | Blocked | |
| 7zip\xz\ | Sample-D | Blocked | |
| 7zip\xz\ | Sample-E | Blocked | |
| 7zip\xz\ | Sample-F | Blocked | |
| ALZIP\ | Sample-A | | Missed |
| ALZIP\ | Sample-B | | Missed |
| ALZIP\ | Sample-C | | Missed |
| ALZIP\ | Sample-D | | Missed |
| ALZIP\ | Sample-E | | Missed |
| ALZIP | Sample-F | | Missed |
| AshampooZip | Sample-A | Blocked | |
| AshampooZip\ | Sample-B | Blocked | |
| AverZip\ | Sample-B | Blocked | |
| AverZip\ | Sample-C | Blocked | |
| Bandizip\ | Sample-C | Blocked | |
| Bandizip\ | Sample-D | Blocked | |
| FilZip\ | Sample-D | Blocked | |
| FilZip\ | Sample-E | Blocked | |
| KuaiZip\ | Sample-A | | Missed |
| KuaiZip\ | Sample-B | | Missed |
| KuaiZip\ | Sample-C | | Missed |
| KuaiZip\ | Sample-D | | Missed |
| KuaiZip\ | Sample-E | | Missed |
| KuaiZip\ | Sample-F | | Missed |
| MuZip\ | Sample-E | Blocked | |
| MuZip\ | Sample-F | Blocked | |
| PicoZip\ | Sample-A | Blocked | |
| PicoZip\ | Sample-B | Blocked | |
| PowerArchiver\ | Sample-A | Blocked | |

| | | | |
|---|---|---|---|
| PowerArchiver\ | Sample-B | Blocked | |
| PowerArchiver\ | Sample-C | Blocked | |
| PowerArchiver\ | Sample-D | Blocked | |
| PowerArchiver\ | Sample-E | Blocked | |
| PowerArchiver\ | Sample-F | Blocked | |
| QuickZip\ | Sample-C | Blocked | |
| QuickZip\ | Sample-D | Blocked | |
| SimplyZip\ | Sample-A | | Missed |
| SimplyZip\ | Sample-B | | Missed |
| SimplyZip\ | Sample-C | | Missed |
| SimplyZip\ | Sample-D | | Missed |
| SimplyZip\ | Sample-E | | Missed |
| SimplyZip\ | Sample-F | | Missed |
| ZipitFast\ | Sample-E | Blocked | |
| ZipitFast\ | Sample-F | Blocked | |

**Figure 16 – Compressor Evasion Results**

The **Fortinet FortiGate 500E** v6.0.3 + **FortiClient** v6.0.3.6219 + **FortiSandbox** v6.0.3 blocked 46 compressors on download. In total, 64 compressor samples were tested.

To establish the baseline use case, we downloaded and executed well-known malware to confirm it functioned as expected in the test harness. Next, we introduced the security product and attempted to attempted to download and execute the known malware samples. The expected behavior was confirmed and recorded, namely that the malware was blocked or detected by the security product. Finally, the experimental use case was run, during which we attempted to download the same known malware samples that were compressed using commercially available compressors. The results were recorded. If the download was successful, a manual scan was attempted. Because the baseline sample was blocked or detected, we did not attempt to execute the samples. It is expected that upon extraction the newly uncompressed sample would have been blocked as well.

## Anti-Discovery

For anti-discovery evasions, the malware used several techniques to determine whether or not it was on a user's machine; whether or not a security product was present; whether or not debugging or sandboxing was occurring; etc. If the sample discovered one or more of these scenarios to be present, it hid or remained dormant until the right conditions were met, e.g., user input from a keyboard and/or mouse.

- Samples in the Anti-Sandbox tests utilized varied techniques to determine whether a sandbox environment was present. If no sandbox was present, the malicious routine was executed.
- Samples in the Anti-Debugger tests utilized varied techniques to determine whether debuggers were present in an environment. A malicious routine was executed only if no debuggers were detected.
- Samples in the Anti-Monitor tests utilized varied techniques to determine whether monitoring mechanisms were present in an environment. A malicious routine was executed only if no monitoring was detected.

### Anti-Sandbox

| Anti-Sandbox | Sample Checks/Actions/Tasks | Result |
|---|---|---|
| Argcount | Detects Arguments passed to executables | Blocked |
| Argvalue | Check for the value of argument passed to executables | Blocked |
| Changingwindow | "Real behavior" like a user changing windows | Blocked |
| Checkpath | Common path used in sandbox/testing environment | Blocked |
| Checkselfname | Self-name (some sandbox/testing renames the target file) | Blocked |
| Displaymessagebox | If message box will be clicked properly by a person | Blocked |
| Enumwindows | Enumerate current window classes if name contains sandbox/testing strings | Blocked |
| Enumwindowstitle | Enumerate current window titles if name contains sandbox/testing strings | Blocked |
| Loadeddll | Detects loaded dlls commonly used by sandbox | Blocked |
| Namedpipes | Known named pipes used by sandbox/VM | Blocked |
| Nettraffic | Presence of internet traffic | Blocked |
| Parentexplorer | Parent process if explorer.exe | Blocked |
| Sandboxprocess | Parent process is a known sandbox process | Blocked |

**Figure 17 – Detailed Anti-Sandbox Evasion Results**

### Anti-Debugger

| Anti-Debugger | Sample Checks/Actions/Tasks | Result |
|---|---|---|
| Closehandle | Debugger will exit if argument in Close Handle is invalid | Blocked |
| Debuggerpresent | Is Debugger Present API | Blocked |
| Hardwarebp | Check memory context if hardware breakpoint is set | Blocked |
| Pageguard | Checks for STATUS_GUARD_PAGE_VIOLATION upon triggering a memory jump | Blocked |
| Remotedebugger | Check Remote Debugger Present API | Blocked |

**Figure 18 – Detailed Anti-Debugger Evasion Results**

### Anti-Monitor

| Anti-Monitor | Sample Checks/Actions/Tasks | Result |
|---|---|---|
| Acceleratedsleep | Check if time is being accelerated via APIs | Blocked |
| Analysistools | Check for presence of certain analysis tools in process list | Blocked |
| Mousecursor | Check for movement of mouse | Blocked |
| Ntdelay | Check for time delays using NtDelayExecution | Blocked |
| Sleeploop | Perform loop in a Sleep API to determine minute changes in time | Blocked |
| Sleeppatched | Is Sleep API patched | Blocked |

**Figure 19 – Detailed Anti-Monitor Evasion Results**

### Data Exfiltration

Breaches may occur as a result of a malicious insider or as a result of an outside attacker gaining physical access to a system. Commercial hardware implants are easily available to enable persistence, load malware, offload documents, and backdoor systems.

To test the BPS' capability to detect exfiltration of sensitive data, multiple data points were used, including but not limited to: title, first name, last name, city, street name, zip code, country, email, username, telephone number, birthday, age, credit card type, credit card number, passport number, favorite color, occupation, name of employer, blood type, weight, password, and several types of hashes for passwords. The following file types were used to store sensitive data: .txt, .sqlite, .csv, .htm, .xls, .xlsx.

Prior to testing, we installed a USB mouse, a USB flash/thumb drive, and a Yubikey U2F device and then confirmed each device was functioning properly. This ensured vendors were not simply disabling access via USB without allowing authorized devices to use USB access. Subsequently, keystroke injection attacks, sideloading of malware, main-in-the-middle attacks, and other techniques based on physical access to the system/premises were tested using COTS hardware implants and built-in OS functionality ("living off the land").

We deployed one Arch Linux server running Linux 4.17.5-1-ARCH x86_64 and nine victim machines: five running Arch Linux 4.17.5-1-ARCH x86_64; two running 64-Bit Windows 7 (version 6.1 (Build 7601: SP1); and two running 64-Bit Windows 10 (version 1709 (Build: 16299.125). These tests were conducted using various command shell tools and Bash Bunny (which was configured to emulate USB Keyboard and USB Keyboard + USB Storage).

Figure 20 through Figure 24 depict exfiltration results for the Fortinet FortiGate 500E v6.0.3 + FortiSandbox v6.0.3.

The first test utilized Ncat to open a shell on the remote server. If this attempt was successful, data exfiltration was attempted

| Test | Data Exfiltration |
|---|---|
| Ncat – Data exfiltration | Missed |

**Figure 20 – Ncat Shell and Ncat Data Exfiltration**

The second test utilized HTTP POST to exfiltrate data.

| Test | Result |
|---|---|
| HTTP POST | Missed |

**Figure 21 – HTTP POST Data Exfiltration**

The third test attempted to open a shell on the remote server using SSH. SSH tunnel was used for data exfiltration.

| Test | Result |
|---|---|
| SSH tunnel | Missed |

**Figure 22 – SSH Shell & SSH Tunnel Data Exfiltration**

The fourth test attempted to open a shell on the remote server using an ICMP tunnel. HTTP POST was used for data exfiltration.

| Test | Result |
|---|---|
| ICMP tunnel w/HTTP POST | Blocked |

**Figure 23 – ICMP Shell & ICMP Tunnel/HTTP POST Data Exfiltration**

The fifth method involved the use of a DNS Tunnel to the remote server. HTTP POST was used for data exfiltration.

| Test | Result |
|---|---|
| DNS tunnel w/HTTP POST | Blocked |

**Figure 24 – DNS Tunnel & HTTP POST Data Exfiltration**

Figure 25 through Figure 28 depict exfiltration results for the FortiClient v6.0.3.6219.

The sixth method involved the use of a USB (Bash Bunny) to emulate a USB Keyboard. Ncat was used for data exfiltration.

| Test | Execute Command | Computer Information | Copy Documents | Dump Registry | Screenshot of Desktop | Data Exfiltration |
|---|---|---|---|---|---|---|
| Ncat | Missed | Blocked | Blocked | Blocked | Blocked | Blocked |

**Figure 25 – USB Keyboard & Ncat Data Exfiltration (Windows 7)**

The seventh method involved the use of a USB (Bash Bunny) to emulate a USB Keyboard and storage. USB storage and Ncat was used for data exfiltration.

| Test | Execute Command | Computer Information | Copy Documents | Dump Registry | Screenshot of Desktop | Data Exfiltration |
|---|---|---|---|---|---|---|
| USB/Ncat | Blocked | Blocked | Blocked | Blocked | Blocked | Blocked |

**Figure 26 – USB Keyboard/ Storage & Ncat Data Exfiltration (Windows 7)**

The eighth method involved the use of a USB (Bash Bunny) to emulate a USB Keyboard. Ncat was used for data exfiltration.

| Test | Execute Command | Computer Information | Copy Documents | Dump Registry | Screenshot of Desktop | Windows Vault Credentials | Data Exfiltration |
|---|---|---|---|---|---|---|---|
| Ncat | Missed | Missed | Missed | Missed | Missed | Missed | Missed |

**Figure 27 – USB Keyboard & Ncat Data Exfiltration (Windows 10)**

The ninth method involved the use of a USB (Bash Bunny) to emulate a USB Keyboard and storage. USB storage and Ncat was used for data exfiltration.

| Test | Execute Command | Computer Information | Copy Documents | Dump Registry | Screenshot of Desktop | Windows Vault Credentials | Data Exfiltration |
|---|---|---|---|---|---|---|---|
| USB /Ncat | Blocked | Blocked | Blocked | Blocked | Blocked | Blocked | Blocked |

**Figure 28 – USB Keyboard/ Storage & Ncat Data Exfiltration (Windows 10)**

# Network Device(s) Performance

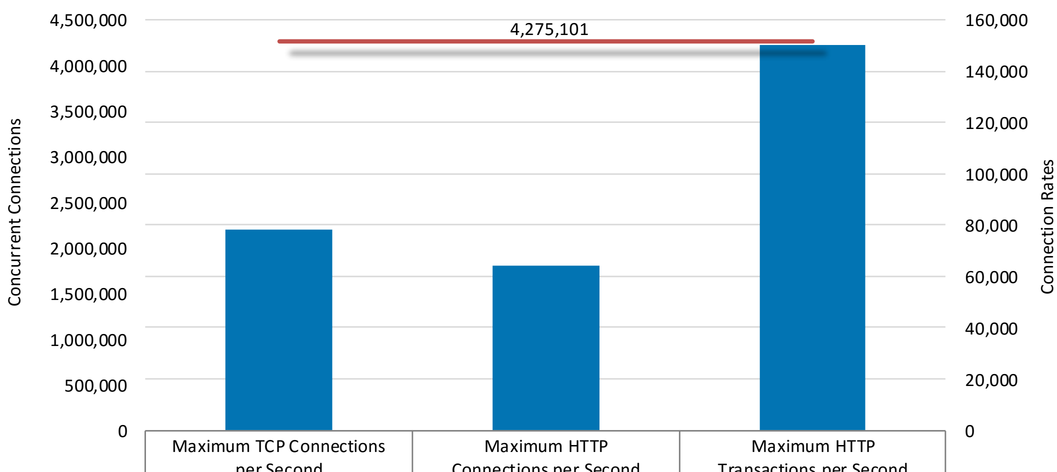
There is frequently a trade-off between security effectiveness and performance; a product's security effectiveness should be evaluated within the context of its performance, and vice versa. Ixia BreakingPoint PS-1 (Software version 8.40.16.19) was used to test performance.

## 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 was to stress the inspection engine and determine how it coped with high volumes of TCP connections per second, application layer transactions per second, and concurrent open connections. All packets contained valid payload and address data, and these tests provided 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—were used:

- **Excessive concurrent TCP connections** – Latency within the BPS is causing an unacceptable increase in open connections.
- **Excessive concurrent HTTP connections** – Latency within the BPS 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 BPS is causing connections to time out.




| | Maximum TCP Connections per Second | Maximum HTTP Connections per Second | Maximum HTTP Transactions per Second |
|---|---|---|---|
| MAX CPS & TPS | 78,420 | 64,000 | 150,200 |
| Max Concurrent Connections | 4,275,101 | 4,275,101 | 4,275,101 |

**Figure 29 – Concurrency and Connection Rates**

## HTTP Capacity

These tests stressed the HTTP detection engine and determined how the system coped with network loads of varying average packet size and varying connections per second. By creating genuine session-based traffic with varying session lengths, the BPS was 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 can be achieved in a lab environment, while also ensuring absolute accuracy and repeatability.

Each transaction consisted of a single HTTP GET request with no transaction delays (that is, the web server responds immediately to all requests). All packets contained valid payload (a mix of binary and ASCII objects) and address data. The test provides an excellent representation of a live network (albeit one biased toward HTTP traffic) at various network loads.

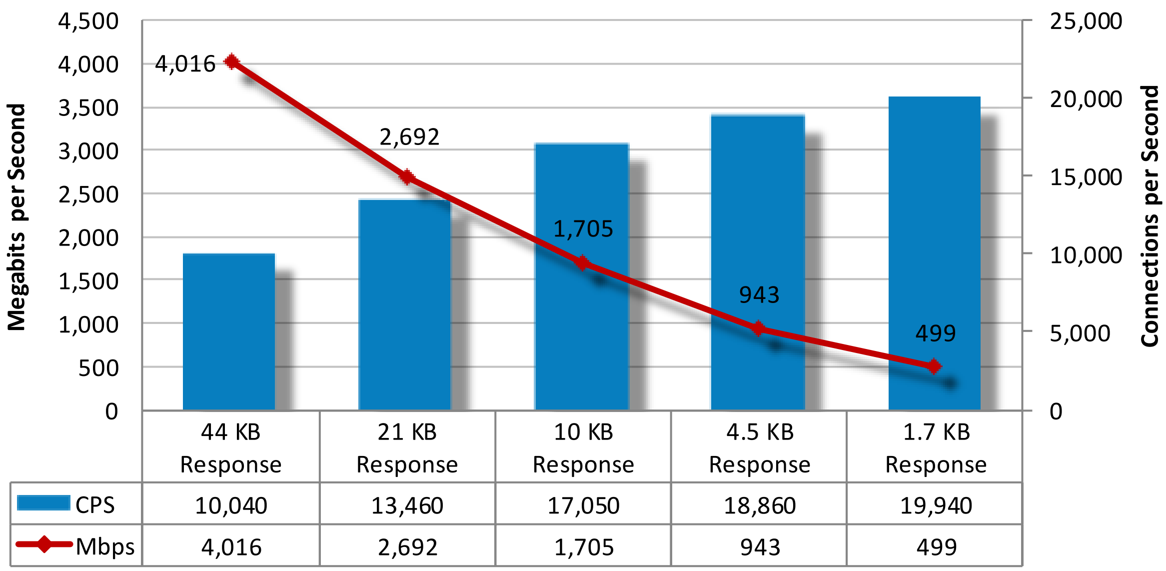



**Figure 30 – HTTP Capacity**

## Application Average Response Time – HTTP

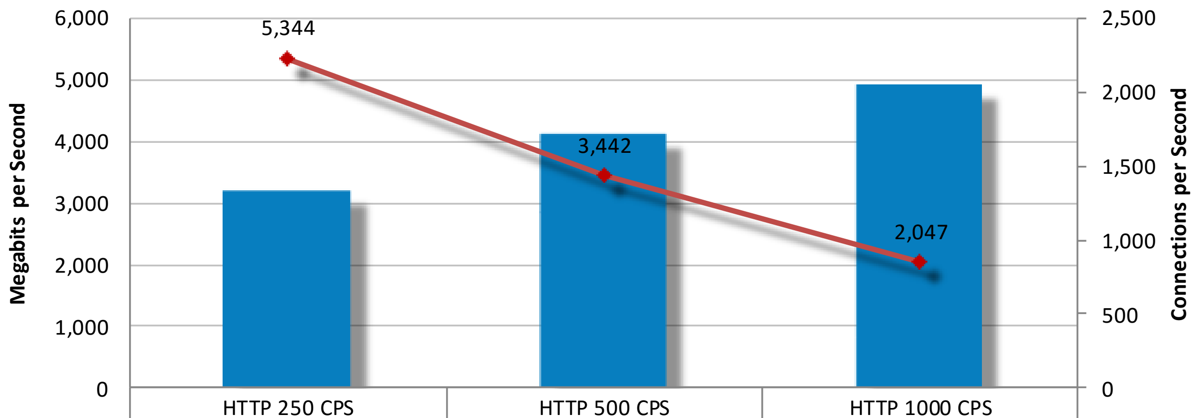
| Application Average Response Time – HTTP (at 90% Maximum Load) | Milliseconds |
|---|---|
| 2.500 Connections per Second – 44 Kbyte Response | 3.4 |
| 5,000 Connections per Second – 21 Kbyte Response | 2.0 |
| 10,000 Connections per Second – 10 Kbyte Response | 2.2 |
| 20,000 Connections per Second – 4.5 Kbyte Response | 1.1 |
| 40,000 Connections per Second – 1.7 Kbyte Response | 0.9 |

**Figure 31 – Average Application Response Time (Milliseconds)**

## HTTP Capacity with HTTP Persistent Connections

These tests determine how the BPS coped with network loads of varying average packet size and varying connections per second while inspecting all traffic. By creating genuine session-based traffic with varying session lengths, the BPS was 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 it is possible to achieve in a lab environment, while ensuring absolute accuracy and repeatability.

This test used HTTP persistent connections, with each TCP connection containing 10 HTTP GETs and associated responses. All packets contained valid payload (a mix of binary and ASCII objects) and address data. The test provides an excellent representation of a live network at various network loads. The stated response size was the total of all HTTP responses within a single TCP session.

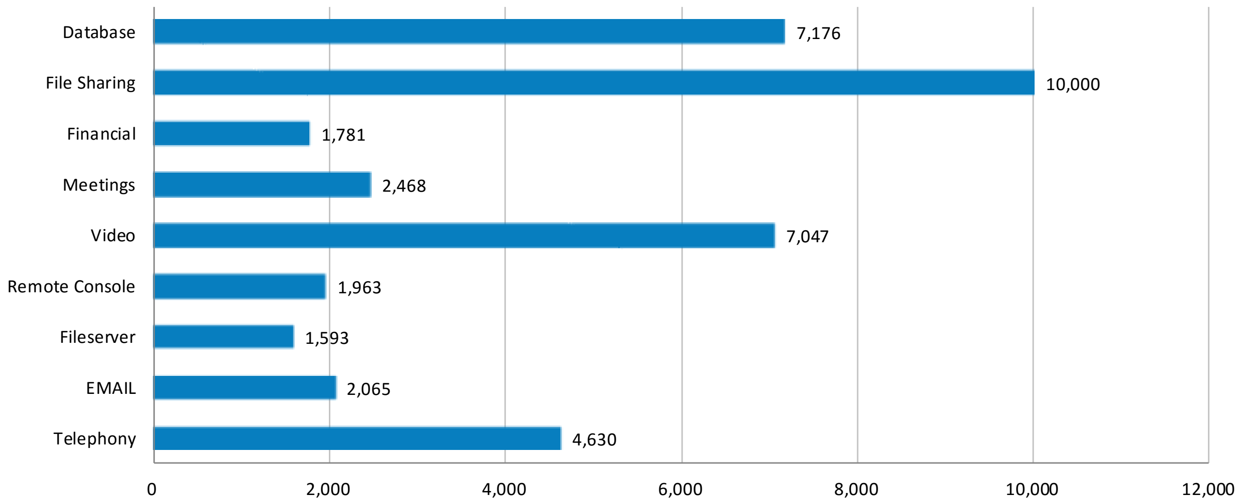



| | HTTP 250 CPS | HTTP 500 CPS | HTTP 1000 CPS |
|---|---|---|---|
| CPS | 1,336 | 1,721 | 2,047 |
| Mbps | 5,344 | 3,442 | 2,047 |

**Figure 32 – HTTP Capacity with HTTP Persistent Connections**

## Single Application Flows

*NSS-Tested Throughput* is rated at 5,717Mbps and is calculated as a weighted average of the traffic that we expected the BPS to experience in an enterprise environment. For more details on weighting and single application flow testing, please see the Appendix: Product Scorecard and the NSS Labs Breach Prevention Test Methodology, available at www.nsslabs.com.




| | Telephony | EMAIL | Fileserver | Remote Console | Video | Meetings | Financial | File Sharing | Database |
|---|---|---|---|---|---|---|---|---|---|
| Mbps | 4,630 | 2,065 | 1,593 | 1,963 | 7,047 | 2,468 | 1,781 | 10,000 | 7,176 |

**Figure 33 – Single Application Flows**

# Total Cost of Ownership (TCO)

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

- **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 solution out of the box, configure it, install components in 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 solution configurations, policy updates, policy deployment, alert handling, and so on

## Calculating the Total Cost of Ownership (TCO)

| Users | Mbps per User | Network Device Throughput | Centralized Management |
|---|---|---|---|
| 500 | 2 Mbps | 1,000 Mbps | 1 |

Figure 34 – Number of Users

When procuring a BPS for the enterprise, it is essential to factor in both bandwidth and number of users. NSS research has shown that, in general, enterprise network administrators architect their networks for up to 2 Mbps of sustained throughput per employee. For example, to support 500 users, an enterprise must deploy 500 agents and/or one network device of 1,000 Mbps capacity.

## Installation Time

| Product | Installation |
|---|---|
| **Fortinet FortiGate 500E** v6.0.3 + FortiClient v6.0.3.6219 + FortiSandbox v3.0.2 | 8 hours |

Figure 35 – Installation Time (Hours)

The table reflects the amount of time that NSS engineers, with the help of vendor engineers, needed to install and configure the BPS to the point where it operated successfully in the test harness, passed legitimate traffic, and blocked and detected any prohibited or malicious traffic. This closely mimics a typical enterprise deployment scenario for a single system. 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.

## 3-Year Total Cost of Ownership

| Details | Cost |
|---|---|
| Initial purchase price | $6,809 |
| Annual cost of support/ maintenance | $5,265 |
| Other Annual cost (AV, IPS, Cloud etc.) | $0 |
| Total cost year 1 | $12,073 |
| Total cost year 2 | $5,265 |
| Total cost year 3 | $5,265 |
| Total cost for all 3 years | $22,603 |

**Figure 36 – 3-Year TCO (US$)**

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 reflected as submitted by the vendor for the purpose of this test; costs for central management solutions (CMS) may be extra.

- **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.

# Appendix: Product Scorecard

| Description | Result | | |
|---|---|---|---|
| **Security Effectiveness** | | | |
| False Positives | **0.0%** | | |
| **Exploits** | **Blocked** | **Subsequently Detected** | |
| Drive-by Exploits | 100.0% | 0.0% | |
| Social Exploits | 100.0% | 0.0% | |
| **Malware** | **Blocked** | **Subsequently Detected** | |
| Delivered over Email (IMAP) | 98.9% | 0.5% | |
| Delivered over HTTP | 100.0% | 0.0% | |
| **Offline Infections (Employee Use Case)** | **Download** | **Execution** | **Reconnecting to the Network** |
| Sample 1 | Blocked | | |
| Sample 2 | Blocked | | |
| Sample 3 | | Blocked | |
| Sample 4 | | Blocked | |
| Sample 5 | Blocked | | |
| Sample 6 | | Blocked | |
| Sample 7 | | Blocked | |
| Sample 8 | | Blocked | |
| Sample 9 | | Blocked | |
| Sample 10 | | Blocked | |
| Sample 11 | | Blocked | |
| Sample 12 | | Blocked | |
| Sample 13 | Blocked | | |
| Sample 14 | Blocked | | |
| Sample 15 | Blocked | | |
| Sample 16 | Blocked | | |
| Sample 17 | Blocked | | |
| Sample 18 | Blocked | | |
| Sample 19 | Blocked | | |
| Sample 20 | Blocked | | |
| **Offline Infections (Contractor Use Case)** | | | **PASS** |
| **Packers** | | **Download** | **Execution** |
| Anskaya | | Blocked | |
| Anskaya | | Blocked | |
| Anskaya | | Blocked | |
| Exestealth | | Blocked | |
| Krypton | | Blocked | |
| Winkypt | | Blocked | |
| Winupack | | Blocked | |
| excalibur | | Blocked | |
| exefog | | Blocked | |

| | | |
|---|---|---|
| fearzpacker | Blocked | |
| fearzpacker | Blocked | |
| fishPE | Blocked | |
| fishPE | Blocked | |
| hidepx | Blocked | |
| hidepx | Blocked | |
| kkrunchy | Blocked | |
| kkrunchy | Blocked | |
| mew | Blocked | |
| petite24 | Blocked | |
| petite24 | Blocked | |
| telock | Blocked | |
| upx | Blocked | |
| upx | Blocked | |
| yc | Blocked | |
| yc | Blocked | |
| yc | Blocked | |

| Compressors | Download | Manual Scan |
|---|---|---|
| 7zip | Blocked | |
| 7zip | Blocked | |
| 7zip | Blocked | |
| 7zip | Blocked | |
| 7zip | Blocked | |
| 7zip | Blocked | |
| ALZIP | | Missed |
| ALZIP | | Missed |
| ALZIP | | Missed |
| ALZIP | | Missed |
| ALZIP | | Missed |
| ALZIP | | Missed |
| AshampooZip | Blocked | |
| AshampooZip | Blocked | |
| AverZip | Blocked | |
| AverZip | Blocked | |
| Bandizip | Blocked | |
| Bandizip | Blocked | |
| bzip2 | Blocked | |
| bzip2 | Blocked | |
| bzip2 | Blocked | |
| bzip2 | Blocked | |
| bzip2 | Blocked | |
| bzip2 | Blocked | |
| FilZip | Blocked | |

| | | |
|---|---|---|
| FilZip | Blocked | |
| gzip | Blocked | |
| gzip | Blocked | |
| gzip | Blocked | |
| gzip | Blocked | |
| gzip | Blocked | |
| gzip | Blocked | |
| KuaiZip | | Missed |
| KuaiZip | | Missed |
| KuaiZip | | Missed |
| KuaiZip | | Missed |
| KuaiZip | | Missed |
| KuaiZip | | Missed |
| MuZip | Blocked | |
| MuZip | Blocked | |
| PicoZip | Blocked | |
| PicoZip | Blocked | |
| PowerArchiver | Blocked | |
| PowerArchiver | Blocked | |
| PowerArchiver | Blocked | |
| PowerArchiver | Blocked | |
| PowerArchiver | Blocked | |
| PowerArchiver | Blocked | |
| QuickZip | Blocked | |
| QuickZip | Blocked | |
| SimplyZip | | Missed |
| SimplyZip | | Missed |
| SimplyZip | | Missed |
| SimplyZip | | Missed |
| SimplyZip | | Missed |
| SimplyZip | | Missed |
| xz | Blocked | |
| xz | Blocked | |
| xz | Blocked | |
| xz | Blocked | |
| xz | Blocked | |
| xz | Blocked | |
| ZipitFast | Blocked | |
| ZipitFast | Blocked | |
| **Sandbox Evasion** | **Download** | **Execution** |
| ASB_argcount.exe | Blocked | |
| ASB_argvalue.exe | Blocked | |
| ASB_changingwindow.exe | Blocked | |

| | | |
|---|---|---|
| ASB_checkpath.exe | Blocked | |
| ASB_checkselfname.exe | Blocked | |
| ASB_displaymessagebox.exe | Blocked | |
| ASB_enumwindows.exe | Blocked | |
| ASB_enumwindowstitle.exe | Blocked | |
| ASB_loadeddll.exe | Blocked | |
| ASB_namedpipes.exe | Blocked | |
| ASB_nettraffic.exe | Blocked | |
| ASB_parentexplorer.exe | Blocked | |
| ASB_sandboxprocess.exe | Blocked | |
| **Anti-Debugger Evasion** | **Download** | **Execution** |
| AD_closehandle.exe | Blocked | |
| AD_debuggerpresent.exe | Blocked | |
| AD_hardwarebp.exe | Blocked | |
| AD_pageguard.exe | Blocked | |
| AD_remotedebugger.exe | Blocked | |
| **Anti-Monitoring Evasion** | **Download** | **Execution** |
| AM_acceleratedsleep.exe | Blocked | |
| AM_analysistools.exe | Blocked | |
| AM_mousecursor.exe | Blocked | |
| AM_ntdelay.exe | Blocked | |
| AM_sleeploop.exe | Blocked | |
| AM_sleeppatched.exe | Blocked | |

| **Data Exfiltration** | **Results** |
|---|---|
| Ncat (Arch Linux) | Missed |
| HTTP POST (Arch Linux) | Missed |
| SSH Tunnel (Arch Linux) | Missed |
| ICMP Tunnel w/HTTP POST (Arch Linux) | Blocked |
| DNS tunnel w/HTTP POST (Arch Linux) | Blocked |
| Ncat (Windows 7) | Blocked |
| USB/Ncat (Windows 7) | Blocked |
| Ncat (Windows 10) | Missed |
| USB /Ncat (Windows 10) | Blocked |

| **IP Fragmentation** | **Results** |
|---|---|
| Small IP fragments; overlapping duplicate fragments with garbage payloads | PASS |
| Small IP fragments in reverse order | PASS |
| Small IP fragments in random order | PASS |
| Small IP fragments; delay first fragment | PASS |
| Small IP fragments in reverse order; delay last fragment | PASS |
| Small IP fragments; interleave chaff after (invalid IP options) | PASS |
| Small IP fragments in random order; interleave chaff sandwich (invalid IP options) | PASS |
| Small IP fragments in random order; interleave chaff sandwich (invalid IP options); delay random fragment | PASS |
| Small IP fragments; interleave chaff before (invalid IP options); DSCP value 16 | PASS |

| | |
|---|---|
| Small IP fragments in random order; interleave chaff after (invalid IP options); delay random fragment; DSCP value 34 | PASS |
| **TCP Segmentation** | **Results** |
| Small TCP segments; overlapping duplicate segments with garbage payloads | PASS |
| Small TCP segments in reverse order | PASS |
| Small TCP segments in random order | PASS |
| Small TCP segments; delay first segment | PASS |
| Small TCP segments in reverse order; delay last segment | PASS |
| Small TCP segments; interleave chaff after (invalid TCP checksums); delay first segment | PASS |
| Small TCP segments in random order; interleave chaff before (invalid TCP checksums); delay random segment | PASS |
| Small TCP segments in random order; interleave chaff sandwich (out-of-window sequence numbers); TCP MSS option | PASS |
| Small TCP segments in random order; interleave chaff after (requests to resynch sequence numbers mid-stream); TCP window scale option | PASS |
| Small TCP segments in random order; interleave chaff sandwich (requests to resynch sequence numbers mid-stream); TCP window scale option; delay first segment | PASS |
| Small overlapping TCP segments | PASS |
| Small TCP segments; small IP fragments | PASS |
| Small TCP segments; small IP fragments in reverse order | PASS |
| Small TCP segments in random order; small IP fragments | PASS |
| Small TCP segments; small IP fragments in random order | PASS |
| Small TCP segments in random order; small IP fragments in reverse order | PASS |
| Small TCP segments in random order; interleave chaff sandwich (invalid TCP checksums); small IP fragments in reverse order; interleave chaff after (invalid IP options) | PASS |
| Small TCP segments; interleave chaff after (invalid TCP checksums); delay last segment; small IP fragments; interleave chaff before (invalid IP options) | PASS |
| Small TCP segments; interleave chaff sandwich (invalid TCP checksums); small IP fragments; interleave chaff sandwich (invalid IP options); delay last fragment | PASS |
| Small TCP segments in random order; interleave chaff before (out-of-window sequence numbers); TCP MSS option; small IP fragments in random order; interleave chaff before (invalid IP options); delay random fragment | PASS |
| Small TCP segments in random order; interleave chaff sandwich (requests to resynch sequence numbers mid-stream); TCP window scale option; delay first segment; small IP fragments | PASS |
| Small overlapping TCP segments; small fragments | PASS |
| Small overlapping TCP segments; delay last segment; small fragments; delay last fragment | PASS |
| **HTTP Obfuscation** | **Results** |
| Declared HTTP/0.9 response; but includes response headers; chunking declared but served without chunking | PASS |
| HTTP/1.1 chunked response with chunk sizes preceded by multiple zeros (hex '30') | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by backspace (hex '08') | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by end of text (hex '03') | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by escape (hex '1b') | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by null (hex '00') | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by a space (hex '20') then a zero (hex '30') | PASS |
| HTTP/1.1 chunked response with final chunk size of '0000000000000000000000000000000000000000000000000000000000000000' (rather than '0') | PASS |
| HTTP/1.1 response with line folded transfer-encoding header declaring chunking ('Transfer-Encoding: ' followed by CRLF (hex '0d 0a') followed by 'chunked' followed by CRLF (hex '0d 0a'); served without chunking | PASS |

| | |
|---|---|
| HTTP/1.1 response with transfer-encoding header declaring chunking with lots of whitespace ('Transfer-Encoding:' followed by 8000 spaces (hex '20' * 8000) followed by 'chunked' followed by CRLF (hex '0d 0a'); served chunked | PASS |
| HTTP/1.0 response declaring chunking; served without chunking | PASS |
| HTTP/1.0 response declaring chunking with invalid content-length header; served without chunking | PASS |
| HTTP/1.1 response with "\tTransfer-Encoding: chunked"; served chunked | PASS |
| HTTP/1.1 response with "\tTransfer-Encoding: chonked" after custom header line with "chunked" as value; served without chunking | PASS |
| HTTP/1.1 response with header with no field name and colon+junk string; followed by '\tTransfer-Encoding: chunked' header; followed by custom header; served chunked | PASS |
| HTTP/1.1 response with "\r\rTransfer-Encoding: chunked"; served chunked | PASS |
| HTTP/1.1 response with using single "\n"'s instead of "\r\n"'s; chunked | PASS |
| HTTP/1.1 response with \r\n\r\n before first header; chunked | PASS |
| HTTP/1.1 response with "SIP/2.0 200 OK\r\n" before status header; chunked | PASS |
| HTTP/1.1 response with space+junk string followed by \r\n before first header; chunked | PASS |
| HTTP/1.1 response with junk string before status header; chunked | PASS |
| HTTP/1.1 response with header end \n\014\n\n; chunked | PASS |
| HTTP/1.1 response with header end \r\n\016\r\n\r\n; chunked | PASS |
| HTTP/1.1 response with header end \n\r\r\n; chunked | PASS |
| HTTP/1.1 response with header end \n\017\018\n\n; chunked | PASS |
| HTTP/1.1 response with header end \n\030\n\019\n\n; chunked | PASS |
| HTTP/1.1 response with status code -203.030; with message-body; chunked | PASS |
| HTTP/1.1 response with status code 402; with message-body; chunked | PASS |
| HTTP/1.1 response with status code 403; with message-body; chunked | PASS |
| HTTP/1.1 response with status code 406; with message-body; chunked | PASS |
| HTTP/1.1 response with status code 505; with message-body; chunked | PASS |
| HTTP/1.1 chunked response with no status indicated | PASS |
| No status line; chunking indicated; served unchunked | PASS |
| HTTP/1.1 response with invalid content-length header size declaration followed by space and null (hex '20 00') | PASS |
| HTTP/1.01 declared; served chunked | PASS |
| HTTP/01.1 declared; served chunked | PASS |
| HTTP/2.B declared; served chunked | PASS |
| HTTP/9.-1 declared; served chunked | PASS |
| Double Transfer-Encoding: first empty; last chunked. Served with invalid content-length; not chunked. | PASS |
| Relevant headers padded by preceding with hundreds of random custom headers | PASS |
| **HTTP Compression** | **Results** |
| HTTP/1.1 response compressed with gzip; invalid content-length | PASS |
| HTTP/1.1 response declaring gzip followed by junk string; invalid content-length; served uncompressed | PASS |
| HTTP/1.1 response compressed with deflate; invalid content-length | PASS |
| HTTP/1.1 response declaring deflate followed by junk string; invalid content-length; served uncompressed | PASS |
| HTTP/1.1 response with content-encoding declaration of gzip followed by space+junk string; served uncompressed and chunked | PASS |
| HTTP/1.1 response with content-encoding header for deflate; followed by content-encoding header for gzip; served uncompressed and chunked | PASS |
| HTTP/1.1 chunked response with chunk sizes preceded by multiple zeros (hex '30'); compressed with gzip | PASS |

| | |
|---|---|
| HTTP/1.1 chunked response with chunk sizes followed by backspace (hex '08'); compressed with gzip | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by end of text (hex '03'); compressed with gzip | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by escape (hex '1b'); compressed with gzip | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by null (hex '00'); compressed with gzip | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by a space (hex '20') then a zero (hex '30'); compressed with gzip | PASS |
| HTTP/1.1 chunked response with chunk sizes preceded by multiple zeros (hex '30'); compressed with deflate | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by backspace (hex '08'); compressed with deflate | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by end of text (hex '03'); compressed with deflate | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by escape (hex '1b'); compressed with deflate | FAIL |
| HTTP/1.1 chunked response with chunk sizes followed by null (hex '00'); compressed with deflate | PASS |
| HTTP/1.1 chunked response with chunk sizes followed by a space (hex '20') then a zero (hex '30'); compressed with deflate | FAIL |
| **HTML Obfuscations** (*Not included in the evasion calculations) | **Results** |
| js-binary-obfuscation* | FAIL |
| babel-minify* | PASS |
| closure* | PASS |
| code-protect* | FAIL |
| confusion* | FAIL |
| jfogs* | FAIL |
| jfogs-reverse* | FAIL |
| jjencode* | PASS |
| jsbeautifier* | PASS |
| jsmin* | PASS |
| js-obfuscator* | FAIL |
| qzx-obfuscator* | FAIL |
| chunked and gzip compressed js-binary-obfuscation* | FAIL |
| chunked and deflate compressed js-binary-obfuscation* | FAIL |
| UTF-8 encoding | PASS |
| UTF-8 encoding with BOM | PASS |
| UTF-16 encoding with BOM | PASS |
| UTF-8 encoding; no http or html declarations | PASS |
| UTF-8 encoding with BOM; no http or html declarations | PASS |
| UTF-16 encoding with BOM; no http or html declarations | PASS |
| UTF-16-LE encoding without BOM | FAIL |
| UTF-16-BE encoding without BOM | FAIL |
| UTF-16-LE encoding without BOM; no http or html declarations | FAIL |
| UTF-16-BE encoding without BOM; no http or html declarations | FAIL |
| UTF-7 encoding | PASS |
| UTF-8 encoding | PASS |
| UTF-8 encoding | PASS |
| EICAR string included at top of HTML | PASS |
| Hex encoded script decoded using JavaScript unescape* | PASS |
| Unicode encoded script decoded using JavaScript unescape* | FAIL |

| | |
|---|---|
| Hex encoded script as variable decoded using JavaScript unescape* | PASS |
| Unicode encoded script as variable decoded using JavaScript unescape* | FAIL |
| padded with <=5MB | PASS |
| padded with <=25MB | PASS |
| padded with >25MB | PASS |
| padded with <=5MB; chunked and compressed with gzip | PASS |
| padded with <=25MB; chunked and compressed with gzip | PASS |
| padded with >25MB; chunked and compressed with gzip | PASS |
| padded with <=5MB; chunked and compressed with deflate | PASS |
| padded with <=25MB; chunked and compressed with deflate | PASS |
| padded with >25MB; chunked and compressed with deflate | PASS |
| **Protection Resiliency** (*Not included in the evasion calculations) | **Results** |
| Hex encoded VBScript decoded using JavaScript unescape* | PASS |
| Hex encoded VBScript as variable decoded using JavaScript unescape* | PASS |
| External VBScript file loaded from HTML* | PASS |
| Multiple VBScript files loaded from HTML* | PASS |
| Multiple VBScript files loaded with external JavaScript file* | PASS |
| VBScript interspersed randomly with null bytes* | PASS |
| nishang bind shell obfuscated with Unicorn* | PASS |
| native Unicorn generated bind shell* | PASS |
| nishang bind shell obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| custom bind shell shellcode obfuscated with Invoke-Obfuscation* | PASS |
| custom bind shell shellcode with password prompt obfuscated with Invoke-Obfuscation* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; nishang bind shell obfuscated with Unicorn* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; nishang bind shell obfuscated with Unicorn; chunked* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; native Unicorn generated bind shell* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of | PASS |

| | |
|---|---|
| chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; native Unicorn generated bind shell; chunked and gzip compressed* | |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; nishang bind shell obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H using online vbscript obfuscator; both spaces and linefeeds replaced with multiples of each; nishang bind shell obfuscated with PowerSploit's Out-EncodedCommand; chunked and deflate compressed* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand; chunked* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; custom bind shell shellcode obfuscated with Invoke-Obfuscation* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; custom bind shell shellcode obfuscated with Invoke-Obfuscation; chunked and gzip compressed* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; custom bind shell shellcode with password prompt obfuscated with Invoke-Obfuscation* | PASS |
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; combine 'myarray' instantiation into single line; combine powershell command into single line; Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script; Some strings split with "+" and "&"; some lines split with "_"; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; custom bind shell shellcode with password prompt obfuscated with Invoke-Obfuscation; chunked and deflate compressed* | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; | PASS |

| | |
|---|---|
| hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; nishang bind shell obfuscated with Unicorn* | |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; nishang bind shell obfuscated with Unicorn; chunked* | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; native Unicorn generated bind shell* | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; native Unicorn generated bind shell; chunked and gzip compressed* | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; nishang bind shell obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; nishang bind shell obfuscated with PowerSploit's Out-EncodedCommand; chunked and deflate compressed* | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand; chunked* | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; custom bind shell shellcode obfuscated with Invoke-Obfuscation* | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; custom bind shell shellcode obfuscated with Invoke-Obfuscation; chunked and gzip compressed | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; custom bind shell shellcode with password prompt obfuscated with Invoke-Obfuscation* | PASS |
| procedures and variables renamed; some script commands/strings converted to series of chr()/Clng/&H; both spaces and linefeeds replaced with multiples of each; numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values; Some strings split with "+" and "&"; some lines split with "_"; custom bind shell shellcode with password prompt obfuscated with Invoke-Obfuscation; chunked and deflate compressed* | PASS |
| both spaces and linefeeds replaced with multiples of each* | PASS |
| procedures and variables renamed* | PASS |

| | |
|---|---|
| numeric values/equations modified and/or inserted; hexadecimal values replaced with decimal values* | PASS |
| change all chr() to chrw() and vice versa where possible* | PASS |
| change chr() and chrw() to chrb()* | PASS |
| some script commands/strings converted to series of chr()/Clng/&H using online vbscript obfuscator* | PASS |
| Veil Ordnance bind shell shellcode dropped into PowerSploit's Invoke-Shellcode; then obfuscated with PowerSploit's Out-EncodedCommand* | PASS |
| Use wscript to call original payload (PoshRat method) * | PASS |
| nishang bind shell obfuscated with Unicorn* | PASS |
| Remove runmumaa and add to setnotsafemode function; move setnotsafemode function to bottom of script* | PASS |
| Some strings split with "+" and "&"; some lines split with "_"* | PASS |
| combine 'myarray' instantiation into single line; combine powershell command into single line* | PASS |
| Combination of techniques used in res-ren-001 and res-chr-003* | PASS |
| Combination of techniques used in res-ren-001; res-chr-003; and res-wsp-001* | PASS |
| Combination of techniques used in res-ren-001; res-chr-003; res-wsp-001; and res-pay-004* | PASS |
| Combination of techniques used in res-ren-001 and res-pay-007* | PASS |
| Combination of techniques used in res-ren-001; res-chr-003; res-wsp-001; res-pay-007; and res-mth-001* | PASS |
| Combination of techniques used in res-mth-001 and res-mrg-001* | PASS |
| Combination of techniques used in res-mth-001; res-mrg-001; and res-ord-001* | PASS |
| Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; and res-pay-008* | PASS |
| Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-008; and res-spl-001* | PASS |
| Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-008; res-spl-001; and res-chr-003* | PASS |
| Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-008; res-spl-001; and res-chr-003; plus removal of all CLng's* | PASS |
| Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-008; and res-chr-003* | PASS |
| Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-008; res-spl-001; res-chr-003; res-wsp-001; plus removal of all CLng's* | PASS |
| Combination of techniques used in res-mth-001; res-mrg-001; res-ord-001; res-pay-009; res-spl-001; res-chr-003; res-wsp-001; res-ren-001; plus removal of all CLng's; replace 'LANGUAGE="VBScript"' with 'type="text/vbScript"'* | PASS |
| UTF-8 encoding; HTTP/1.1 chunked response with chunk sizes preceded by multiple zeros (hex '30'); small TCP segments; small IP fragments; padding | PASS |
| UTF-8 encoding with BOM; HTTP/1.1 chunked response with chunk sizes followed by backspace (hex '08'); small TCP segments; small IP fragments in reverse order; padding | FAIL |
| UTF-16 encoding with BOM; HTTP/1.1 chunked response with chunk sizes followed by end of text (hex '03'); small TCP segments in random order; small IP fragments; padding | FAIL |
| UTF-8 encoding; no http or html declarations; HTTP/1.1 chunked response with chunk sizes followed by escape (hex '1b'); small TCP segments; small IP fragments in random order; padding | FAIL |
| UTF-8 encoding with BOM; no http or html declarations; HTTP/1.1 chunked response with chunk sizes followed by null (hex '00'); small TCP segments in random order; small IP fragments in reverse order; padding | PASS |

Performance

| Raw Packet Processing Performance (UDP Traffic) | Mbps | NSS-Rated Throughput Weighting |
|---|---|---|
| 64 Byte Packets | 20,000 | 0.4% |
| 128 Byte Packets | 20,000 | 0.6% |
| 256 Byte Packets | 20,000 | 1.2% |

| | | | |
|---|---|---|---|
| 512 Byte Packets | 20,000 | | 1.2% |
| 1024 Byte Packets | 20,000 | | 2.5% |
| 1514 Byte Packets | 20,000 | | 3.3% |
| **Latency – UDP** | **Microseconds** | | |
| 64 Byte Packets | 2 | | |
| 128 Byte Packets | 2 | | |
| 256 Byte Packets | 2 | | |
| 512 Byte Packets | 2 | | |
| 1024 Byte Packets | 2 | | |
| 1514 Byte Packets | 3 | | |
| **Maximum Capacity** | **CPS** | | |
| Theoretical Max. Concurrent TCP Connections | 4,275,101 | | |
| Maximum TCP Connections per Second | 78,420 | | |
| Maximum HTTP Connections per Second | 64,000 | | |
| Maximum HTTP Transactions per Second | 150,200 | | |
| **HTTP Capacity with No Transaction Delays** | **CPS** | **Mbps** | **NSS-Rated Throughput Weighting** |
| 2,500 Connections per Second – 44 Kbyte Response | 10,040 | 4,016 | 8.0% |
| 5,000 Connections per Second – 21 Kbyte Response | 13,460 | 2,692 | 7.5% |
| 10,000 Connections per Second – 10 Kbyte Response | 17,050 | 1,705 | 7.0% |
| 20,000 Connections per Second – 4.5 Kbyte Response | 18,860 | 943 | 7.0% |
| 40,000 Connections per Second – 1.7 Kbyte Response | 19,940 | 499 | 4.0% |
| **Application Average Response Time – HTTP (at 90% Max Load)** | **Milliseconds** | | |
| 2.500 Connections per Second – 44 Kbyte Response | 3.4 | | |
| 5,000 Connections per Second – 21 Kbyte Response | 2.0 | | |
| 10,000 Connections per Second – 10 Kbyte Response | 2.2 | | |
| 20,000 Connections per Second – 4.5 Kbyte Response | 1.1 | | |
| 40,000 Connections per Second – 1.7 Kbyte Response | 0.9 | | |
| **HTTP Capacity with HTTP Persistent Connections** | **CPS** | | **Mbps** |
| 250 Connections per Second | 1,336 | | 5,344 |
| 500 Connections per Second | 1,721 | | 3,442 |
| 1,000 Connections per Second | 2,047 | | 2,047 |
| **Single Application Flows** | **Mbps** | | **NSS-Rated Throughput Weighting** |
| Telephony | 4,630 | | 16.9% |
| EMAIL | 2,065 | | 12.0% |
| Fileserver | 1,593 | | 0.4% |
| Remote Console | 1,963 | | 0.8% |
| Video | 7,047 | | 16.2% |
| Meetings | 2,468 | | 0.8% |
| Financial | 1,781 | | 0.0% |
| File Sharing | 10,000 | | 7.2% |
| Database | 7,176 | | 3.0% |

| Total Cost of Ownership | |
|---|---|
| **Ease of Use** | |
| Initial Setup (Hours) | 8 |
| **Expected Costs** | |
| Initial purchase price | $6,809 |
| Annual cost of support/ maintenance | $5,265 |
| Other Annual cost (AV, IPS, Cloud etc.) | $0 |
| | |
| Total cost year 1 | $12,073 |
| Total cost year 2 | $5,265 |
| Total cost year 3 | $5,265 |
| Total cost for all 3 years | $22,603 |

**Figure 37 – Scorecard**

# Test Environment

- VmWare vCenter (Version 6.5.0 Build 5973321)
- VmWare vSphere (Version 6.5.0.10000 Build 5973321)
- VmWare ESXi (Version 6.5.0-20170702001-standard)
- BaitNET (NSS Labs Proprietary)
- Evader++ (NSS Labs Proprietary)
- Hak5 Bash Bunny (1.5_298)
- Ixia BreakingPoint Perfect Storm (Version 8.40.16.19)
- Rapid7 Metasploit (v5.0.3-dev)
- 32-bit Microsoft Windows 7 (Version 6.1 (Build 7601: SP1) with Internet Explorer 9 (Version 9.0.8112.16421 – Update version 9.0.26)
- 64-bit Microsoft Windows 7 (Version 6.1 (Build 7601: SP1) and Internet Explorer 11 (Version 11.0.9600.17843 – Update version 11.0.20)
- 64-bit Microsoft Windows 10 (version 1607 (Build: 14393.0), Internet Explorer 11 (Version 11.0.14393.0 – Update version 11.0.33)
- Microsoft Office 2016 with Microsoft Word (Version 1609, Build 7369.2038)
- Microsoft Silverlight 5.1.20125
- Adobe Reader 9.3.4
- Adobe Flash Player 18.0.0.160
- Adobe Flash Player 29.0.0.171
- Adobe Reader 9.40
- Adobe Reader DC 2017.012.20093
- Oracle Java 6 Update 27
- Oracle Java 6 Update 27
- Oracle Java 8 Update 171
- Oracle Java 8 Update 181
- Nitro Pro PDF Reader 11.0.3.173
- WinRar 4.20
- Kali (Kernel release 4.19.0-kali1-amd64)
- CentOS 7 (Kernel release 3.10.0-957.5.1.el7.x86_64)
- Arch Linux (Kernel release 4.17.5-1-ARCH)
- FreeBSD (Kernel release 11.1-RELEASE-p1)

Published: August 7, 2019
Authors – Jessica Williams, Ryan Turner, Tim Otto

# Test Methodology

NSS Labs Breach Prevention Systems (BPS) Test Methodology v2.0

NSS Labs Evasions Test Methodology v1.2

Copies of the test methodologies are available at www.nsslabs.com.

# Contact Information

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

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



Please read the disclaimer in this box because it contains important information that binds you. If you do not agree to these conditions, you should not read the rest of this report but should instead return the report immediately to us. "You" or "your" means the person who accesses this report and any entity on whose behalf he/she has obtained this report.

1. The information in this report is subject to change by us without notice, and we disclaim any obligation to update it.

2. The information in this report is believed by us to be accurate and reliable at the time of publication but is not guaranteed. All use of and reliance on this report are at your sole risk. We are not liable or responsible for any damages, losses, or expenses of any nature whatsoever arising from any error or omission in this report.

3. NO WARRANTIES, EXPRESS OR IMPLIED ARE GIVEN BY US. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, ARE HEREBY DISCLAIMED AND EXCLUDED BY US. IN NO EVENT SHALL WE BE LIABLE FOR ANY DIRECT, CONSEQUENTIAL, INCIDENTAL, PUNITIVE, EXEMPLARY, OR INDIRECT DAMAGES, OR FOR ANY LOSS OF PROFIT, REVENUE, DATA, COMPUTER PROGRAMS, OR OTHER ASSETS, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

4. This report does not constitute an endorsement, recommendation, or guarantee of any of the products (hardware or software) tested or the hardware and/or software used in testing the products. The testing does not guarantee that there are no errors or defects in the products or that the products will meet your expectations, requirements, needs, or specifications, or that they will operate without interruption.

5. This report does not imply any endorsement, sponsorship, affiliation, or verification by or with any organizations mentioned in this report.