TEST METHODOLOGY

SSL/TLS Performance

December 10, 2019

v1.51

Table of Contents

1 Introduction

1.1 The Need for SSL/TLS Performance Testing

Use of the Secure Sockets Layer (SSL) protocol and its current iteration, Transport Layer Security (TLS), is rising dramatically in response to an ever-increasing need for online privacy. In August 2019, NSS Labs found that 58% of web traffic is being sent over HTTPS. SSL/TLS is prone to various security attacks that can occur at multiple levels of communication in the network—attacks have been observed on the handshake protocol, record protocol, application data protocol, and PKI, to name just a few. To address the growing threat of focused attacks using the most common web protocols and applications, NSS has developed a methodology that tests the capabilities and performance of devices providing visibility into the SSL/TLS protocols.

1.2 About This Test Methodology

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

- Verification of SSL/TLS functional capabilities
- SSL/TLS performance
- A device claiming SSL/TLS visibility should possess the following capabilities:
- Perform SSL/TLS decryption
- Negotiate commonly used ciphers and key sizes
- Set policies excluding some subset of traffic from decryption
- Perform either of the following:
    - Natively detect and block TLS encapsulated attacks, including exploits and evasions
    - Feed external security solutions with decrypted stream of traffic for the purpose of detecting and blocking TLS encapsulated attacks, including exploits and evasions

NSS Labs test methodologies are continually evolving in response to feedback. If you would like to provide input, please contact advisors@nsslabs.com. For a list of changes, please reference the Change Log in the Appendix.

1.3 Inclusion Criteria

To encourage participation and to allay concerns of bias, NSS invites all security vendors claiming SSL/TLS capabilities to submit their products at no cost. Vendor participation is not limited by market share. Examples of technologies offering SSL/TLS capabilities include next generation firewall (NGFW), next generation intrusion prevention system (NGIPS), secure web gateway (SWG), and SSL/TLS offload solutions. This test will support both explicit and transparent proxies and will have the flexibility to test devices with onboard inspection engines as well as those designed to send decrypted traffic through one or more external inspection solutions.

2 SSL/TLS Functionality Testing

2.1 Decryption Validation

To confirm that the device under test is correctly decrypting and (if applicable) inspecting SSL/TLS traffic, a validation test will be performed prior to conducting functional or performance testing. The device will be expected to cover all test cases in this methodology with a single configuration.

For devices providing both decryption and inspection, this test will consist of a known exploit embedded in encrypted traffic passed through the device. NSS has an extensive library of well-known malicious files and exploits suitable for this purpose. Devices in this category will be expected to decrypt the stream, detect the exploit, and block or alert as appropriate. The purpose of this test is not to evaluate security effectiveness of the device but rather to validate that the device is decrypting and inspecting traffic.

For devices providing decryption without an onboard inspection engine (i.e., SSL/TLS offload devices), the same test is performed. However, instead of requiring the device to block or alert on the encapsulated payload, the evaluation methodology will include manual analysis via differential comparison of the traffic pre- and post-proxy.

2.2 Cipher Selection

To determine the most commonly employed cipher suites for inclusion in testing, ciphers were selected from the 12/31/2017 results of the Alexa Top 1 Million Analysis.[1] The top 30 ciphers from these data were selected for use in functional capability testing and the top four (representing more than 90% of the distribution) for performance.

2.3 Cipher Support

The device is expected to be capable of negotiating a wide range of commonly used SSL/TLS ciphers in order to increase the security visibility of potential threats encapsulated in real-world SSL/TLS traffic. This test will cover the top 30 cipher suites as determined in section 2.3.1 of this methodology. Unless otherwise specified, the functional tests will use the most common key sizes for RSA (2,048 bit) and ECDSA (256 bit).

2.3.1 Top 24 Cipher Suites from the Alexa Top 1 Million, as of 12/31/2017:

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA

[1] Alexa Top 1 Million Analysis performed on 10/03/2019 by Scott Helme

TLS_RSA_WITH_AES_128_CBC_SHA	TLS_DHE_RSA_WITH_SEED_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_SEED_CBC_SHA
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256

2.3.2 Support for Emergent Ciphers

In addition to the top 30 ciphers specified in section 2.3.1, support for the following emergent ciphers and parameters will be tested:

- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- x25519 Elliptic Curve Key Exchange

2.3.3 Prevention of Weak Ciphers

The device will be expected to protect against the use of ciphers that are known to offer either weak protection or none at all, including (but not limited to):

- Null ciphers (no encryption of data provided)
- Anonymous ciphers (no authentication provided)

2.3.4 Support for TLS 1.3

Within the past year, TLS 1.3 was approved by the IETF to be the new standard for securing connections. TLS 1.3 includes several controversial changes over TLS 1.2 along with a few new handshake protocols. Certain elements of TLS 1.3 such as certificate pinning prevent the third-party inspection of traffic required to protect users from attacks. As such, secure and desirable behavior is for a device to force a downgrade from TLS 1.3 to TLS 1.2 and maintain inspection rather than lose all visibility and protections. The device will be expected to **either** decrypt and inspect TLS 1.3 connections **or** block TLS 1.3 and force a downgrade to TLS 1.2, and then decrypt and inspect traffic for functional testing as well as performance testing.

- TLS13_AES_256_GCM_SHA384
- TLS13_AES_128_GCM_SHA256
- TLS13_CHACHA20_POLY1305_SHA256
- TLS13_AES_128_CCM_8_SHA256
- TLS13_AES_128_CCM_SHA256

Results will be reported, so that customers will know what behavior(s) to expect from the tested device.

2.4 Decryption Bypass Exceptions

The device will be expected to support the configuration of policies that permit conditional bypass of decryption in order to preserve privacy, either for regulatory or other reasons. The device must maintain decryption capabilities as tested under section 2.3 concurrently with these conditional bypass rules; i.e., turning off all decryption on a device is not an acceptable method for meeting requirements under this section. The device will be tested for decryption bypass capabilities under various conditions, including:

- Layer 3 information (i.e., bypass based on source or destination IP address)

- Layer 4 information (i.e., bypass based on TCP port number)
- Server Name Indication (SNI) TLS extension information
- Site category based on Common Name (CN) and/or Subject Alternative Name (SAN)

2.5 Certificate Validation

The device will be expected to validate the status of all SSL/TLS certificates presented, except in cases where decryption bypass is enabled. When presented with an invalid certificate, the device must either prevent the establishment of a connection or replicate the original invalid status in the proxied/resigned certificate presented to the client, such that the client is aware of the potential risk.

2.6 TLS Session Re-use

In order to improve performance and reduce the overhead associated with conducting the full handshake for each session, the TLS protocol allows for abbreviated handshakes, which re-use previously established sessions. The two primary methods for session re-use are session IDs and session tickets. Whereas session IDs are included in the main TLS specification, session tickets are an extension of the specification, detailed in a separate RFC. Support for both of these methods will be tested under this section.

3 SSL/TLS Performance

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

Note: For these tests, the same configuration established for testing under Section 2 of this methodology will be used This will ensure that the device is not bypassing the decryption/inspection process for the purpose of better performance.

Baseline HTTP tests using no encryption will be performed for each category under this section in order to establish comparative metrics against which the performance decrease under load of SSL/TLS decryption/inspection may be measured.

The same cipher selection methodology outlined in Section 2.2 will be used to determine testing targets under this section. The top four ciphers as listed in Section 2.3.1:

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 using 2,048 bit and 4,096 bit keys
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 using a 2,048 bit key
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 using a 256 bit key and the secp256r1 curve
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 using a 2,048 bit key
- TLS13_AES256_GCM_SHA384

3.1 Maximum HTTP(S) Connections per Second

This test is designed to determine the maximum HTTPS connection rate of the device with a one-byte response size. This type of traffic is atypical of a normal network, but the negligible payload size provides a means to measure the device's SSL/TLS handshake performance independent of throughput performance. An increasing number of new sessions is established through the device until a maximum is reached and each session is immediately closed upon successful negotiation of the SSL/TLS handshake and transfer of the payload.

3.2 HTTP(S) Capacity

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

Each transaction consists of a single (1) HTTP(S) GET request ,and there are no transaction delays (i.e., the web server responds immediately to all requests). All packets contain valid payload (a mix of binary and ASCII objects) and address data, and this test provides a feasible representation of a live network (albeit one biased towards HTTPS traffic) at various network loads.

3.2.1 2,880 KB HTTP(S) Response Size – 40 Connections per Second

Maximum 40 new connections per second per Gigabit of traffic with a 2,880 KB HTTP(S) response.

3.2.2 768 KB HTTP(S) Response Size – 150 Connections per Second

Maximum 150 new connections per second per Gigabit of traffic with a 768 KB HTTP(S)response.

3.2.3 192 KB HTTP(S) Response Size – 600 Connections per Second

Maximum 600 new connections per second per Gigabit of traffic with a 192 KB HTTP(S) response.

3.2.4 44 KB HTTP(S) Response Size – 2,500 Connections per Second

Maximum 2,500 new connections per second per Gigabit of traffic with a 44 KB HTTPS response.

3.3 HTTP(S) Capacity with HTTP(S) Persistent Connections

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

This test will use HTTP persistent connections, with each TCP connection containing ten (10) HTTP(S) GETs and associated responses. All packets contain valid payload (a mix of binary and ASCII objects) and address data as a feasible representation of a live network (albeit one biased towards HTTPS traffic) at various network loads. The stated response size is the total of all HTTP responses within a single TCP session.

3.3.1 2,880 KB HTTP(S) Response Size – 400 Transactions per Second

Maximum 40 new connections per second (400 transactions per second) per Gigabit of traffic with a 2,880 KB HTTP(S) response.

3.3.2 768 KB HTTP(S) Response Size – 1,500 Transactions per Second

Maximum 150 new connections per second (1,500 transactions per second) per Gigabit of traffic with a 768 KB HTTP(S) response.

3.3.3 192 KB HTTP(S) Response Size – 6,000 Transactions per Second

Maximum 600 new connections per second (6,000 transactions per second) per Gigabit of traffic with a 192 KB HTTP(S) response.

3.3.4 44 KB HTTP(S) Response Size – 25,000 Transactions per Second

Maximum 2,500 new connections per second (25,000 transactions per second) per Gigabit of traffic with a 44 KB HTTP(S) response.

3.4 Application Average Response Time: HTTP(S)

The test traffic is passed through the device and directly back to the test generator. Time to First Byte (TTFB) and Time to Last Byte (TTLB) results are recorded for each response size (44 KB, 192 KB, 768 KB, and 2,880 KB HTTP(S)) from sections 3.2 and 3.3 and at a load level of 95% of the maximum throughput with zero packet loss. For section 3.3, these data points represent the average response time for each transaction within the persistent connection.

Appendix A: Cipher Selection Details

In the Alexa Top 1 Million Analysis,[2] cipher frequency was determined via a custom PHP script using cURL for crawling with the following parameters:

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HEADER, 1);
curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36');
curl_setopt($ch, CURLOPT_TIMEOUT, 10);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_URL, $address);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_CERTINFO, 1);
curl_setopt($ch, CURLOPT_VERBOSE, true);
$response = curl_exec($ch);
```

The crawler initiated a GET request for the domain, followed redirects to completion, and extracted the headers at the final location. A certificate was also extracted to obtain the issuer string. For TLS ciphers, OpenSSL 1.0.2g1 (Mar 2016) was employed using a standard connection with the default configuration to allow the server to choose the suite, and from the debug output, results were parsed to populate values:

```
openssl s_client -connect $hostname:443 -servername $hostname -debug
```

[2] Alexa Top 1 Million Analysis. Scott Helme

Appendix B: Change Log

Version 1.51 – December 10, 2019

- Updated Section 2.3.4

Version 1.5 – October 24, 2019

- Added TLS 1.3 functional tests
- Added TLS 1.3 performance tests
- Revised cipher support list by removing RC4 and 3DES
- Updated Scott Helme source information

Version 1.4 – December 3, 2018

- Section 1.2 inclusion and feedback guidance added. Additional edits for style, clarity.

Version 1.3 – February 2, 2018

- Added cipher selection methodology as new Section 2.2
- Restructured cipher test targets for both functional and performance testing
- Moved Cipher Negotiation section 2.2 to Cipher Support section 2.3 and reorganized subsections
- Section 2.3: Functionality separated into sections 2.4 – 2.6
- Switched sections 3.3 and 3.4
- Added clarifying verbiage in various sections

Version 1.2 – January 27, 2017

- Removed fourteen (14) cipher suites from Section 2.2
- Removed ChaCha20-Poly1305 from Section 2.3
- Clarified known weak cipher suites with examples
- Corrected two (2) cipher suites in Section 3
- Changed load level for scoring in section 3.3

Version 1.1 – November 4, 2016

- Added Functionality subsection to Section 2
- Added cipher suites to Section 2
- Clarified versions of SSL to be tested in Section 2
- Clarified types of devices to be tested in Section 1

Version 1.0 – September 6, 2016

Contact Information

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

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

© 2019 NSS Labs, Inc. All rights reserved. No part of this publication may be reproduced, copied/scanned, stored on a retrieval system, emailed or otherwise disseminated or transmitted without the express written consent of NSS Labs, Inc. ("us" or "we").

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.

6. All trademarks, service marks, and trade names used in this report are the trademarks, service marks, and trade names of their respective owners.