

Global Information Assurance Certification Paper

Copyright SANS Institute Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering "Network Monitoring and Threat Detection In-Depth (Security 503)" at http://www.giac.org/registration/gcia

Benefits and Adoption Rate of TLS 1.3

GIAC (GCIA) Gold Certification

Author: Ben Weber, ben.weber@student.sans.edu Advisor: Johannes Ullrich

Accepted: July 8th, 2020

Abstract

The cybersecurity industry is often reluctant to adopt new technologies due to perceived complications, assumed dependencies, and unclear information about the benefits. Digital communication protections are not exempt from this phenomenon and are often overlooked when maintaining a secure environment. Adopting new technologies is essential to utilize recent advancements in speed, security, and other newly available features. RFC 8446, better known as TLS 1.3, was released in August of 2018 and included enhancements to the speed and security of a TLS session. Older versions of TLS that still exist, however, fall short when compared to TLS 1.3. This paper provides data testing the speed and security of TLS 1.3 compared to TLS 1.2 across major TLS libraries and a point-in-time measurement of TLS 1.3 adoption across the top 500 websites in the business, retail, technology, and news sectors.

1. Introduction

Information Technology practitioners are often reluctant to make changes to an environment or technology without a driving force supporting the migration. Sometimes, that driving force is performance-related: for example, an organization may make a change because the organization expects a 10% increase in performance. Other times, the driving force is security related: for example, an organization may make a change because the organization must resolve a vulnerability. Transport Layer Security version 1.3 (TLSv1.3) is described as a solution that brings both performance and security improvements with minimal risk of negative impact. This research will investigate the performance impact in enabling TLS 1.3 when compared to TLS 1.2 across commonly used TLS libraries. Additionally, this research will measure the current adoption rate among various industry sectors. The results of this research and related experiments will provide evidence related to implementing TLS 1.3 in an environment.

1.1. An overview of TLS

Transport Layer Security (TLS) is a widely adopted security protocol designed to facilitate privacy and data security for communications over the internet ("Transport Layer Security TLS", n.d.). TLS is the successor to SSL; SSL Version 1 was first developed by Netscape in 1995 but was never released because it was riddled with serious security flaws (Onelski, 2020). SSL Version 2 was the first official version of SSL released in late 1995, and SSL Version 3 was released in 1996 (later documented under RFC6101). TLS was first released as TLS 1.0 in January 1999 under RFC2246 (IETF) – TLS 1.0 was an upgrade from SSL Version 3 and the differences were not dramatic (wolfSSL, "Differences between SSL and TLS Protocol Versions (#TLS13)", 2018). New versions have been released since then: TLS 1.1 released in April 2006 under RFC2246, TLS 1.2 released in August 2008 under RFC5246, and finally, TLS 1.3 released in August 2018 under RFC8446. TLS protects data between two endpoints by providing a method to set up an encrypted, authenticated, and integrity-checked communication channel.

Ben Weber, ben.weber@student.sans.edu

1.2. TLS Cipher Suites

All versions of TLS leverage cipher suites, which describe how the encrypted channel is initiated. The cipher suite is split into four main sections: the algorithm used for key exchange, the algorithm used for authentication, the algorithm used for bulk encryption, and the algorithm used for hashing. In the example 'ECDHE_ECDSA_WITH_AES_128_GCM_SHA256', ECDHE describes the key exchange, ECDSA describes the authentication, AES_128_GCM describes the bulk authentication and SHA256 for hashing. Various cipher suites are considered depreciated due to weaknesses in the cipher – for example by using weak encryption or hashing. The TLS ciphers used can directly impact the confidentiality and integrity of an encrypted communication.

1.3. Perfect Forward Secrecy

Perfect Forward Secrecy (PFS) describes a configuration of TLS, which prevents previous TLS sessions from being decrypted should the server's private key ever be compromised (Enabling Perfect Forward Secrecy, n.d.). If RSA is the key exchange algorithm, during the session negotiation, a link is created between the servers' key pair and the session key generated for each unique session (Enabling Perfect Forward Secrecy, n.d.). Thus, if an attacker is ever able to get hold of the server's private key, they can decrypt your TLS session and any saved TLS sessions (Enabling Perfect Forward Secrecy, n.d.). In contrast, if PFS is enabled, the link between the servers' private key and the session key is broken, making it impossible to decrypt prior TLS sessions (Enabling Perfect Forward Secrecy, n.d.). To enable Perfect Forward Secrecy, you must do the following: Reorder your cipher suites to place the ECDHE (Elliptic Curve Diffie-Hellman) suites at the top of the list, followed by the DHE (Diffie-Hellman) suites (Enabling Perfect Forward Secrecy, n.d.). The downside to PFS is that traditional intrusion detection solutions will not be able to decrypt traffic with the private key of the server. Instead, the protection will need to terminate the TLS connection, inspect the traffic, and re-encrypt the traffic.

1.4. TLS version negotiation and its significance

TLS version negotiation describes a step within the TLS session setup process which compares supported TLS versions and ciphers on both client and server to determine what versions are supported by both. For example, if a web browser supports TLS 1.1 and TLS 1.2 and the webserver supports TLS 1.3 and TLS 1.2, the protocol will 'negotiate' and use TLS 1.2 (that is, the highest supported version that both web servers have in common). The same process happens with TLS Ciphers. This process can be seen below in Figure 1.4.1 in the first two communications between client and server (called the handshake).



Figure 1.4.1: The TLS Handshake Process (Kemmerer, 2015)

TLS negotiation is crucial because it allows web servers to support the "Latest and Greatest" TLS version and ciphers even if all clients do not support them yet. Clients will negotiate down to the best supported TLS version and cipher until they can support a better combination.

5

2. Comparison of TLS 1.2 and TLS 1.3

2.1. TLS 1.2 - Session Setup Process

TLS 1.2 leverages two round trips to set up the TLS session. The first round trip (Steps 1 and 2 from Figure 2.1.1) is responsible for negotiating a TLS version and cipher, transferring the server certificate to the client, and transferring the "Server Random" used during key derivation ("What Happens in a TLS Handshake?," n.d.). In the second round trip (Steps 3 and 4 from Figure 2.1.1), the client uses the cipher suite to generate the pre-master secret and sends the pre-master secret back to the server. The server then acknowledges the key back to the client. Now the TLS tunnel is completed, and application traffic can flow (in this example, HTTP).







2.2. TLS 1.3 – Session Setup Process

TLS 1.3 differs from TLS 1.2 in that TLS 1.3 requires only one round trip to set up the TLS tunnel (as seen in Steps 1 and 2 of Figure 3). The version negotiation, cipher negotiation, and key sharing are completed in a single round trip meaning the setup time is reduced by a full round trip (2 trips for TLS 1.2, 1 trip for TLS 1.3). This is a primary use case for enabling TLS 1.3 as it should be more performant. Test 1 below will test this claim.

Figure 3 ("What's new in TLS 1.3?," n.d.)

2.3. Cipher Support

TLS 1.3 (Full Handshake)

An important difference between TLS 1.2 and TLS 1.3 is the volume of supported cipher suites. In almost 12 years of support, TLS 1.2 has varying types of key exchange, authentication, and hashing algorithms; some are deprecated due to vulnerabilities and some are deprecated due to weak algorithms in today's standards. In contrast, TLS 1.3 supports only five ciphers all of which enable PFS and strong protections.

2.4. Browser Support

Below is a table of common web browsers and if/when TLS 1.2 and TLS 1.3 are supported.

	Google Chrome	Internet Explorer	Firefox	Safari
TLS 1.2	v29	v11	v27+	v7+
TLS 1.3	v70	Not supported	v63	v12.1+ on
		as of		OSX 10.14
		publication.		+and above ¹

1 TLS 1.3 support is disabled by default and must be manually enabled.

2.5. Common Webserver Support

Below is a table of common webservers and if/when TLS 1.2 and TLS 1.3 are supported.

	NGINX	Apache	IIS
TLS 1.2	Release 13+	V2.2+	V7.5+
TLS 1.3	Release 17+	V2.4+	Not supported as of publication.

It is important to note the versions listed above represent out-of-the-box support for TLS 1.2 and TLS 1.3. It is possible to manually recompile NGINX and Apache to support TLS 1.3 with possible mixed results.

2.6. Common TLS library support

Below is a table of common TLS libraries and if/when TLS 1.2 and TLS 1.3 are supported.

	OpenSSL	GnuTLS	NSS	LibreSSL	SCHANNEL
TLS 1.2	v1.0.1+	v1.7.0+	v3.15.1+	Supported at	Windows
				release	2008 SP2+
TLS 1.3	v1.1.1+	3.5.x+	vV3.29.0+	V3.1.1 (client	Not
				only)	supported
					as of
					publication.

3. Lab Setup

3.1. Setup for Test 1 – TLS 1.2 vs. TLS 1.3 session speed

A major difference in TLS 1.2 vs. TLS 1.3 is the difference in the session setup

process – As Section 2.1 and 2.2 explained, TLS 1.3 has an advantage due to the one

less round trip needed. In Test 1, TLS 1.2 and TLS 1.3 will be compared to measure how many TLS Sessions can be set up per second. To do this, three machines were used:

- Machine 1: The client
 - o 4 CPU, 8GB Ram
 - o CentOS v8.1.1911. Kernel 4.18.0-147.8.1.el8_1.x86_64
 - Openssl version 1.1.1c FIPS
 - o LibreSSL 3.1.2
- Machine 2: TLS Library Host
 - 4 CPU, 8GB Ram
 - o CentOS v8.1.1911. Kernel 4.18.0-147.8.1.el8_1.x86_64
 - o OpenSSL 1.1.1c FIPS
 - GNUTls v3.6.8

The host for these 3 virtual machines is a Dell R620. For capturing the speed, we will use the 's_time' function of openssl and LibreSSL: "openssl s_time -connect \$ip-address\$:443 -new". There will be eight datasets captured in total:

- 1.) Openssl connecting to openssl over TLS1.3 using TLS_AES_256_GCM_SHA384
- 2.) Openssl connecting to openssl over TLS1.2 using ECDHE-RSA-AES256-GCM-SHA384
- 3.) Openssl connecting to GnuTLS over TLS1.3 using TLS_AES_256_GCM_SHA384
- 4.) Openssl connecting to GnuTLS over TLS1.2 using ECDHE-RSA-AES256-GCM-SHA384
- 5.) LibreSSL connecting to openssl over TLS1.3 using TLS_AES_256_GCM_SHA384
- 6.) LibreSSL connecting to openssl over TLS1.2 using ECDHE-RSA-AES256-GCM-SHA384
- 7.) LibreSSL connecting to GnuTLS over TLS1.3 using TLS_AES_256_GCM_SHA384

8.) LibreSSL connecting to GnuTls over TLS1.2 using ECDHE-RSA-AES256-GCM-SHA384

Each dataset consists of 500 datapoints on the average number of TLS sessions setup in a 30-second period. At the time of testing, LibreSSL did not have a method to stand up a simple testing server like Openssl and GnuTls so LibreSSL was excluded from server-side testing.

3.2. Setup for Test 2 – Simulating a stolen private key and testing PFS configuration

Test 2 will simulate a stolen private key to test PFS in both a weak TLS 1.2 configuration and a default TLS 1.3 configuration. For this test, we used Machine 2 from Test 1 and configured NGINX to use a cipher that does not leverage PFS (TLS_RSA_WITH_AES_256_GCM_SHA384) with TLS 1.2. Using the attack machine, HTTP over TLS traffic was generated and captured using Wireshark. After loading the private key into Wireshark, we checked to see if only the private key could be used to decrypt the traffic. A second sample was performed using a default configuration with Nginx and TLS 1.3 and results are compared.

3.3. Setup for Test 3 – Scanning sites for TLS support

During Test 3 information will be captured to document the current adoption rate of TLS 1.3 in various business sectors. This will be accomplished using the top 500 websites from Amazon's Alexa service in the Business, Retail, Technology, News, and "Kids and Teens" sections. With the lists, Qualys' SSL Labs API was used to scan each site for supported TLS protocols and exported the findings. The script ran on a Dell XPS15 laptop.

4. Analysis

4.1. Test 1 Analysis, Graphs, and key takeaways

This test will measure the speed with which a TLS session can be set up in TLS 1.2 and TLS 1.3 using three major TLS libraries. Looking back at the comparison

between session setup in TLS 1.2 and TLS 1.3 above (2.1 and 2.2 respectively), the expected outcome is that TLS 1.3 is faster than TLS 1.2 by a factor of the latency between client and server because of the reduced round trips needed.

4.1.1. Libressl connecting to GnuTLS

During Test 1, Libressl is used to connect to GnuTls. Using TLS1.2 shown in Figure 4.1.1.1, the average number of connections per second is 39.5, with a standard deviation of 2.09. Figure 4.1.1.2 shows when TLS1.3 is used, the average increases to 43.3, and the standard deviation is reduced to 1.69. Comparing the two, the average number of sessions per second increased by 9.62%, and the standard deviation reduced by 19.14%. In other words, **TLS1.3 is 9.62% faster on average than TLS1.2 and 19.14% more consistent using Libressl and GnuTls.**



Figure 4.1.1.1

Ben Weber, ben.weber@student.sans.edu



Figure 4.1.1.2

4.1.2. LibressI connecting to OpenssI

In the second set of Test 1, Libressl is used to connect to GnuTls. Using TLS1.2, the average number of connections per second is 72.8, with a standard deviation of 2.59. These results are shown in Figure 4.1.2.1. Figure 4.1.2.2 represents when TLS1.3 is used, the average increases to 76.6, with a standard deviation of 1.59. This results in a 5.22% increase in speed and a 38.61% decrease in standard deviation. Using LibreSSL and Openssl, TLS1.3 is 5.22% faster and 38.61% more consistent than TLS1.2.







Figure 4.1.2.2





Libressl to Openssl using TLS1.3

4.1.3. Openssl connecting to GnuTLS

When using Openssl to connect to GnuTLS, TLS1.2 has an average of 41.9 and a standard deviation of 2.71 (Figure 4.1.3.1) compared to an average of 45.3 and a standard deviation of 2.02 in TLS1.3 (Figure 4.1.3.2). In this dataset, TLS1.3 is 8.11% faster and 25.46% more consistent than TLS1.2.



Figure 4.1.3.1





4.1.4. Openssl connecting to Openssl

Using Openssl to connect to Openssl, TLS1.2 has an average of 76.4 and a standard deviation of 2.64 (Figure 4.1.4.1) compared to 80.8 and 2.35, respectively in TLS1.3 (Figure 4.1.4.2). **TLS1.3 is 5.76% faster and 10.98% more consistent in this case.**







Ben Weber, ben.weber@student.sans.edu

4.2. Test 2 Analysis, Graphs, and key takeaways

Test 2 simulates a compromised private key in both a PFS and non-PFS configuration. In Figure 4.2.1 below represents a compromised private key, and a server configured with TLS1.2 without PFS ciphers enabled. In this example, an adversary was able to capture network traffic between a client and a webserver. The image shows Wireshark loaded with the private key of the server. Once loaded, the traffic is decrypted and the HTTP verbs and resources requested are seen in plaintext.



When a similar simulation is performed with TLS1.3 instead of TLS1.2 in Figure 4.2.2, Wireshark is unable to decrypt the traffic. This is because all ciphers supported in TLS 1.3 enable PFS by default.

File Edit	View Go Capture Ar	nalyze Statistics Tele	ephony Tools Internals Help		Capturing from v	lp2s0 (host 192.168.1.20)				0
• •	🛋 🔳 🔬 🕫	🗋 × C C	2 ← → ∩ ⊼ ±		• 🖭 🙀 🕅	• 🗃				
Filter:			▼ Expression Clear	r Apply Save						
No. S	Source	Destination	Protocol Request Method	d Cipher Suite		Info				
1 1	0.0.3.23	192.168.1.20	TCP			55176 - 443 [SYN] Seq=0 Win=6424	0 Len=0 MSS=1460 SACK	PERM=1 TSval=3429558415 T	Secr=0 WS=128	
2 1	.92.168.1.20	10.0.3.23	TCP			443 → 55176 [SYN, ACK] Seq=0 Ack	=1 Win=28960 Len=0 MSS	S=1460 SACK_PERM=1 TSval=3	170807557 TSecr	=3429558415 WS=1
3 1	0.0.3.23	192.168.1.20	TCP			55176 → 443 [ACK] Seq=1 Ack=1 Wi	n=64256 Len=0 TSval=34	29558428 TSecr=3170807557		
4 1	.0.0.3.23	192.168.1.20	TLSv1.3	TLS_AES_256_GCM_SH	A384,TLS_CHACHA20_POL	Y1305_SHA25 Client Hello				
5 1	.92.168.1.20	10.0.3.23	TCP			443 → 55176 [ACK] Seq=1 Ack=518	Win=30080 Len=0 TSval=	=3170807583 TSecr=34295584	40	
0 1	92.108.1.20	10.0.3.23	11591.3	TLS_AES_200_GCM_SH	A384	Server Hello, Change Cipher Spec	, Application Data, Ap	optication Data, Applicati	on Data, Applic	ation Data
/ 1	0.0.3.23	192.108.1.20	TLCv1 2			SS1/0 → 445 [ACK] SEQ=S10 ACK=14 Change Cipher Cost Application	45 WIN=04126 Len=0 151	/at=3429556453 15ec1=31/00	07230	
9 1	0.0.3.23	192.100.1.20	TISVI 3			Application Data	Vata			
10-1	92 168 1 26	10 0 3 23	TCP			443 a 55176 (ack) Sena1445 arka6	90 Win-30080 Len-0 TS		98494	
1 1	92.168.1.20	10.0.3.23	TLSv1.3			Application Data				
12 1	0.0.3.23	192.168.1.20	TCP			55176 → 443 [ACK] Seg=696 Ack=17	16 Win=64128 Len=0 TSV	/al=3429558458 TSecr=31708	07597	
18 1	92.168.1.20	10.0.3.23	TLSv1.3			Application Data				
14 1	0.0.3.23	192.168.1.20	TCP			55176 → 443 [ACK] Seq=696 Ack=19	87 Win=64128 Len=0 TSV	/al=3429558458 TSecr=31708	07597	
15 1	92.168.1.20	10.0.3.23	TLSv1.3			Application Data				
15 1	0.0.3.23	192.168.1.20	TCP			55176 → 443 [ACK] Seg=696 Ack=28	59 Win=64128 Len=0 TS	/al=3429558458 TSecr=31708	07597	
19-1	0.0.3.23	192.168.1.29	TLSv1.3			Application Data		CCI Deservely DeeGI	a Defeult	
19 1	9.9.3.23 9.0.3.23 92.168.1.20	192.168.1.20 192.168.1.20 10.0.3.23	TLSv1.3 TCP TCP			Application Data 55176 - 443 [FIN, ACK] Seq=720 A 443 - 55176 [FIN ACK] Seq=2859	ck=2859 Ack=728	SSL Decrypt - Profil	e: Default	0
1 <mark>7 1</mark> 18 1 19 1	0.0.3.23 0.0.3.23 92 168 1 28	192.168.1.20 192.168.1.20 10 A 3.23	TLSv1.3 TCP TCP			Application Data 55176 - 443 [FIN, ACK] Seq#720 A 443 - 55176 [FIN ACK] Seq=2859	ck=2859 Ack=728	SSL Decrypt - Profil	e: Default	- × ×
1 <mark>7 1</mark> 18 1 19 1	0.0.3.23 0.0.3.23 92.168.1.20	192.168.1.20 192.168.1.20 10 A 3.23	TLSv1.3 TCP TCP Wireshark: Preferences - Pr	rofile: Default		Application Data 55176 - 443 [FIN, ACK] Seq=720 A 443 - 55176 [FIN ACK] Seq=2859 - 8 O	ck=2859 Ark=728	SSL Decrypt - Profil	e: Default	- V O
10-1	0.0.3.23 0.0.3.23 92.168.1.20	192.168.1.20 192.168.1.20 .10.0.3.23	TLSv1.3 TCP TCP Wireshark: Preferences - Pr	rofile: Default		Application Data 55176 - 443 [FIN, ACK] Seq=720 A 443 - 55176 [FIN, ACK] Sen=7859 - 0	ck=2859 Ack=724	SSL Decrypt - Profil IP address Port Protoco 192.168.1.20 443 http	e: Default Key File /tmp/nginx-ss.key	- × O
10-1 18 1 19 1 	0.0.3.23 0.0.3.23 92.168.1.28	192.168.1.20 192.168.1.20 - 10.0.3.23	TLSv1+3 TCP TCP Wireshark: Preferences - Pr	rofile: Default RSA keys list:		Application Data 55176 - 443 [EN. ACK] Seq=720 A 443 - 55176 [ETN ACK] Seq=7260 - * © de_	ck=2859 Ark=77A Up Down	SSL Decrypt - Profil IP address Port Protoco 192.168.1.20 443 http	e: Default Key File /tmp/nginx-ss.key	Password
10-1 18 1 19 1 SMU2 SNA	0.0.3.23 .0.0.3.23 92 168 1 20	192.168.1.20 192.168.1.20 10.0.3.23	TLSv1.3 TCP TCP Wireshark: Preferences - Pr	rofile: Default RSA keys list:	E	Application Data 55176 - 443 [FIN, ACK] Seq=720 A 443 - 55176 [FIN ACK] Seq=726 A - * O dt	ck=2859 Ark=720 Up Down	SSL Decrypt - Profil IP address Port Protoco 192.168.1.20 443 http	e: Default Key File /tmp/nginx-ss.key	Password
10-1 18 1 19 1 SMU2 SNA SNA	0.0.3.23 0.0.3.23 92 168 1 20	192.168.1.20 192.168.1.20 18.6.3.23	TLSV1.3 TCP TCP Wireshark: Preferences - Pr	rofile: Default RSA keys list: SSL debug file:	E	Application Data 55176 - 443 [EH] AKI Sem-720 A 44% 55176 JEFN ArKI Sem-2850 dt	ck#2859 Ark=720 Up Down	SSL Decrypt - Profil IP address Port Protoco 192.168.1.20 443 http	e: Default Key File /tmp/nginx-ss.key	Password
10-1 18 1 19 1 SMU2 SNA SNA SNA SNA	0.0.3.23 00.0.3.23 09.168.1.20 X	192.168.1.20 192.168.1.20 19.0.3.23	TLSv1.2 TCP TCP Wireshark: Preferences - Pr	rofile: Default RSA keys list: SSL debug file:	Ε	Aptication bata 53/17 - 443 [FBI, ACK] See728 A 433 - 5177 JFBI ACK] See728 A - * • dRBrows	ck=2859 Ark=730 Up Down New Edlt	SSL Decrypt - Profil IP address Port Protoco 192.168.120 443 http	e: Default Key File /tmp/nginx-ss.key	Password
10-1 18 1 19 1 SMU2 SNA SNA SNA SNA SNA SNA SNA	0.0.3.23 0.0.3.23 92.168.1.20 X P t t	192,168,1,20 192,168,1,20 19,8,3,23	TLEV1,3 TCP Wireshark: Preferences - Pr Reassemble 55L records spanning r	rofile: Default RSA keys list: SSL debug file: multiple TCP segments:	3	Application beta	ck=2859	SSL Decrypt - Profil IP address Port Protoco 192.168.1.20 443 http	e: Default Key File /tmp/nginx-ss.key	- × C
10-1 18 1 19 1 SMU2 SNA SNME Snort Socks Solar	0.0.3.23 00.0.3.23 02.168.1.20 X X P t t t t t t t t t t t ge	<u>192,168,1,20</u> 192,168,1,20 <u>10,4,3,23</u>	TLSV1.3 TCP Wireshark: Preferences - Pr Reassemble 55L records spanning r	rofile: Default RSA keys list: SSL debug file: multiple TCP segments: g multiple SSL records:	2 2	Aptication Data 55176 - 445 [FB], ACK Seqr220 A 443 - 5417 [FB], ACK Seqr220 A 643 - 5417 [FB], ACK Seqr220 A 642 - 645 -	Ck=2855 Ark=779 Down Down Edk Copy	SSL Decrypt - Profil IP address Port Protoco 192.168.120 443 http	e: Default K ey File /tmp/nginx-ss.key	- × C
10-1 18 1 19 1 SMU2 SNA SNME Snort Socks Solar SoulS	0.0.3.23 00.0.3.23 02.168.1.20 X X P t t 5 5 5 60 6 6 6 6 8	<u>192,168,1,20</u> 192,168,1,20 19 A 3 23	TLSV1.3 TCP TCP Wireshark: Preferences - Pr Reassemble 55L records spanning r sesemble 55L records spanning r Mescape Authentization Code Mar	rofile: Default RSA keys list: SSL debug file: multiple SSL records: g multiple SSL records:	2 2 2	Application beta 5570 - 443 [FB], ACK] Segr20 A 4570 - 443 [FB], ACK] Segr20 A 68. 0rowse.	Chr2855 Ark=736 Up Down Edk Edk Copy Delete	SSL Decrypt - Profil Paddress Port Protoco 192.166.1.20 443 http	e: Default Key File /tmp/nginx-ss.key	- V C
10-1 18 1 19 1 SMUD SNA SNA SNA SNA SNA SNA SNA SNA	0.0.3.23 00.3.23 97 168 1 28 X P t t t t t t t t t t t t t t t t t t	<u>192,168,1,20</u> 192,168,1,20 <u>18,8,3,23</u>	TLSV1.3 TCP TCP Wireshark: Preferences - Pr Reassemble 55L records spanning or ssemble 55L Application Data spannin Message Authentication Code (MA	rofile: Default RSA keys list: SSL debug file: multiple TCP segments: g multiple SSL records: g multiple SSL records: (C), ignore "mac failed":	3	Aptication Data 55176 - 445 [FH], ACK Seq-720 A 441 - 51776 [FH] ACK Seq-720 A 641 - 5176 [FH] ACK Seq-720 A 642 - 5176 [FH] ACK Seq-720 A 642 - 5176 [FH] ACK Seq-720 A Brown.	cke-2855 Ark=270 Down Edit_ Copy Delete	SSL Decrypt - Profil	e: Default Key File /tmp/nginx-ss.key	- · C
10-1 18 1 19 1 SMUX SNA SNME Snort Socks Solari SoulS Soup SPDY	0.0.3.23 0.0.3.23 92 168 1.20 X P t t 5 5 Edge Seek ØinTCP	<u>192,168,1,20</u> 192,168,1,20 <u>18,8,2,23</u>	TLEV1, 3 TCP TPP Wireshark: Preferences - Pr Reassemble SSL records spanning r ssemble SSL Apolication Data spannin Message Authentication Code (MA	RSA keys list: RSA keys list: SSL debug file: multiple FCP segments: g on ultiple SSL records: G on Jonce "mac failed": Pre-Shared-Key:	3	Application Data 55170 - 443 [FB], ACK] Seg-720 A 437 - 443 [FB], ACK] Seg-720 A 68. 010098.	CK+2855 Aek=726 Up Down Edit. Copy Delete Befresh	SSL Decryst - Profil	e: Default Key File /tmp/nginx-ss.key	Password
10-1 18 1 19 1 SMUD SNA SNA SNA SNA SOUS Sour SOUS SOUS SOUS	0.0.3.23 00.0.3.23 92 168 1.28 X P E E S E Goge Beek HinTCP	102.106.1.20 102.106.1.20 10 A 3 23	T, SAVI - 3 TCP TCP TCP TCP TCP TCP TCP TCP TCP TCP	RSA keys list: RSA keys list: SSL debug file: Ing multiple TCP segments: Ing multiple SSL records: Pre-Shared-Key: Pre-Shared-Key:	2 2	Aptication bata 55176 - 445 [FB], ACK Seqr220 A 441 - 5176 [FB], ACK Seqr220 A 642 - 5176 [FB] APK 500-2850 082. Brown.	ck=2855 Ark=220 Up Down Edit Copy Delete Refresh	551. Decryst - Profil	e: Default Key File /tmp/nginx-ss.key	- C
10-1 18 1 19 1 SMUX SNA SNMF Snort Souls Souls Souls Souls SPDY Spice SPRT	0.0.3.23 00.0.3.23 02.168.1.26 22.168.1.26 23.168.1.26 24.168.1.26	192,166,1.20 192,166,1.20 16 A 3 23	T, SAY, J. 2 TCP TCP TCP Wireshaft: Preferences - Pre Reasemble 55, records spanning rp sesemble 55, records spanning rn Kessage Authentication Code (MA	rofile: Default R5A keys list: S54 keys list: S54 keys list: multiple TCP segments: multiple TCP segments: multiple S24 keys: (c), ignore "mac failed": Pre-Shared-Key: ker-Secret log filename:	2	Aprication Data 55176 - 443 [FB], ACK] Segr220 A 457 - 443 [FB], ACK] Segr220 A 68. Browse. Browse.	Arka7285 Arka7298 Up Down Edit_ Copy Delete Refresh Clear	SSL Decypt- Profil	e: Default Key File /tmp/nginx-ss.key	Password
10-1 19 1 SMUX SNA SNMF Socks Solar SoulS Soup SPDY Spice SPRT SRVLI	0.0.3.23 0.3.23 0.3.168.1.28 V V P t t 5 5 5 6 0 6 0 C 0 C	192.166.1.20 192.166.1.20 18 A.3.23	T, SAVI - 3 TCP TCP TCP TCP Writeshark: Perferences - Per Reasemble 55, records spanning Reasemble 55, records spanning Message Authentication Code (MA	rofile: Default RSA keys list: SSL debug file: gmultiple TCP segments: gmultiple SSL records: GL jagnee "mac failed"; Pre-Shared-Key: ter-Secret log filename;	2	Aptication Data 55176 - 445 [FB], ACK Seqr220 A 441 - 51776 [FBH ACK] Seqr220 A 642 - 5176 [FBH ACK] Seqr220 A 642 - 5176 [FBH ACK] Sequences 642 - 5176 [FBH ACK] Sequences Browse.	Cke2855 Arke779 Up Down New Ede. Copy Delete Refresh Clear	551. Decryst - Profil	e: Default Key File /tmp/nginx-ss.key	Password
10-1 10-1 10-1 SMUX SNM SNM SNM SNM SNM SNM SNM SNM	0.0.3.23 00.3.23 00.164.1.26 X P E E E E E E E E E E E E E E E E E E	192.196.1.20 192.196.1.20 10 0.3.23	T, SAY, J. 2 TCP TCP TCP Wireshaft: Preferences - Pr Reasoning of SS, records spanning or sesemble SS, Application Data spannin Message Authentication Code (MA	rofile: Default RSA keys list: SSL debug file: Inutlipile TCP segments: Ig multipile SSL records: Ig multipile SSL records: Oc Jonore "mac Railed": Pre-Shared Key: tere Secret log filename:	2	Aprication Data 55176 - 443 [FB], ACK] Segr220 A 44% - K17 [FB], ACK] Segr220 A 60 Brown Brown	LK02255 Arke2256 Down C.GL Copy Delete Refresh Clear	SSL Decypt- Profil	e: Default Key File /tmp/nginx-skey	Pessword
18-1 18-1 19-1 SMU2 SNA SNMF Snort Sours Sours Sours Sours SPDY Spice SPRT SRVLI Help	6.0.3.23 00.3.23 00.10.4 20 X P E E E E E E E E E E E E E E E E E E	192,168,1.26 192,168,1.20 10 A 3 73	T, SAN J 3 TCP TCP TCP TCP Writeshark: Perferences - Pit Writeshark: Perferences - Pit Ressemble 551, records spanning m sessemble 551, records spanning m Message Authentication Code (MA	RSA keys list: RSA keys list: SSL debug file: multiple TCP segments: gmtliple SSL records: (C), ignore "mac failed": Pre-Shared Key: ter-Secret log filename:	2 2 2	Aptication Data 55176 - 445 [FB], KKI See728 A 441 - 5176 [FB], KKI See728 A 682 - 0 682 - 0 Browne - 0 Browne - 0 Cancel OK	Lev2839 Arke-274 Down Down New Edit Copy Defete Refresh Clear	551. Decryst Profil	e: Default Key File /tmp/nginx-ss.key Apply Can	Password Cel OK
18-1 18-1 19-1 19-1 19-1 19-1 SMU SNU SNU SNU SNU SNU SNU SNU SN	6.0.3.23 60.3.23 80.3.148.1.28 8 8 8 6 6 6 6 6 6 6 6 0 0 0	192.196.1.20 192.196.1.20 10 0 1 23	T, SAY, J. TCP TCP TCP Wireshaft: Preferences - Pr Reassemble 55, records spanning or sesemble 55, records spanning m Kessage Authentication Code (MA (Pre)-Mast	roffie: Default RSA keys list: SSI, debug file: In granultiple SSI, records: granultiple SSI, records: Pre-Shared-Key: tere-Secret log filename:	2 2 2 2 2	Aprication Data 5575 - 445 [FB], ACK Seg-726 A 457 - 415 [FB], ACK Seg-726 A 68	Alkatha Alkatha Down Cdla Copy Delete Refresh Clear	SSL Decypt- Profil	e: Default t Key File /tmp/nginx-ss.key Apply Can	Password Cel OK

It is important to note that TLS 1.2 can be configured to only support PFS-enabled ciphers. However, this leads to the possibility of human error or configuration drift, allowing non-PFS ciphers to be enabled. If PFS is a requirement for an environment and strong configuration management is not in place, TLS 1.3 would be beneficial.

4.3. Test 3 Analysis, Graphs, and key takeaways

After determining that TLS 1.3 provides the benefit of speed of TLS 1.2 in Test 1, and the benefit of security in Test 2, the purpose of Test 3 is to measure the adoption rate of TLS 1.3.

4.3.1. Highest version of TLS Support

Figures 4.3.1.1 through 4.3.1.5 below represent the population of 500 websites in each category and what the highest TLS version enabled on the webserver.





In Figure 4.3.1.1, the Kids and Teens category has 6.7% without any TLS enabled and almost 3% with TLS 1.0. TLS 1.2 is strongly represented, which is no surprise TLS 1.2 the highest adoption rate of any TLS version according to SSLLabs ("SSL Pulse", n.d.). TLS 1.3 has a surprisingly high adoption rate in this category at 34.3%.

20





The Science category results in Figure 4.3.1.2 have the second lowest number of websites supporting no TLS or weak TLS (TLS 1.0) configurations at 7.1%. It is also noteworthy that 0% of the websites tested offer TLS 1.1 as the highest version. TLS 1.2 with 64.6% and TLS 1.3 with 28.4% show strong adoption rates for modern TLS libraries.





The Shopping category has positive results with low numbers in TLS 1.0 and TLS 1.1 - 92.4% of websites tested in this category support either TLS 1.2 or TLS 1.3 as the highest version.





Ben Weber, ben.weber@student.sans.edu

The Computers category has the best overall score, with 0% of the websites tested supporting TLS 1.0 or TLS 1.1 as the highest version. 98.2% support either TLS 1.2 or TLS 1.3 as the highest version.





The Business category has the lowest adoption rate of TLS 1.3 across the five categories of sites tested. However, it also has the largest population of websites supporting TLS 1.2 as the highest version and low numbers for both TLS 1.0 and TLS 1.1.

Further testing in this area could include adding datapoints for the websites tested which have various audit requirements. For example, do companies with PCI environments have better adoption of modern TLS protocols and less support for older protocols? Additionally, it would be significant to measure the population of websites tested which use a hosting provider like Cloudflare. Is there a trend of websites with good TLS 1.3 adoption rates and usage of hosting providers? Do 'selfmanaged' websites have the same adoption rate as 'Third-Party Managed' sites? Answering these questions would help understand if the adoption rates are the result of many different companies following TLS best practices, or if the adoption

Ben Weber, ben.weber@student.sans.edu

rates are the result of many different companies using a third-party who follows TLS best practices.

4.3.2. Lowest Version of TLS Support

The second part Test 3 analyzes the same data used in the first part of Test 3 and extrapolates the lowest version of TLS offered by the domain. In this part of the test, very low numbers are expected for TLS 1.3 due to the unlikeliness that it is the only TLS version offered by a domain. However, it is interesting to compare the deprecation rate of TLS 1.0 and TLS 1.1 across sectors.

There are many reasons to disable deprecated TLS versions – many of which include confidentiality vulnerabilities. The IETF made very clear recommendations in March of 2019: "TLSv1.0 MUST NOT be used. Negotiation of TLSv1.0 from any version of TLS MUST NOT be permitted" ("Deprecating TLSv1.0 and TLSv1.1", 2019) and "TLSv1.1 MUST NOT be used. Negotiation of TLSv1.1 from any version of TLS MUST NOT be used. Negotiation of TLSv1.1 from any version of TLS MUST NOT be used. Negotiation of TLSv1.1 from any version of TLS MUST NOT be used. Negotiation of TLSv1.1 from any version of TLS MUST NOT be permitted." ("Deprecating TLSv1.0 and TLSv1.1", 2019). NIST took a similar stance in its SP 800-52 Rev 2: "Guidelines for TLS Implementations": "Servers that support citizen or business-facing applications ... shall be configured to negotiate TLSv1.2 and should be configured to negotiate TLS 1.3. The use of TLS version 1.1 and 1.0 is generally discouraged" (McKay & Cooper, "Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations").





In Figure 4.3.2.1, 51.4% of domains tested in the Kids and Teens category still support TLS 1.0. 11% of the population have removed support for TLS 1.0 but support TLS 1.1.





Ben Weber, ben.weber@student.sans.edu

In the Science category, almost 54% still support TLS 1.0, with an additional 11.3% still supporting TLS 1.1.





The Shopping category has the best results of all five categories in this test, with only 28.1% supporting TLS 1.0 and 21.8% supporting TLS 1.1. 42.7% of the domains tested have taken the guidance from IETF and NIST to disable TLS 1.0 and TLS 1.1.



The Computers category has the worst results in this test, with 58.5% still supporting TLS 1.0 or TLS 1.1. The computer category has the lowest population of domains without TLS enabled, but the largest population of poor TLS configuration, which was an unexpected result.

27



Finally, the Business category has approximately 54% of domains supporting deprecated protocols and only 37% of the population following IETF and NIST recommendations.

5. Conclusion

The purpose of this research is to provide evidence supporting or refuting the benefits of enabling TLS 1.3. In Test 1, TLS 1.3 was proven to be between 5%-9% faster on TLS session setup time and between 11%-38% more consistent in session setup time when compared to TLS 1.2. These speed-related benefits may not convince low-volume services to enable TLS 1.3 support, however large-volume services and CDNs would likely benefit by adopting TLS 1.3. Additionally, we found the fastest results when openssl is used as the server-side TLS library. Test 2 showed the benefits of TLS 1.3 when PFS is considered valuable to an organization without strong configuration drift management. Finally, Test 3 showed the number of popular high-volume websites that have seen the benefits of enabling TLS 1.3 and

Ben Weber, ben.weber@student.sans.edu

those who have not. Due to the low risk of enabling TLS 1.3 and the benefits illustrated, the recommendation is to configure TLS 1.3 to garner the speed and security benefits but to keep TLS 1.2 enabled for backward compatibility. Security .e. .nd high practitioners who enable TLS 1.3 will gain the above benefits, take steps towards removing support for legacy protocols, and highlight the use of newer, better

- CloudFlare.net. (n.d.). Transport Layer Security TLS. Retrieved May 14, 2020, from https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/
- Dierks, T., & Allen, C. (1999, January). The TLS Protocol Version 1.0. Retrieved May 14, 2020, from <u>https://tools.ietf.org/html/rfc2246</u>
- Enabling Perfect Forward Secrecy. (n.d.). Retrieved May 15, 2020, from <u>https://www.digicert.com/kb/ssl-support/ssl-enabling-perfect-forward-secrecy.htm</u>
- Kemmerer, C. (2015, March 3). The SSL/TLS Handshake: an Overview. Retrieved May 16, 2020, from <u>https://www.ssl.com/article/ssl-tls-handshake-overview/</u>
- (n.d.). Retrieved May 16, 2020, from https://knowledge.digicert.com/generalinformation/INF03299.html#Chro me
- What's new in TLS 1.3? (n.d.). Retrieved May 16, 2020, from https://www.cloudflare.com/learning-resources/tls-1-3/
- What Happens in a TLS Handshake? (n.d.). Retrieved May 16, 2020, from <u>https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/</u>

SSL Pulse. (n.d.). Retrieved May 26, 2020, from https://www.ssllabs.com/ssl-pulse/

- Deprecating TLSv1.0 and TLSv1.1. (n.d.). Retrieved May 29, 2020, from <u>https://tools.ietf.org/id/draft-ietf-tls-oldversions-deprecate-</u>02.html#rfc.section.4
- McKay, K. A., & Cooper, D. A. (n.d.). Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations. Retrieved May 29, 2020, from Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations
- Onelski, J. (2020, February 13). SSL vs TLS What's the Difference? Retrieved May 31, 2020, from <u>https://www.globalsign.com/en/blog/ssl-vs-tls-difference</u>
- wolfSSL. (2018, September 13). Differences between SSL and TLS Protocol Versions (#TLS13). Retrieved June 15, 2020, from https://www.wolfssl.com/differences-between-ssl-and-tls-protocolversions-3/