22 June 2026

TLS 1.3 - Smashing Transport Security

Overview

Secure web communications have come a long way since the 1990s. From a rocky start to implementing protections against quantum computing, SSL and TLS have a

storied history that affects everyone from the developer to the executive in C suites. The modern implementations of the TLS protocol suites are a first line of defense to many organizations' enterprise IT presence. More importantly it represents one of the strongest layers of defense your organization has to offer in protecting its crown jewels - proprietary and in some cases extremely sensitive data. In this article I will go over some of the history of TLS, why adoption remains an issue, why it's important to include this in your organization's roadmap, and what can be done to protect your organization; whether you are a small business or large government agency.

History

Before we address TLS 1.3 and its benefits, we should take a walk down memory lane and review the history of TLS and SSL. In 1994, Netscape was the dominant browser in the burgeoning world wide web. Initially, they were concerned about the need for secure online communication and concentrated on securing traffic within the browser to websites. They developed SSL 1.0 in an effort to meet this requirement.

SSL 1.0 was not released publicly due to a variety of basic security flaws including the lack of sequence numbers to protect against relay attacks and the use of insecure ciphers, such as RC4 (Oppliger, 2009). While SSL 1.0 was considered a failure, Netscape made improvements and corrections, releasing SSL 2.0 to the public in 1995. Unfortunately SSL 2.0 was plagued by its own set of security flaws. On top of the security flaws it was only able to accept a single domain certificate; making the protocol practically useless for websites being hosted as virtual web servers. (Subramanium, n.d.).

SSL 3.0 was developed to address the issues with SSL 2.0. It was released in 1996 and became the standard bearer for almost a decade. Furthermore, SSL 3.0 drastically improved its usefulness by bringing compatibility for multiple SAN (Subject Alternative Name) certificates and improved ciphers, data integrity, and man-in-the-middle protection. SSL 3.0 became the standard bearer for the better part of 10 years, then the industry began paying attention to something called a POODLE attack. It's interesting to note that it took 10 years for the industry at large to acknowledge the vulnerability.

Introduction of Transport Layer Security

By the late 1990s, the Internet Engineering Task Force (IETF) became the primary author for SSL/TLS standards. In 1999, citing significant improvements to SSL 3.0, IETF approved the TLS 1.0 standard. As opposed to previous version changes that included both security and functionality changes, TLS 1.0's major differences were all security related. The key derivation and message authentication code mechanisms were greatly improved. Hash based Message Authentication Codes (HMAC) were implemented as a

far more secure method that did not depend on a single hashed value of the message and secret key. HMAC brought about a more complex construction that involved two passes of the hashing function, along with the use of both inner and outer padding. (What is the difference between a MAC and HMAC, and how does HMAC enhance the security of MACs?, 2023)

TLS 1.1 was released in 2006, bringing about two significant changes to the standard. First, TLS 1.1 added protections against cipher-block chaining attacks by changing padding error behaviors. Secondly, notably the addition of an explicit Initialization Vector to protect against BEAST attacks. (Banach, 2024) Two years later, TLS 1.2 was released with a large number of changes including the implementation of AES ciphers and upgraded hashing algorithms.

In 2014, SSL 3.0 and TLS 1.0 and 1.1 were widely used across many environments. The publication of a successful exploitation of a POODLE attack rattled cyber security experts and IT shops around the world. The specifics of the POODLE attack involved how the protocol managed padding schemes and cipher-block chaining attacks. This issue was further exacerbated by the fact that the TLS versions of the time allowed downgrading of SSL/TLS versions to support SSL 3.0; effectively negating any and all of the security benefits of later TLS implementations. The attacker effectively instructed a TLS server that the client only supported SSL 3.0 in order to complete a padding attack. (SSL 3.0 Protocol Vulnerability and POODLE Attack, 2016)

The response to this realization resulted in a wide spread effort to remove SSL 3.0 completely in sensitive companies and government agencies. The reaction was so great that retroactive RFC updates removed backwards compatibility in all TLS versions, effectively ending any implied support for SSL 3.0. This was the first time, in SSL/TLS history, that the IETF adopted a mindset that in order to leverage new security features, backwards compatibility would be explicitly denied. The outcome was dramatic, companies and organizations with any type of web presence started moving toward TLS and away from the deprecated SSL standards, evidence of an intensified awareness toward cyber threats.

TLS 1.3

TLS 1.3 was released in 2018 as the modern implementation of transport security. We can find widespread adoption of mobile and desktop clients consuming all types of cloud services; not just web traffic but mail, file transfer and everything in between. TLS is widely considered a hard and fast requirement ahead of going to production with any type of service.

The TLS 1.3 standard, a giant leap forward in terms of security, was a response to the new cyber security challenges in 2018. caused a giant leap forward in terms of security.

This is factored into the with. First, TLS 1.3 removed a litany of older cipher suites and encryption mechanisms, not considered secure.

Granted, Artificial Intelligence and quantum computing still stood over secure computing in 2018 threatening to break through previously considered secure ciphers and encryption standards. Developers of the standard paid particular attention to this threat. Specifically, they analyzed TLS 1.2's vulnerabilities and developed the standard accordingly:

- **TLS 1.2's Vulnerability:** Quantum computers can potentially break the RSA and ECC algorithms that TLS 1.2 relies on for key exchange. This means that a quantum computer could decrypt communications that were encrypted with TLS 1.2.
- **TLS 1.3's Resilience:** TLS 1.3 uses stronger key exchange algorithms that are believed to be more resistant to quantum attacks. Additionally, TLS 1.3 has removed support for weaker algorithms and has a more streamlined handshake process, which reduces the attack surface.

Once again, a line was drawn in the sand, this time backwards compatibility for any older version of TLS was disallowed - this time data security was no longer simply a goal, it was a mandate. However, as with the previous removal of backward compatibility, organizations faced a number of challenges:

Legacy Systems: Older systems and software may not support TLS 1.3. This can lead to connection failures or fallback to older, less secure TLS versions.

Middleboxes: Some network devices, such as firewalls and load balancers, may not be configured to properly handle TLS 1.3 traffic, potentially causing disruptions.

Application Compatibility: Certain applications may have dependencies on specific TLS features or extensions that are not supported or have changed in TLS 1.3.

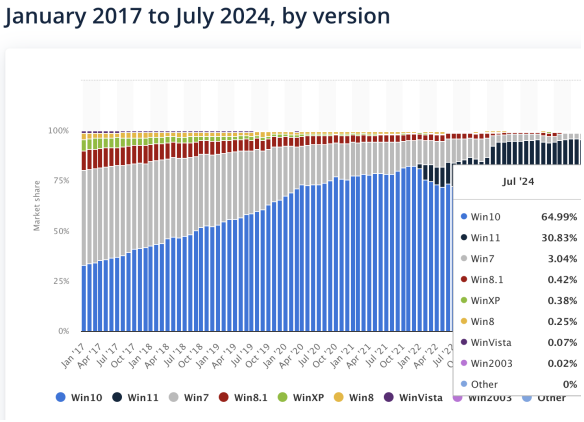
Why now?

It's mid 2026 - why are we still talking about SSL/TLS/Security practices? Unfortunately, many organizations and companies have not been able or willing to migrate to the new standard. More appalling, Alexa reports that over 20% of the sites on the internet do not use SSL or TLS, in spite of a laundry list of security concerns. (Nohe, 2018)

The lack of widespread implementation of TLS protocols is concerning and is evidence of the backwards compatibility that has been removed in the various standards. While all modern operating systems support TLS 1.0 and above; modern is still a relative term.

Windows Vista for example is only 18 years old, but doesn't support TLS 1.2. (Vera, 2024)

Achieving consistent security requirements is not only difficult for companies, but government agencies charged with a mandate to protect personal information such as PII and PHI data transmitted to requesters. Sites and services available to end users who rarely have a requirement to upgrade their personal computing devices (smart phones and personal computers). In 2024 the reported distribution of Windows 8 and older Windows operating systems, that by default supported TLS 1.2 was around 5% (Sherif, 2024)



Blame for the lack of migration to the new standard isn't solely on users who fail to migrate to the latest and greatest devices. Many end user devices run TLS 1.3 supporting operating systems. Companies and government organizations are some of the biggest drivers of adoption, first as a method of best practice but also through regulatory requirements. Many of these regulatory requirements dictate that governments and organizations must begin moving toward TLS 1.3, either implicitly or explicitly. Here in the UK, organizations are subject to broader data protection and cyber security laws that, while not as forthright in their specifics, still indirectly require the use of up-to-date and secure TLS versions:

- **Data Protection Act 2018** This Act requires organizations to implement appropriate technical and organizational measures to ensure the security of personal data. Using outdated and insecure TLS versions could be considered a failure to meet these obligations.
- **Network and Information Systems (NIS) Regulations 2018** These regulations apply to operators of essential services and digital service providers. They require organizations to take appropriate security measures to manage risks to network and information systems. This would likely include using secure TLS versions.

This is compared to those across the pond in the United States, FIPS, NIST, and OMB clearly emphasize a preference and support toward moving to TLS 1.3:

- **Federal Information Processing Standards (FIPS) Publication 140-2:** This standard specifies security requirements for cryptographic modules protecting sensitive information. It indirectly impacts TLS by requiring the use of approved cryptographic algorithms, which are often tied to specific TLS versions.
- **National Institute of Standards and Technology (NIST) Special Publications:** NIST provides guidance on various aspects of information security, including the use of TLS. NIST SP 800-52 Revision 2 specifically addresses TLS implementation and recommends using the latest TLS versions (currently 1.2 and 1.3) and disabling outdated and insecure versions (SSL 2.0, SSL 3.0, TLS 1.0, and TLS 1.1).
- **Office of Management and Budget (OMB) Circular A-130:** This circular outlines management of federal information resources and emphasizes the importance of security controls, including the use of appropriate encryption protocols like TLS.

What Can You Do

Like any good recovery program, the first step is identifying that you have a problem. Traditionally, this is a task for IT administrators, Engineers, and Cyber Security professionals; however, end users can perform this task on their own machines. Mileage will vary depending upon their specific operating system. The easiest way to identify the sites SSL version is to utilize the openssl library to interrogate their web sites and resources:

```
~/Downloads openssl s_client -connect www.plaidsecurity.co.uk:443 | grep -i protocol

Connecting to 51.159.13.88
depth=3 C=US, O=Internet Security Research Group, CN=ISRG Root X1
verify return:1
depth=2 C=US, O=ISRG, CN=Root YR
verify return:1
depth=1 C=US, O=Let's Encrypt, CN=YR1
verify return:1
depth=0 CN=www.plaidsecurity.co.uk
verify return:1
Protocol: TLSv1.3

    Protocol  : TLSv1.3
    Protocol  : TLSv1.3
```

In the above example, we can see that the Scaleway hosted website for Plaid Security Ltd is leveraging TLS 1.3. By comparison, Speakeasy only implements TLS 1.2:

```
~/repos openssl s_client -connect www.speakeasy.net:443 | grep -i protocol

Connecting to 69.50.99.219
depth=2 C=US, O=DigiCert Inc, OU=www.digicert.com, CN=DigiCert Global Root G2
verify return:1
depth=1 C=US, O=DigiCert Inc, OU=www.digicert.com, CN=RapidSSL TLS RSA CA G1
verify return:1
depth=0 CN=*.speakeasy.net
verify return:1
40889DF501000000:error:0A00018A:SSL routines:tls_process_ske_dhe:dh key too small:ssl/statem/statem_clnt.c:2315:
Protocol: TLSv1.2
    Protocol  : TLSv1.2

~/repos
```

This isn't to blame any single company or group of users, but to make more readily available the knowledge of how these security mechanisms work and how to help people improve them. TLS 1.3, for example, is preferred, but any type of encrypted communication is better than none at all.

```
~/repos openssl s_client -connect www.ilfornopizzeria.net:443 | grep -i protocol

Connecting to 199.34.228.70
40889DF501000000:error:0A000410:SSL routines:ssl3_read_bytes:ssl/tls alert handshake failure:ssl/record/rec_layer_s3.c:908:SSL alert number 40
```

Once we've identified whether or not we have a problem, it's time to look at the specific technology stack and begin engineering a resolution. In many cloud cases, this is a non-issue. Vendors such as Microsoft, Google, and AWS have already implemented and defaulted to secure protocols, such as TLS 1.3 (*API Gateway now supports TLS 1.3*, 2024). Many of our US Government customers implement web applications and services using .Net. While .Net application loads are not solely limited to Windows platforms, they are definitely more predisposed to running on top of Windows Servers. For Windows 2008 R2 and later, TLS 1.2 can be enabled by default. (Rahul, January 2024) In the vast majority of Microsoft Windows Server implementations the steps are the same to enable various TLS versions. Open registry editor and update the following registry keys:

- [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2]
- [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Client]
  "DisabledByDefault"=dword:00000000 "Enabled"=dword:00000001
- [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server]
  "DisabledByDefault"=dword:00000000 "Enabled"=dword:00000001

These keys can be rinsed and repeated for other SSL/TLS protocols (eg. SSL 1.0, 2.0, and so on depending on platform compatibility). Unfortunately, Windows operating systems, including Windows 2016, do not support TLS 1.3.

In Unix/Linux environments the prevailing web service mechanism is Nginx and Apache. Regardless of the software generally the solution is pretty straightforward. Implementation of the relevant sections within the configuration file to support the correct protocol, as shown in the below Nginx example (Vivek, 2024) (Rahul, June 2024):

ssl_protocols TLSv1.2 TLSv1.3;

SSLProtocol -all +TLSv1.2 +TLSv1.3

In many cases this might require a package upgrade that may cause dependency issues with other installed software. This is where many of our customers start to see complexity that initially slows and can even stall their progress to leveraging the latest version of TLS. In examples such as application or web servers that can't be easily updated without refactoring of applications or application components, choke points in the usage of TLS 1.3 start to develop. Consider a distributed application that depends on a Windows 2016 system to host one of the overalls applications web components. While other components such as the CDN can be built to leverage TLS 1.3 the older Windows hosted components can only be upgraded to TLS 1.2, resulting in the inability for the organization to utilize TLS 1.3 exclusively for their client machines.

In some cases, these issues can be effectively resolved by the use of proxies, Web Application Firewalls (WAF), or certain types of load balancers. In many of these examples, client SSL traffic can be terminated at the appliance or WAF. This allows components that may not be directly TLS 1.3 compatible participate in an overall application system that does support the upgraded protocol. Like web server configurations; configuring TLS 1.3 on devices, like the F5, is a relatively trivial matter. Specifically, all the device required is configuration SSL Server profiles cipher group to only support TLS 1.3. (K39580786: Configure the SSL profile to allow TLS 1.2 and 1.3 only, 2023). F5 is not the only vendor currently offering customers the ability to restrict their TLS versions to 1.3. Many such as ZScaler also have best practice articles that demonstrate how the vendor community is ready to provide mechanisms by which organizational and modernization shortcomings can be solved, at least in the short term.

It's important to recognize that while switching to TLS 1.3 is the best-case scenario, in many cases where this is simply not a solution, the strategy that IT professionals should consider centers around identification, communication and management of the problem. It's important to identify and track points of failure that can stop your organization from implementing the best security posture possible. Significantly enough this mirrors industry wide practices, such as Risk Mitigation Framework, wherein risk is identified, quantified, and tracked until the risk can be completely mitigated. It's important to remember that IT modernization is a marathon, not a sprint, and identifying these components so that a strategy can be developed and incorporated into your organization's roadmap. Successful implementation of modern secure communication

protocols, such as TLS 1.3, are integral parts of this roadmap and critical to protecting your organizations, and its customers, most important product - their data.

References

Oppliger, R. (2009). *SSL and TLS: Theory and Practice*.

Subramanium, G (n.d.). *About SSL and Flaws in SSL 2.0.* Retrieved February 8, 2025, from https://www.seekahost.com/flaws-in-ssl-version-2/

K39580786: Configure the SSL profile to allow TLS 1.2 and 1.3 only. (2023, March 8). F5 https://my.f5.com/manage/s/article/K39580786

SSL 3.0 Protocol Vulnerability and POODLE Attack.(2016, September 30). CISA. https://www.cisa.gov/news-events/alerts/2014/10/17/ssl-30-protocol-vulnerability-and-poodle-attack

What is the difference between a MAC and HMAC, and how does HMAC enhance the security of MACs? (2023, August). EITCA. https://eitca.org/cybersecurity/eitc-is-acc-advanced-classical-cryptography/message-authentication-codes/mac-message-authentication-codes-and-hmac/examination-review-mac-message-authentication-codes-and-hmac/what-is-the-difference-between-a-mac-and-hmac-and-how-does-hmac-enhance-the-security-of-macs/

Banach, Z (2024, November 14). *How the BEAST attack works: Reading encrypted data without decryption.* https://www.invicti.com/blog/web-security/how-the-beast-attack-works/

Nohe, P (2018, December 12). *Nearly 21% of the world's top 100,000 websites still aren't using HTTPS.* https://www.thesslstore.com/blog/nearly-21-of-the-worlds-top-100000-websites-still-arent-using-https/

Vera (2024, November 26). *How to Enable TLS 1.2 in Windows 10/11, Windows 7 & Server 2012.* https://www.minitool.com/news/enable-tls-1-2-windows-10-11-7.html

Sherif, A (2024, August 8). *Windows operating systems market share of desktop PCs worldwide 2017-2024.* https://www.statista.com/statistics/993868/worldwide-windows-operating-system-market-share/

API Gateway now supports TLS 1.3. (2024, February 15). https://aws.amazon.com/about-aws/whats-new/2024/02/api-gateway-tls-1-3/

Rahul (2024, January 22). *How to Enable TLS 1.2 on Windows Server - TecAdmin.* https://tecadmin.net/enable-tls-on-windows-server-and-iis/

Rahul (2024, June 6). *How to enable TLS 1.3, TLS1.2 only in Apache.* https://tecadmin.net/enable-tls-version-apache/

Vivek, G (2024, September 10). *How To Configure Nginx to use TLS 1.2/1.3 only.* https://www.cyberciti.biz/faq/configure-nginx-to-use-only-tls-1-2-and-1-3/

Further Reading

- https://isc.sans.edu/diary/29908
- https://www.ssldragon.com/blog/history-of-ssl/
- https://www.digicert.com/blog/evolution-of-ssl
- https://www.acunetix.com/blog/articles/history-of-tls-ssl-part-2/