

Where The TLS Ends

An exercise in HTTP, HTTPS/TLS, and C#

4-Jan-03

202

What is TLS?

- Cryptographic protocol to provide communications security over a computer network
- Runs in the presentation layer (OSI model) but is composed of two layers: TLS record and TLS handshake protocols (when two devices/programs communicate)
- TLS is what SSL has become. SSL is deprecated!

SSL/TLS Protocols

SSL and TLS protocols

Protocol ↕	Published ↕	Status ↕
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011 (RFC 6176)
SSL 3.0	1996	Deprecated in 2015 (RFC 7568)
TLS 1.0	1999	Deprecated in 2021 (RFC 8996) ^{[20][21][22]}
TLS 1.1	2006	Deprecated in 2021 (RFC 8996) ^{[20][21][22]}
TLS 1.2	2008	In use since 2008 ^{[23][24]}
TLS 1.3	2018	In use since 2018 ^{[24][25]}

Intent To Deprecate TLS 1.0/1.1

- This happened in 2018
- <https://security.googleblog.com/2018/10/modernizing-transport-security.html>
 - *TLS (Transport Layer Security) is the protocol which secures HTTPS. It has a long history stretching back to the nearly twenty-year-old [TLS 1.0](#) and its even older predecessor, SSL. Over that time, we have learned a lot about how to build secure protocols.*
 -
 - *[TLS 1.2](#) was published ten years ago to address weaknesses in TLS 1.0 and 1.1 and has enjoyed wide adoption since then. Today only 0.5% of HTTPS connections made by Chrome use TLS 1.0 or 1.1.*

Support For TLS Versions

- The underlying system must support the algorithms and the ciphers that the protocol requires
- Must allow for some failover, but in many cases a failure in negotiation will terminate the connection.
 - For instance, you may need to define the algorithms/ciphers that a specific connection requires.

```
Host mydomain.local
  HostkeyAlgorithms ssh-dss,ssh-rsa
  KexAlgorithms +diffie-hellman-group1-sha1,diffie-hellman-group14-sha1
  Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc
```


















What is this even doing?

- Negotiation phase:
 - Client send ClientHello with highest TLS version it supports (and more)
 - Server responds with ServerHello, with the chosen protocol (and more), finishing with is ServerHelloDone.
 - Client responds with ClientKeyExchange potentially with a PreMasterSecret encrypted with the public key of the server certificate
 - The client and server use random numbers and PreMasterSecret to compute a common secret (“master secret”)
- Client sends a ChangeCipherSpec which means that everything going forward will be authenticated (and potentially encrypted).
- Server sends and ChangeCipherSpec stating similar terms.
- The “handshake” is complete and the application protocol is enabled.

Risks for TLS - The Backbone of a Secure Network

- Forcing downgrades of communications protocols (say from TLS 1.3 -> 1.2)
- Forcing weak ciphers
 - Updates are constant. For instance, CVE-2011-3389 was first published in September 2011 but most recently updated in November 2022
 - <https://nvd.nist.gov/vuln/detail/CVE-2011-3389>
- Some folks ignore internal services, or consider internal services unnecessary to update on schedule since these systems are not exposed. Smart? Or Not?

Support For TLS Versions

Windows Desktop	Minimum Build	TLS 1.0	TLS 1.2*	TLS 1.3
Windows XP	2600 (SP3)	✓		
Windows Vista	6002 (SP2)	✓		
Windows 7	7601 (SP1)	✓	✓	
Windows 8	9200	✓	✓	
Windows 8.1	9600	✓	✓	
Windows 10	18362 (1903)	✓	✓	✓
Windows 11	21H2	✓	✓	✓
Windows Server				
Windows Server 2003	3790	✓		
Windows Server 2003 R2	3790	✓		
Windows Server 2008	6003	✓		
Windows Server 2008 R2	7601	✓	✓	
Windows Server 2012	9200	✓	✓	
Windows Server 2012 R2	9600	✓	✓	
Windows Server 2016	14393	✓	✓	
Windows Server 2019	18362 (1903)	✓	✓	✓
Windows Server 2022	20348	✓	✓	✓

* Operating system support for TLS 1.2 also includes TLS 1.1.

Windows 7 and Windows Server 2008 R2 are the minimum supported platforms for secure connections. Because most servers today will reject connections which attempt to use TLS 1.0 or TLS 1.1, secure connections will fail on unsupported versions of Windows.

Support For TLS Versions

- Linux:
 - `nmap --script ssl-enum-ciphers -p 443 <hostname or IP>`
- Other ways:
 - <https://www.ssllabs.com/>
- HTTP/2
 - Consider that although HTTP/2 does not require usage of encryption, all major browsers have stated they will only support HTTP/2 over TLS

What About Some Code?

- Look at some terraform and the resulting AWS resources created
- Consider a static website (hosted from S3)
- Consider a standalone EC2 instance

My AWS Preferences

- Use Route53 (nameserver) to manage subdomains
- Use Certificate Manager for certificates. Support auto-renew when managed with Route53
- Use Application Load Balancers with AWS certificates assigned
 - In front of ECS, EC2, and potentially any other services with multiple access points
- Prefer public certificates over self hosted for ease of use
 - Private CAs are not super easy
 - Better to not use wildcard certificates, so automation is key