**I recently wrote a blog about**[**why TLS 1.3 is such a huge improvement**](https://www.venafi.com/blog/why-tls-13-huge-improvement)**over previous versions. Basically, it was rewritten from the ground up to secure new encryption environments. While this radical rethinking is incredibly valuable to our future security, it may also require us to rethink the ways that we implement features such as**[**SSL inspection**](https://www.venafi.com/education-center/ssl/what-is-ssl)**. As you may have guessed, I’m talking about forward secrecy, which is featured in TLS version 1.3. But before I dive deep on that, I’ll explain a bit about the evolution of TLS, and why it’s so important.**

**All about TLS 1.3**

**TLS 1.3 was originally defined in RFC 8446 in August 2018, about half a year ago. So the new internet encryption standard is still pretty new. It improves upon TLS 1.2 in a number of different ways. It removed support for weaker hashing algorithms, MD5 and SHA-224 specifically. Support was also dropped for weaker and lesser-used elliptic curves. TLS 1.3 also no longer supports non-AEAD ciphers, non-PFS key exchanges, Change Cipher Spec protocol, Hello message UNIX time, compression, and renegotiation. Those were all features of TLS 1.2 which are now easier for cyber attackers to exploit to perform man-in-the-middle attacks on encrypted web traffic. TLS 1.2 is still better than TLS 1.1, and pretty much any form of data encryption is safer to transmit through the internet than plaintext. Nonetheless, cyber attackers are finding more effective ways to exploit those older TLS features. It’s our responsibility to implement newer TLS features which make data much more difficult for cyber attackers to decrypt.**

**Apple, Google, Microsoft, and Mozilla will depreciate TLS 1.1 completely in March 2020, and wise organizations are quickly replacing TLS 1.2 implementations with TLS 1.3. All recent versions of popular web browsers like Mozilla Firefox, Google Chrome, and Apple Safari now support TLS 1.3, so I hope web servers, email servers, and other sorts of cloud servers will completely drop TLS 1.2 for TLS 1.3 over the duration of 2019. TLS 1.3 was available for many months before it was defined in RFC 8446.**

**I’ve mentioned many of TLS 1.3’s improvements, but perhaps the most significant reason why TLS 1.3 is more secure is how it features forward secrecy.**

**What’s forward secrecy?**

**Here’s the simplest way I can describe forward secrecy. Imagine if a new key was generated each time I went to unlock the front door of my apartment, with a changing lock to match. When I unlock the front door and go into my apartment with all of my heavy grocery shopping, I drop my key into my mailbox. That’s okay because next time I lock my front door and go out into the city to do more grocery shopping, a newly generated key appears in my purse, and the lock on my front door again changes to match. It doesn’t matter if a burglar acquires my old house key, it was only good for the last time I entered my apartment. Cool system, eh?**

**There was the possibility for forward secrecy in older TLS versions, but now it’s mandatory! Sometimes security features won’t be used thoroughly until they’re made to be mandatory, so it’s an excellent development.**

**Here’s a more technical explanation, pertaining to how forward secrecy is implemented in TLS 1.3.**

**TLS 1.3 uses the Ephemeral Diffie-Hellman key exchange protocol, which generates a one-time key that’s used only for the current network session. At the end of the session, the key is discarded. Cyber attackers can still record TLS 1.3 ciphertext and try to crack the encryption later on, but it’s a very difficult process that could require loads of computing power over the duration of months or longer.**

**Inspection complications**

**The mandatory forward secrecy in TLS 1.3 makes your network transferred data more secure from cyber attackers. But there are some downsides to perfect forward secrecy.**

**Network security devices such as components of intrusion prevention systems inspect packets which travel through them, looking for malware or other types of cyber attacks. An intrusion prevention system will typically use decryption keys used throughout the network to decrypt packets so their contents can be inspected. Web firewalls often look for indications of SQL injection attacks with packet inspection. But if specific keys are only used for limited data transmission sessions, packet inspection becomes very challenging indeed.**

**If only the endpoints of TLS 1.3 sessions, the specific clients and servers, have decryption keys, the security appliances that exist somewhere in between become blind to the content of their packets.**

**The financial services industry in particular considers it to be important that their security appliances can inspect packets throughout their networks. So they may stick with TLS 1.2 for the time being, where forward secrecy is only optional.**

**TLS 1.3 proxy servers are another possible option for some organizations. With proxy servers, TLS 1.3 is communicated to the end user, and then the proxies establish new TLS 1.2 sessions through the datacentre to servers. That’s probably the best way to implement TLS 1.3 while enabling packet inspection, and the IETF TLS committee endorses that approach.**

**In the cryptography community, many people expect that quantum computers that can crack encryption will debut in the 2020s, very soon indeed. With NIST support, it’s possible that TLS 2.0 will be available in a few years with post-quantum capabilities. So some organizations may be waiting until the soon-coming quantum age. Unlike a binary digit, a qubit could be 0, 1, or both 0 *and* 1. The implications of qubits are mindblowing, and I can’t completely grasp it.**

**As far as web servers and email servers are concerned, I still think as much TLS 1.3 adoption as possible is the best way to go right now.**