UNITED STATES

HOW-TO

What is SSL, TLS? And how this encryption protocol works

The SSL/TLS protocol encrypts internet traffic of all types, making secure internet communication (and therefore internet commerce) possible. Here are the basics of how it works and what comes next.

By Josh Fruhlinger

CSO |

DEC 4, 2018 12:22 PM PST

Since the early days of the web, the SSL protocol and its descendent, TLS, have provided the encryption and security that make modern internet commerce possible. The decades-long history of these protocols has been marked by continuous updates that aim to keep pace with increasingly sophisticated attackers. The next major version of the protocol, TLS 1.3, will soon be finalized — and most anyone who runs a website will want to upgrade, because cybercriminals are catching up.

What is SSL?

Secure Sockets Layer, or SSL, was the original name of the protocol when it was developed in the mid-1990s by Netscape, the company that made the most popular Web browser at the time. SSL 1.0 was never released to the public, and SSL 2.0 had serious flaws. SSL 3.0, released in 1996, was completely revamped, and set the stage for what followed.

[All SSL certs are not created equal. Here's what the different certificates actually mean | Get the latest from CSO by signing up for our newsletters.]

TLS vs. SSL

When the next version of the protocol was released in 1999, it was standardized by the Internet Engineering Task Force (IETF) and given a new name: *Transport Layer Security,* or TLS. As the TLS specification notes, "the differences this protocol and SSL 3.0 are not dramatic." Thus, it's not really a matter of TLS vs. SSL; rather, the two form a continuously updated series of protocols, and are often lumped together as SSL/TLS.

The TLS protocol encrypts internet traffic of all types. The most common is web traffic; you know your browser is connected via TLS if the URL in your address starts with "https," and there's an indicator with a padlock telling you the connection is secure, as in this screenshot from Chrome:

But TLS can be used by other applications as well, including e-mail and usenet.

How SSL works

Encryption is necessary in order to communicate securely over the internet: if your data isn't encrypted, anyone can examine your packets and read confidential information. The safest method of encryption is called *asymmetrical cryptography;* this requires two cryptographic *keys* — pieces of information, usually very large numbers — to work properly, one public and one private. The mathematics here are complex, but in essence, you can use the public key to *encrypt* the data, but need the private key to *decrypt* it. The two keys are related to each other by some complex mathematical formula that is difficult to reverse-engineer by brute force. Think of the public key as information about the location of a locked mailbox with a slot on the front, and the private key as the key that unlocks the mailbox. Anyone who knows where the mailbox is can put a message in it; but for anyone else to read it, they need the private key.

Because asymmetrical cryptography involves these difficult mathematical problems, it takes a lot of computing resources, so much so that if you used it to encrypt all the information in a communications session, your computer and connection would grind to a halt. TLS gets around this problem by only using asymmetrical cryptography at

the very beginning of a communications session to encrypt the conversation the server and client have to agree on a single *session key* that they'll both use to encrypt their packets from that point forward. Encryption using a shared key is called *symmetrical cryptography,* and it's much less computationally intensive than asymmetric cryptography. Because that session key was established using asymmetrical cryptography, the communication session as a whole is much more secure than it otherwise would be.

The process by which that sessions key is agreed upon is called a *handshake,* since it's the moment when the two communicating computers introduce themselves to each other, and it's at the heart of the TLS protocol.

SSL handshake process

The handshake process is quite complex, and there are a number of variations allowed by the protocol. The following steps provide a broad outline that should give you a sense of how it works.

1. The client contacts the server and requests a secure connection. The server replies with the list of *cipher suites* — algorithmic toolkits of creating encrypted connections — that it knows how to use. The client compares this against its own list of supported cipher suites, selects one, and lets the server know that they'll both be using it.
2. The server then provides its *digital certificate,* an electronic document issued by a third-party authority confirming the server's identity. We'll discuss digital certificates in more detail in a moment, but for now the most important thing you need to know about them is that they contain the server's public cryptographic key. Once the client receives the certificate, it confirms the certificate's authenticity.

3. Using the server's public key, the client and server establish a session key that both will use for the rest of the session to encrypt communication. There are several techniques for doing this. The client may use the public key to encrypt a random number that's then sent to the server to decrypt, and both parties then use that number to establish the session key. Alternately, the two parties may use what's called a Diffie–Hellman key exchange to establish the session key.

This article at SSL.com has a great diagram outlining each communication step over the course of the TLS handshake process.

As its name implies, the session key is only good for the course of a single, unbroken communications session. If for some reason communications between client and server are cut off — due to a network problem, for instance, or because the client is idle for too long — a new handshake will be required to establish a new session key when communication is re-established.

What is an SSL certificate?

Let's return to the concept of an *SSL certificate*. As the description in the previous section made clear, these certificates are at the heart of the SSL/TLS protocol: they provide the client with the public cryptographic key necessary to initiate secure connections. But their purpose goes beyond just supplying the key itself; they also authenticate that the key is in fact associated with organization offering it to the client.

How does this work? Certificates are issued by *Certificate Authorities* (CAs), who serve as the equivalent of a passport office when it comes to confirming identities. Organizations that want to offer services encrypted by TLS must purchase certificates from CAs, who in turn verify that the organizations are who they claim to be. For instance, if you wanted to buy a certificate to secure a website at example.com, you'd have to take some steps to prove to the CA that you control the example.com domain. That way, if someone connects to example.com and gets a

valid SSL certificate issued by a trusted CA, they can be sure that they're communicating with the legitimate owner of example.com. This can prevent man in the middle attacks.

Notice that we used the phrase "trusted CA" in that last paragraph. Anyone can set themselves up as a certificate authority; how can you tell which ones perform the due diligence needed to authenticate their customers? Fortunately, the job of figuring that out is mostly taken care of by software manufacturers. The Mozilla Foundation maintains a list of CAs that Firefox will trust; Apple and Microsoft also maintain lists that they implement at the OS level for Windows, macOS, and iOS, which Chrome uses on those platforms. The decisions on which CAs to trust have high stakes, as a 2017 showdown between Google and Symantec over what Google felt were Symantec's lax standards made clear.

The standard that defines SSL certificates is called *X.509.* This standard allows certificates to carry a lot of information beyond just the public key and the confirmed identity of the certificate owner; DigiCert is a CA whose knowledge base has a detailed breakdown of the standard.

SSL checkers

Almost all of the exchange and confirmation of information detailed above takes place behind the scenes as you communicate with servers that offer TLS-encrypted connections. If you want to get a bit more transparency, you can enter the URL of an SSL/TLS-encrypted site into an *SSL checker* website. The checker will return a host of information about the tested site's certificate, including the server type, which web browsers will and won't trust the certificate, the issuer, the serial number, and the expiration date.

Most SSL checkers are free services offered by CAs as marketing tools for their wares; many will, for instance, allow you to set an alert for when an inspected certificate will expire, on the assumption that it's your certificate and you'll be in the

market for a new one as that date approaches. If you're looking for a somewhat less commercial alternative, check out the SSL checker from Qualys SSL Labs, which provides a particularly robust collection of information on inspected websites.

TLS 1.2 and TLS 1.2 vulnerabilities

TLS 1.2 is the most current defined version of the protocol, and it has been for several years. It established a host of new cryptographic options for communication. However, like some previous versions of the protocol, it also allowed older cryptographic techniques to be used, in order to support older computers. Unfortunately, that opened it up to vulnerabilities, as those older techniques have become more vulnerable as time has passed and computing power has become cheaper.

In particular, TLS 1.2 has become increasingly vulnerable to so-called "man-in-the-middle" attacks, in which a hacker intercepts packets in mid-communication and sends them on after reading or altering them. It's also open to the POODLE, SLOTH, and DROWN attacks. Many of these problems have arisen in the last two years, increasing the sense of urgency for updating the protocol.

TLS 1.3

Fortunately, help is on the way. Version 1.3 of the TLS protocol, currently in draft form but soon to be finalized, plugs a lot of these holes by jettisoning support for legacy encryption systems. There is backwards compatibility in the sense that connections will fall back to TLS 1.2 if one end isn't capable of using the newer encryptions systems on the 1.3 approved list. However, if, for instance, a man-in-the-middle attack tries to force a fallback to 1.2 in order to snoop on packets, that will be detected and the connection dropped.

There are still servers out there that are using versions of TLS even older than 1.2 — some are still using the original SSL protocol. If your sever is one of those, you should upgrade now, and just leap ahead and upgrade to the draft 1.3 spec.

TLS crimeware

One last note on TLS and security: the good guys aren't the only ones who use it! Many cybercriminals use TLS to encrypt command-and-control traffic between their servers and malware installed on their victim's computers. This ends up inverting the usual state of affairs and leaves the victims of cybercrime looking for a way to decrypt traffic. There are a number of techniques for dealing with this kind of encrypted attack, including using network metadata about the encrypted traffic to get a sense of what attackers are doing without actually reading any of it.

More on SSL/TLS:

- **All you need to know about the move from SHA-1 to SHA-2 encryption**
- **Security vs. visibility: Why TLS 1.3 has data center admins worried**
- **Know the limits of SSL certificates**
- **Why SSL/TLS attacks are on the rise**
- **Challenges Associated with SSL/TLS traffic decryption and security inspection**

Next read this

- *8 key security considerations for protecting remote workers*
- *The 10 most powerful cybersecurity companies*
- *6 security metrics that matter – and 4 that don't*
- *A security guide for pandemic planning: 7 key steps*
- *8 PCI DSS questions every CISO should be able to answer*
- *5 standout products from RSA 2020*
- *2020 cybersecurity trends: 9 threats to watch*
- *Do you trust your admins? 5 tips to manage administrator access*
- *How to write an effective information security policy*

Josh Fruhlinger is a writer and editor who lives in Los Angeles.

Follow

The 10 most powerful cybersecurity companies