6.4 SSL/TLS

A new type of firewall uses SSL (Secure Sockets Layer) or TLS (Transport Layer Security) to provide VPN access through a web portal. Essentially, TLS and SSL are the protocols used to secure websites. If you see a website beginning with HTTPS, then traffic to and from that website is encrypted using SSL or TLS. Today, we almost always mean TLS when we say SSL. This is due to the fact that “SSL” has caught on with people and so this is how they refer to it. This becomes clear from the brief history of SSL/TLS presented here:

- Unreleased SSL v1 (Netscape).
- Version 2 released in 1995 but had many flaws.
- Version 3 released in 1996 (RFC 6101).
- Standard TLS 1.0, RFC 2246, released in 1999.
- TLS 1.1 defined in RFC 4346 in April 2006.
- TLS 1.2 defined in RFC 5246 in August 2008. It is based on the earlier TLS 1.1 spec.
- As of July 2017, TLS 1.3 is a draft and details have not been fixed yet.

In some VPN solutions the user logs in to a website, one that is secured with SSL or TLS. Following that, the user is given access to a virtual private network. However, visiting a website that uses SSL or TLS does not mean you are on a VPN. As a general rule most websites, such as banking websites, give you access only to a very limited set of data, such as your account balances. A VPN gives you access to the network, the same or similar access to what you would have if you were physically on that network.

Whether you are using SSL to connect to an e-commerce website or to establish a VPN, the SSL handshake process is needed to establish the secure/encrypted connection:

1. The client sends the server the client’s SSL version number, cipher settings, session-specific data, and other information that the server needs to communicate with the client using SSL.

2. The server sends the client the server’s SSL version number, cipher settings, session-specific data, and other information that the client needs to communicate with the server over SSL. The server also sends its own certificate, and if the client requests a server resource that

requires client authentication, the server requests the client's certificate.

3. The client uses the information sent by the server to authenticate the server—e.g., in the case of a web browser connecting to a web server, the browser checks whether the received certificate's subject name actually matches the name of the server that is being contacted, whether the issuer of the certificate is a trusted certificate authority, whether the certificate has expired, and, ideally, whether the certificate has been revoked. If the server cannot be authenticated, the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established. If the server can be successfully authenticated, the client proceeds to the next step.

4. Using all data generated in the handshake thus far, the client (with the cooperation of the server, depending on the cipher in use) creates the pre-master secret for the session, encrypts it with the server's public key (obtained from the server's certificate, sent in step 2), and then sends the encrypted pre-master secret to the server.

5. If the server has requested client authentication (an optional step in the handshake), the client also signs another piece of data that is unique to this handshake and known by both the client and server. In this case, the client sends both the signed data and the client's own certificate to the server along with the encrypted pre-master secret.

6. If the server has requested client authentication, the server attempts to authenticate the client. If the client cannot be authenticated, the session ends. If the client can be successfully authenticated, the server uses its private key to decrypt the pre-master secret, and then performs a series of steps (which the client also performs, starting from the same pre-master secret) to generate the master secret.

7. Both the client and the server use the master secret to generate the session keys, which are symmetric keys used to encrypt and decrypt information exchanged during the SSL session as well as to verify its integrity (that is, to detect any changes in the data between the time that it is sent and the time it is received over the SSL connection).

8. The client sends a message to the server so as to inform that any future messages from the client will be encrypted with the session key.

Following that, it sends a separate (encrypted) message indicating that the client’s portion of the handshake has finished.

9. The server sends a message to the client so as to inform that any future messages from the server will be encrypted with the session key. Following that, it sends a separate (encrypted) message indicating that the server’s portion of the handshake has finished.