Proxying SSL
Why It's Important

Transport Layer Security (TLS) and its predecessor Secure Sockets Layer (SSL) are cryptographic protocols. They are designed to permit the authentication of clients to servers, and vice-versa, as well as protect the communications from such threats as eavesdropping, tampering, and replay. The protocols have been around for more than 10 years now, and are seeing near ubiquitous usage as the protocol of choice for secure communications of many sorts.

The Certificate

At the core of TLS/SSL is the certificate. This is based on RSA asymmetric key cryptography, where a key pair (public and private) is generated such that one key is used to encrypt while the other is used to decrypt. So, for example, we can encrypt some text with the public key, and then only the holder of the private key can decrypt it, or vice-versa. Such asymmetric key systems can be used for encryption/ decryption as well as signing/verifying (I sign with one key, then you can verify with the other).

To generate a certificate, two entities are involved.

1. Firstly, the owner of the certificate generates a random public+private key pair, and then puts the public key plus his identifying information into a certificate request, and sends it to a trusted certificate authority.

2. Secondly, the certificate authority generates its own public+private key pair (or more commonly uses one previously generated) and then uses its own private key to sign the owner's public key in the certificate request.

The result of this process is a certificate containing the owner's identifying information and public key, signed with the private key of the trusted certificate authority.

In addition, the trusted certificate authority puts its own public key into a certificate and typically self-signs it with its own private key (or in some cases gets it signed by another upstream trusted certificate authority), then publishes that certificate.

Certificate Verification

Now, when a client connects to a server talking TLS/SSL, the server can provide its own certificate to the client.

When the client receives this certificate, it can verify the authenticity by checking against a list of trusted certificate authorities it maintains. By using the certificate authority's public key (inside a certificate), the client can verify the signature on the certificate presented by the server. You can see that for this to work, the certificate authority who signed the server's certificate must also be trusted by the client (i.e. in the list of trusted certificate authorities installed on that client).

Once the certificate has been verified, key exchange can be safely performed using the public key inside the certificate (so that only the server that has the matching private key can decrypt the information).

Of course, the above is a vastly simplified overview of a complex process, but you should be able to see the basic mechanics of how this works. The key points to take away from this are:

1. Data encrypted with a public key can only be decrypted with the matching private key. This is asymmetric key cryptography, and works both ways public → private and private → public.
2. The certificate contains a public key, as well as matching identification information for the owner.
3. The certificate is signed by the private key of a certificate authority trusted by both the client and the server.
4. The client can verify the signature using the public key of the trusted certificate authority.
5. The client can send data securely to the server using the verified public key of the server (as only the server has the matching private key used to decrypt).

Man in the Middle

So, say that as an organizational policy, we want to be able to inspect TLS/SSL protected traffic. We want to ensure that TLS/ SSL protected websites being visited over the HTTPS protocol conform to organizational policy, and that the files transferred do not contain malware. We also want to ensure that proper validation of the TLS/SSL certificate is performed, and protocol exploits protected against.

The way to do this is with a so-called man-in-the-middle interception.

The idea is that when the client connects to the server, that connection is intercepted by the proxy. The proxy then maintains two separate TLS/SSL connections - one from the client to proxy and the other from the proxy to the server. This is performed as follows:

1. The proxy itself has its own certificate (with public and private keys) and this certificate is installed as a trusted certificate authority into the client.
2. The proxy negotiates TLS/SSL with the server, and validates and ensures the correctness of this connection.
3. The proxy takes the server certificate, replaces the public key with one of its own, and then self-signs that certificate with its own certificate authority private key.
4. The proxy uses this modified certificate to negotiate a TLS/SSL connection with the client.

When talking to the server the proxy sees the server's certificate signed by a certificate authority trusted by both the server and the proxy. The server's public key (contained in that certificate) can be used to securely communicate with the server. But, the client sees the server's certificate signed by the intercepting proxy with the proxy's public key, and the proxy's public key (contained in that modified certificate) can be used to securely communicate with the proxy.

With such an arrangement, the proxy decrypts the traffic from the client, examines it for policy enforcement, then encrypts it and forwards it on to the server. Return traffic from the server can be decrypted by the proxy, examined for policy enforcement, then re-encrypted and forwarded on to the client. The communications client-proxy and proxy-server are secure, but the traffic is subject to policy enforcement.

Strengths and Weaknesses

Such an arrangement removes the weak link (the end-user policy decision) from the TLS/SSL protocol, by allowing policy enforcement to be made by administrators at the network gateway. It also allows for protocol upgrading, and protection against browser exploits and other client-side vulnerabilities. A good example of this is the recent FREAK vulnerability - Network Box 5 SSL proxy users have been protected against that exploit from day #1, as the Network Box 5 SSL proxy already enforces secure cryptographic cipher selection at the gateway (irrespective of the vulnerability at the client).

However, there are two main areas that such an arrangement may not work:

1. Client-Side Certificates. The TLS/SSL protocol allows for client side certificates. If these are used, the traffic cannot be intercepted (as the client certificate provided to the server could not be modified without access to the server's trusted certificate authority list - which is not practical).

2. Certificate Pinning. Some client side applications using TLS/SSL perform an extra validation step which is to verify the identity of the certificate authority used to sign the server's certificate. For example, requiring the certificates presented by server X to be signed by authority Y. Such traffic cannot be intercepted (as the client side application would object to the certificate authority used by the intercepting proxy).

In such cases, policy rules are normally put in place, at the proxy, to bypass such sites from TLS/SSL interception.

The other restriction is that the certificate authority used by the intercepting proxy must be trusted by the clients (normally meaning that the proxy's SSL certificate must be installed into the trusted certificate store of the client). We are often asked why this is a requirement, and the answer is simple - if it wasn't a requirement, then we could simply and easily intercept, monitor and modify TLS/SSL communications transparently to the client and the security and integrity of the world's financial networks would be destroyed.

How Network Box supports SSL in Network Box 5

SSL as a Filter

The first key concept is that in Network Box 5, TLS/SSL is treated as an encapsulation layer. HTTPS is purely the HTTP protocol encapsulated in an SSL stream. Similarly SMTPS is SMTP in SSL, POP3S is POP3 in SSL, and IMAP4S is IMAP in SSL. In general, any arbitrary data stream or protocol can be encapsulated in SSL for protection.

Network Box 5 treats SSL interception as a filter on the communication channels. The device can be configured to add or remove SSL from either the input or output sides of a communication. If SSL is chosen to be removed, then the resulting plain text can be analyzed further.

The SSL negotiation is always initiated from the client side, and can be started in one of two ways:

1. Fixed Port (for example, TCP/443 for HTTPS). The client starts the negotiation of SSL immediately after the network connection has been established.

2. Command based (for example, CONNECT for HTTP proxy, STARTTLS for SMTP, etc). The client first initiates a switch to SSL, using a protocol-level command such as CONNECT or STARTTLS. Then, once the server acknowledges that, both clients assume the connection is now ready for SSL and the client starts the negotiation of SSL.

For either type, SSL identification and decoding starts with the client-initiated SSL negotiation.

Client Side vs Server Side

Typically, there are two scenarios where Network Box SSL proxying can be involved:

1. **Client Side.** Here, clients (typically in the LAN/DMZ) are connecting out to external public SSL servers. In this case, the certificate for the SSL server is not under the control of the Network Box admin. SSL Man in the Middle style interception and proxying must be used to decrypt the traffic.

2. **Server Side.** Here, clients (typically on the Internet) are connecting in to internal SSL servers (typically in the DMZ). In this case, the certificate for the SSL server is under control of the Network Box admin, but the client policies are not. The server certificate and private key would typically be installed on the Network Box, so that the Network Box SSL proxy can use those to act on behalf of the server and decrypt the incoming SSL sessions.

As you can see, the technologies used for both are very different, so let's discuss them further individually.

Client Side SSL Man in the Middle Proxying

The Network Box 5 SSL Man in the Middle proxy works by intercepting SSL communications between the client and the server. Two separate SSL sessions are maintained - one between the client and the proxy, and the other between the proxy and the server.

There are various policy rule points in the Network Box proxy, including:

- The decision on whether to decode the SSL traffic is made shortly after connection time, once the client SSL negotiation message has been received. The options available are (a) bypass the traffic without decode, (b) permit the traffic and decode it, or (c) deny the connection.

- Once the server certificate has been received, it is validated. Checks are made for items such as certificate signing chain, as well as issue and expiration dates. The result of those checks is then available as a policy decision to (a) permit it anyway, or (b) deny the connection.

- When inspecting a communication channel for SSL traffic, at some point a decision is made whether there is actually an SSL negotiation being attempted. In the case where the traffic is not SSL, a policy decision is made whether to (a) bypass the traffic and allow it, or (b) deny the connection. An example here is the HTTP proxy CONNECT statement. That should be used to switch to SSL, and a policy decision can be made if the client does not actually negotiate SSL after the CONNECT statement (as they are most likely trying to bypass firewall policy).

Server Side SSL Fronting

Network Box 5 can be used to 'front' SSL communications at the gateway. Rather than deploying a SSL enabled server, the SSL certificate and private key can be installed on the Network Box itself, and the SSL connection terminated there. With this arrangement, the connection from client to proxy is SSL protected. The connection from proxy to server can either be plaintext, or re-encoded as SSL again.

The advantage of SSL fronting is that it allows protection technologies such as WAF and anti-DDoS to be deployed even for SSL protected services.

Why Decode SSL?

So, given the above technologies and capabilities, we come to the core question of why would we want to decode SSL?

The simple answer is that we want to enforce policy control and have the capability to apply protection technologies for traffic protected by SSL, in the same way we do for plaintext traffic. As more and more of the Internet's communications move to SSL, less and less is subject to policy control and protection, unless we deploy SSL decoding capabilities. An example is Google Safe Search. Sure, we can enforce safe search at the proxy, and it works well in school and home environments. But, Google Search has now moved to SSL, and without SSL decoding, safe search cannot be enforced at the proxy.

Another reason is that the SSL certificate verification process occurs largely at the client workstation, with the end-user being responsible for the decision as to whether to accept/ reject a problem with a particular connection. End-users are typically not trained in the security implications of such decisions, and in general it is better to leave such security policies up to the administrator, to be enforced at the gateway.

Protocol Promotion

One unique feature of the Network Box 5 SSL proxy, that comes from our implementation as a filter, is that the SSL decode mechanism can be combined with other functions such as application identification. We can intercept network communications generically, use the SSL filter to decode the SSL stream, and then apply application identification to that decoded stream. With such a capability, we can detect the full suite of 1,000+ applications, even if they are encrypted in SSL.

Moreover, once we've used the filter to decode SSL, and identified an application such as SMTP, HTTP, POP3, etc, we can 'promote' the stream to a protocol specific module for further protection. An example of this would be to intercept all outbound TCP traffic, filter out SSL if detected, then if the application type is detected to be HTTP (originally HTTPS when protected by SSL), transparently switch the traffic through to the webclient module for URL policy enforcement, anti-malware scanning, and safe search enforcement.

We hope that you now have an understanding of the SSL/TLS protocol, and how Network Box 5 can filter it out of your network communications, to allow you to apply policy control and other protection technologies to the underlying data stream.