9.10.3 Secure Protocol Facts

When many protocols were created, they were designed with little or no security controls. An unsecured protocol is one that does not provide authentication or encryption, or that uses plain text for passing authentication information or data. Security services (authentication and encryption) are often added to new or existing protocols using one of the following secure protocols:

Protocol	Description
Secure Sockets Layer (SSL)	Secure Socket Layer (SSL) secures messages being transmitted on the internet. SSL: - Uses the SSL Handshake Protocol to establish the secure channel. - Requires the server to have a certificate issued by a CA and uses asymmetric encryption. 1. The client checks the server's certificate validity period. The authentication process stops if the current date and time fall outside of the validity period. 2. The client compares the name on the certificate with the name on the URL. 3. The client verifies that the issuing Certificate Authority (CA) is on its list of trusted CA's. 4. The client uses the CA's public key to validate the CA's digital signature on the server certificate. If the digital signature can be verified, the client accepts the server certificate as a valid certificate issued by a trusted CA. 5. A session key is used between the client and the server for the duration of the SSL session. 6. To protect against man-in-the-middle attacks, the client compares the actual DNS name of the server to the DNS name on the certificate. 7. If all checks are successful, the client continues with the SSL handshake process. - Uses RSA or the Key Exchange Protocol (KEA) for secure exchanging of encryption keys. - Operates at the Session layer (layer 5) of the OSI model. - Uses port 443 for encrypted traffic. Most firewalls allow port 443 traffic even when other traffic is blocked. For this reason, technologies that can use SSL are more likely to be allowed through firewalls than technologies that require other ports to be opened. - Has different versions, with the later versions being more secure. Secure Sockets Layer (SSL) 3.0 was the last SSL version. - Employs session keys in 40-bit, 56-bit, 128-bit, and 256-bit lengths. - Provides an end-to-end encrypted tunnel that is impossible to monitor, scan, or sniff. - The advantage is that it increases security. - The disadvantages are that: - Security software cannot detect embedded attacks in transit. - Internal users can use SSL to bypass proxy servers or internet content filtering systems that have been set up by organizations to control internet usage and content. - SSL inspection uses security software on a proxy server. The proxy server intercepts and inspects traffic between a client and web server. This is similar to a man-in-the middle attack, but for positive use. In SSL inspection: - The client establishes a SSL tunnel with the proxy server which then decrypts the SSL session, scans the content, repackages the SSL session and sends the transmission to the web server via an SSL tunnel. - The process is reversed when the web server establishes an SSL tunnel with the proxy server which decrypts, scans and repackages the SSL session before sending the transmission to the client. - The proxy server blocks the transmission of inappropriate or unauthorized content in either direction. - Can be used to secure LDAP communication (LDAPS) and FTP (FTPS).
Transport Layer Security (TLS)	Transport Layer Security (TLS) is the successor to SSL 3.0. - TLS and SSL are similar but not interoperable, although most applications can use both SSL and TLS. - Applications that can use both SSL and TLS negotiate which protocol to use during the handshake process. - An SSL session begins when the client sends a client hello message to the server. - The client hello message specifies the highest SSL/TSL version that the client supports. - The message also contains a random number, a list of ciphers and suggested compression methods. - The server responds with a server hello message. - The server hello message specifies the protocol version, a different random number, and the selected cipher and compression method. - The server sends a certificate message followed by a server hello done message. - The client responds with a client key exchange message. - The random numbers exchanged earlier are used to compute the master secret. - All further key data for the connection is derived from the master secret. - The client then sends a change cipher spec message which indicates that further communication will be encrypted. - The client then sends a finished message. - The finished message contains a hash and a MAC. - The server attempts to decrypt the finished message and verify the hash and MAC. - If the server fails to decrypt the message, the connection is ended. - If the server succeeds in decrypting the message, the server sends the client a *change cipher spec* message indicating that further transmission will be encrypted. - The server then sends a finished message to the client. - The finished message contains a hash and a MAC.

	- The client attempts to decrypt the finished message and verify the hash and MAC. - If the client fails to decrypt the message, the connection is ended. - If the client succeeds, the handshake is considered complete. - Many secure connections that are described as using SSL might actually be using TLS instead. - TLS uses Diffie-Hellman or RSA to exchange session keys. - TLS is implemented through two protocols: - TLS Record provides connection security with encryption (with DES for example). - TLS Handshake provides mutual authentication and choice of encryption method.
Secure Shell (SSH)	SSH allows for secure interactive control of remote systems. - SSH uses RSA public key cryptography for both connection and authentication. - SSH uses the IDEA algorithm for encryption by default, but is able to use Blowfish and DES. - SSH is a secure and acceptable alternative to Telnet. - SSH is used by unsecured protocols to establish a secure channel. For example, SFTP and SCP are secure file copy protocols that use SSH.

A common unsecured protocol is the Hyper Text Transfer Protocol (HTTP). HTTP is used for exchanging web content and passes data in clear text. HTTP uses TCP port 80 and is stateless, which means by default it doesn't keep track of clients. To solve this problem, cookies can be used to keep track of the client's behavior. To secure HTTP, use one of the following protocols:

Protocol	Description
HTTPS	Hyper Text Transfer Protocol Secure (HTTPS) is a secure form of HTTP that uses either SSL or TLS to encrypt sensitive data before it is transmitted. HTTPS: - Is stateful, which means that it keeps track of the client. To do this, the client must communicate with the same HTTPS server for the duration of the session. Load balancing is not possible during the connection, and is available only to initially determine which server will handle the client's session. - Requires TCP port 443 inbound on the web server to be open. - Can be identified by verifying that the URL starts with https://, or by looking for a lock symbol in the browser. Double clicking on the lock icon will display the certificate.
S-HTTP	Secure Hypertext Transfer Protocol (S-HTTP) is an alternate protocol that is not widely used because it is not as secure as HTTPS. S-HTTP : - Is connectionless, unlike SSL which is connection-oriented. - Provides only message security, unlike HTTPS which provides a full secure channel for all messages. - Does not use port 443.