

Overview of SSL and SSL Authentication

By Edward Ray
Chief Information Security Officer
MMICMAN, LLC
ewray@mmicman.com

Overview of Topics

- What is SSL?
- SSL/TLS Description
- SSL/TLS Handshake
- History and Development of SSL/TLS
- SSL/TLS Security Features
- How SSL/TLS Provides Authentication, Confidentiality and Integrity
- Cipher Suites and Cipher Specs
- Digital Certificates & PKI
- Security Vulnerabilities in SSL/TLS
- Conclusion & Questions

What Is SSL?

- **Transport Layer Security (TLS)** and its predecessor, **Secure Sockets Layer (SSL)**, are cryptographic protocols that provide communication security.
- Can be used both for Internet and Local Area Network (LAN) communication security.
- TLS and SSL encrypt the segments of network connections above the Transport Layer by using:
 - Asymmetric cryptography used for key exchange
 - Symmetric encryption for privacy
 - Message authentication codes for message integrity
- Used in applications such as web browsing, electronic mail, wireless communications, instant messaging and voice-over-IP (VoIP)

SSL/TLS Description

- The SSL/TLS protocol allows client-server applications to communicate across a network in a way designed to prevent eavesdropping and tampering.
- Since most protocols can be used either with or without TLS (or SSL) it is necessary to indicate to the server whether the client is making a TLS connection or not. There are two ways to accomplish this:
 - use a different port number for SSL/TLS connections (i.e. TCP port 443 for HTTPS)
 - use the regular port number and have the client request that the server switch the connection to SSL/TLS using a protocol specific mechanism (for example STARTTLS for mail and news protocols)
- Once the client and server have decided to use TLS they negotiate a stateful connection by using a handshaking procedure.

SSL/TLS Description con.

- During this handshake, the client and server agree on various parameters used to establish the connection's security.
 - The handshake begins when a client connects to a TLS-enabled server requesting a secure connection and presents a list of supported cipher suites (ciphers and hash functions).
 - From this list, the server picks the strongest cipher and hash function that it also supports and notifies the client of the decision.
 - The server sends back its identification in the form of a digital certificate. The certificate usually contains the server name, the trusted certificate authority (CA) and the server's public encryption key.
 - The client may contact the server that issued the certificate (the trusted CA as above) and confirm the validity of the certificate before proceeding.
 - In order to generate the session keys used for the secure connection, the client encrypts a random number with the server's public key and sends the result to the server. Only the server should be able to decrypt it, with its private key.
 - From the random number, both parties generate key material for encryption and decryption.

SSL/TLS Description con.

- This concludes the handshake and begins the secured connection, which is encrypted and decrypted with the key material until the connection closes.
- **If any one of the above steps fails, the SSL/TLS handshake fails and the connection is not created.**

SSL/TLS Handshake in Detail

(1) The SSL client sends a "client hello" message that lists cryptographic information such as the SSL version and, in the client's order of preference, the CipherSuites supported by the client. The message also contains a random byte string that is used in subsequent computations. The SSL protocol allows for the "client hello" to include the data compression methods supported by the client, but current SSL implementations do not usually include this provision.

(2) The SSL server responds with a "server hello" message that contains the CipherSuite chosen by the server from the list provided by the SSL client, the session ID and another random byte string. The SSL server also sends its digital certificate. If the server requires a digital certificate for client authentication, the server sends a "client certificate request" that includes a list of the types of certificates supported and the Distinguished Names of acceptable Certification Authorities (CAs).

(3) The SSL client verifies the digital signature on the SSL server's digital certificate and checks that the CipherSuite chosen by the server is acceptable.

(4) The SSL client sends the random byte string that enables both the client and the server to compute the secret key to be used for encrypting subsequent message data. The random byte string itself is encrypted with the server's public key.

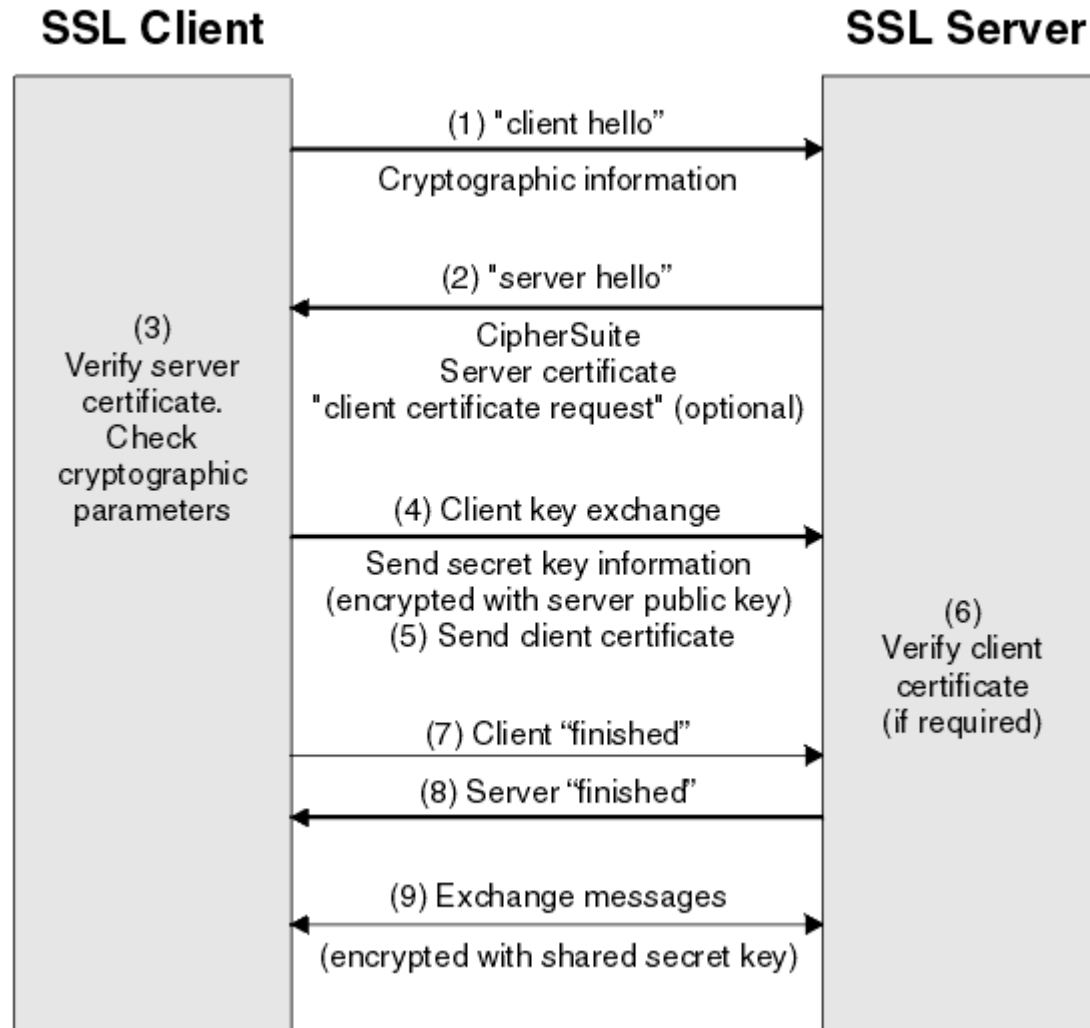
(5) If the SSL server sent a "client certificate request", the SSL client sends a random byte string encrypted with the client's private key, together with the client's digital certificate, or a "no digital certificate alert". This alert is only a warning, but with some implementations the handshake fails if client authentication is mandatory.

(6) The SSL server verifies the signature on the client certificate.

(7) The SSL client sends the SSL server a "finished" message, which is encrypted with the secret key, indicating that the client part of the handshake is complete.

(8) The SSL server sends the SSL client a "finished" message, which is encrypted with the secret key, indicating that the server part of the handshake is complete.

(9) For the duration of the SSL session, the SSL server and SSL client can now exchange messages that are symmetrically encrypted with the shared secret key.



History and Development of SSL/TLS

- **Secure Network Programming API**
 - Early research efforts toward transport layer security included the **Secure Network Programming** (SNP) application programming interface (API) to facilitate retrofitting preexisting network applications with security measures
- **SSL 1.0, 2.0 and 3.0**
 - The SSL protocol was originally developed by Netscape. Version 1.0 was never publicly released; version 2.0 was released in February 1995 but "contained a number of security flaws which ultimately led to the design of SSL version 3.0".^[6] SSL version 3.0, released in 1996, was a complete redesign of the protocol. Newer versions of SSL/TLS are based on SSL 3.0.
 - SSLv3 suffers from long known Cipher Block Chaining (CBC) vulnerability.
- **TLS 1.0**
 - Differs from SSLv3 in ways that make them inoperable with each other:
 - SSL connections begin with security and proceed directly to secured communications, TLS connections first begin with an insecure "hello" to the server and only switch to secured communications after the handshake between the client and the server is successful
 - TLS allows both secure and insecure connections over the same port, whereas SSL requires a designated secure-only port.

History and Development of SSL/TLS con

- **TLS 1.0 con**

- If the TLS handshake fails for any reason, the connection is never created.
- Open Standard: Defined in RFC 2246 in January 1999
- Suffers from same susceptibility to CBC attacks as SSLv3

- **TLS 1.1**

- Open Standard: Defined in RFC 4346 in April 2006
- Update from TLS 1.0
- Added protection against CBC attacks

- **TLS 1.2**

- Open Standard: Defined in RFC 4346 in April 2008, RFC 6176 in March 2011
- Based on TLS 1.1 specification
- Support for more secure authentication encryption ciphers
- Improved security by redacting its backward compatibility with SSL such that TLS sessions will never negotiate the use of Secure Sockets Layer (SSL) version 2.0.

SSL/TLS Security Features

- Protection against a downgrade of the protocol to a previous (less secure) version or a weaker cipher suite.
- Numbering subsequent Application records with a sequence number and using this sequence number in the message authentication codes (MACs).
- Using a message digest enhanced with a key (so only a key-holder can check the MAC). The HMAC construction used by most TLS cipher suites is specified in RFC 2104 (SSL 3.0 used a different hash-based MAC).
- The message that ends the handshake ("Finished") sends a hash of all the exchanged handshake messages seen by both parties.

SSL/TLS Security Features con.

- The pseudorandom function splits the input data in half and processes each one with a different hashing algorithm (MD5 and SHA-1), then XORs them together to create the MAC. This provides protection even if one of these algorithms is found to be vulnerable. **TLS only.**
- SSL 3.0 improved upon SSL 2.0 by adding SHA-1 based ciphers and support for certificate authentication.
- From a security standpoint, SSL 3.0 should be considered less desirable than TLS 1.0.
 - The SSL 3.0 cipher suites have a weaker key derivation process; half of the master key that is established is fully dependent on the MD5 hash function, which is not resistant to collisions and is, therefore, not considered secure.
 - Under TLS 1.0, the master key that is established depends on both MD5 and SHA-1 so its derivation process is not currently considered weak. It is for this reason that SSL 3.0 implementations cannot be validated under FIPS 140-2

How SSL/TLS provides authentication

- During both client and server authentication there is a step that requires data to be encrypted with one of the keys in an asymmetric key pair and decrypted with the other key of the pair.
- For server authentication, the client uses the server's public key to encrypt the data that is used to compute the secret key. The server can generate the secret key only if it can decrypt that data with the correct private key.
- For client authentication, the server uses the public key in the client certificate to decrypt the data the client sends during step 5 of the handshake. The exchange of finished messages that are encrypted with the secret key (steps 7 and 8 on “SSL/TLS Handshake in Detail” slide 7) confirms that authentication is complete.
- If any of the authentication steps fail, the handshake fails and the session terminates.

How SSL/TLS provides authentication con.

- The exchange of digital certificates during the SSL handshake is part of the authentication process. The certificates required are as follows, where CA X issues the certificate to the SSL client, and CA Y issues the certificate to the SSL server
- For server authentication only,
 - the SSL server needs:
 - The personal certificate issued to the server by CA Y
 - The server's private key
 - And the SSL client needs:
 - The CA certificate for CA Y or the personal certificate issued to the server by CA

How SSL/TLS provides authentication con.

- If the SSL server requires client authentication, the server verifies the client's identity by verifying the client's digital certificate with the public key for the CA that issued the personal certificate to the client, in this case CA X. For both server and client authentication,
 - the SSL server needs:
 - The personal certificate issued to the server by CA Y
 - The server's private key
 - The CA certificate for CA X or the personal certificate issued to the client by CA X
 - And the SSL Client needs:
 - The personal certificate issued to the client by CA X
 - The client's private key
 - The CA certificate for CA Y or the personal certificate issued to the server by CA Y
- Both the SSL server and the SSL client might need other CA certificates to form a certificate chain to the root CA certificate.

How SSL/TLS provides confidentiality and integrity

- SSL uses a combination of symmetric and asymmetric encryption to ensure message privacy.
 - During the SSL handshake, the SSL client and SSL server agree an encryption algorithm and a shared secret key to be used for one session only.
 - All messages transmitted between the SSL client and SSL server are encrypted using that algorithm and key, ensuring that the message remains private even if it is intercepted.
 - Because SSL uses asymmetric encryption when transporting the shared secret key, there is no key distribution problem with SSL.
- SSL provides data integrity by calculating a message digest. A message digest might be effective against casual or indiscriminate tampering, but it does not prevent the more informed individual from changing or replacing the message, and generating a completely new digest for it.

CipherSuites and CipherSpecs

- A CipherSuite is a suite of cryptographic algorithms used by an SSL connection. A suite comprises three distinct algorithms:
 - The key exchange and authentication algorithm, used during the SSL handshake
 - The encryption algorithm, used to encipher the data
 - The MAC (Message Authentication Code) algorithm, used to generate the message digest
- There are several options for each component of the suite, but only certain combinations are valid when specified for an SSL connection. The name of a valid CipherSuite defines the combination of algorithms used. For example, the CipherSuite `SSL_RSA_WITH_RC4_128_MD5` specifies:
 - The RSA key exchange and authentication algorithm
 - The RC4 encryption algorithm, using a 128-bit key
 - The MD5 MAC algorithm

CipherSuites and CipherSpecs con.

- Several algorithms are available for key exchange and authentication, but the RSA algorithm is currently the most widely used. There is more variety in the encryption algorithms and MAC algorithms that are used.
- A CipherSpec identifies the combination of the encryption algorithm and MAC algorithm. Both ends of an SSL connection must agree the same CipherSpec to be able to communicate.
- Examples of CipherSuites include:
 - TLS_RSA_WITH_AES_128_CBC_SHA256
 - TLS_RSA_WITH_AES_256_CBC_SHA256
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_RC4_128_SHA

What is a digital certificate?

- Digital certificates used by organization should comply with the X.509 standard, which specifies the information that is required and the format for sending it (X.509 is the Authentication framework part of the X.500 series of standards. X.500 is the OSI Directory Standard).
- Digital certificates contain at least the following information about the entity being certified:
 - The owner's public key
 - The owner's Distinguished Name
 - The Distinguished Name of the CA that is issuing the certificate
 - The date from which the certificate is valid
 - The expiry date of the certificate
 - A version number
 - A serial number

What is a digital certificate?

- When you receive a certificate from a CA, the certificate is signed by the issuing CA with a digital signature. You verify that signature by using a CA certificate, from which you obtain the public key for the CA.
- You can use the CA public key to validate other certificates issued by that authority.
- Recipients of your certificate use the CA public key to check the signature.
- Digital certificates do not contain your private key. You must keep your private key secret.

Where would I use a Digital Certificate?

- You would use an SSL/TLS Certificate anywhere that you wish to transmit information securely. For example:
 - Securing communication between your web site and your customer's Internet browser.
 - Securing internal communications on your corporate intranet.
 - Securing email communications sent to and from your network (*or private email address*).
 - Securing information between servers (*both internal and external*).
 - Securing information sent and received via mobile devices.

Types of SSL/TLS Certificates

- There are a number of different SSL Certificates on the market today:
 - The first type of SSL Certificate is a **self-signed certificate**. This is a certificate that is generated for internal purposes and is *not* issued by a CA.
 - A **Domain Validated Certificate** is considered an entry-level SSL Certificate and can be issued quickly. The only verification check performed is to ensure that the applicant owns the domain (web site address) where they plan to use the certificate.
 - A **fully authenticated SSL Certificate** is the first step to true online security and confidence building. Taking slightly longer to issue, these certificates are only granted once the organization passes a number of validation procedures and checks to confirm the existence of the business, the ownership of the domain, and the user's authority to apply for the certificate.

Digital Certificates advantages

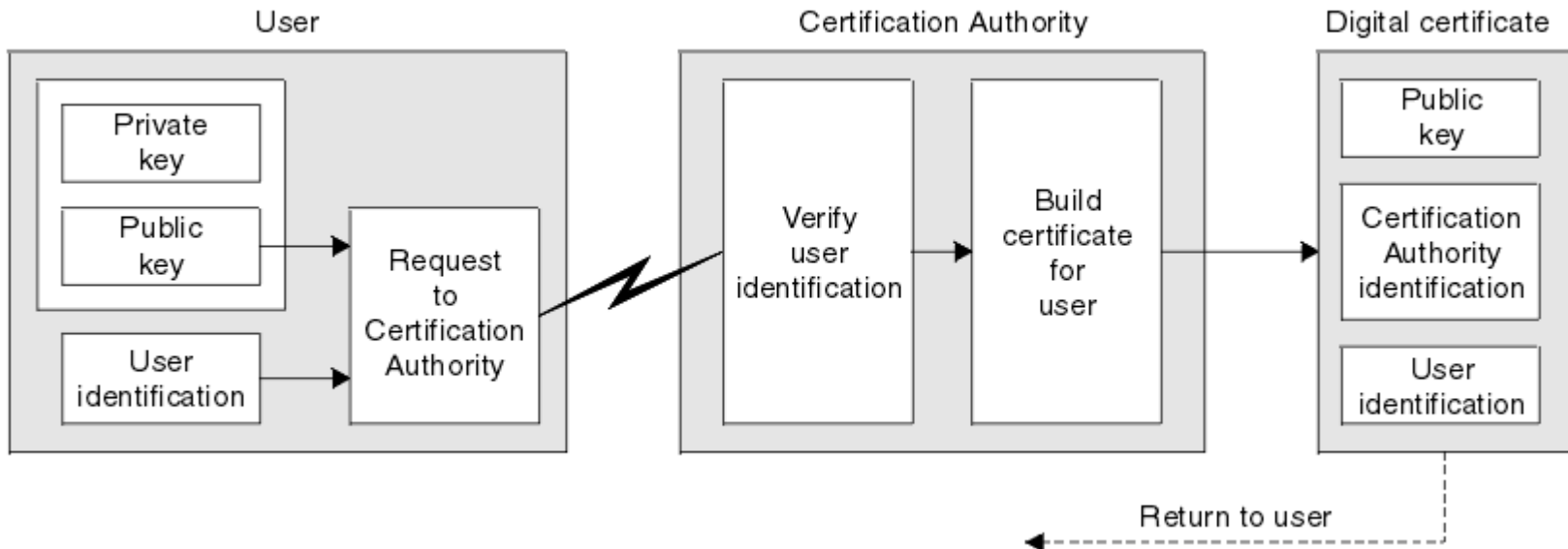
- Digital certificates provide protection against impersonation, because a digital certificate binds a public key to its owner, whether that owner is an individual, a queue manager, or some other entity.
- Digital certificates are also known as public key certificates, because they give you assurances about the ownership of a public key when you use an asymmetric key scheme.
- A digital certificate contains the public key for an entity and is a statement that the public key belongs to that entity:
 - When the certificate is for an individual entity, the certificate is called a *personal certificate* or *user certificate*.
 - When the certificate is for a Certification Authority, the certificate is called a *CA certificate* or *signer certificate*.

Digital Certificates con.

- If public keys are sent directly by their owner to another entity, there is a risk that the message could be intercepted and the public key substituted by another. This is known as a *man in the middle attack*.
- The solution to this problem is to exchange public keys through a trusted third party, giving you a strong assurance that the public key really belongs to the entity with which you are communicating.
- Instead of sending your public key directly, you ask the trusted third party to incorporate it into a digital certificate. The trusted third party that issues digital certificates is called a Certification Authority (CA).
- The roles of a CA are:
 - On receiving a request for a digital certificate, to verify the identity of the requestor before building, signing and returning the personal certificate
 - To provide the CA's own public key in its CA certificate
 - To publish lists of certificates that are no longer trusted in a Certificate Revocation List (CRL).

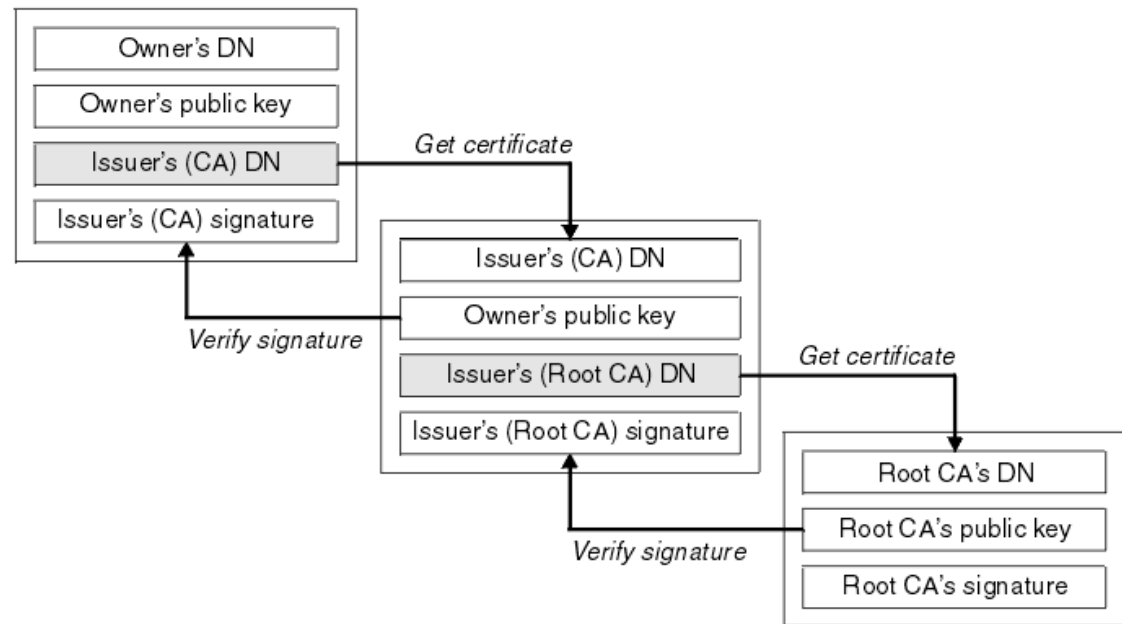
Obtaining digital certificates

When you obtain a certificate from a trusted external CA, you pay for the service. When you are testing your system, or you need only to protect internal messages, you can create self-signed certificates. These are created and signed by the certificate management tool your system uses. Self-signed certificates cannot be used to authenticate certificates from outside your organization.



How Certificates work

- When you receive the certificate for another entity, you might need to use a *certificate chain* to obtain the *root CA* certificate.
- The certificate chain, also known as the *certification path*, is a list of certificates used to authenticate an entity.
- The chain, or path, begins with the certificate of that entity, and each certificate in the chain is signed by the entity identified by the next certificate in the chain.
- The chain terminates with a root CA certificate.
- The root CA certificate is always signed by the CA itself.
- The signatures of all certificates in the chain must be verified until the root CA certificate is reached.



When Certificates Are No Longer Valid

- Digital certificates are issued for a fixed period and are not valid after their expiry date. Certificates can also become untrustworthy for various reasons, including:
 - The owner has moved to a different organization
 - The private key is no longer secret
- A Certification Authority can revoke a certificate that is no longer trusted by publishing it in a Certificate Revocation List (CRL).

Public Key Infrastructure (PKI)

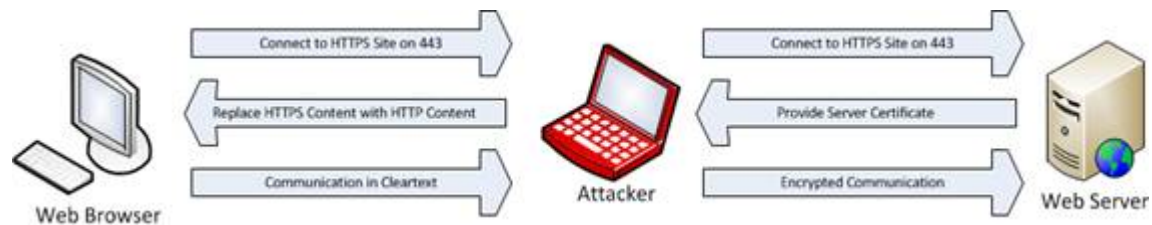
- A Public Key Infrastructure (PKI) is a system of facilities, policies, and services that supports the use of public key cryptography for authenticating the parties involved in a transaction. A PKI typically comprises Certification Authorities and other Registration Authorities (RAs) that provide the following services:
 - Issuing digital certificates
 - Validating digital certificates
 - Revoking digital certificates
 - Distributing public keys
- The X.509 standard is a Public Key Infrastructure.
- RAs verify the information provided when digital certificates are requested. If the RA verifies that information, the CA can issue a digital certificate to the requester.
- A PKI might also provide tools for managing digital certificates and public keys..

Security Vulnerabilities in SSL/TLS

- the **man-in-the-middle attack** (often abbreviated **MITM**), is a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker.
- The attacker must be able to intercept all messages going between the two victims and inject new ones, which is straightforward in many circumstances (for example, an attacker within reception range of an unencrypted Wi-Fi wireless access point, can insert himself as a man-in-the-middle).
- A man-in-the-middle attack can succeed only when the attacker can impersonate each endpoint to the satisfaction of the other—it is an attack on mutual authentication

Security Vulnerabilities in SSL/TLS: MiTM Example

- Traffic between the client and web server is intercepted.
- When an HTTPS URL is encountered the attacker replaces it with an HTTP link and keeps a mapping of the changes.
- The attacking machine supplies certificates to the web server and impersonates the client.
- Traffic is received back from the secure website and provided back to the client.



Security Vulnerabilities in SSL/TLS con.

- A proper web browsing client will warn the user of a certificate problems if any of the following are *not* true:
 - the certificate has been signed by a recognized certificate authority
 - the certificate is currently valid and has not expired
 - the common name on the certificate matches the DNS name of the server
- Defenses against this type of attack include PKI and stronger mutual authentication (i.e. secret keys, passwords)

Security Vulnerabilities in SSL/TLS con.

- SSL 2.0 is flawed in a variety of ways:
 - Identical cryptographic keys are used for message authentication and encryption.
 - SSL 2.0 has a weak MAC construction that uses the MD5 hash function with a secret prefix, making it vulnerable to length extension attacks.
 - SSL 2.0 does not have any protection for the handshake, meaning a [man-in-the-middle](#) downgrade attack can go undetected.
 - SSL 2.0 uses the TCP connection close to indicate the end of data. This means that truncation attacks are possible: the attacker simply forges a TCP FIN, leaving the recipient unaware of an illegitimate end of data message (SSL 3.0 fixes this problem by having an explicit closure alert).
 - SSL 2.0 assumes a single service and a fixed domain certificate, which clashes with the standard feature of virtual hosting in Web servers. This means that most websites are practically impaired from using SSL.
- SSL 2.0 is disabled by default in Internet Explorer (IE) 7, IE 8, IE9, Mozilla Firefox (FF) 2, FF 3, FF 4, Opera and Safari.

Security Vulnerabilities in SSL/TLS con.

- A vulnerability of the renegotiation procedure was discovered in August 2009 that can lead to plaintext injection attacks against SSL 3.0 and all current versions of TLS.
 - it allows an attacker who can hijack an https connection to splice their own requests into the beginning of the conversation the client has with the web server.
 - The attacker can't actually decrypt the client-server communication, so it is different from a typical man-in-the-middle attack.
 - A short-term fix is for web servers to stop allowing renegotiation, which typically will not require other changes unless client certificate authentication is used.
 - To fix the vulnerability, a renegotiation indication extension was proposed for TLS. It will require the client and server to include and verify information about previous handshakes in any renegotiation handshakes.
 - When a user doesn't pay attention to their browser's indication that the session is secure (typically a padlock icon), the vulnerability can be turned into a true man-in-the-middle attack.
 - This extension has become a proposed standard and has been assigned the number [RFC 5746](#)

Security Vulnerabilities in SSL/TLS con.

- There are some attacks against the implementation rather than the protocol itself:
 - In the earlier implementations, some CAs did not explicitly set basicConstraints CA=FALSE for leaf nodes. As a result, these leaf nodes could sign rogue certificates. In addition, some early software (including IE6 and Konqueror) did not check this field altogether. This can be exploited for man-in-the-middle attack on all potential SSL connections.
 - Some implementations (including older versions of Microsoft Cryptographic API, Network Security Services (NSS) and GnuTLS) stop reading any characters that follow the null character in the name field of the certificate, which can be exploited to fool the client into reading the certificate as if it were one that came from the authentic site, e.g. paypal.com\0.badguy.com would be mistaken as the site of paypal.com rather than badguy.com.
 - Browsers implemented SSL/TLS protocol version fallback mechanisms for compatibility reasons. The protection offered by the SSL/TLS protocols against a downgrade to a previous version by an active MITM attack can be rendered useless by such mechanisms.

BEAST

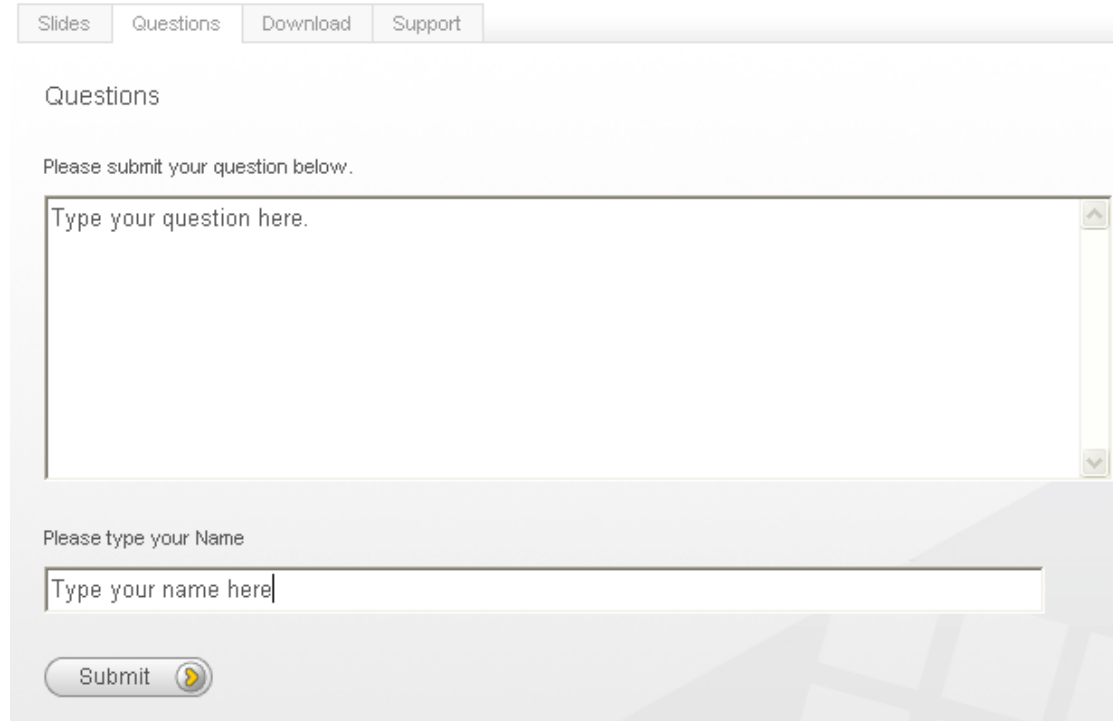
- On September 23, 2011 researchers Thai Duong and Juliano Rizzo demonstrated a "proof of concept" called BEAST (using a Java Applet to violate "same origin policy" constraints) for a long-known Cipher block chaining (CBC) vulnerability in TLS 1.0.
- Practical exploits had not been previously demonstrated for this vulnerability, which was originally discovered by Phillip Rogaway in 2002.
- Mozilla updated the development versions of their NSS libraries to mitigate BEAST-like attacks. NSS is used by Mozilla Firefox and Google Chrome to implement SSL. Some web servers that have a broken implementation of the SSL specification may stop working as a result.
- Microsoft released Security Bulletin MS12-006 on January 10, 2012, which fixed the BEAST vulnerability by changing the way that the Windows Secure Channel (SChannel) component transmits encrypted network packets.
- As a work-around, the BEAST attack can also be prevented by removing all CBC ciphers from one's list of allowed ciphers—leaving only the RC4 cipher, which is still widely supported on most websites. Users of Windows 7 and Windows Server 2008 R2 can enable use of TLS 1.1 and 1.2, but this work-around will fail if it is not supported by the other end of the connection and will result in a fall-back to TLS 1.0

Conclusions

- SSL/TLS provides an excellent (but not flawless) method to secure communications.
- Use at least SSLv3/TLSv1 implementations. Prohibit use of SSLv2 and below. Deprecate use of IE 6.
- Only use self-signed certificates for testing purposes
- Use domain validated certificates for communications between known organizations and for non-ecommerce use (i.e. VPNs, wireless)
- Update/Renew certificates every 1-2 years
- Only get fully authenticated certificates from known good sources.
- Stay up-to-date on known security vulnerabilities and apply patches and/or workarounds when needed to mitigate risk.

Questions?

Click on the questions tab on your screen, type in your question, name and e-mail address; then hit submit.



The screenshot shows a web interface with a navigation bar at the top containing four tabs: "Slides", "Questions", "Download", and "Support". The "Questions" tab is selected. Below the navigation bar, the heading "Questions" is displayed. A message reads "Please submit your question below." followed by a large text area with the placeholder text "Type your question here." and a vertical scrollbar on the right. Below the text area, the prompt "Please type your Name" is shown above a text input field with the placeholder "Type your name here". At the bottom of the form is a "Submit" button with a right-pointing arrow.