


PURDUE UNIVERSITY

Computer Security

CS 426

Lecture 24

Security of Distributed Systems



Elisa Bertino
Purdue University
IN, USA
bertino@cs.purdue.edu

Center for Education and Research
in Information Systems and Security

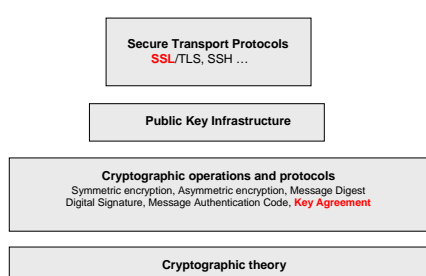
PURDUE UNIVERSITY

Topics

- Secure Socket Layer (SSL)
- Kerberos
- Firewalls and Intrusion Detection Systems

PURDUE UNIVERSITY

Framework



Secure Transport Protocols
SSL/TLS, SSH ...

Public Key Infrastructure

Cryptographic operations and protocols
Symmetric encryption, Asymmetric encryption, Message Digest
Digital Signature, Message Authentication Code, **Key Agreement**

Cryptographic theory

PURDUE UNIVERSITY

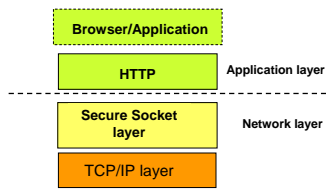
Secure Socket Layer (SSL)

A bit of history:

- SSL was originally developed by Netscape in 1996
- SSL 3.0 released on 1996
- It served as the basis for IETF RFC 2246 Transport Layer Security (TLS) proposed standard

SSL

SSL is a *protocol* layer which may be placed between a reliable connection-oriented network layer protocol (e.g. TCP) and the application protocol layer (e.g. HTTP).



SSL

SSL provides:

- **SSL server authentication** - allows a client to confirm a server's identity. SSL-enabled client software use public-key cryptography to check that a server's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the client's list of trusted CAs.
- **SSL client authentication (optional)** - allows a server to confirm a client's identity. Using the same techniques as those used for server authentication, SSL-enabled server software can check that a client's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the server's list of trusted CAs.
- **An encrypted SSL connection** between the client and the server software, thus providing a high degree of confidentiality. In addition, all data sent over an encrypted SSL connection is protected with a mechanism for detecting tampering, that is, for automatically determining whether the data has been altered in transit.

SSL sessions and connections

- A session is an association between a client and a server
- To exchange data, the two parties must establish a connection (a transport in the OSI model):
 - The two parties exchange two random numbers and create (using the master secret) the keys and the parameters needed for exchanging messages
- A session can consist of many connections
- A session avoids the expensive negotiation of new security parameters for each connection
- After a session is established, the two parties have the following common information:
 - The session identifier
 - The certificate(s) authenticating each of them
 - The compression method (if needed)
 - The cipherSuite
 - The master secret (used to create keys for message encryption)

SSL CipherSuites

A CipherSuite includes 3 components:

- Key Exchange Protocol
 - defines how the shared secret symmetric cryptography key used for application data transfer will be agreed upon by client and server
- Symmetric encryption algorithm
- Message Digest (for creating the Message Authentication Code)

Key-exchange protocols

- A key exchange protocol is cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communication channel
- SSL supported key-exchange protocols
 - RSA (the secret key is encrypted with the receiver's RSA public key. This requires that a public key certificate for the receiver's key must be made available)
 - Fixed Diffie-Hellman - the server certificates contains the public parameters of the Diffie-Hellman protocol, i.e. two integers p and g (see next slide)
 - Ephemeral Diffie-Hellman
 - Anonymous Diffie-Hellman
 - Fortezza
 - KEA - Key Exchange Algorithm, an algorithm used for key exchange by the U.S. Government

Diffie-Hellman protocol

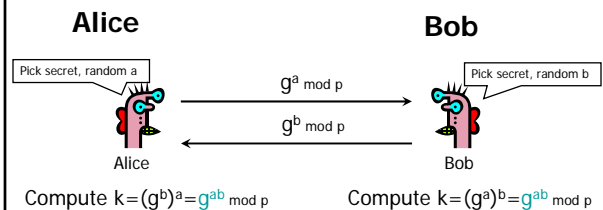
- The protocol has two system parameters p and g
- p and g are both public and may be used by all the users in a system
- parameter p is a large prime number
- parameter g is a generator of Z_p^*

$$Z_p^* = \{1, 2, \dots, p-1\}; \forall a \in Z_p^* \exists i \text{ such that } a = g^i \pmod p$$

Diffie-Hellman protocol

- Alice and Bob want to agree on a shared secret key using the Diffie-Hellman key agreement protocol. They proceed as follows:
 - First, Alice generates a *random private* value a and Bob generates a *random private* value b . Both a and b are drawn from the set of integers
 - Then they derive their public values using parameters p and g and their private values:
 - Alice's public value is $g^a \pmod p$, and
 - Bob's public value is $g^b \pmod p$
 - They then exchange their public values
 - Finally:
 - Alice computes $g^{ab} = (g^b)^a \pmod p$, and
 - Bob computes $g^{ab} = (g^a)^b \pmod p$
 - Since $g^{ab} = g^{ba} = k$, Alice and Bob now have a shared secret key k

Diffie-Hellman protocol



Note: a key of the form g^a is called a Diffie-Hellman key

Diffie-Hellman protocol

Three variations:

- fixed Diffie-Hellman
 - the server has fixed DH parameters contained in a certificate signed by a CA
 - the client may have fixed DH parameters certified by a CA or it may send an unauthenticated one-time DH public value in the client_key_exchange message
- ephemeral Diffie-Hellman
 - both the server and the client generate one-time DH parameters
 - the server signs its DH parameters with its private RSA key
 - the client may authenticate itself (if requested by the server) by signing the hash of the handshake messages with its private RSA key
- anonymous Diffie-Hellman
 - both the server and the client generate one-time DH parameters
 - they send their parameters to the peer without authentication

SSL supported Encryption algorithms

- DES (40 & 56 bits) - Data Encryption Standard, an encryption algorithm used by the U.S. Government
- Triple-DES (168 bits) - DES applied three times
- Fortezza
- RC4(40 & 128 bits) - Rivest encryption ciphers developed for RSA Data Security

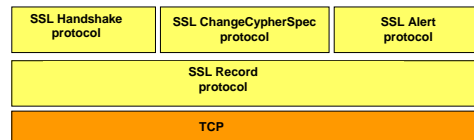
SSL supported Cryptographic algorithms

- Administrators can enable or disable any of the supported cipher suites for both clients and servers
- When a particular client and server exchange information during the SSL handshake, they identify the strongest enabled cipher suites they have in common and use those for the SSL session
- Which cipher suites should be enabled?
 - it depends on trade-offs among the sensitivity of the data involved, and the speed of the cipher

SSL protocol

SSL protocol is composed by 2 protocol layers

- the SSL Handshake, Change Cypher and Alert protocol, used in the management of SSL exchanges
- the SSL Record Protocol



SSL Handshake Protocol at glance

- The SSL Handshake protocol:
 - establishes a session between the client and the server, i.e. the client and the server negotiate the creation of symmetric keys used for encryption, decryption, and tamper detection during the session that follows
- In the SSL Handshake protocol:
 - the server authenticates itself to the client using public-key techniques
 - Optionally, the handshake protocol also allows the client to authenticate itself to the server

SSL Handshake Protocol at glance

The SSL ChangeCipherSpec Protocol is used to inform the parties that the Handshake is concluded and they can begin to exchange messages using the agreed ciphersuite

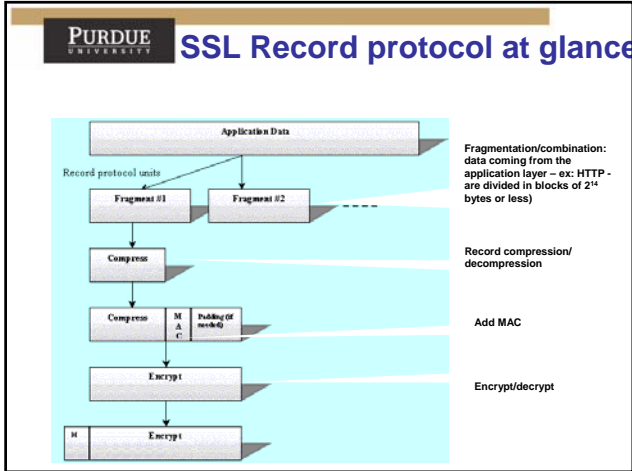
SSL Alert Protocol at glance

The SSL Alert Protocol is used for reporting errors and abnormal conditions

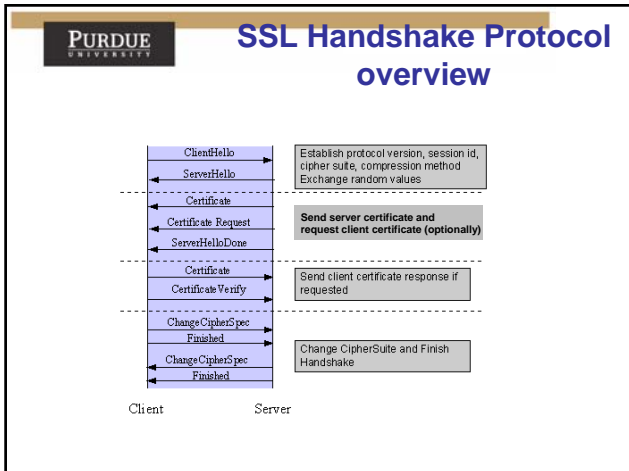
SSL Record protocol at glance

It provides:

- Fragmentation (data coming from the application layer – ex: HTTP - are divided in blocks of 2^{14} bytes or less)
- Record compression and decompression
- Record payload protection
 - All records are protected using the encryption and MAC algorithms defined in the current CipherSpec (the one agreed in the Handshake protocol)



- ## SSL Handshake Protocol
- The SSL Handshake protocol is initiated when a client and a server first start communicating
 - The SSL Handshake Protocol:
 - Allows to negotiate an encryption algorithm and cryptographic keys (CipherSuite) before the application protocol transmits or receives its first byte of data
 - When a particular client and a server exchange information during the SSL handshake, they identify the strongest enabled cipher suites they have in common and use those for the SSL session
 - Authenticates the server to the client
 - Optionally authenticates the client to the server
 - Establishes an encrypted SSL connection



- ## SSL Handshake Protocol steps
- Client Hello:** The client sends the server the client's SSL version number, cipher settings, randomly generated data, and other information the server needs to communicate with the client using SSL.

Example of randomly generated data (Unix env):

```
struct { uint32 gmt_unix_time; opaque random_bytes[28]; } Random;
```

 - gmt_unix_time**
 - The current time and date in standard UNIX 32-bit format according to the sender's internal clock. Clocks are not required to be set correctly by the basic SSL Protocol; higher level or application protocols may define additional requirements.
 - random_bytes**
 - 28 bytes generated by a secure random number generator.

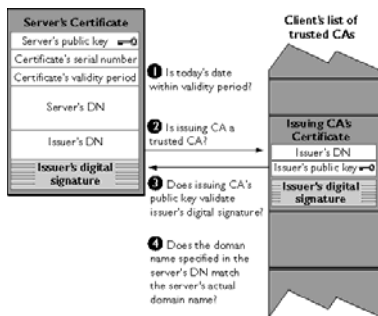
SSL Handshake Protocol steps

2. Server Hello: The server sends the client the server's chosen SSL protocol version number, randomly generated data, cipher suite, and compression method from the choices offered by the client. The server also sends its own certificate and, if the client is requesting a server resource that requires client authentication, requests the client's certificate.

SSL Handshake Protocol steps

3. The client authenticates the server. If the server cannot be authenticated, the application 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 goes on to Step 4.

More on Server Authentication (by Client)



X.500 Distinguished Name

DN Field	Abbrev.	Description	Example
Common Name	CN	Name being certified	CN=Joe Average
Organization or Company	O	Name is associated with this organization	O=Snake Oil, Ltd.
Organizational Unit	OU	Name is associated with this organization unit, such as a department	OU=Research Institute
City/Locality	L	Name is located in this City	L=Snake City
State/Province	ST	Name is located in this State/Province	ST=Desert
Country	C	Name is located in this Country (ISO code)	C=XZ

SSL Handshake Protocol steps

- Using all data generated in the handshake so far, the client (with the cooperation of the server, depending on the cipher being used) creates the premaster secret for the session, encrypts it with the server's public key (obtained from the server's certificate, sent in Step 2), and sends the encrypted premaster secret to the server

SSL Handshake Protocol steps

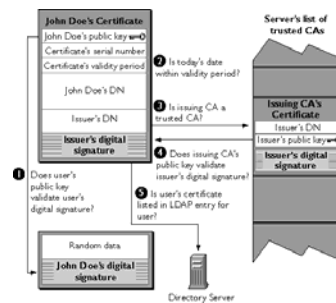
- If the Server requires to authenticate the Client:
 - the client sends the server both a certificate and a separate piece of digitally signed data to authenticate itself

The SSL protocol requires the client to create a digital signature by creating a one-way hash from data generated randomly during the handshake and known only to the client and server. The hash of the data is then encrypted with the private key that corresponds to the public key in the certificate being presented to the server

SSL Handshake Protocol steps

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

More on Client Authentication (by Server)



SSL Handshake Protocol steps

7. Both the client and the server use the master secret to generate the session keys
Session keys - symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity

SSL Handshake Protocol steps

8. The client sends a message to the server informing it that future messages from the client will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the client portion of the handshake is finished.

SSL Handshake Protocol steps

9. The server sends a message to the client informing it that future messages from the server will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the server portion of the handshake is finished.

SSL Handshake Protocol steps

10. The SSL handshake is now complete, and the SSL session begins. The client and the server use the session keys to encrypt and decrypt the data they send to each other and to validate its integrity

SSL authentication – CA vs. self-signed certificates

- The SSL authentication process uses certificates that are issued by a certificate authority. The same process applies if the certificates are issued by a certificate generation utility or if self-signed certificates are used (ref: keytool utility in Java environment)
- A self-signed certificate contains:
 - a public key, information about the owner of the certificate, and the owner's signature.
 - It has an associated private key, but it does not verify the origin of the certificate through a third-party certificate authority (CA)

Self-signed certificates

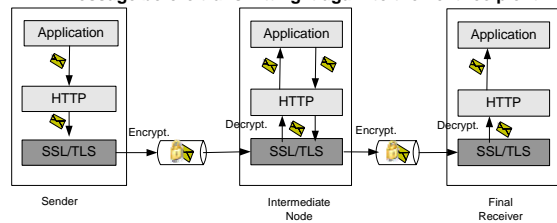
- When and how to use self-signed certificates?
 - It depends on security requirements
 - To achieve the highest level of authentication between critical software components, do not use self-signed certificates.
- Self-signed certificates can be used to test an SSL configuration before creating and installing a signed certificate issued by a certificate authority

Key points: what SSL provides

- SSL provides:
 - server authentication to the client and optionally client authentication to the server
 - confidentiality using symmetric encryption
 - message integrity using a message authentication code
- SSL includes protocol mechanisms to enable the client and the server (both are TCP users) to determine the security mechanisms and services they will use

Key points: what SSL does not provide

- **No end-to-end communication protection**
- **At the receiving end the message is delivered decrypted to the application layer:**
 - The intermediary application/service might then, inadvertently or maliciously, examine or even modify the message before transmitting it again to the next recipient.



- Material to read

- **SSL tutorials:**

1. <http://www2.rad.com/networks/2001/ssl/index.htm>
2. http://httpd.apache.org/docs/2.2/ssl/ssl_intro.html

References

1. IETF RFC 4246 The Transport Layer Security (TLS) Protocol Version 1.1 <http://tools.ietf.org/html/rfc4346>
2. IETF RFC 3546 Transport Layer Security (TLS) Extensions <http://tools.ietf.org/html/rfc3546>