Transport Layer Security Protocols

- Secure Socket Layer (SSL)
  - Originally developed to secure http
  - Version 3 was developed with public review
  - Application independent
    - Can be used for any application protocol
    - Examples: telnet, pop3, imap, ftp, etc.
- Transport Layer Security (TLS)
  - TLS 1.0 very close to SSL 3.1
  - Backward compatible with SSL v3.
SSL Architecture

- A two-layered protocol.
- Rely on TCP for a reliable communication.

<table>
<thead>
<tr>
<th>SSL Handshake Protocol</th>
<th>SSL Change Cipher Spec Protocol</th>
<th>SSL Alert Protocol</th>
<th>HTTP and other protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL Record Protocol</td>
<td>TCP</td>
<td>IP</td>
<td></td>
</tr>
</tbody>
</table>

SSL Protocol Stack

SSL Services

- Peer entity and data authentication
- Data confidentiality
- Data integrity
- Compression/decompression
- Generation/distribution of session keys
  - Integrated to protocol
  - A different approach from IPSec
- Security parameter negotiation.
SSL Connection and Session

- Each SSL session can be used for multiple SSL connections.
- SSL Session
  - An association between a client and a server.
  - Created by handshake protocol.
  - Are used to avoid negotiation of new security parameters for each connection.
- SSL Connection
  - A connection is a transport that provides a suitable type of service.
  - Peer-to-peer, transient
  - Each connection is associated with one session.

SSL Session

- We can view an SSL session as an SSL security association.
- A SSL session consists of
  - Session ID
  - X.509 public-key certificate of peer (could be null)
  - Compression algorithm
  - Cipher spec:
    - Encryption algorithm, message digest algorithm, etc.
  - Master secret: 48 byte secret shared between the client and server
  - Is reusable
SSL Connection

• An SSL Connection consists of
  – Server and client random
  – Server write MAC secret
  – Client write MAC secret
  – Server write key
  – Client write key
  – Server IV
  – Client IV
  – Sequence number

SSL Record Protocol

• Four steps by sender (reversed by receiver)
  – Fragmentation
    • $2^{14}$ bytes
  – Compression (optional)
  – MAC
  – Encryption
SSL Record Protocol Operation

SSL Record Format
## SSL Record Protocol Payload

<table>
<thead>
<tr>
<th>1 byte</th>
<th>3 bytes</th>
<th>≥ 0 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Length</td>
<td>Content</td>
</tr>
</tbody>
</table>

### (a) Change Cipher Spec Protocol

### (b) Alert Protocol

<table>
<thead>
<tr>
<th>1 byte</th>
<th>1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
<td>Alert</td>
</tr>
</tbody>
</table>

### (c) Handshake Protocol

<table>
<thead>
<tr>
<th>≥ 1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpaqueContent</td>
</tr>
</tbody>
</table>

### (d) Other Upper-Layer Protocol (e.g., HTTP)

---

## Handshake Protocol

- Initially SSL session has null compression and encryption algorithm.
- Both are set by the handshake protocol at the beginning of session.
- Handshake protocol may be repeated during the session.
- Four phases
  - Establish Security Capabilities
  - Server Authentication and Key Exchange
  - Client Authentication and Key Exchange
  - Finish
Phase 1. Establish Security Capabilities

Client | Server

- Client_hello*
- Server_hello*

Message marked by * are mandatory; Other messages are optional.

Phase 1 (Cont’d)

- **Client_hello**
  - Version: The highest SSL version understood by the client
  - Random: 4-byte timestamp + 28-byte random number.
  - Session ID: zero for new session, non-zero for a previous session
  - CipherSuite: list of supported algorithms
  - Compression Method: list of supported compression methods
Phase 1 (Cont’d)

- **Server_hello**
  - Version: min (client_hello version, highest version supported by the server)
  - Random: 4-byte timestamp + 28-byte random number.
    - Generated by the server
  - Session ID:
  - CipherSuite: selected from the client’s list by the server
  - Compression method: selected from the client’s list by the server

Phase 2: Server Authentication and Key Exchange

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate</td>
<td>Server_key_exchange</td>
</tr>
<tr>
<td>Certificate_request</td>
<td></td>
</tr>
<tr>
<td>Server_done*</td>
<td></td>
</tr>
</tbody>
</table>

Certificate is almost always used.
Certificate message

- Required for any agreed-on key exchange method except for anonymous Diffie-Hellman.
  - Anonymous D-H
    - Problem?
- Contains one or a chain of X.509 certificates.

Server_key_exchange message

- Not required if
  - The server has sent a certificate with fixed D-H parameters, or
  - RSA key exchange is to be used.
- Needed for
  - Anonymous D-H
  - Ephemeral D-H
  - RSA key exchange, in which the server is using RSA but has a signature-only RSA key.
  - Fortezza
Certificate_request message

- Request a certificate from the client
- Two parameters
  - Certificate_type
    - RSA, signature only
    - DSS, signature only
    - ...
  - CertificateAuthorities

Server_done message

- Indicate the end of server hello and associated messages.
Phase 3. Client Authentication and Key Exchange

- **Certificate**
  - One or a chain of certificates.

- **Client_key_exchange**
  - RSA: encrypted pre-master secret with the server’s public key.
  - D-H: client’s public key.

- **Certificate_verify**
  - Only sent following any client certificate that has signing capability
  - Proves the client is the valid owner of the certificate.
Phase 4. Finish

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text*{\textit{Change_cipher_spec}}</td>
<td>\text*{\textit{Finished}}</td>
</tr>
<tr>
<td>\text*{\textit{Finished}}</td>
<td>\text*{\textit{Change_cipher_spec}}</td>
</tr>
</tbody>
</table>

Master Secret Creation

- The master secret is a one-time 48-byte value.
  - Pre-master secret: by RSA or D-H
  - Master secret is computed from the pre-master secret, client random and server random.
Generation of Cryptographic Parameters

- Generated from the master secret, client random, and server random.
  - Client write MAC secret
  - Server write MAC secret
  - Client write key
  - Server write key
  - Client write IV
  - Server write IV

Change Cipher Spec Protocol

- Session State
  - Current state
    - The session state in effect
  - Pending state
    - The session being negotiated.

- Change Cipher Spec Protocol
  - Cause the pending state to be copied into the current state.
Alert Protocol

• Convey SSL related alerts to the peer.
• Compressed and encrypted.
• Two types of alerts
  – Fatal
    • SSL immediately terminates the connection.
    • Examples
      – Unexpected message
      – Bad_record_mac
  – Warning
    • Examples
      – Close_notify
      – No_certificate

Application Ports Used with SSL

• https  443
• smtps  465
• nntps  563
• ldaps  636
• ldaps  636
• pop3s  995
• ftp-datas  889
• ftplib  990
• imaps  991