Announcements

• Project 3 has been graded:
  – Average: 89
  – Std. Dev.: 5.8

• Reminder:
  – Final Exam. Tuesday, Dec. 21st
  – 11:00-1:00
  – Towne 311

• Course evaluations will be given out on Friday
  – Please attend class and give me feedback.
Transport Layer Security (TLS)

- Designed by the IETF (Internet Engineering Task Force)
  - Version 1.0 proposed in 1999
- Based on SSL v3.0 (Secure Socket Layer)
- Motivated by web security (HTTPS)
Secure Socket Layer (SSL)

- Session protocol with:
  - Server authentication.
  - Client authentication optional.
  - Integrity checksum.
  - Confidentiality.

- Possibly the most important security-related e-commerce protocol.

- Connection sets up security parameters.
- Many sessions possible within a given connection.
# SSL Protocol Stack

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

<table>
<thead>
<tr>
<th>Content Type</th>
<th>Major Version</th>
<th>Minor Version</th>
<th>Length</th>
</tr>
</thead>
</table>

Payload
Content Types

- Handshake: 1 byte, 3 bytes, ≥ 0 bytes
- Change Cipher Spec: 1 byte
- Alert: 1 byte, 1 byte
- Opaque Content: ≥ 1 bytes

Higher-Level Protocol Content
What Is Needed?

• Negotiation of cryptographic protocols.
• Initial authentication.
• Key exchange to set up:
  – Bulk encryption.
  – Message Authenticity Codes (MACs).
How is this Done?

- The handshake protocol negotiates the protocols to be used for authentication and bulk encryption.
- The handshake protocol carries out initial authentication.
- The handshake protocol establishes a 48 byte pre-master secret.
  - Client and server use this to derive a 48 byte master secret.
  - The master secret is used to generate a key block of sufficient length to supply all needed cryptographic parameters.
SSL Negotiated Parameters

• Key Exchange Protocols
  – RSA
  – Fixed Diffie-Hellman
  – Ephemeral Diffie-Hellman
  – Anonymous Diffie-Hellman
  – Fortezza

• SSL Bulk Encryption Protocols
  – Protocol (type, key length)
    • IDEA (block, 128)
    • RC2-40 (block, 40)
    • DES-40 (block, 40)
    • DES (block, 56)
    • 3DES (block, 168)
    • Fortezza (block, 80)
    • RC4-40 (stream, 40)
    • RC4-128 (stream, 128)
SSL Handshake Protocol Phases

- Establish Security Capabilities
- Server Authentication and Key Exchange
- Client Authentication and Key Exchange
- Finish
Establish Security Capabilities

Client Hello

Client Hello

Server Hello

Client

Server

Time
Client Hello

1. Highest version number understood by client.
2. 32-bit timestamp and 28-bit nonce.
3. Session identifier.
   - Nonzero value: update parameters of existing connection.
   - Zero value: new connection and session.
4. CipherSuite list in order of preference.
   - Identifies a key exchange protocol and other parameters.
5. List of compression methods.
CipherSuite

- BulkCipherAlgorithm
  - as given earlier
- MACAlgorithm
  - SHA-1 or MD5
- CipherType
  - stream or block
- IsExportable
  - true or false
- HashSize:
  - 0, 16 (for MD5), or 20 (for SHA-1) bytes
Server Hello

1. Highest version number supplied by client and acceptable to server.
2. Time and nonce from server
   - independent of same from client
3. Session identifier.
   - If non-zero from client then same value from server.
   - Otherwise proposed by server.
4. First cipher suite proposed by client and supported by server.
5. First compression method proposed by client and supported by server.
Server Auth & Key Exchange

Client

Server

Time

Certificate

Server Key Exchange

Certificate Request

Server Hello Done

Optional

CSE331 Fall 2004
x.509 Certificates

• List of x.509 certificates.
  – Originally intended to support access control for the directory as part of the Directory Access Protocol (DAP).
  – Dominant candidate now for PKI to support electronic commerce, although adoption has been slow.

• Required for all protocols except Anonymous Diffie-Hellman.
• http://www.ietf.org/html.charters/pkix-charter.html
X.509 Certificates

X.509 certificates bind a subject to a public key. This binding is signed by a Certificate Authority (CA).

- Subject Name
- Subject Public Key
- CA Name
- CA Signature
Server Key Exchange

- Protocol-dependent keying material.
- Needed for all except RSA and fixed Diffie-Hellman.
- Example: in anonymous Diffie-Hellman this message consists of a prime number, a primitive root for it, and the Diffie-Hellman public key for the server.
Other Server Messages

• Certificate Request
  – Requests a certificate type
  – Specifies a list of certificate authorities.

• Server Hello Done
  – Indicates that all messages in the server authentication phase have been sent and server is now awaiting a client response.
Client Auth & Key Exchange

Client

Server

Certificate

Client Key Exchange

Certificate Request

Optional

Optional

Time
Client response

- **Certificate**
  - Client sends an X.509 certificate as requested by server or sends a No Certificate Alert.

- **Client key exchange**
  - Protocol-dependent keying material.
  - Example: In RSA the client generates a 48 byte secret and encrypts it using the public key of the server (from the Server Key Exchange message).

- **Certificate Verification**
  - Contains explicit verification of client certificate.
  - Client signs the hash of a concatenation of the master secret and handshake messages.
Client Auth & Key Exchange

Client

Server

Time

Change Cipher Spec

Finish

Change Cipher Spec

Finish
Client and Server

- The Change CipherSpec message causes the CipherSpec to become active. This is not a handshake record, but a record in the Change CipherSpec protocol.
- Each party sends the Finished message under the new CipherSpec.
- The message contains hashes (using MD5 or SHA-1) of the master secret and the handshake messages.
Master Secret

- The Master Secret is computed from the Pre-Master Secret as follows:

\[
\text{MasterSecret} = \\
\text{MD5} (\text{PreMasterSecret} \ , \ \text{SHA}(“A” \ , \ \text{PreMasterSecret} \ , \ \text{ClientHello.random} \ , \ \text{ServerHello.random})) \ , \\
\text{MD5} (\text{PreMasterSecret} \ , \ \text{SHA}(“BB” \ , \ \text{PreMasterSecret} \ , \ \text{ClientHello.random} \ , \ \text{ServerHello.random})) \ , \\
\text{MD5} (\text{PreMasterSecret} \ , \ \text{SHA}(“CCC” \ , \ \text{PreMasterSecret} \ , \ \text{ClientHello.random} \ , \ \text{ServerHello.random}))
\]
Cryptographic Parameters

• The remaining cryptographic parameters are computed by iterating the following pattern.

KeyBlock =
MD5( MasterSecret , SHA(“A”, MasterSecret ,
ClientHello.random , ServerHello.random)),
MD5( MasterSecret , SHA(“BB”, MasterSecret ,
ClientHello.random , ServerHello.random)),
MD5( MasterSecret , SHA(“CCC”, MasterSecret ,
ClientHello.random , ServerHello.random)),,
Creating Opaque Content

• Begin with application data.
• Fragment it into blocks of $2^{14}$ bytes or less.
• Optionally compress the fragments.
• Add a message authentication code (MAC) to the compressed data to ensure integrity.
• Encrypt the data to ensure confidentiality.
• Add SSL record header (4 fields).
MAC Calculation

- A MAC write secret, **MACWS**, is extracted from the key block.
- A hash function **H** is selected: either MD5 or SSH.
- The **MAC** is defined as follows.

\[
H(\text{MACWS}, \text{pad2}, \\
H(\text{MACWS}, \text{pad1}, \text{seq_no}, \text{SSLCompressed}.type, \\
\text{SSLCompressed}.length, \text{SSLCompressed}.fragment))
\]
Alerts

- Unexpected message.
- Bad MAC.
- Decompression failure.
- Handshake failure, unable to negotiate an acceptable set of security protocols.
- Illegal parameter in handshake message.
- Close notify.
Alerts, Continued

- No certificate.
- Bad certificate.
- Unsupported certificate.
- Certificate revoked.
- Certificate expired.