Secure Socket Layer (SSL) / TLS (Transport Layer Security)

Haipeng Dai

haipengdai@nju.edu.cn
313 CS Building
Department of Computer Science and Technology
Nanjing University
SSL/TLS

- By far, SSL/TLS is the dominant security technology on the web
- Provide transport layer security
  - Any TCP/IP application can be protected by using SSL. For example, FTP communication or Telnet or HTTP communication can be protected by using SSL.
  - HTTPS is HTTP over SSL
- Responsible for the emergence of e-commerce, other security sensitive services on the web
- Beneficiary of several years of public scrutiny
SSL / TLS in the Real World

Account Summary

<table>
<thead>
<tr>
<th>Wells Fargo Accounts</th>
<th>OneLook Accounts</th>
</tr>
</thead>
</table>

Tip: Select an account's balance to access the Account History.

Extended Tables

<table>
<thead>
<tr>
<th>Account</th>
<th>Account Number</th>
<th>Available Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checking</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To end your session, be sure to Sign Off.

© 1995 - 2003 Wells Fargo. All rights reserved.
History of the Protocol

- **SSL 1.0**
  - Internal Netscape design. Never deployed.

- **SSL 2.0**
  - Published by Netscape, November 1994
  - Several weaknesses

- Microsoft improved upon SSL v2, and call it PCT (Private Communication Technology).

- **SSL 3.0**
  - Designed by Netscape and Paul Kocher, November 1996
  - SSL v3 is deployed in nearly every Web browser.

- **TLS (Transport Layer Security) 1.0** by IETF.
  - Internet standard based on SSL 3.0, January 1999
  - Not interoperable with SSL 3.0
RFC: “Request for Comments”

- Network protocols are usually disseminated in the form of an RFC
- TLS version 1.0 is described in RFC 2246
- Intended to be a self-contained definition of the protocol
  - Describes the protocol in sufficient detail for readers who will be implementing it and those who will be doing protocol analysis
  - Mixture of informal prose and pseudo-code
Why SSL? SSL Provides ...

- Confidentiality (Privacy)
- Data integrity (Tamper-proofing)
  - done as part of digital signing.
- Server authentication (Proving a server is what it claims it is)
- Used in typical B2C transactions
- Optional client authentication
  - Would be required in B2B (or Web services environment in which program talks to program)
  - Why not B2C: server use passwords to authenticate clients, not certificates. Clients normally do not have certificates.
TLS Basics

- TLS consists of two protocols
  - Handshake protocol
    - Use public-key cryptography to establish a shared secret key between the client and the server
  - Record protocol
    - Use the secret key established in the handshake protocol to protect communication between the client and the server
- We will focus on the handshake protocol
TLS Handshake Protocol

- Two parties: client and server
- Negotiate version of the protocol and the set of cryptographic algorithms to be used
  - Interoperability between different implementations of the protocol
- Authenticate server and client (optional)
  - Use digital certificates to learn each other’s public keys and verify each other’s identity
- Use public keys to establish a shared secret
Is this vulnerable to man-in-the-middle attacks?
— Yes.
Man-in-the-middle Attack on the Early Version of SSL

Alice

I am Robert

{K_{ar}}_{PUR}

Bob

I am Alice

{K_{rb}}_{PUB}

Robert

{n}_{K_{ar}}

{n}_{K_{rb}}

{Certificate}_{Alice, \{n\}_{PRA}}_{K_{ar}}

{Certificate}_{Alice, \{n\}_{PRA}}_{K_{rb}}
SSL version 3 / TLS

- Why client sends “ciphers I support”? Also, why clients sends highest version of its SSL implementation?
  - Client and server may have different implementations running on different systems.
What’s the purpose of server’s certificate?

- Allow client to authenticate the server.
- Allow client to know server’s public key.
SSL version 3 / TLS

Who make the decision on the cipher suite?

- SSL v2: Client makes the decision. So, server sends back a list of cipher suites that it supports.
- This is silly because client already tell server all the cipher suites that it has.
- SSL v3: Server makes the decision. So, server only needs to send back the cipher suite that it chooses.
SSL version 3 / TLS

- Why HMAC all previous handshake messages?
Why HMAC all previous handshake messages?

- To prevent “cipher downgrade” and “version downgrade” attacks
  - Attackers removes stronger crypto algorithms and change SSL from Version 3 to 2

- Why do people release new versions of security protocols?
  - Because the old version got broken!

- New version must be **backward-compatible**
  - Not everybody upgrades right away

- Attacker can fool someone into using the old, broken version and exploit known vulnerability
  - Similar: fool victim into using weak crypto algorithms

- Defense is hard: must authenticate version early

- Many protocols had “version rollback” attacks
  - SSL, SSH, GSM (cell phones)

- HMAC on all previous handshake messages prevents that.
Why add “CLNT” and “SRVR” in HMAC?

— Otherwise the two message will be the same!
— If the HMAC message is replayed to the client, although the server can detect that the integrity of handshake messages has been compromised, the client cannot. Thus, attackers can change “cipher I choose” to a weak one, then the client may send sensitive data encrypted using a weak encryption algorithm. The first few packets may be decrypted by attackers.
How to ensure $n_c$ and $n_v$ never repeat?

- First 4 bytes are Unix time (seconds since Jan. 1, 1970).
SSL version 3 / TLS

- Why need the “finish” message?
  - TCP connection close message is not encrypted.
  - An attacker can close the connection by sending bogus TCP close message. Thus, C and S will not know that the connection was abnormally closed.
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites;
    CompressionMethod compression_methods;
} ClientHello
ClientKeyExchange (RFC)

struct {
    select (KeyExchangeAlgorithm) {
        case rsa: EncryptedPreMasterSecret;
        case diffie_hellman: ClientDiffieHellmanPublic;
    } exchange_keys
} ClientKeyExchange

struct {
    ProtocolVersion client_version;
    opaque random[46];
} PreMasterSecret

Random bits from which symmetric keys will be derived (by hashing them with nonces)
SSL 2.0 Weaknesses (Fixed in 3.0)

- Cipher suite preferences are not authenticated
  - “Cipher suite rollback” attack is possible
- Weak MAC construction
- SSL 2.0 uses padding when computing MAC in block cipher modes, but padding length field is not authenticated
  - Attacker can delete bytes from the end of messages
- No support for certificate chains or non-RSA algorithms, no handshake while session is open
SSL/TLS Record Protection - Integrity

Use symmetric keys established in handshake protocol
Session Resumption

- SSL/TLS assumes that a session is a long-lived thing.
- This is because SSL was designed to work with HTTP 1.0, which opens many TCP connections between the same client and server (one item on the page per connection).
Session Resumption

- If the server allows clients to have multiple connections based on the same SSL session, the server sends the client a session-ID, and then stores (session-ID, K).

\[ K = f(S, n_c, n_v) \]

\[ \{S\}_{PUV} \cdot \text{HMAC}(K, "CLNT", \text{handshake messages}) \]

\[ \text{HMAC}(K, "SRVR", \text{handshake messages}) \]

\[ \{\text{data}\}_K \]

\[ \{\text{finish}\}_K \]
Session Resumption

- When the client build a new connection based on the same SSL session:

  **Session-ID**, Ciphers I support, SSL highest version,  \( n_c \)

  **Session-ID**, Certificate, Ciphers I choose, SSL highest version,  \( n_v \),

  \( \text{HMAC}(K, \text{"SRVR"}, \text{handshake messages}) \)

  \( \text{HMAC}(K, \text{"CLNT"}, \text{handshake messages}) \)

  \( \{\text{data}\}_K \)

  \( \{\text{finish}\}_K \)