CS 494/594
Computer and Network Security

Dr. Jinyuan (Stella) Sun
Dept. of Electrical Engineering and Computer Science
University of Tennessee
Fall 2010
Web Security: SSL/TLS
What is SSL/TLS?

- Transport Layer Security protocol, version 1.0
  - De facto standard for Internet security
  - “The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications”
  - In practice, used to protect information transmitted between browsers and Web servers
- Based on Secure Sockets Layers protocol, ver 3.0
  - Same protocol design, different algorithms
- Deployed in nearly every Web browser
SSL / TLS in the Real World
Application-Level Protection

Protects against application-level threats (e.g., server impersonation), **NOT** against IP-level threats (spoofing, SYN flood, DDoS by data flood)
History of the Protocol

- **SSL 1.0**
  - Internal Netscape design, early 1994?
  - Lost in the mists of time

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

- **SSL 3.0**
  - Designed by Netscape and Paul Kocher, November 1996

- **TLS 1.0**
  - Internet standard based on SSL 3.0, January 1999
  - Not interoperable with SSL 3.0
    - TLS uses HMAC instead of MAC; can run on any port
“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
TLS Basics

- TLS consists of **two** protocols
  - Familiar pattern for key exchange 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
SSL Protocol Architecture

SSL Handshake Protocol
SSL Change Cipher Spec Protocol
SSL Alert Protocol
HTTP, other apps

SSL Record Protocol

TCP
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 client and server (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
Handshake Protocol Structure

C (Client) → S (Server)

- ClientHello
- ServerHello, [Certificate], [ServerKeyExchange], [CertificateRequest], ServerHelloDone
- [Certificate], ClientKeyExchange, [CertificateVerify]
- switch to negotiated cipher
- Finished
- Record of all sent and received handshake messages
- switch to negotiated cipher
- Finished
ClientHello

Client announces (in plaintext):
- Protocol version he is running
- Cryptographic algorithms he supports
ClientHello (RFC)

```c
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites;
    CompressionMethod compression_methods;
} ClientHello
```

- `ProtocolVersion client_version`: Highest version of the protocol supported by the client.
- `Random random`: Session id (if the client wants to resume an old session).
- `SessionID session_id`: Set of cryptographic algorithms supported by the client (e.g., RSA or Diffie-Hellman).
ServerHello

Server responds (in plaintext) with:

- Highest protocol version supported by both client and server
- Strongest cryptographic suite selected from those offered by the client

\[ \text{C, Version}_c, \text{ suite}_c, N_c \]
Server sends his public-key certificate containing either his RSA, or his Diffie-Hellman public key (depending on chosen crypto suite)
ClientKeyExchange

C, Version\textsubscript{c}, suite\textsubscript{c}, N\textsubscript{c}

Version\textsubscript{s}, suite\textsubscript{s}, N\textsubscript{s}, sig\textsubscript{ca}(S,K\textsubscript{s}), “ServerHelloDone”

Client generates some secret key material and sends it to the server encrypted with the server’s public key (if using RSA)
“Core” SSL 3.0 Handshake

C, Version$_c$ = 3.0, suite$_c$, N$_c$

Version$_s$ = 3.0, suite$_s$, N$_s$,
sig$_{ca}(S,K_s)$,
“ServerHelloDone”

${\text{Secret}}_c$$K_s$ switch to key derived from secret$_c$, N$_c$, N$_s$

If the protocol is correct, C and S share some secret key material (secret$_c$) at this point

switch to key derived from secret$_c$, N$_c$, N$_s$
“Core” SSL 3.0 Handshake (Cont’d)

The book shows the above protocol which is the same as the protocol in the previous slide. Find out the correspondence between the different notations that denote the same thing in these two protocols.
Version Rollback Attack

C, Version = 2.0, suite_c, N_c

Server is fooled into thinking he is communicating with a client who supports only SSL 2.0

Version_s = 2.0, suite_s, N_s, sig_{ca}(S,K_s), “ServerHelloDone”

{Secret_c}_K_s

C and S end up communicating using SSL 2.0 (weaker earlier version of the protocol that does not include “Finished” messages)
SSL 2.0 Weaknesses (Fixed in 3.0)

- Cipher suite preferences are not authenticated
  - “Cipher suite rollback” attack is possible
- Truncation attack
- Weak MAC construction
- MAC hash uses only 40 bits in export mode
- No support for certificate chains or non-RSA algorithms, no handshake while session is open
“Chosen-Protocol” Attacks

- 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)
Version Check in SSL 3.0

If the protocol is correct, C and S share some secret key material $\text{secret}_c$ at this point.

**C**

Version $c=3.0$, suite $c$, $N_c$

$\{\text{Version}_c, \text{Secret}_c\}_{K_s}$

**S**

Version $s=3.0$, suite $s$, $N_s$,

$\text{sig}_{ca}(S,K_s)$,

“ServerHelloDone”

“Embed” version number into secret

Check that received version is equal to the version in ClientHello

switch to key derived from $\text{secret}_c$, $N_c$, $N_s$

switch to key derived from $\text{secret}_c$, $N_c$, $N_s$
SSL/TLS Record Protection

Use symmetric keys established in handshake protocol
SSL Handshake Protocol – Additional Features

- SSL Handshake Protocol supports *session resumption* and *ciphersuite re-negotiation*.
  - Allows authentication and shared secrets to be reused across multiple connections.
    - Eg, next webpage from same website.
  - Allows re-keying of current connection using fresh nonces.
  - Allows change of ciphersuite during session.
  - `ClientHello` *quotes old* `SessionID`.
  - Both sides contribute new nonces, update `secret_c`.
  - All protected by existing Record Protocol.
Session resumption

- Public key operation is avoided

**Protocol 19-3.** Session resumption if both sides remember session-id
Other SSL Protocols

- Alert protocol.
  - Management of SSL session, error messages.
  - Fatal errors and warnings.

- Change cipher spec protocol.
  - Not part of SSL Handshake Protocol.
  - Used to indicate that entity is changing to recently agreed ciphersuite.

- Both protocols run over Record Protocol (so peers of Handshake Protocol).
SSL/TLS Applications

- Secure e-commerce using SSL/TLS.
  - Client authentication not needed until client decides to buy something.
  - SSL provides secure channel for sending credit card information.
  - Client authenticated using credit card information, merchant bears (most of) risk.
  - Wildly successful (amazon.com, on-line supermarkets,…)
  - Some famous disasters (boo.com, webvan), nothing to do with security though.
SSL/TLS Applications

- Secure e-commerce: some issues.
  ◦ No guarantees about what happens to client data (including credit card details) after session: may be stored on insecure server.
  ◦ Does client understand meaning of certificate expiry and other security warnings?
  ◦ Does client software actually check complete certificate chain?
  ◦ Does the name in certificate match the URL of e-commerce site? Does the user check this?
  ◦ Is the site the one the client thinks it is?
  ◦ Is the client software proposing appropriate ciphersuites?
SSL/TLS Applications

- Secure electronic banking.
  - Client authentication may be enabled using client certificates.
    - Issues of registration, secure storage of private keys, revocation and re-issue.
  - Otherwise, SSL provides secure channel for sending username, password, mother’s maiden name,…
    - What else does client use same password for?
  - Does client understand meaning of certificate expiry and other security warnings?
  - Is client software proposing appropriate ciphersuites?
    - Enforce from server.
Some SSL/TLS Security Flaws

- **(Historical) flaws in random number generation for SSL**
  - Low quality RNG leads to predictable session keys.
    - Goldberg and Wagner, Dr. Dobb’s Journal, Jan. 1996.

- **Flaws in error reporting.**
  - (differing response times by server in event of padding failure and MAC failure) + (analysis of padding method for CBC-mode) = recovery of SSL plaintext.

- **Timing attacks.**
  - analysis of OpenSSL server response times allows attacker in same LAN segment to derive server’s private key!
    - Boneh and Brumley, 12th Usenix Security Symposium.
Comparing IPSec and SSL/TLS

- Both have initial (authenticated) key establishment then key derivation.
  - IKE in IPSec
  - Handshake Protocol in SSL/TLS (can be unauthenticated!)
- Both protect ciphersuite negotiation.
- Both use keys established to build a ‘secure channel’.
Comparing IPSec and SSL/TLS

- Operate at different network layers.
  - This brings pros and cons for each protocol suite.
  - Recall ‘Where shall we put security?’ discussion.
  - Naturally support different application types, can all be used to build VPNs.

- Both practical, but not simple.
  - Complexity leads to vulnerabilities.
  - Complexity makes configuration and management harder.
  - Complexity can create computational bottlenecks.
  - Complexity necessary to give both flexibility and security.
Comparing IPSec and SSL/TLS

- Security of both undermined by:
  - Implementation weaknesses.
  - Weak server platform security.
    - Worms, malicious code, rootkits,…
  - Weak user platform security.
    - Keystroke loggers, malware,…
  - Limited deployment of certificates and infrastructure to support them.
    - Especially client certificates.
  - Lack of user awareness and education.
    - Users click-through on certificate warnings.
    - Users fail to check URLs.
    - Users send sensitive account details to bogus websites (“phishing”) in response to official-looking e-mail.
Secure Protocols – Last Words

Eugene Spafford:

“Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit-card information from someone living in a cardboard box to someone living on a park bench.”
Reading Assignment

- [Kaufman] Chapter 19