Course Logistics

- HW 4 due today.
- HW 5 will be posted later today. Due in a week. Group homework.
- DoD Scholarships?
- NSF Scholarships?
- Read Chapter 7 Section 2 of text for SSL.
Where do we provide security?

- Firewall
- Application layer
  - S-MIME, PGP, Kerberos etc.
- Transport layer
  - SSL, TSL
- Network layer
  - VPN’s, IPSEC
Transport Layer Security

Standard TCP/IP Protocol Suite

HTTP / FTP / SMTP / etc
TCP
IP

SSL Implementation

HTTP
SSL
TCP
IP
Secure Sockets Layer (SSL)

- Platform and Application Independent
- Operates between application and transport layers

<table>
<thead>
<tr>
<th>Web Applications</th>
<th>HTTP</th>
<th>NNTP</th>
<th>FTP</th>
<th>Telnet</th>
<th>Etc.</th>
<th>New Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP/IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## SSL over TCP over IP

<table>
<thead>
<tr>
<th>IP Header</th>
<th>IP Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>Dst</td>
</tr>
<tr>
<td>Src Port</td>
<td>Dst Port</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SSL – Secure Socket Layer Protocol

- Provides privacy and reliability between two communicating applications (Focus on Web).
- Requires reliable transport protocol (TCP).
- Only protects data in transit.
- Limited by cryptographic tools it uses.
- Does not provide non-repudiation or traffic flow confidentiality.
SSL History

Initial Design 1995
SSLRef 2.0 1996
SSL v3.0 1996
TLS Draft 1999

Independent Implementations
Hardware, Toolkits, Applications
SSL - Secure Socket Layer Protocol

- A server running SSL provides
  - Confidentiality
  - Integrity
  - Authenticity

- SSL is designed to operate in a number of different modes, depending on the requirement of the network connection
  - Encrypted communication only
  - Encryption and authentication of the server
  - Encryption and authentication of client and server
SSL - Overview

- Two layers of protocols.
- Establishes peer-to-peer “session” which could consist of multiple connections.
SSL Record Protocol.

- Provides confidentiality and message integrity services.

Application Data

Fragment

Compress

Add MAC

Encrypt

Append SSL Record Header

Fragment - Blocks of $2^{14}$ bytes.

MAC - HMAC like, using SHA-1 or MD5.

Encrypt - DES, 3DES, IDEA, RC4, RC5, with different key lengths.

Compress - none.
SSL Record Format

- **Type**: Which higher layer protocol. Only one type in each message.
- **Major, Minor version**: SSL version.
- **Length of message**
SSL – Handshake Protocol.

- Allows server and client to authenticate one another and establish security parameters.
- Consists of following phases:
  - Establish security capabilities.
  - Server authentication and key exchange.
  - Client authentication and key exchange.
  - Finish.
- Message format

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>3 bytes</td>
<td>&gt;= 0 bytes</td>
</tr>
</tbody>
</table>
SSL – Alert Protocol

- Alert Protocol – Used for SSL related alert and error messages

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

Messages include: unexpected_message, bad_record_mac, handshake_failure, illegal_parameter, bad_certificate etc.
SSL - Cipher Change Spec Protocol

- Causes pending state to be copied into current state.

1 byte

- Why separate protocol?
  - Because of Record Layer encapsulation.
  - Security services (e.g., Encryption) applied to all of Record Layer Message at once. Hence cipher change cannot be mixed with other messages.
  - Separate protocol ensures above condition.
SSL Example Exchange

THE SSL PROTOCOL DEFINES VARIOUS MESSAGE TYPES EXCHANGED BETWEEN CLIENT AND SERVER

1. ClientHello
2. ServerHello
3. Certificate
4. ServerKeyExchange
5. ServerHelloDone
6. ClientKeyExchange
7. ChangeCipherSpec
8. Finished
9. ChangeCipherSpec
10. Finished
SSL Cipher Suite

- For public-key, symmetric encryption and certificate verification we need
  - public-key algorithm
  - symmetric encryption algorithm
  - message digest (hash) algorithm
- This collection is called a cipher suite
- SSL supports many different suites
- Client and server must decide on which one to use
- The client offers a choice; the server picks one.
# SSL- Encryption Techniques

<table>
<thead>
<tr>
<th>Block Cipher</th>
<th>Algorithm</th>
<th>Key Size</th>
<th>Stream Cipher</th>
<th>Algorithm</th>
<th>Key Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IDEA</td>
<td>128</td>
<td>RC4-40</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RC2-40</td>
<td>40</td>
<td>RC4-128</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DES-40</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DES</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3DES</td>
<td>168</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fortezza</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SSL- Message Authentication Code

hash(MAC_write_secret || pad_2 ||
hash(MAC_write_secret || pad_1|| seq_num || SSL
Compressed.type ||
SSLCompressed.length || SSLCompressed.fragment))

where
MAC_write_secret = shared secret key
hash = cryptographic hash algorithm; either MD5 or SHA-1
pad_1= the byte 0x36(0011 0110)repeated 48*(384 bits) for MD5 and 40*(320 bits) for SHA-1
pad_2= the byte 0x5C (0101 1100) repeated 48*for MD5 and 40*for SHA-1
seq_num = the sequence number for this message
SSLCompressed.type = the higher-level protocol used to process this fragment
SSLCompressed.length = the length of the compressed fragment
SSLCompressed.fragment = the compressed fragment(if compression is not used, the plaintext fragment)
Example Cipher Suites

SSL_NULL_WITH_NULL_NULL = { 0, 0 }

PUBLIC-KEY ALGORITHM
SYMMETRIC ALGORITHM
HASH ALGORITHM

SSL_RSA_WITH_NULL_MD5 = { 0, 1 }
SSL_RSA_WITH_NULL_SHA = { 0, 2 }
SSL_RSA_EXPORT_WITH_RC4_40_MD5 = { 0, 3 }
SSL_RSA_WITH_RC4_128_MD5 = { 0, 4 }
SSL_RSA_WITH_RC4_128_SHA = { 0, 5 }
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5 = { 0, 6 }
SSL_RSA_WITH_IDEA_CBC_SHA = { 0, 7 }
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA = { 0, 8 }
SSL_RSA_WITH_DES_CBC_SHA = { 0, 9 }
SSL_RSA_WITH_3DES_EDE_CBC_SHA = { 0, 10 }

INITIAL (NULL) CIPHER SUITE

CIPHER SUITE CODES USED IN SSL MESSAGES
SSL Encryption

- **Premaster secret**
  - Created by client; used to “seed” calculation of encryption parameters
  - Very simple: 2 bytes of SSL version + 46 random bytes
  - Sent encrypted to server using server’s public key

- **Master secret**
  - Generated by both parties from premaster secret and random values generated by both client and server

- **Key material**
  - Generated from the master secret and shared random values

- **Encryption keys**
  - Extracted from the key material
Forming the Master Secret

SERVER’S PUBLIC KEY IS SENT BY SERVER IN ServerKeyExchange

CLIENT GENERATES THE PREMASTER SECRET

ENCRIPTS WITH PUBLIC KEY OF SERVER

CLIENT SENDS PREMASTER SECRET IN ClientKeyExchange

MASTER SECRET IS 3 MD5 HASHES CONCATENATED TOGETHER = 384 BITS
Forming the Key Material

JUST LIKE FORMING THE MASTER SECRET

EXCEPT THE MASTER SECRET IS USED HERE INSTEAD OF THE PREMASTER SECRET
Obtaining Keys from the Key Material

Source: Thomas, SSL and TLS Essentials

SECRET VALUES INCLUDED IN MESSAGE AUTHENTICATION CODES

SYMMETRIC KEYS

INITIALIZATION VECTORS FOR DES CBC ENCRYPTION
SSL- Handshake Protocol

The handshake protocol is used before any application data is transmitted.

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello_request</td>
<td>null</td>
</tr>
<tr>
<td>client_hello</td>
<td>version, random, session id, cipher suite, compression method</td>
</tr>
<tr>
<td>server_hello</td>
<td>version, random, session id, cipher suite, compression method</td>
</tr>
<tr>
<td>certificate</td>
<td>chain of X.509v3 certificates</td>
</tr>
<tr>
<td>server_key_exchange</td>
<td>parameters, signature</td>
</tr>
<tr>
<td>certificate_request</td>
<td>type, authorities</td>
</tr>
<tr>
<td>server_done</td>
<td>null</td>
</tr>
<tr>
<td>certificate_verify</td>
<td>signature</td>
</tr>
<tr>
<td>client_key_exchange</td>
<td>parameters, signature</td>
</tr>
<tr>
<td>finished</td>
<td>hash value</td>
</tr>
</tbody>
</table>
SSL Handshake

1. Client sends ClientHello message.
2. Server acknowledges with ServerHello message.
4. Server requests client’s certificate.
5. Client sends its certificate.
6. Client sends ClientKeyExchange message.
8. Both send ChangeCipherSpec messages.
9. Both send Finished messages.

SECURETRANSMISSION BEGINS HERE
SSL- Hello Parameters

- **Session ID**: An arbitrary byte sequence chosen by the server to identify an active or resumable session state
  - Could be created by using Diffie-Hellman.
  - Or by the hash of the shared secret and salt.
- **Peer certificate**: An X509.v3 certificate of the peer. This element of the state may be null.
- **Compression method**: The algorithm used.
- **Cipher spec**: Specifies the bulk data encryption algorithm (such as null, DES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size.
SSL Messages

CLIENT SIDE

OFFER CIPHER SUITE
MENU TO SERVER

SEND ENCRYPTED
SYMMETRIC KEY

ACTIVATE
ENCRYPTION

CLIENT PORTION
DONE

( CLIENT CHECKS OPTIONS )

SERVER SIDE

SELECT A CIPHER SUITE

SEND CERTIFICATE AND
CHAIN TO CA ROOT

SEND PUBLIC KEY TO
ENCRYPT SYMM KEY

SERVER NEGOTIATION
FINISHED

( SERVER CHECKS OPTIONS )

ACTIVATESERVER
ENCRYPTION

SERVER PORTION
DONE

NOW THE PARTIES CAN USE SYMMETRIC ENCRYPTION
Encryption and authentication of server - Making a purchase from Barnes & Nobles

Server sends ServerKeyExchange along with Server Certificate

1. Client Hello
2. Server Hello
3. Server Certificate
4. Server Hello Done
5. Client Key Exchange
6. Change Cipher Spec
7. Finished
8. Change Cipher Spec
9. Finished
SSL- Other Modes

- Encryption and authentication of both the client and server
  - Both client and server send certificates.
- Encrypted Communication only
  - No certificates are sent.
  - ServerKeyExchange replaces server certificate in the 3rd step.
SSL: Attacks - Man-in-the-Middle attack

V2 ClientHello
(with v3 hints)

V2 ServerHello

V2 ClientHello
(hints removed)

V2 ServerHello
SSL- Detection of Man-in-the-Middle attack:

- Duel version client uses special padding values in the ClientKeyExchange message.
- It sets the last 8 bytes to 00000011
- If the server finds this value, then an attack is occurring.

- Note: Attacker will not be able to modify this value as it is encrypted with server’s public key.
SSL: Traffic Analysis Attacks

- Attacker may learn a lot about the target just by observing the traffic to and from that target, even if he/she cannot decrypt it.
- SSL only provides padding for block ciphers and not stream ciphers.
- So when SSL uses stream ciphers for encryption, an attacker can easily find the size of the unencrypted data. All he has to do is subtract the size of the MAC from the encrypted message.
SSL: The Bleichenbacher Attack

- In SSL the encoded data always begins with the two bytes 00 and 02.
- The attacker constructs many artificial ciphertext blocks and sends them to the target.
- The target decrypts these blocks. As a result one of the following might happen.
  - The resulting plaintext does not have the above mentioned encoding format. So the target generated an error or ignores the communication.
SSL: The Bleichenbacher Attack continued

- The resulting plaintext will begin with 00 and 02, hence the target considers the decryption to be successful.
- The attacker observes the target reaction to each ciphertext block it sends, noting which block gets decrypted successfully (according to the target) and which does not.
- Over a period of time and with mathematical analysis the attacker might be able to decrypt one of the cipher blocks that was sent to the target during a legitimate communication.
SSL- Differences between V2 and V3

- Version 3 supports more cryptographic functions in its cipher suite.
- The ClientHello message slightly varies in v2 and v3.
TLS - Transport Layer Security

- TLS is very similar to SSLv3.
- TLS makes use of HMAC algorithm defined as:

$$HMAC_K = H[(K^+ \text{ xor } opad) \ || \ H[(K^+ \text{ xor } ipad) \ || \ M]]$$

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>embedded hash function (for TLS, either MD5 or SHA-1)</td>
</tr>
<tr>
<td>M</td>
<td>message input to HMAC</td>
</tr>
<tr>
<td>K^+</td>
<td>secret key padded with zeros to block length of hash code</td>
</tr>
<tr>
<td>Ipad</td>
<td>00110110 (36 in hexadecimal) repeated 64 times (512 bits)</td>
</tr>
<tr>
<td>Opad</td>
<td>01011100 (5C in hexadecimal) repeated 64 times (512 bits)</td>
</tr>
</tbody>
</table>
SSLv3 and TLS Differences

- SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length.
- The MAC calculation covers all of the fields covered by the SSLv3 calculation, plus the field TLSCompressed.version, which is the version of the protocol being employed.
- TLS supports all of the alert codes defined in SSLv3 with the exception of no_certificate.
SSLv3 and TLS Differences (Contd.)

- Key Exchange: TLS supports all of the key exchange techniques of SSLv3 with the exception of Fortezza.

- Symmetric Encryption Algorithms: TLS includes all of the symmetric encryption algorithms found in SSLv3, with the exception of Fortezza.
SSLv3 and TLS Differences (Contd.)

- In SSL, the padding added prior to encryption of user data is the minimum amount required so that the total size of the data to be encrypted is a multiple of the cipher's block length.

- In TLS, the padding can be any amount that results in a total that is a multiple of the cipher's block length, up to a maximum of 255 bytes.
Further Reading

- RFC on TLS at http://www.ietf.org/rfc/rfc2246.txt