



Slide 3

Application Layer approach (SSL)

Advantages:
 don't need to modify operating system

Disadvantages:
 Need to modify application to use SSL

Attacker can add false packets to packet stream, that pass TCP checksum and sequence number checks. — > Real packets lost, fake packets passed to SSL

SSL 3.0/TLS

Slide 1

SSL 3.0/TLS

De facto standard for secure internet communication, particularly between browser client C and server S .
 Development initiated by Netscape, SSLV2 1995
 TLS an IETF standardised version, but incompatible with SSLV3

Objectives:

- authenticate C and S to each other,
- negotiate cryptosystem to be used,
- generate a shared secret key,
- roll back to SSL 2.0 if either C or S is using that version.

Network protocol approach (IPSec)

Advantages:

Prevent false packet attack

Don't need to modify application

Disadvantages:

need to modify operating system

hides information that firewalls would like to see

(in current version): IPSec can authenticate users, but API can can only pass IP addresses to application, not users

Slide 5

Information exchanged in the SSL protocol:

- Ver_i : SSL version numbers of i
- $Suite_i$: cryptographic preferences of i
- $Cipher_i$: algorithm selected from the client Suite by Server
- N_i : Nonce generated by i
- $Secret_i$: random secret generated by i
- $Messages$: all messages up to this point.

“Specification of SSL” (Inferred from description)

At the end of the handshake:

1. If $Secret_C, Secret_S$ are what S and C consider to be the shared secret, then $Secret_C = Secret_S$.
2. the shared secret is not visible to an intruder
3. the parties agree on each other's identity and protocol completion status
4. the crypto system agreed upon should be the strongest one available to both parties
5. the parties have a consistent opinion of the version of SSL being run.

Slide 7

SSL Protocol

$C \rightarrow S : C, Ver_C, Suite_C, N_C$

$S \rightarrow C : Ver_S, (sessionid), Cipher, N_S, sign_{CA}(S, K_S)$

$C \rightarrow S :$

$sign_{CA}(C, V_C), \{Secret_C\}_{K_S}, sign_C(Hash(Messages(1 - 2)))$

(change to negotiated cipher)

$S \rightarrow C : \{Hash(Messages(1 - 3))\}_{MasterKey}$

$C \rightarrow S : \{Hash(Messages(1 - 4))\}_{MasterKey}$

where $MasterKey = f(Secret_C, N_C, N_S)$

$Messages(1-n)$ is all messages in first n steps.

Client certificate $sign_{CA}(C, V_C)$ is optional, and not much used.

Slide 9

Session Resumption

Subsequent connections (e.g. to fetch another web page from the server) use keys computed from $MasterKey$, provided Server has allowed this by sending $sessionId$ and saving $(sessionId, Secret)$ at the time of the initial connection

$$C \rightarrow S : sessionId, Ver_C, Suite_C, N_C^i$$

$$S \rightarrow C :$$

$$sessionId, Ver_S, \{Cipher, N_S^i, \{Hash(Messages(1 - 1))\}_{MasterKey^i}\}$$

$$C \rightarrow S : \{Hash(Messages(1 - 2))\}_{MasterKey^i}$$

$$\text{where } MasterKey^i = f(Secret_C, N_C^i, N_S^i)$$

(This establishes the new connection in 3 messages rather than 5)

Slide 11

A Weakness with session resumption

The session resumption protocol specification implicitly allows that the new connection can be resumed with version 2 rather than version 3. In fact an adversary can force the version to be rolled back to version 2.

(See papers by Wagner and Schneier, 1996, and by Mitchell and Shmatikov, 1998.)

Version 2 had a number of weaknesses:

1. **Downgrade:** An attacker could force the use of the weakest cipher in the client supported ciphersuite

2. **Truncation:** An attacker could terminate a TCP connection,

closing the session potentially without the client/server noticing anything abnormal.

Slide 13