


Real-Time Communication Security: SSL/TLS

Guevara Noubir
noubir@ccs.neu.edu
 Network Security

Reading Assignment: chapters 16, 19


Network Security SSL – TLS 1



Some Issues with Real-time Communication

- Session key establishment
- Perfect Forward Secrecy
 - Diffie-Hellman based PFS
 - Escrow-foilage:
 - If keys are escrowed Diffie-Hellman protects against passive attacks
 - Signature keys are usually not escrowed
- Preventing Denial of Service
 - SYN attack on TCP: use stateless cookies = hash(IP addr, secret)
 - Puzzles: e.g., what 27-bit number has an MD = x?
 - These techniques do not fully protect against DDOS launched through viruses
- Hiding endpoint identity:
 - DH + authentication allows anonymous connection or detects man-in-the-middle
- Live partner reassurance:
 - Modify DH to include a nonce in the computation of the session key
- Optimization using parallel computation, session resumption, deniability

Network Security SSL – TLS 2



Securing Networks

- Where to put the security in a protocol stack?
- Practical considerations:
 - End to end security
 - No modification to OS/network stack

Control/Management (configuration)	Applications Layer	Network Security Tools: Monitoring/Logging/Intrusion Detection
	telnet/ftp, ssh, http: https , mail: PGP	
	(SSL/TLS)	
	Transport Layer (TCP)	
	(IPSec, IKE)	
	Network Layer (IP)	
	Link Layer	
	(IEEE802.1x/IEEE802.10)	
	Physical Layer	
	(spread-Spectrum, quantum crypto, etc.)	

Network Security SSL – TLS 3



SSL vs. IPsec

- **SSL:**
 - Avoids modifying "TCP stack" and requires minimum changes to the application
 - Mostly used to authenticate servers
- **IPsec**
 - Transparent to the application and requires modification of the network stack
 - Authenticates network nodes and establishes a secure channel between nodes
 - Application still needs to authenticate the users



General Description of SSL/ TLS

- **Terminology:**
 - SSL: Secure Socket Layer
 - TLS: Transport Layer Security
- **Concept:** secure connections on top of TCP
 - OS independent
 - TCP instead of UDP
 - Cons: Rogue packet problem
 - Pro: SSL/TLS doesn't have to deal with packet retransmission
- **History:**
 - SSLv2 proposed and deployed in Netscape 1.1 (1995)
 - PCT (Private Communications Technology) by Microsoft
 - SSLv3: most commonly used (1995)
 - TLS proposed by the IETF based on SSLv3 but not compatible (1996)
 - Uses patent free DH and DSS instead of RSA which patent didn't expire yet



SSL Architecture

- **SSL session**
 - An association between client & server
 - Created by the Handshake Protocol
 - Defines a set of cryptographic parameters
 - May be shared by multiple SSL connections
- **SSL connection**
 - A transient, peer-to-peer, communications link
 - Associated with 1 SSL session



SSL/TLS Basic Protocol

- SSL/TLS partitions TCP byte stream into records:
 - A record has: header, cryptographic protection => provides a reliable encrypted, and integrity protected stream of octet
 - Record types:
 - User data
 - Handshake messages
 - Alerts: error messages or notification of connection closure
 - Change cipher spec
- Basic Protocol:
 - $A \rightarrow B$: I want to talk, ciphers I support, R_A
 - $B \rightarrow A$: certificates, cipher I choose, R_B
 - $A \rightarrow B$: $\{S\}_B$ {keyed hash of handshake msgs}
 - $B \rightarrow A$: {keyed hash of handshake msgs}
 - $A \leftrightarrow B$: data encrypted and integrity checked with keys derived from K
 - Keyed hashes use $K = f(S, R_A, R_B)$

Network Security

SSL – TLS

7



SSL/TLS Basic Protocol (Cont'd)

- How do you make sure that keyed hash in message 3 is different from B 's response?
 - Include a constant *CLNT/client finished* (in SSL/TLS) for A and *SRVR/server finished* for B
- Keyed hash is sent encrypted and integrity protected for no real reason
- Keys: derived by hashing K and R_A and R_B
 - 3 keys in each direction: encryption, integrity and IV
 - Write keys (to send: encrypt, integrity protect)
 - Read keys (to receive: decrypt, integrity check)

Network Security

SSL – TLS

8



What's still missing?

- SSL/TLS allowed to authenticate the server
- How would the server authenticate the user?
 - SSL/TLS allows clients to authenticate using certificates:
 - B requests a certificate in message 2
 - A sends: certificate, signature of hash of the handshake messages

Network Security

SSL – TLS

9



Session Resumption

- Many secure connections can be derived from the session
 - Cheap: how?
- Session initiation: modify message 2
 - $B \rightarrow A$: session_id, certificate, cipher, R_B
- A and B remember: (session_id, master key)
- To resume a session: A presents the session_id in message 1
 - $A \rightarrow B$: session_id, ciphers I support, R_A
 - $B \rightarrow A$: session_id, cipher I choose, R_B , {keyed hash of handshake msgs}
 - $A \rightarrow B$: {keyed hash of handshake msgs}
 - $A \leftrightarrow B$: data encrypted and integrity checked with keys derived from K



Computing the Keys

- S : pre-master secret (forget it after establishing K)
- $K = f(S, R_A, R_B)$
- 6 keys = $g(K, R_A, R_B)$
- R s: 32 bytes (usually the first 4 bytes are Unix time)



PKI in SSL

- Client comes configured with a list of “trusted organizations”: CA
- What happens when the server sends its certificate?
- When the server wishes to authenticate the client:
 - Server sends a list of CA it trusts and types of keys it can handle
- In SSLv3 and TLS a chain of certificates can be sent

Negotiating Cipher Suites

- A cipher suite is a complete package:
 - (encryption algorithm, key length, integrity checksum algorithm, etc.)
- Cipher suites are predefined:
 - Each assigned a unique value (contrast with IKE)
 - SSLv2: 3 bytes, SSLv3: 2 bytes => upto 65000 combinations
 - 30 defined,
 - 256 reserved for private use: FFxx (risk of non-interoperability)
- Selection decision:
 - In v3 A proposes, B chooses
 - In v2 A proposes, B returns acceptable choices, and A chooses
- Suite names examples:
 - SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
 - SSL2_RC4_128_WITH_MD5

Network Security

SSL – TLS

13

Attacks fixed in v3

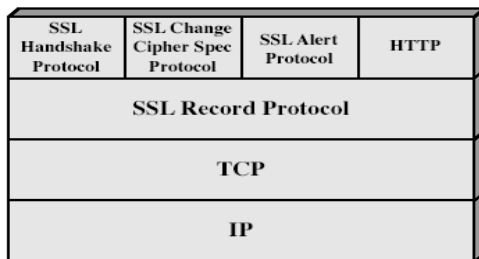
- Downgrade attack:
 - In SSLv2 there is no integrity protection for the initial handshake
 - Active attacker can remove strong crypto algorithm from proposed cipher suite by A => forcing A and B to agree on a weak cipher
 - Fixed by adding a *finished* message containing a hash of previous messages
- Truncation attack:
 - Without the *finished* message an attacker can send a TCP FIN message and close the connection without communicating nodes detecting it
- Attacks not fixed: session resumption ...

Network Security

SSL – TLS

14

SSL Stack



Network Security

SSL – TLS

15



SSL Record Protocol

- SSL Record Protocol defines these two services for SSL connections:
 - **Confidentiality**
 - Using symmetric encryption with a shared secret key defined by Handshake Protocol
 - IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
 - CBC mode (except for RC4)
 - Message is compressed before encryption
 - **Message integrity**
 - Using a MAC with shared secret key
 - Based on HMAC and MD5 or SHA (with a padding difference due to a typo in an early draft of HMAC RFC2104)
- Records sent after *ChangeCipherSpec* record are cryptographically protected
- Record header:
 - [record type, version number, length]
 - ChangeCipherSpec = 20, Alert = 21, Handshake = 22, Application_data = 23

Network Security

SSL – TLS

16



SSL Change Cipher Spec Protocol

- One of 3 SSL-specific protocols which use the SSL Record Protocol
- Single message
 - Causes pending state to become current
- ⇒ all records following this will be protected with the ciphers agreed upon

Network Security

SSL – TLS

17



SSL Alert Protocol

- Conveys SSL-related alerts to peer entity
- Severity
 - warning or fatal
- Specific alerts
 - Unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
 - Close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- Compressed & encrypted

Network Security

SSL – TLS

18



SSL Handshake Protocol

- Allows server & client to:
 - Authenticate each other
 - Negotiate encryption & MAC algorithms
 - Negotiate cryptographic keys to be used
- Comprises a series of messages in phases
 - Establish Security Capabilities
 - Server Authentication and Key Exchange
 - Client Authentication and Key Exchange
 - Finish

Network Security

SSL – TLS

19



Handshake Messages

- **ClientHello** message:
 - [type=1, length, version number, R_p , length of session_id, session_id, length of cipher suite list, sequence of cipher suites, list of compression methods]
- **ServerHello**: [type=2, length, version number, R_p , length of session_id, session_id, chosen cipher, chosen compression method]
- **Certificate**: [type=11, length, length of first certificate, first certificate, ...]
- **ServerKeyExchange**: (for export: ephemeral public key)
 - [type=12, length, length of modulus, modulus, length of exponent, exponent]
- **CertificateRequest**: [type=13, length, length of key type list, list of types of keys, length of CA name list, length of first CA name, 1stCA name, ...]
- **ServerHelloDone**: [type=14, length=0]
- **ClientKeyExchange**: [type=16, length, encrypted pre-master secret]
- **CertificateVerify**: [type=15, length, length of signature, signature]
- **HandshakeFinished**: [type=20, length=36 (SSL) or 12 (TLS), digest]

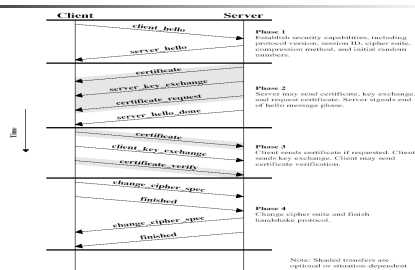
Network Security

SSL – TLS

20



SSL Handshake Protocol



Network Security

SSL – TLS

21



Exportability Issues

- Exportable suites in SSLv2:
 - 40 secret bits out of 128 in symmetric keys
 - 512-bits RSA keys
- Exportability in SSLv3:
 - Integrity keys computed the same way
 - Encryption keys: 40 bits secret
 - IV non-secret
 - When a domestic server (e.g., 1024-bit RSA key) communicates with an external client the server creates an ephemeral key of 512-bits and signs it with its 1024-bit key

Network Security

SSL – TLS

22



TLS (Transport Layer Security)

- IETF standard RFC 2246 similar to SSLv3
- Minor differences
 - Record format version number
 - HMAC for MAC
 - Pseudo-random function to expand the secrets
 - Additional alert codes
 - Changes in supported ciphers
 - Changes in certificate negotiations
 - Changes in use of padding

Network Security

SSL – TLS

23
