Embedded SSL

Christophe Kiennert, Pascal Urien
Introduction

• TLS/SSL is the Holy Grail of WEB security
  – Many applications may be secured by SSL
    • HTTP, FTP, SIP, SMTP, POP, …

• TLS is secured, but what about trust?
  – *Branch Prediction Attacks* (2006) may recover an RSA key during a single calculation
  – *Instruction Cache Attacks* (2005) may recover an AES key in 65 milliseconds
  – These attacks work with OpenSSL, which runs on more than 60 percent of the world’s server installations.

• TLS/SSL stacks are running on untrustworthy computers
Security issues to be addressed

• Certificates are checked by host computers
• Unknown certificates are approved by users
• RSA private key may be recovered (client’s side)
• TLS master secret may be recovered
  – Session hijacking
• No plug and play architecture
  – Host computer must be configured with the CA Certificate
About TLS

- Server authentication or mutual authentication
- Authentication based on X509 certificates
- Once the certificates have been verified, keying material is generated on both client and server for encryption of subsequent messages
- HTTP and SSL can be split in two parts
  - An HTTPS session begins by a pure SSL exchange, 4 ways (full session) or 3 ways (resume session) handshake
  - Afterwards HTTP messages are tunneled in SSL packets

- Embedded SSL exploits this idea
TLS Stack Structure

Encrypted Payload

HMAC(Ki, Seq-num | Header | Message)

EP = {Message | HMAC | Padding-Bytes | PAD-Length} Kc

Header | EP

Embedded SSL - Christophe Kiennert, Pascal Urien
Basic key calculations

• For Full Sessions
  – A PreMasterSecret is sent by the client encrypted with the server public key
    • { PreMasterSecret } KPubS
  – master-secret = PRF(PreMasterSecret, "master secret", ClientRandom | ServerRandom)
  – If a client certificate is required (mutual authentication) it is forwarded to the server, and authenticated by a signature generated with the client private key.

• For Resume Sessions
  – A master secret has already been computed, it is identified by an index, named the Session-ID
  • Negotiated cryptographic algorithms are identified by a two bytes value labeled CipherSuite.
  • key-block = PRF(master-secret, "key expansion", ServerRandom | ClientRandom)
Embedded SSL main idea

• A TLS session is split in two phases.
  – First (*Phase I*) deals with authentication and cryptographic key calculations
  – Second (*Phase II*) takes advantage of the previously created secure channel, in order to exchange information between applications in a safe context.

• Phase I performed in smartcard
• Phase II performed on client computer
  – If performed in smartcard, throughput is about 10 Kbit/s, incompatible with common multimedia files
The Embedded SSL Platform

• A Browser
  • A Proxy Software
    – Runs the SSL record layer
    – http://127.0.0.1:8080/reader/apdu?=data
  • A Smart Card
    – Runs the SSL stack
• A WEB Server
  – Configured for SSL session with mutual authentication

Classical architecture vs our approach

HTTP

SSL

CA Certificate

Client Certificate

Driver

Proxy

SSL

Client Certificate

CA Certificate
Dual SSL Stack (TLS-Tandem)

Client

- Client hello (ClientRandom)
- Server Hello (Session-id, ServerRandom)
  - Certificate
  - * CertificateRequest
  - ServerHelloDone
  - * Certificate
  - Certificate Verify
  - ChangeCipherSpec
    - (Encrypted) Finished
  - ChangeCipherSpec
    - (Encrypted) Finished

Server

- Client hello (Session-id, ClientRandom)
- Server Hello (Session-id, ServerRandom)
  - ChangeCipherSpec
    - (Encrypted) Finished

ClientKeyExchange

- \{PreMasterSecret\}_{KpubS}
- * Certificate Verify
- ChangeCipherSpec
  - (Encrypted) Finished

Record Layer in Ciphered Mode

* Optional messages

Server

- ChangeCipherSpec
  - (Encrypted) Finished

Record Layer in Ciphered Mode

* Optional messages

PHASE I

PHASE II

Key-block CipherSuite

Embedded SSL - Christophe Kiennert, Pascal Urien
Specificities & Benefits

• Based on EAP-TLS smartcards
  – EAP-TLS is a transparent encapsulation of TLS

• Typical performances
  – TLS full mode, 4 s
  – TLS resume mode, 2s

• SSL smart cards present the following security benefits:
  – The server’s certificate is checked in a trusted computing environment, *anti-phising feature*.
  – Client RSA keys are handled by a trusted computing environment, in *state-full* way
  – *Full and Resume* sessions are managed by a trusted computing environment.
Tandem Overview

http://127.0.0.1:8080/ ~url=server.com/file

SSL - hello

Certificate Request

User’s Certificate

Finished

Finished

Encrypted HTTP Request, GET /File.....

Encrypted file
TLS-Tandem Card Commands

- **Verify-PIN**, unlocks the application via a user’s PIN.
- **Set-Identity**, activates an electronic identity e.g. all credentials needed by the TLS session for mutual authentication (Certification Authority certificate, user’s certificate, user’s RSA private key...).
- **Reset**, resets the EAP-TLS state machine.
- **Process-EAP-TLS**, processes a TLS packet and returns the associated response.
- **Get-CipherSuite**, reads the *CipherSuite* value (a two bytes value)
- **Get-Key-Block**, collects the list of ciphering and integrity keys (typically four values of 16 bytes).
TLS-Tandem API

• The TLS-Tandem API offers three high levels procedures,
  – TLS-Tandem.connect() realizes Phase I operations, either in full or resume mode.
  – TLS-Tandem.write() encrypts and sends data in the Phase II context.
  – TLS-Tandem.read() reads and decrypts data in the Phase II context.

• Close to the OPENSSL paradigm:

  // creates an SSL context, e.g. sets CA certificate, client’s certificate
  ssl= new (ctx);
  // creates a sbio object linked to socket s
  sbio = BIO_new_socket(s, BIO_NOCLOSE);
  // links ssl and sbio objects to socket s SSL_set_bio(ssl,sbio,sbio);
  // performs TLS PhaseI
  SSL_connect(ssl);
  // reads data from TLS peer in Phasell
  length= SSL_read(ssl, buffer,sizeof(buffer));
  // sends data to TLS peer in Phasell.
  error = SSL_write(ssl,buffer,length);
  // releases an SSL context
  destroy(ctx);
Security issues

• The trust relies on the proxy application integrity
• A corrupted software could induce hijacking
• Possible protections:
  – Digest control
  – Signature checking
  – Downloading from trusted servers
Conclusion

• TLS-based applications may be secured with smart cards

• Today, Phase II cannot reasonably be managed by smartcards (throughput is less than 2000 bytes/s)

• Embedded SSL is applied to identity management (such as OpenID), suppressing the login/password concept

• http://autoconnect.me/