IIG University of Freiburg

Web Security, Summer Term 2012

Secure HyperText Transfer Protocol

Dr. E. Benoist

Sommer Semester

Table of Contents

- Attacks using HTTP
  - Where do we Need Security?
- Symetric and Asymetric Cryptography
  - Signing a message
- Public Key Infrastructure
  - Efficiency of Cryptography
- SSL and TLS
  - Presentation of phases
  - Properties of a TLS connextion
- How to Configure your Web Server
  - Create a HTTPS server
  - Host Multiple Servers on One Machine
  - Double Side Authentication
  - Apache Configuration
- Limitations
- References

Attacks Using HTTP

- **HTTP is transfered Cleartext over the internet**
  - Request / Response sent unencrypted
- **Eavesdropping**
  - Any attacker can read everything transfering on the Internet.
  - Includes GET and POST parameters
  - For instance Username / Password or Session Cookie
- **Message Modification**
  - Message can be manipulated
  - Request : add some tracking information
  - Response: modify the page.

    Examples:
    - Add some Javascript for sending information to a third party
    - Change the action of a form (to redirect the user to a physhing site)
    - . . .

Needs for security

- **Confidentiality**
  - Nobody can read the message I send
  - For both Security and Privacy
- **Authentication of the partner**
  - Am I realy talking with the server I am supposed to?
  - Am I realy the person I am supposed to be?
- **Integrity of the Message**
  - Is the message the one that my partner sent?

Symetric Cryptography

- **Symetric Cryptography**
  - Alice and Bob share the same Key K (which is secret)
  - Alice encrypts the message with K
  - Bob decrypts the message with K
  - If Charly doesn't have K, he can not read the message
- **Efficiency**
  - This type of crypto is very efficient
- **Problem**
  - How to exchange the key if you do not meet your correspondant
  - Alice and Bob need a secure chanel to exchange the key

Asymetric Cryptography

- **Knowledge of Keys is Asymetric**
  - Alice wants to send a message M to Bob
  - Alice has access to the public key K_{Bpub} of Bob
  - Bob knows a pair (K_{Bpub}, K_{Bpriv})
- **Encryption of a message**
  - Alice encrypts the message using Bob's Public key K_{Bpub}
  - Bob decrypts the message using his private key K_{Bpriv}
- **Problem**
  - How can Bob be sure it is Alice who sent the message?
  - Charlie may have intercepted the message and replaced by another one

Signing a message

- **Bob wants to be certain the message was sent by Alice**
  - He wants to check the *integrity* of the message
- **Signing of the message**
  - Alice also has a pair of keys:(K_{Apub}, K_{Apriv})
  - Bob knows the public key of Alice K_{Apub}
  - Alice uses her private key to sign the message sent to Bob
  - Bob uses the public key to verify the signature of Alice
  - Since Charly does not know the private key, he can not forge such a message
  - Bob is convinced that Alice has sent this message
- **Combining both : encrypting and signing**
  - Alice writes a message M
  - She creates a signature $\sigma(M)$ with her private key K_{Apriv}
  - She encrypts both M and $\sigma(M)$ with Bob's public key K_{Bpub}
  - Bob receives the encrypted message,

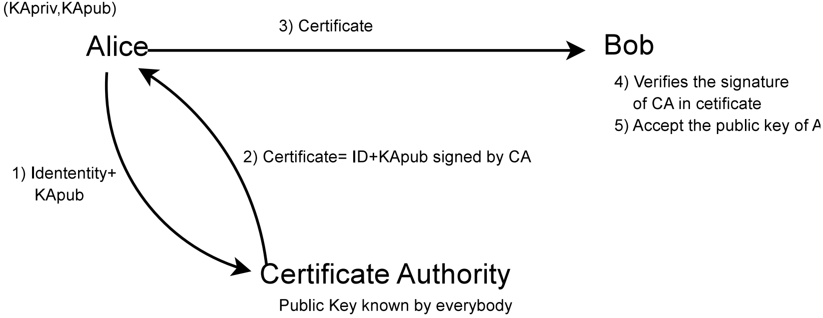
Public Key Infrastructure

- **A lot of keys have to be exchanged**
  - Alice needs the public key of Bob
  - Bob needs the public key of Alice
  - etc.
- **How to exchange keys in a secure way?**
  - Alice and Bod never met eachother
  - They trust the same third party (called Certificate Authority - CA)
  - They both have received (in a secure way) the public key of the CA

Certificate

- **Alice wants to receive Bob's public key**
  - Bob creates his key pair
  - Bob is identified by CA and gives his Name and public key to the CA
  - CA signes a "certificate" containing the following information
    - Name of the Certificate Authority
    - Name of the owner of the certificate (Bob)
    - Address, . . .
    - Public key of Bob
- **So if Alice trusts the verification of CA, she trusts the public key of Bob.**
- **Problems in real life**
  - Alice and Bob may not have the same certificate authority: We have a chain of trust (or web of trust)
  - The Public Key Infrastructure PKI uses a Root Certificate who anybody trusts.
  - You need a way to revoke compromised keys
  - . . .

Public Key Infrastructure (simplified)


(KApriv,KApub)
Alice
3) Certificate
Bob
4) Verifies the signature
of CA in cetificate
5) Accept the public key of A
2) Certificate= ID+KApub signed by CA
1) Idententity+
KApub
Certificate Authority
Public Key known by everybody

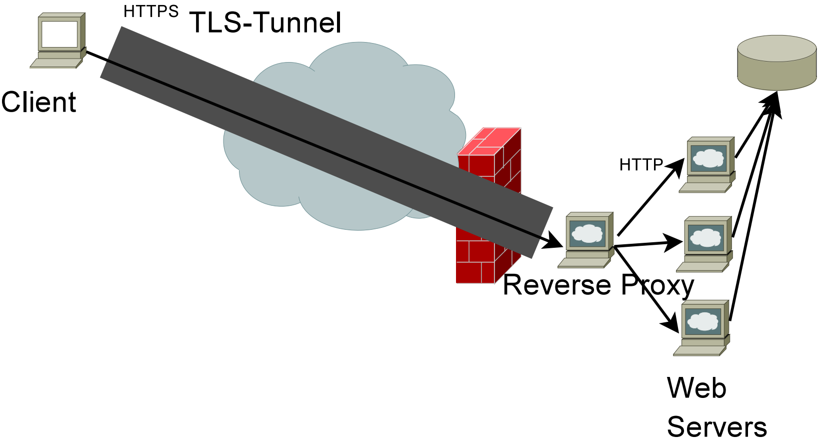
Efficiency of Cryptography

- **Public Key Cryptography**
  - Very useful between unknown persons
  - Requires long keys
  - Is too slow
  - Can not be used for big transfer
- **Symetric Key Cryptography**
  - Reserved for people that know each other
  - Can be much more efficient
  - Should be used to transfer large data
- **Solution**
  - Use the two systems
  - First “Hand Shaking” using a public key
  - Then (once we know each other) use a symetric key algorithm

SSL and TLS

- **TCP/IP socket can be read and modified**
  - TCP/IP does not contain any security mechanism
  - Idea: we trust the others
- **SSL and TLS**
  - Create a socket that can neither be read nor modified
  - *"Tuneling"*
- **Content is protected from its origin to its destination**
  - Content is encrypted and signed
  - Protocol prevents any modification

Tunnel



Three Phases of TLS

1. **Peer negotiation for algorithm support**
   - For key Exchange: RSA, Difie-Hellman, DSA, SRP, PSK
   - For symetric ciphers: RC4, Triple DES, AES or Camellia
   - For crypto hash function (Message authentication codes - MAC): HMAC-MD5 or HMAC-SHA
2. **Key exchange and authentication**
   - Exchange of Certificate(s) (normally X.509, draft for OpenPGP)
   - Verification of the certificate(s) (can ask the CA if it is still valid)
   - Exchange of a new secret key for symetric encryption
3. **Symmetric cipher encryption and message authentication**
   - Symetric encryption is faster

Advantages of TLS

- **Eavesdropping**
  - The data transfered on the net are crypted. It is not possible to read it.
- **Data Modification**
  - Since consitancy of data is checked using MAC hash functions, content can not be modified
- **Man in the middle attack**
  - The client is certain to be faced with the server possessing the certificate.

Problems with HTTPS / TLS-SSL

- **Relies on the PKI infrastructure**
  - Revocation of Certificates have to be tested
  - Certificate Authorities have to be known (and trusted)
  - X.509 relies on a Root certificate, should not be protected
- **Man in the Middle attack**
  - Possible: the user is warned and clicks the button *OK*
- **Interception / Modification**
  - Much more complicated than with HTTP

Create your own HTTPS server

- **Create a Certificate** (Standard X.509)
  Contains
  - Identity and Address of the subject
  - Validity (not before, not after)
  - Public Key Information (algorithm and key)
- **Let a CA sign your certificate** You add the following information in your certificate
  - Identity and Address of the issuer of the certificate
  - Signature (algorithm and fingerprint)
- **Configure the port 443 of your server**
  - Update the configuration of your server such that it listens to this port using HTTPS.

Sample Certificate

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                   ...
                   d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                   e8:35:1c:9e:27:52:7e:41:8f
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
       92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
       ...
       8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
       68:9f
```

Hosting multiple servers on one computer

- **Virtual Hosts very common on Apache**
  - One IP-address can correspond to many names (DNS's point at the same address)
  - One program listens on the port 80 of the computer
  - HTTP header contains a field (mandatory) `Host:`
  - The program can create virtual hosts for each of the host names.
- **Problem with virtual hosts for HTTPS**
  - A https server listens to the port 443
  - Encrypted content arrives,
  - It can not be redirected to the right server for authentication
  - The requests are all directed toward one single server.
- **Solution: Having One IP-Address per Virtual-host**
  - Virtual hosts can listen to the different ports for the different IP addresses.

Double Side Authentication

- **HTTPS means : authentification of the server**
  - The client verifies that the server he/she talks to is the right one
  - The classic web programming is used for identification / authentification of the client: Username + password
  - Why not use the same mechanism in both directions
- **Users May be authenticated by a certificate**
  - The browser may send its own certificate to the server
  - Certificate and Private key may be contained inside the browser, on a chip card or USB stick.

Apache Configuration

- **You have to create a Certificate**
  - `server.crt` your certificate (signed by a CA or self-signed)
  - `server.key` your private key
- **Install SSL module**
  - Module takes care of the protocole
  - Load specific configuration for the module
- **Create a Virtual Host for your SSL server**
  - CipherSuite (which protocols are supported)
  - Address of the certificate and Key
  - Port (or IP-address and Port) to be listened

Apache Configuration

For authenticating the users also

- **Definition of the CA's certificates (CRT)**

```
#SSLCACertificatePath /opt/lampp/etc/ssl.crt
#SSLCACertificateFile /opt/lampp/etc/ssl.crt/ca-bundle.
```

- **Definition of revocation lists (CRL)** (for the CA's)

```
#SSLCARevocationPath /opt/lampp/etc/ssl.crl
#SSLCARevocationFile /opt/lampp/etc/ssl.crl/ca-bundle.c
```

- **Set properties for clients**
  - Protect one directory
  - Protect the whole server
  - . . .

Limitations: Server Problems

- **Private Key may be compromized** (stollen, copied, changed, . . .)
  - Man in the middle attack possible
  - Revocation list
- **Content of the site may have been changed**
  - By malicious administrators
  - By visitors

Limitations: Client Problems

- **Client may not pay attention to security warnings**
  - Manual Verification of Certificate
- **Client may be compromized**
  - Virus,
  - Trojan,
  - Worm,
  - may infect the client computer
  - Strength of the crypto depends on the client (in the handshaking part of the protocol)
- **Client may be malicious**
  - You do have 0 control on the client
  - Never trust the client side verification (javascript for instance)

References

- **Apache SSL/TLS Encryption**
  http://httpd.apache.org/docs/2.2/ssl/
- **Wikipedia**
  http://en.wikipedia.org/wiki/Transport_Layer_Security
  http://en.wikipedia.org/wiki/X.509
- **Other resources**
  http://sebsauvage.net/comprendre/encryptage/crypto_asy.html
  http://www.openssl.org
  http://www.authsecu.com/ssl-tls/ssl-tls.php
  http://www.hsc.fr/ressources/presentations/pki/