Cryptography – SSL/TLS
History

• Secure Sockets Layer was developed by Netscape in 1994 as a protocol which permitted persistent and secure transactions.

• In 1997 an Open Source version of Netscape’s patented version was created, which is now OpenSSL.

• In 1999 the existing protocol was extended by a version now known as Transport Layer Security (TLS).

• By convention, the term "SSL" is used even when technically the TLS protocol is being used.
SSL/TLS Protocol

Application (HTTP)
Encrypted SSL/TLS data [HTTP]
Transport (TCP)
Internet
Network Access
TLS/SSL: What it does

• Confidentiality
  – Encryption

• Integrity
  – Keyed hash (HMAC): TLS (authentication!)
  – Hash (MAC): SSL

• Authentication
  – certificates
SSL/TLS Operations

• Client connects to the server
  – To access a resource

• Public-key cryptography during initial handshake to authenticate and exchange session keys
  – PKI (X.509 Certificates)

• Symmetric key cryptography to encrypt and hash data
  – Master secret (shared secret) generated
  – Separate **Encryption** and **Hashing** keys from the master secret
How SSL/TLS Works – Part 1

SSL Handshake Phase

1. Sends Hello Message
   - Supported algorithms, random number

2. Algorithms, random number
   - Sends Hello Message

3. Certificate
   - Sends Certificates

4. Authentication Server
   - Generates random value (pre-master secret & encrypts it with the server’s public key)

5. Encrypted pre-master secret
   - Decerts to retrieve pre-master secret

6. Encrypted pre-master secret
   - Decrypts to retrieve pre-master secret

7. Calculates Keys
   - Master Secret (shared)
     - Encryption key
     - Hashing key

8. Sends finished message
   - Sends finished message

9. SSL Data Transfer Phase
   - Data
   - Encrypted data
**PKI – public key infra**

- Digital (X.509) certificates
  - associates a public key with an individual or organization
    - public key of the subject!

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Version of X.509</td>
</tr>
<tr>
<td>Serial Number</td>
<td>Uniquely identifies the certificate</td>
</tr>
<tr>
<td>Signature Algorithm</td>
<td>Algorithms used by the CA to sign the cert</td>
</tr>
<tr>
<td>Issuer Name</td>
<td>Id of the CA (that issued the cert)</td>
</tr>
<tr>
<td>Validity Period</td>
<td>Cert validity</td>
</tr>
<tr>
<td>Subject Name</td>
<td>The cert owner</td>
</tr>
<tr>
<td><strong>Subject Public Key</strong></td>
<td>Owner’s public key</td>
</tr>
<tr>
<td>Issuer ID</td>
<td>Extra info (Issuer of the cert)</td>
</tr>
<tr>
<td>Subject ID</td>
<td>Extra info (owner of the cert)</td>
</tr>
<tr>
<td>Extensions (CRL)</td>
<td></td>
</tr>
<tr>
<td>CA Digital Signature</td>
<td>Digital Signature of the CA</td>
</tr>
</tbody>
</table>
PKI – Chain of Trust

• Root CA
  – Self-signed
  – Issue and sign ICA’s certificate

• Intermediate CA
  – Issue and sign EE certificate

• End Entity
PKI – Example

- Client (browser) sends https request to google.com
  - browsers have trusted CA certificates stored

- Web server sends back google.com’s certificate
  - Signed by Google ICA, plus
  - Google ICA’s certificate signed by root CA (GeoTrust)

- Verify the certificates up the chain of trust
  - Once successfully verified, use the public key
How SSL/TLS Works – Part 2

**SSL Handshake Phase**

1. **CLIENT** sends Hello Message
2. **SERVER** sends Hello Message
3. **CLIENT** sends Certificate
4. **SERVER** sends Certificate
5. **CLIENT** generates random value (pre-master secret & encrypts it with the server’s public key)
6. **SERVER** decrypts to retrieve pre-master secret
7. **CLIENT** calculates Keys
8. **SERVER** calculates Keys
9. **CLIENT** sends finished message
10. **SERVER** sends finished message

**Symmetric Encryption**

- **Master Secret (shared)**
  - Encryption key
  - Hashing key

**SSL Data Transfer Phase**

11. **CLIENT** sends data
12. **SERVER** sends data

**Encrypted data**
Symmetric Encryption

• Once the server’s public key is verified up the chain of trust
  – The client generates a pre-master secret
    • C-random & S-random
  – Sends to the server encrypted with server’s public key

• Both client and server generates the Master Secret
  – Uses the pre-master secret, C-random, and S-random with the agreed key exchange cipher (eg: DH)

• Separate Encryption and Hashing keys generated from the Master secret
  – All future communication hashed and encrypted using the symmetric keys
Trusted vs Non-trusted Certificate

This Connection is Untrusted
You have asked Firefox to connect securely to owncloud.rg.net, but we can’t confirm that your connection is secure.

What Should I Do?
If you usually connect to this site without problems, this error could mean that someone is impersonating the site, and you shouldn’t continue.

Technical Details
I Understand the Risks

Page Info - https://www.facebook.com/
Website Identity
Website: www.facebook.com
Owner: This website does not supply ownership information.
Verified by: DigiCert Inc

Privacy & History
Have I visited this website prior to today? Yes, 95 times
Is this website storing information (cookies) on my computer? Yes
Have I saved any passwords for this website? No

Technical Details
Connection Encrypted: High-grade Encryption (TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, 128 bit keys)
The page you are viewing was encrypted before being transmitted over the Internet. Encryption makes it very difficult for unauthorized people to view information traveling between computers. It is therefore very unlikely that anyone read this page as it traveled across the network.
## Certificate Authority

You have certificates on file that identify these certificate authorities:

<table>
<thead>
<tr>
<th>Certificate Name</th>
<th>Security Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeoTrust Inc.</td>
<td></td>
</tr>
<tr>
<td>GeoTrust Global CA</td>
<td>Built-in Object Token</td>
</tr>
<tr>
<td>GeoTrust Global CA 2</td>
<td>Built-in Object Token</td>
</tr>
<tr>
<td>GeoTrust Universal CA</td>
<td>Built-in Object Token</td>
</tr>
<tr>
<td>GeoTrust Universal CA 2</td>
<td>Built-in Object Token</td>
</tr>
<tr>
<td>GeoTrust Primary Certification Authority</td>
<td>Built-in Object Token</td>
</tr>
<tr>
<td>GeoTrust Primary Certification Authority - G3</td>
<td>Built-in Object Token</td>
</tr>
<tr>
<td>GeoTrust Primary Certification Authority - G2</td>
<td>Built-in Object Token</td>
</tr>
<tr>
<td>GeoTrust DV SSL CA - G3</td>
<td>Built-in Object Token</td>
</tr>
<tr>
<td>GeoTrust Extended Validation SHA256 SSL CA</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>RapidSSL SHA256 CA</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>GeoTrust DV SSL CA - G4</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>RapidSSL SHA256 CA - G2</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>RapidSSL SHA256 CA - G4</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>GeoTrust EV SSL CA - G4</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>RapidSSL SHA256 CA - G3</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>GeoTrust SSL CA - G3</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>GeoTrust SHA256 SSL CA</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>Google Internet Authority G2</td>
<td>Software Security Device</td>
</tr>
</tbody>
</table>
WoSign

MONDAY, AUGUST 29, 2016

Chinese CA WoSign faces revocation after issuing fake certificates of Github, Microsoft and Alibaba

One of the largest Chinese root certificate authority WoSign issued many fake certificates due to an vulnerability. WoSign's free certificate service allowed its users to get a certificate for the base domain if they were able to prove control of a subdomain. This means that if you can control a subdomain of a major website, say percy.github.io, you're able to obtain a certificate by WoSign for github.io, taking control over the entire domain.

In deed, this has been seen in the wild in multiple instances as reported in the thread, aggregated here. I've notified related parties about the possible fake certs.

Possible fake cert for Github -- confirmed fake
https://crt.sh/?id=29647048
https://crt.sh/?id=29805567

Update: crt.sh is down after my post. Google's CT log here https://www.google.com/transparencyreport/https/ct/#domain=github.io&incl_exp=false&incl_sub=false&issuer=IPrsb9Gbn4s%3D

Possible fake cert for Alibaba, the largest commercial site in China -- confirmed fake
https://crt.sh/?id=29884704

https://groups.google.com/forum/m/#!topic/mozilla.dev.security.policy/k9PBmyLCi8I/discussion
Introducing Let’s Encrypt

Let’s Encrypt is a new Certificate Authority: **It’s free, automated, and open.**

- An open source CA
  - Proof your domain to get your digital (TLS/SSL) certificate
  - [https://letsencrypt.org](https://letsencrypt.org)
Let’s Encrypt chain

- Let’s Encrypt ICA (X3) cross-signed by DST (IdenTrust)
  - Until ISRG (Internet Security Research Group) is trusted by everyone
Introducing Let’s Encrypt

• Browsers and OS
  – https://community.letsencrypt.org/t/which-browsers-and-operating-systems-support-lets-encrypt/4394

• Check your browser
  – https://wiki.apnictraining.net
  – (signed by Let’s Encrypt)
Lab Exercise
Setup

• Follow lab manual ssl_lab.pdf