Introduction to Information Security

SSL & TLS
Story of a protocol

Itamar Gilad (infosec15 at modprobe dot net)

Motivations for SSL/TLS

- ~~Military & state secrets~~
- Medical & "regular" business confidential information
- **Online banking & online shopping**
- Secure access to online services (webmail, social networks and many different cloud services)
- Protecting against phishing attacks, traffic manipulation, cookie stealing and many other forms of attack

- Basic privacy – Who did I speak to, when, about what and why?

Definitions & History

- HTTPS = HTTP Secure ("HTTP over SSL")
- SSL = Secure Socket Layer
  - ~~1.0 <created by Netscape, never released>~~
  - 2.0 1995
  - 3.0 1996 (comprehensive re-design!)
  - superseded by TLS, but v3 was often enabled and used until 2014 (!)
- TLS = Transport Layer Security
  - v1.0 (1999)
  - v1.1 (2006)
  - v1.2 (2008) – current stable version
  - v1.3 (2015?) – draft status
- Leading implementations –
  - S-Channel (Windows)
  - OpenSSL, LibreSSL (community fork), BoringSSL (fork by Google)
  - GnuTLS (Linux and others)
  - Mozilla NSS (Many), BouncyCastle (Java, C#, etc.), many more…

Early SSL History

- Leading design criteria: protection against a passive attacker
- Simplified session stages –
  - Alice & Bob exchange session keys using Diffie-Hellman / RSA
  - They also negotiate a symmetric cipher algorithm they both can use
  - Alice uses her session key to encrypt connection data, and sends her data to Bob
  - Bob does the same in the other direction
  - And they both lived happily ever after…
  - Or did they?

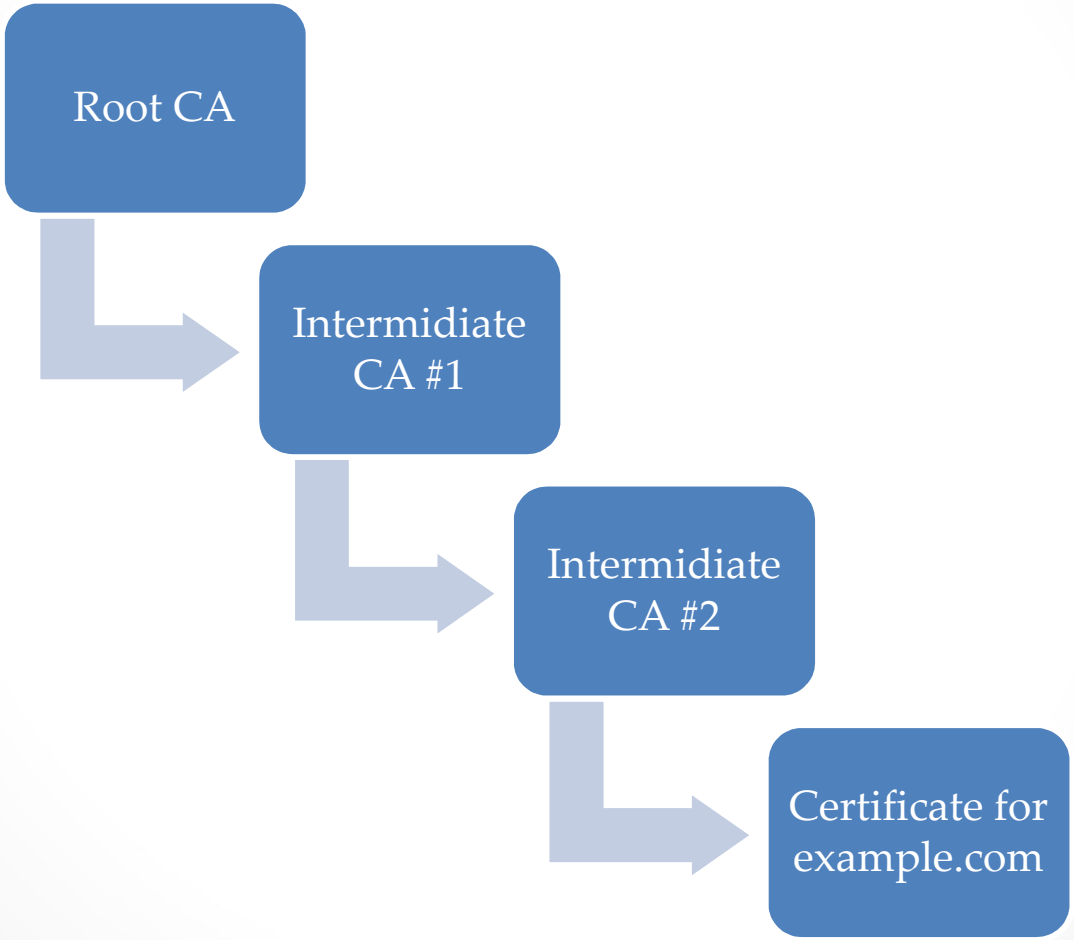
Eve is Passive-Aggressive

- Turns out that Eve can perform a MITM (Man-In-The-Middle) attack
  - Eve can pretend to be Bob as far as Alice is concerned
  - Eve can pretend to be Alice as far as Bob is concerned
  - Therefore – Eve can place herself in the middle of the connection in a transparent manner, and have access to the secret contents of the communication
- These attacks are not theoretical. They can be carried out –
  - By a rogue sys-admin / ISP / Internet exchange
  - By a malicious/compromised user on the local network
- The key issue is a lack of authentication
-

Solutions to the MITM Issue

- Have a pre-shared secret between both parties!
- But that's really hard…
- Solution: Authenticate using a common third party both sides can trust
- And thus, the CA (Certificate Authority) model was born
- Basic concept –
  - Alice trusts the CA
  - Bob presents a certificate to Alice, signed by the CA
  - Alice and Bob use information from the certificate to set up an authenticated session
  - Eve can no longer MITM, because Alice will know and promptly terminate the session
  - (Often, only the server will be authenticated explicitly, but there is support for client authentication as well)

More CAs


Root CA
Intermidiate CA #1
Intermidiate CA #2
Certificate for example.com

Being a CA is hard

- Problem –
  - As a CA, you have to verify each certificate request and stake your reputation on it
- Solution –
  - More CAs were created
  - New, more thorough validation standards added (EV = Extended Validation)
  - CAs delegated further. Most OS and browsers will trust 10-100 root CAs out of the box, and 100-1000 CAs implicitly

Honest Achmed 2011-04-06 02:31:02 PDT Description

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.2.16) Gecko/20110319 Firefox/3.6.16
Build Identifier:

This is a request to add the CA root certificate for Honest Achmed's Used Cars and Certificates. The requested information as per the CA information checklist is as follows:

1. Name

Honest Achmed's Used Cars and Certificates

2. Website URL

www.honestachmed.dyndns.org

3. Organizational type

Individual (Achmed, and possibly his cousin Mustafa, who knows a bit about computers).

4. Primary market / customer base

Absolutely anyone who'll give us money.

5. Impact to Mozilla Users

Achmed's business plan is to sell a sufficiently large number of certificates as quickly as possible in order to become too big to fail (see "regulatory capture"), at which point most of the rest of this application will become irrelevant.

Kyle Hamilton 2011-04-13 13:49:27 PDT Comment 1

Honest Achmed is at least more honest than Comodo.

Issues with the CA model

- Each and every CA your system trusts can be evil (or at least – incompetent) and **compromise everything**
  - CAs can be fooled into signing bad certificates
  - Some CAs were compromised (DigiNotar)
  - Some CAs are "evil"/forced into being evil (TurkSign, CNNIC)
  - Even worse – even when shown to be bad, some CAs were not removed (Comodo) because they were "too big to fail". More info
- The root cause –
  - The CA model was designed when the Internet was a very small and innocent place
  - Designers of this system never imagined having to verify and issue 10M+ certificates

Fixing the CA model

- Certificate Pinning –
  - Storing a digest of the certificate inside the browser/application, and validation the otherwise trusted certificate yourself
  - Very effective in preventing tampering, but can cause other issues (e.g.: availability)
- Signing restrictions –
  - Defining authoritative bodies for different domain suffixes to limit the effect of rogue bodies (i.e.: CNNIC can only sign for *.cn)
- Certificate transparency –
  - Each CA publishes a list of all the certificates it has ever signed
  - Only Certificates on that list are trusted
  - Anyone can audit the published list, and any wrongdoings can result in removing trust from that CA
  - Can prevent "back-room" signing (but not fix everything)

Alternatives to the CA model

- DANE (DNS-based Authentication of Named Entities) –
  - Relying on DNSSEC to solve the trust issue, which in turn means DNS zone controllers also control TLS authentication
- Web-of-trust model
  - Shifting trust to a continuum rather than a binary value
  - Users define a level of trust per peer, which make a weighted graph
  - End-to-end trust level is a product of all trust levels on the arcs of the best path

Certificates

- Each domain (or HTTP origin) must have a certificate (X.509 ITU-T)
- Basic structure:
  - Version, Serial Number
  - Algorithm ID
  - Issuer
  - Validity - Not Before & Not After dates
  - Subject (target)
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Certificate Signature Algorithm
  - Certificate Signature

Certificate Lifetime

- On some occasions, the certificate must be invalidated earlier than expected (e.g.: after a server holding the private key is known to be compromised)
- This is done using a protocol known as OCSP (Online Certificate Status Protocol)
- Basically, when creating a new TLS connection, the client must communicate with a "OCSP Responder" for that CA to verify that the certificate is still valid
- The OCSP response is signed by the CA and and verified by the client

Problems with OCSP

- If the attacker can already MITM the traffic, they can fail the OCSP checks (which usually fail-unsafe)
- OCSP overload –
  - OCSP is traffic **heavy** (an OCSP request should in theory be made before each HTTPS / TLS connection on the Internet...)
  - As an attacker, DDoSing the OCSP responders is very useful
- Performing an extra lookup to the OCSP responder means the responder can learn which hosts a client is communicating with
- Solution –
  - "OCSP Stapling" (TLS Certificate Status Request), which works by including a **recently** signed OCSP response in the TLS Handshake
  - Cannot be forged by the original certificate owner, because the OCSP response is signed by the CA, not the certificate holder