Application-Layer Security: SSL/TLS

SSL/TLS

- Most widely used application-layer security protocol
  - Used to protect web traffic
- Encrypts everything in the application layer
  - With a few exceptions
- Original: SSL 2.0 in 1995; now TLS 1.3
  - Each version change was due to security concerns
  - 1.2 is still common due to ossification
  - 1.3 cuts a round trip and strictly controls which protocols are allowed

SSL/TLS

Objectives:

1. Encryption, to protect confidentiality of network *payloads*
   - Achieved with AES (block cipher) in a decent mode of operation
2. Integrity, to ensure that information cannot be changed
   - Achieved with keyed hashes in Message Authentication Codes
   - Combined with 1) in AEAD, required for TLS 1.3
3. Authenticity, to ensure that users are talking to the correct destination (visiting the actual website)
   - Achieved with Certificate Authorities as its Public Key Infrastructure

Certificate Authorities

- A certificate authority sends a website a certificate signed with the CA's private key:

$$\text{Sign}_{CA}(\text{Verify}_{website})$$

- The web server's ability to sign with the private key corresponding to the public key is its proof of identity
- The web server sends the following to the client:

$$\text{Verify}_{website}, \text{Sign}_{CA}(\text{Verify}_{website}), \text{Sign}_{website}(\text{Message})$$

- The verification keys of all root CAs come with your browser
- Intermediate CAs would also need to obtain a signature from a root CA of its verification key

Certificate Authorities

- Used to be expensive for personal websites, so most such websites were not encrypted
- Politically, many countries have their own root CAs
  - Still, any root CA can sign any website
  - e.g. TurkTrust gave certificates for Google to unknown users
- Nowadays, 60%+ of certificates are signed by Let's Encrypt

2011 DigiNotar compromise

- DigiNotar was a Dutch root CA
- Signed fraudulent certificates, likely used for MITM attacks against Gmail in Iran
- Attack undisclosed for several months
- Revealed supply chain attacks as a serious issue for CAs

HTTP Public Key Pinning (Certificate Pinning)

- Right after the DigiNotar compromise in August 2011, Google required that Chrome would only accept certificates from Google's CA for google.com
- Later extended to Certificate Pinning mechanism:
  - Servers would declare which pinned certificates (=public key hashes) were usable to access that website
  - No other certificate was accepted (default expiry of 60 days)
  - Interacted poorly with revocation due to compromise and renewal
    - If the attacker manages to pin their key for your website, they could demand a ransom to buy their key; failure to comply means 60 days of downtime
  - Deprecated around 2019

Certificate Transparency

- Replacement for Certificate Pinning
- CAs publish all signed certificates to an append-only, globally shared log
- Browsers also check the log to see if certificates have been published
- Responsibility of an organization to check that its own domains have not had suspicious CA signatures

TLS Extensions: Server Name Indication

- In initial ClientHello, client can indicate which domain it wants to connect to
- This allows multiple web servers to live on the same IP address (and use multiple TLS certificates)
- SNI is not encrypted, which leaks information
  - How could it be encrypted? It is the first message of the handshake

TLS Extensions: Encrypted Client Hello

- We can actually encrypt the ClientHello
  - The key is distributed by DNS
- Splits ClientHello into Outer and Inner portions; Outer is for compatibility, while Inner is encrypted and contains the true SNI
- Initially implemented in 2018, default in most browsers since Sept 2023
- Banned in multiple countries

TLS 1.3 round trip technique

- Original TLS has two round trips:
  - 1. ClientHello: “Here are my protocols A B and C”
  - 1. ServerHello: “I will choose C.” + Certificate + ServerKeyExchange: “Here is my initial key material for protocol C.”
  - 2. ClientKeyExchange: “Here is my initial key material for protocol C.” + ChangeCipherSpec: “Let’s confirm we did things correctly.”
  - 2. Server also responds with ChangeCipherSpec
- TLS 1.3 has the client **guess** one or multiple protocols that the server would support and send key material in the original hello

DNS over HTTPS

- Separately, DNS still reveals domain names even if HTTPS did not with ECH
- DNS over HTTPS (DoH) allows the client to talk to recursive resolvers over HTTPS
  - DoH is both encrypted and authenticated
  - Default in some browsers

Attacks

- Flame: Surveillance malware (2012) that obtained a seemingly-valid SSL certificate for Microsoft
- Used an old Microsoft server that signed certificates with MD5, without breaking into it
- MD5 enables a length extension attack:

  Given h(key||x), it is easy to find h(key||x||y)

- If x is a valid piece of code signed by Microsoft, you can append y to that code and replace the hash to obtain another valid certificate
- Published in academia in 2008

Attacks

- Encryption modes such as CBC are no longer supported due to cryptographic vulnerabilities
- SSL2 used short “export-grade” keys and servers often allowed downgrade to SSL2 even when they supported TLS
- Side channel includes compression size (“CRIME attack”):
  - Victim sends secret cookie in HTTP header; attacker can inject plaintext in header (e.g. when visiting attacker website)
  - If the real key is “123” and attacker injection is “456”, no compression occurs (size = 6)
  - If attacker injection is “123”, compression occurs (size = 5) -> guess is correct
- Another side channel is timing (“Lucky Thirteen Attack”)