CS 253: Web Security
Transport Layer Security
The problem with HTTP

- HTTP is not "secure"
  - Why?
HTTP request-response
POST /login HTTP/1.1
user=alice&pass= hunter2
POST /login HTTP/1.1
user=alice\&pass=hunter2

HTTP/1.1 200 OK
Set-Cookie: sessionId=1234;
<!doctype html> good html
POST /login HTTP/1.1
user=alice&pass=hunter2
POST /login HTTP/1.1
user=alice&pass=hunter2

HTTP/1.1 200 OK
Set-Cookie: sessionId=1234;
<!doctype html> good html
Active attacker
GET / HTTP/1.1

HTTP/1.1 200 OK
<!doctype html> evil html

Inject attack code

HTTP/1.1 200 OK
<!doctype html> good html

GET / HTTP/1.1
What is the threat model?

- **Network attackers** control network infrastructure like routers or DNS servers
- Network attackers may eavesdrop, inject, block, or modify packets
- Potential network attackers occur anywhere there is an untrusted router or ISP
  - Wireless networks at cafes or hotels
  - Border gateways between countries
Goal: Secure communication

- Secure communication requires three properties
  - **Privacy:** No eavesdropping
  - **Integrity:** No tampering
  - **Authentication:** No impersonation
Goal: Secure communication
GET / HTTP/1.1

Client

Attacker

Server
example.com

Feross Aboukhadijeh
Client

GET / HTTP/1.1

Attacker

GET / HTTP/1.1

Server

text

HTTP/1.1 200 OK

Response is from example.com? OK!
Transport Layer Security (TLS)

- Hypertext Transfer Protocol Secure (HTTPS) keeps browsing safe by securely connecting the browser with the website server.
- HTTPS relies on Transport Layer Security (TLS) encryption to secure connections.
- TLS is used with web traffic, email, instant messaging, voice over IP (VoIP), and many other protocols.
  - When TLS is used with HTTP, we call it HTTPS.
Anonymous Diffie-Hellman key exchange
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

$a \leftarrow \{1, \ldots, q\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$a \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

Server

$b \leftarrow \{1, \ldots, q\}$

example.com
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$a \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$B = g^b \in G$

Server

$b \leftarrow \{1, \ldots, q\}$

example.com
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$\text{a } \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$B = g^b \in G$

Server

$\text{b } \leftarrow \{1, \ldots, q\}$

DHKey = $g^{ab}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$\text{DHKey} = B^a$

$A = g^a \in G$

Server

$B = g^b \in G$

$\text{DHKey} = g^{ab}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$a \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$B = g^b \in G$

Server

$b \leftarrow \{1, \ldots, q\}$

DHKey $= B^a$

DHKey $= g^{ab}$

DHKey $= A^b$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

$a \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$b \leftarrow \{1, \ldots, q\}$

$B = g^b \in G$

$\text{DHKey} = g^{ab}$

$(g^b)^a = B^a = (g^a)^b = A^b$
Anonymous key exchange

- Problem: Client doesn't know with which server it performed key exchange
  - It's possible that the client securely derived a key with the network attacker instead of the intended server!
  - While the communication is technically private (secure against eavesdropping), it lacks authentication
- Key idea: Without authentication, you can't actually have privacy
Man-in-the-middle attack on Anonymous Diffie-Helman key exchange
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$\text{a} \leftarrow \{1, \ldots, q\}$

Attacker

$c \leftarrow \{1, \ldots, q\}$

Server

example.com
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

- Client:
  - $a \leftarrow \{1, \ldots, q\}$
  - $A = g^a \in G$

- Attacker:
  - $c \leftarrow \{1, \ldots, q\}$

- Server:
  - $b \leftarrow \{1, \ldots, q\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$a \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

Attacker

$c \leftarrow \{1, \ldots, q\}$

$C = g^c \in G$

Server

$b \leftarrow \{1, \ldots, q\}$

example.com
Group $G = \{1, g, g^2, g^3, ..., g^{q-1}\}$

Client

\[ a \in \{1, ..., q\} \]

\[ a \xleftarrow{} A = g^a \in G \]

\[ c \in \{1, ..., q\} \]

\[ c \xleftarrow{} C = g^c \in G \]

\[ \text{DHKey}_1 = g^{ac} \]

Attacker

Server

\[ b \in \{1, ..., q\} \]

\[ \text{example.com} \]
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$a \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$C = g^c \in G$

$\text{DHKey}_1 = C^a$

$\text{DHKey}_1 = g^{ac}$

Attacker

$c \leftarrow \{1, \ldots, q\}$

Server

$b \leftarrow \{1, \ldots, q\}$

example.com
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$a \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$C = g^c \in G$

$\text{DHKey}_1 = C^a$

Attacker

$c \leftarrow \{1, \ldots, q\}$

$\text{DHKey}_1 = A^c$

Server

$b \leftarrow \{1, \ldots, q\}$

example.com

52 Feross Aboukhadijeh
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$a \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$C = g^c \in G$

$\text{DHKey}_1 = C^a$

$\text{DHKey}_1 = g^{ac}$

$\left(g^c\right)^a = C^a = \left(g^a\right)^c = A^c$

Attacker

$c \leftarrow \{1, \ldots, q\}$

$\text{DHKey}_1 = A^c$

Server

$b \leftarrow \{1, \ldots, q\}$

example.com

53 Feross Aboukhadiejh
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$a \leftarrow \{1, \ldots, q\}

A = g^a \in G$

C = g^c \in G

DHKey$_1 = C^a$

DHKey$_1 = g^{ac}$

\[(g^c)^a = C^a = (g^a)^c = A^c\]

Attacker

$c \leftarrow \{1, \ldots, q\}$

$\text{DHKey}_1 = A^c$

Server

$b \leftarrow \{1, \ldots, q\}$

\[B = g^b \in G\]

$C = g^c \in G$

55 Feross Aboukhadijeh
Group $G = \{1, g, g^2, g^3, ..., g^{q-1}\}$

Client

- $a \leftarrow \{1, ..., q\}$
- $A = g^a \in G$
- $C = g^c \in G$
- $\text{DHKey}_1 = C^a = (g^a)^c = A^c$

Attacker

- $c \leftarrow \{1, ..., q\}$
- $\text{DHKey}_1 = g^{ac}$

Server

- $b \leftarrow \{1, ..., q\}$
- $B = g^b \in G$
- $C = g^c \in G$
- $\text{DHKey}_2 = g^{bc}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client:
- $a \leftarrow \{1, \ldots, q\}$
- $\text{DHKey}_1 = C^a$

Attacker:
- $c \leftarrow \{1, \ldots, q\}$
- $\text{DHKey}_1 = g^{ac}$
- $\text{DHKey}_1 = A^c$
- $\text{DHKey}_1 = g^{ac}$
- $\text{DHKey}_2 = B^c$
- $\text{DHKey}_2 = g^{bc}$

Server:
- $b \leftarrow \{1, \ldots, q\}$
- $B = g^b \in G$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$\begin{align*}
a &\in \{1, \ldots, q\} \\
A &= g^a \in G \\
C &= g^c \in G \\
\text{DHKey}_1 &= g^{ac} \\
(g^c)^a &= C^a = (g^a)^c = A^c
\end{align*}$

Attacker

$\begin{align*}
c &\in \{1, \ldots, q\} \\
\text{DHKey}_1 &= A^c \\
\text{DHKey}_2 &= B^c \\
C &= g^c \in G \\
B &= g^b \in G \\
\text{DHKey}_2 &= g^{bc} \\
\text{DHKey}_2 &= C^b
\end{align*}$

Server

$b \in \{1, \ldots, q\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
How do we get authentication?

- **Goal:** If the client could authenticate the server it is performing key exchange with, then it could securely derive a shared key with that (and only that) server

- **Solution:** Use **public-key cryptography** for authentication
  - Remember **signing** from lecture 3 on cookies?
  - Let's review
Review: Signature schemes

- Triple of algorithms \((G, S, V)\)
  - \(G() \rightarrow (pk, sk)\) - generator returns public key and secret key
  - \(S(sk, x) \rightarrow t\) - signing returns a tag \(t\) for input \(x\)
  - \(V(pk, x, t) \rightarrow \text{accept} | \text{reject}\) - checks validity of tag \(t\) for given input \(x\)

- Algorithm properties
  - **Correctness property:** \(V(pk, x, S(sk, x)) = \text{accept}\) should always be true
  - **Security property:** \(V(pk, x, t) = \text{accept}\) should almost never be true when \(x\) and \(t\) are chosen by the attacker
Authenticated Diffie-Hellman key exchange
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$pk$

$a \leftarrow \{1, \ldots, q\}$

Server

example.com

$sk$

$b \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$pk$

$a \leftarrow \{1, \ldots, q\}$

Server

example.com

$sk$

$b \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$S(sk, \text{transcript}) \rightarrow t$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

- $pk$
- $a \leftarrow \{1, \ldots, q\}$

Server

- $sk$
- $b \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$B = g^b \in G, t$

$S(sk, \text{transcript}) \rightarrow t$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

pk

\[ a \leftarrow \{1, \ldots, q\} \]

\[ A = g^a \in G \]

$S(sk, \text{transcript}) \rightarrow t$

\[ B = g^b \in G, t \]

V(pk, \text{transcript}, t) \rightarrow \text{ok?}
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$pk$

$a \leftarrow \{1, \ldots, q\}$

Server example.com

$sk$

$b \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$B = g^b \in G, t$

$S(sk, \text{transcript}) \rightarrow t$

$V(pk, \text{transcript}, t) \rightarrow \text{ok?}$ OK!
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

- $a \leftarrow \{1, \ldots, q\}$
- $A = g^a \in G$

Server

- $b \leftarrow \{1, \ldots, q\}$
- $B = g^b \in G, t$
- $S(sk, \text{transcript}) \rightarrow t$

$V(pk, \text{transcript}, t) \rightarrow \text{ok?}$

$\text{OK!}$

$DHKey = g^{ab}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

- $pk$
- $a \leftarrow \{1, \ldots, q\}$
- $V(pk, \text{transcript}, t) \rightarrow \text{ok?}$

Server example.com

- $sk$
- $b \leftarrow \{1, \ldots, q\}$
- $S(sk, \text{transcript}) \rightarrow t$

$A = g^a \in G$

$B = g^b \in G, t$

DHKey = $B^a = g^{ab}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

1. $a \leftarrow \{1, \ldots, q\}$
2. $b \leftarrow \{1, \ldots, q\}$
3. $A = g^a \in G$
4. $B = g^b \in G, t$

$V(pk, transcript, t) \rightarrow ok? \quad \text{OK!}$

$DHKey = B^a$

$S(sk, transcript) \rightarrow t$

$DHKey = A^b = g^{ab}$
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

$pk$

$a \leftarrow \{1, \ldots, q\}$

$A = g^a \in G$

$B = g^b \in G, t$

$S(sk, \text{transcript}) \rightarrow t$

$V(pk, \text{transcript}, t) \rightarrow \text{ok?} \rightarrow \text{OK!}$

DHKey = $B^a$

Server example.com

$sk$

$b \leftarrow \{1, \ldots, q\}$

$S(sk, \text{transcript}) \rightarrow t$

DHKey = $A^b$

$(g^b)^a = B^a = (g^a)^b = A^b$
How does the client get the server's public key?

- Idea: Build in every website's public key into the browser
  - Would be a huge list, constantly changing, cannot connect to server if list is out-of-date
- Idea: Server can send the public key to the client during the key exchange
  - Back to the same problem as anonymous key exchange!
  - What's to stop an active network attacker from send their own public key in the exchange?
Group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Client

- $pk$
- $a \leftarrow \{1, \ldots, q\}$
- $A = g^a \in G$
- $V(pk, transcript, t) \rightarrow ok?$  
  \text{OK!}$
- $DHKey = B^a$

Server

- $sk$
- $b \leftarrow \{1, \ldots, q\}$
- $B = g^b \in G, t$
- $S(sk, transcript) \rightarrow t$
- $DHKey = A^b$

$(g^b)^a = B^a = (g^a)^b = A^b$
Certificate authorities (CAs)

- A certificate authority (CA) is an entity that issues digital certificates.

- A certificate certifies that a named subject is the owner of a specific public key.
  - "I, CERTIFICATE_AUTHORITY, certify that SUBJECT_NAME is the owner of public key PUBLIC_KEY"
Common name rules

- Subject's **CommonName** can be:
  - an explicit name, e.g. `cs.stanford.edu`
  - a wildcard cert, e.g. `*.stanford.edu` or `cs*.stanford.edu`

- Matching rules
  - The * must occur in the leftmost subdomain component
  - The * does not match . characters
  - Example: `*.a.com` matches `x.a.com` but not `y.x.a.com`
Who does your browser trust?

<table>
<thead>
<tr>
<th>Certificate Name</th>
<th>Security Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-Trust GmbH</td>
<td></td>
</tr>
<tr>
<td>Dhimydis</td>
<td></td>
</tr>
<tr>
<td>DigiCert Inc</td>
<td></td>
</tr>
<tr>
<td>Digital Signature Trust Co.</td>
<td></td>
</tr>
<tr>
<td>Dirig a.s.</td>
<td></td>
</tr>
<tr>
<td>E-Türe EBO Bilişim Teknolojileri ve Hizmetleri A.Ş.</td>
<td></td>
</tr>
<tr>
<td>eMudhra Inc</td>
<td></td>
</tr>
<tr>
<td>eMudhra Technologies Limited</td>
<td></td>
</tr>
<tr>
<td>Entrust, Inc.</td>
<td></td>
</tr>
<tr>
<td>Entrust.net</td>
<td></td>
</tr>
<tr>
<td>FNMT-RCM</td>
<td></td>
</tr>
<tr>
<td>GeoTrust Inc.</td>
<td></td>
</tr>
<tr>
<td>GlobalSign</td>
<td></td>
</tr>
<tr>
<td>GlobalSign nv-sa</td>
<td></td>
</tr>
<tr>
<td>GoDaddy.com, Inc.</td>
<td></td>
</tr>
<tr>
<td>Google Trust Services LLC</td>
<td></td>
</tr>
<tr>
<td>Government Root Certification Authority</td>
<td></td>
</tr>
<tr>
<td>GUANG DONG CERTIFICATE AUTHORITY CO., LTD.</td>
<td></td>
</tr>
<tr>
<td>Hellenic Academic and Research Institutions Cert...</td>
<td></td>
</tr>
<tr>
<td>Hongkong Post</td>
<td></td>
</tr>
<tr>
<td>IdenTrust</td>
<td></td>
</tr>
<tr>
<td>Internet Security Research Group</td>
<td></td>
</tr>
</tbody>
</table>
Certificate exchange
Client

pk_{CA}

Server

example.com

G() → (pk, sk)

pk, proof "I am example.com"

Certificate Authority

sk_{CA}
Client

$pk_{CA}$

Server

eexample.com

$G() \rightarrow (pk, sk)$

pk, proof "I am example.com"

Check proof

$S(sk_{CA}, 'example.com key is pk') \rightarrow cert$
Client

pk_{CA}

Server
example.com

G() \rightarrow (pk, sk)

pk, proof "I am example.com"

Check proof

S(sk_{CA}, 'example.com key is pk') \rightarrow cert

cert

One time process!
Client

$pk_{CA}$

Server

example.com

$G() \rightarrow (pk, sk)$

pk, proof "I am example.com"

Certificate Authority

$sk_{CA}$

Check proof

S($sk_{CA}$, 'example.com key is pk') $\rightarrow$ cert

cert

One time process!
Client

$pk_{CA}$

Server
example.com

$G() \rightarrow (pk, sk)$

Certificate Authority

$sk_{CA}$

pk, proof "I am example.com"

Check proof

$S(sk_{CA}, 'example.com key is pk') \rightarrow cert$

cert

One time process!

V($pk_{CA}, cert$) $\rightarrow$ ok?

OK!

cert is for example.com?
Client

\(pk_{CA}\)

<table>
<thead>
<tr>
<th>Server example.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G() \rightarrow (pk, sk))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Certificate Authority</th>
</tr>
</thead>
<tbody>
<tr>
<td>(sk_{CA})</td>
</tr>
</tbody>
</table>

pk, proof "I am example.com"

Check proof

S\(sk_{CA}, \text{'example.com key is pk'}\) \rightarrow cert

cert

One time process!

\(V(pk_{CA}, cert) \rightarrow ok?\) OK!

cert is for example.com? OK!
Client

pk_{CA}

Server

example.com

G() → (pk, sk)

pk, proof "I am example.com"

Check proof

S(sk_{CA}, 'example.com key is pk') → cert

Certificate Authority

sk_{CA}

cert

One time process!

V(pk_{CA}, cert) → ok? OK!

cert is for example.com? OK!

Reuse cert for all clients
TLS 1.3

- TLS 1.3 is the latest version of TLS which replaces TLS 1.2, which replaced TLS 1.1, 1.0, SSL 3.0, 2.0, 1.0.

- **Goal:** "provide privacy and reliability between two communicating applications"

- Two phase protocol
  - **Handshake protocol:** Establish a shared secret key using public-key cryptography
  - **Record protocol:** Transmit data using the negotiated key
HTTPS requirements for lock icon

- All elements on the page must be fetched using HTTPS
- For all elements
  - HTTPS certificate must be issued by a CA trusted by browser
  - HTTPS certificate must not be expired
  - HTTPS certificate `CommonName` or `SubjectAlternativeName` must match the URL
TLS 1.3 session setup (simplified)

ClientHello: nonce_C, KeyShare

ServerHello: nonce_S, KeyShare, Enc[cert_S,...]

CertVerify: Enc[\text{Sig}_S(data)], Finished

Finished

session-keys ← HKDF(DHkey, nonce_C, nonce_S)

Encrypted ApplicationData

Most common: server authentication only
TLS 1.3 properties

- **Nonces**: Prevent replay of an old session
- **Forward secrecy**: server compromise does not expose old sessions
- **Some identity protection**: certificates are sent encrypted
- **One-sided authentication**: Client authenticates the server using the server's certificate
  - TLS has support for mutual authentication ("client certificates") but it is rarely used
HTTPS adoption

- Survey of top 100 non-Google sites on the internet, which account for 25% of website traffic worldwide

- Sites that **work** on HTTPS
  - ? / 100

- Sites that **default** to HTTPS
  - ? / 100
HTTPS adoption

- Survey of **top 100** non-Google sites on the internet, which account for 25% of website traffic worldwide

- Sites that **work** on HTTPS
  - 96 / 100

- Sites that **default** to HTTPS
  - 90 / 100
% pages loaded over HTTPS in Chrome by platform

Fragment navigations, history push state navigations, and all schemes besides HTTP/HTTPS (including new tab page navigations) are not included.
% Google pages loaded over HTTPS

- Across Google

Jan 01, 2014 to May 01, 2019
Why not 100%?

- Old excuses (not true anymore)
  - Crypto is slow
  - Ad networks do not support HTTPS
- "Mobile devices account for the vast majority of unencrypted end user traffic that originates from a given set of surveyed Google services. Some older devices cannot support modern encryption, standards, or protocols. Unfortunately, these devices may no longer support software updates and, as a result, may never support encryption"
Treatment of HTTP pages

Current (Chrome 67)  example.com

July 2018 (Chrome 68)  Not secure example.com
TLS certificate chains

- How many CAs are there?
  - Top-level CAs = ~60
  - Intermediate CAs = ~1200

- If any single CA is compromised, security of all websites on the internet could be compromised – yikes!
Hackers spied on 300,000 Iranians using fake Google certificate

Investigation reveals month-long, massive Gmail snooping campaign

By Gregg Keizer
Senior Reporter, Computerworld  |  SEP 6, 2011 5:43 AM PST

About 300,000 Iranians had their Gmail accounts compromised and their messages read by hackers, according to a forensics firm that has investigated the theft of hundreds of digital certificates from a Dutch company.

Although the report did not identify the hacker, or hackers, who may have spied on the Iranian users, security researchers have pointed to Iran’s government, which has been linked to other attempts to intercept the communications of activists and protesters.
Technology

Trustico revokes 23,000 SSL certificates due to compromise
Security

New hack on Comodo reseller exposes private data

And then there were four

By Dan Goodin 24 May 2011 at 19:58

Yet another official reseller of SSL certificate authority Comodo has suffered a security breach that allowed attackers to gain unauthorized access to data.

Brazil-based ComodoBR is at least the fourth Comodo partner to be compromised this year. In March, the servers of a separate registration authority were hacked by attackers who used their access to forge counterfeit certificates signed with Comodo's root signing key. Comodo admitted that two more of its resellers were hit in similar attacks, although no keys were issued.

Comodo has so far declined to name the resellers.

The SQL-injection attack on ComodoBR exploited vulnerabilities in the company's web applications that allowed the hackers to pass database commands to the website's backend server. The attackers posted two data files that appeared to show information related to certificate signing requests, in addition to email addresses, user IDs, and password information for a limited number of employees.
Comodo reseller hack

- The attackers registered fraudulent certificates for gmail.com, google.com, login.yahoo.com, login.skype.com, addons.mozilla.com, and login.live.com

- Quote from Comodo president and CEO
  - "So as a summary: its an SQL attack (fairly common) on a company in Brazil who sells some of our products." he wrote in an email. "Nothing to report really."
Distrust of Symantec TLS Certificates

A Certification Authority (CA) is an organization that browser vendors (like Mozilla) trust to issue certificates to websites. Last year, Mozilla published and discussed a set of issues with one of the oldest and largest CAs run by Symantec. The discussion resulted in the adoption of a consensus proposal to gradually remove trust in all Symantec TLS/SSL certificates from Firefox. The proposal includes a number of phases designed to minimize the impact of the change to Firefox users:

- January 2018 (Firefox 58): Notices in the Browser Console warn about Symantec certificates issued before 2016-06-01, to encourage site owners to replace their TLS certificates.
- May 2018 (Firefox 60): Websites will show an untrusted connection error if they use a TLS certificate issued before 2016-06-01 that chains up to a Symantec root certificate.
- October 2018 (Firefox 63): Distrust of Symantec root certificates for website server TLS authentication.
HTTPS attack: TLS Strip

- This attack is commonly known as "ssl strip"
- Most servers which support HTTPS implement an HTTP to HTTPS redirect
- When user omits protocol, the browser assumes http:// protocol
- What if the attacker intercepts the first unencrypted HTTP request?
  - Then they can man-in-the-middle all the traffic to rewrite the HTML to keep the user on the HTTP version of the site
Redirect HTTP to HTTPS
Client

Server
domain.example.com

118 Feross Aboukhadjiex
GET / HTTP/1.1

HTTP/1.1 301 Moved Permanently
Location: https://example.com
<!doctype html> Page has moved!
HTTP/1.1 301 Moved Permanently
Location: https://example.com
<!doctype html> Page has moved!

GET / HTTP/1.1

HTTP/1.1 200 OK
<!doctype html> good html
TLS Strip attack

Client

Active Attacker

Server

eexample.com

123 Feross Aboukhadjihe
HTTP/1.1 301 Moved Permanently
Location: https://example.com
<!doctype html> Page has moved!
GET / HTTP/1.1
128 Feross Aboukhadjiyeh
GET / HTTP/1.1

HTTP/1.1 301 Moved Permanently
Location: https://example.com
<!doctype html> Page has moved!

GET / HTTP/1.1

HTTP/1.1 200 OK
<!doctype html> good html
HTTP strict transport security (HSTS)

- To defend against the TLS Strip attack, the server tells the browser "no matter what protocol the user specifies, always use HTTPS"

- **Strict-Transport-Security: max-age=31536000**
  - Use HTTP header to force browser to use HTTPS for one year!

- **Downside:** "Trust on first use model" means that first visit to a site is still not secure against man-in-the-middle!

- Should clearing history also clear the HSTS list? Privacy vs. security
HSTS Preload list

- Browsers offer to hardcode sites which want to always be HTTPS only
- **Strict-Transport-Security**: `max-age=63072000; includeSubDomains; preload`
  - Must send `includeSubDomains` and `preload` options
- Difficult/impossible to remove a domain once hardocded into the browser itself
- Certain TLDs added the whole TLD to the preload list (e.g. `.dev`)
Lots more, but no time today!

- Public Key Pinning (HPKP)
- Certificate Transparency
- DNS Certification Authority Authorization (DNS CAA)
Credits:

Dan Boneh

https://transparencyreport.google.com/https/overview