Web security
HTTPS and the Lock Icon
Goals for this lecture

Brief overview of HTTPS:
  • How the SSL/TLS protocol works (very briefly)
  • How to use HTTPS

Integrating HTTPS into the browser
  • Lots of user interface problems to watch for
Threat Model: Network Attacker

Network Attacker:

• Controls network infrastructure: Routers, DNS
• Eavesdrops, injects, blocks, and modifies packets

Examples:

• Wireless network at Internet Café
• Internet access at hotels (untrusted ISP)
TLS overview: (1) DH key exchange

Anonymous key exchange secure against eavesdropping:

The Diffie-Hellman protocol in a group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$

Browser Alice

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

$$PMS = B^a$$

Server Bob

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

$$PMS = A^b$$

$$A = g^a \in G$$

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

$$PreMasterSecret = g^{ab} = (g^b)^a = B^a = (g^a)^b = A^b$$
How does Alice (browser) obtain $PK_{Bob}$?

Browser Alice

- $PK_{CA}$
- Verify cert

Server Bob

- Choose $(SK, PK)$
- $PK_{CA}$

CA

- $SK_{CA}$
- Check proof "I am Bob"
- Issue Cert with $SK_{CA}$:

Bob uses Cert for an extended period (e.g. one year)

Bob's key is PK
Sample certificate:

- **Organization**: Bank of America Corporation
- **Business Category**: Private Organization
- **Organizational Unit**: eComm Network Infrastructure
- **Serial Number**: 2927442
- **Common Name**: www.bankofamerica.com

**Public Key Info**
- **Algorithm**: RSA (1.2.840.113549.1.1.1)
- **Parameters**: None
- **Public Key**: 256 bytes: BE E5 23 1D 17 9A 68 05 ...
- **Exponent**: 65537
- **Key Size**: 2,048 bits
- **Key Usage**: Encrypt, Verify, Wrap, Derive

**Signature** (by CA)
256 bytes: 39 D0 09 7E 99 C6 B3 01 ...
Certificates on the web

Subject’s CommonName can be:

• An explicit name, e.g. \texttt{cs.stanford.edu}, or
• A wildcard cert, e.g. \texttt{*\.stanford.edu} or \texttt{cs*.stanford.edu}

matching rules:

“*” must occur in leftmost component, does not match “.”

example: \texttt{*\.a.com} matches \texttt{x.a.com} but not \texttt{y.x.a.com}

(as in RFC 2818: “HTTPS over TLS”)
Certificate Authorities

Browsers accept certificates from a large number of CAs

- Top level CAs ≈ 60
- Intermediate CAs ≈ 1200

<table>
<thead>
<tr>
<th>Certificate Authority</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entrust.net Certification Authority (2048)</td>
<td>Jul 24, 2029 7:15:12 AM</td>
</tr>
<tr>
<td>Entrust.net Specification Authority</td>
<td>May 25, 2019 9:39:40 AM</td>
</tr>
<tr>
<td>ePKI Root Certification Authority</td>
<td>Dec 19, 2034 6:31:27 PM</td>
</tr>
<tr>
<td>Equifax Secure Certificate Authority</td>
<td>Aug 22, 2018 9:41:51 AM</td>
</tr>
<tr>
<td>Equifax Secure eBusiness CA-1</td>
<td>Jun 20, 2020 9:00:00 PM</td>
</tr>
<tr>
<td>Equifax Secure eBusiness CA-2</td>
<td>Jun 23, 2019 5:14:45 AM</td>
</tr>
<tr>
<td>Equifax Secu...l eBusiness CA-1</td>
<td>Jun 20, 2020 9:00:00 PM</td>
</tr>
<tr>
<td>Federal Common Policy CA</td>
<td>Dec 1, 2030 8:45:27 AM</td>
</tr>
<tr>
<td>FNMT Clase 2 CA</td>
<td>Mar 18, 2019 8:26:19 AM</td>
</tr>
<tr>
<td>GeoTrust Global CA</td>
<td>May 20, 2022 9:00:00 PM</td>
</tr>
<tr>
<td>GeoTrust Pri...fication Authority</td>
<td>Jul 16, 2036 4:59:59 PM</td>
</tr>
<tr>
<td>Global Chambersign Root</td>
<td>Sep 30, 2037 9:14:18 AM</td>
</tr>
</tbody>
</table>

Dan Boneh
(3) TLS 1.3 session setup (simplified)

ClientHello: nonce_C, KeyShare

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

CertVerify: Enc[Sig_s(data)], Finished

Finished

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

Encrypted ApplicationData

Most common: server authentication only
(3) TLS 1.3 session setup: optimization (and caution)

ClientHello: nonce\textsubscript{C}, KeyShare, Enc[0-RTT data]

ServerHello: nonce\textsubscript{S}, KeyShare, [Enc[cert\textsubscript{S}],...]

CertVerify: Enc[\text{Sig}\textsubscript{S}(data)]

Finished

Client

Server

secret key

cert\textsubscript{S}

Data encrypted using a pre-shared key

**Caution**: 0-RTT data is vulnerable to replay

⇒ data should have no side effects

(i.e. GET but not POST)

Most common: server authentication only
Integrating TLS with HTTP: HTTPS

TCP handshake $\Rightarrow$ TLS handshake

HTTP/2 (2015)

HTTP
TLS
TCP
IP

HTTP/3 (2022)

TLS
QUIC
IP

integrated handshake (encrypting by default)
Integrating TLS with HTTP: HTTPS

A complication:

**Virtual hosting**: many sites hosted at same IP address

Solution since TLS 1.1: SNI (2003)

client_hello_extension: `server_name=cnn.com`

... but SNI defeats privacy benefit of encrypted cert in TLS 1.3.

Solution: **enc. client hello (ECH)** [encrypted with pk in server DNS]
HTTPS for all web traffic?

Old excuse:

• Crypto slows down web servers

⇒ no longer true (thanks to AES-NI)

Since July 2018: Chrome marks HTTP sites as insecure
HTTPS in the Browser
The TLS indicator

(HTTPS site)

Stanford University

← → ☑ stanford.edu

VS.

(HTTP site)

http.badssl.com

← → ☑ Not Secure http.badssl.com

Intended goal:

• Provide user with identity of page origin

• Indicate to user that page contents were not viewed or modified by a network attacker
When is the TLS icon displayed

All elements on the page fetched using HTTPS

For all elements:

• HTTPS cert issued by a CA trusted by browser
• HTTPS cert is valid (e.g. not expired)
• Domain in URL matches: **CommonName** or **SubjectAlternativeName** in cert
Positive security indicators are dangerous

The lock icon is a **positive security indicator**. Problem: picture-in-picture attacks.

Trained users are more likely to fall victim to this  [JSTB’07]
HTTPS and login pages: incorrect usage

Suppose user lands on HTTP login page.

• say, by typing HTTP URL into address bar

View source:

<form method="post" action="https://onlineservices.wachovia.com/..." (old site)
HTTPS and login pages: guidelines

General guideline:

Response to http://login.site.com should be Location: https://login.site.com (redirect)

Should be the response to every HTTP request ...
Problems with HTTPS and the Lock Icon
Problems with HTTPS and the Lock Icon

1. Upgrade from HTTP to HTTPS
2. Forged certs
3. Mixed content: HTTP and HTTPS on the same page
4. Does HTTPS hide web traffic?
   - Problems: traffic analysis, compression attacks
1. HTTP ⇒ HTTPS upgrade

Suppose user does:

- connect to bank site over HTTP; bank redirects to HTTPS

SSL_strip attack: prevent the upgrade [Moxie’08]

```
<form action=https://... >  ⟷  <form action=http://...>
Location: https://...        ⟷        Location: http://...  (redirect)
```

```
<a href=https://...>        ⟷        <a href=http://...>
```

Dan Boneh
UI design flaw in old browsers: location of fav icon

$\Rightarrow$ fav icon no longer presented in address bar

Number of users who detected HTTP downgrade: 0
Defense: Strict Transport Security (HSTS)

Strict-Transport-Security: max-age=63072000; includeSubDomains
(ignored if not over HTTPS)

Header tells browser to always connect over HTTPS

Subsequent visits must be over HTTPS (self signed certs result in an error)
- Browser refuses to connect over HTTP or if site presents an invalid cert
- Requires that entire site be served over valid HTTPS

HSTS flag deleted when user “clears private data”: security vs. privacy
Preloaded HSTS list

https://hstspreload.org/

Enter a domain for the HSTS preload list:

- paypal.com

Strict-Transport-Security: max-age=63072000; includeSubDomains; preload

Preload list hard-coded in Chrome source code. Examples:
  Google, Paypal, Twitter, Simple, Linode, Stripe, Lastpass, ...
CSP: upgrade-insecure-requests

The problem: many pages use `<img src="http://site.com/img">`
• Makes it difficult to migrate a section of a site to HTTPS

**Solution:** gradual transition using CSP

**Content-Security-Policy: upgrade-insecure-requests**

```html
<img src="http://site.com/img">
<img src="http://othersite.com/img">
<a href="http://site.com/img">
<a href="http://othersite.com/img">
```
```
<img src="https://site.com/img">
<img src="https://othersite.com/img">
<a href="https://site.com/img">
<a href="https://othersite.com/img">
```
2. Certificates: wrong issuance

2011: Comodo and DigiNotar CAs hacked, issue certs for Gmail, Yahoo! Mail, ...

2013: TurkTrust issued cert. for gmail.com (discovered by pinning)

2014: Indian NIC (intermediate CA trusted by the root CA IndiaCCA) issue certs for Google and Yahoo! domains

  Result: (1) India CCA revoked NIC’s intermediate certificate
           (2) Chrome restricts India CCA root to only seven Indian domains

2016: WoSign (Chinese CA) issues cert for GitHub domain (among other issues)

  Result: WoSign certs no longer trusted by Chrome and Firefox

⇒ enables eavesdropping w/o a warning on user’s session
Man in the middle attack using rogue cert

GET https://bank.com

ClientHello \rightarrow \text{attacker} \rightarrow \text{bank}

ServerCert \,(\text{rogue})

\text{(cert for Bank by a valid CA)}

\text{TLS key exchange} \quad k_1 \quad k_1

HTTP data enc with \( k_1 \)

\text{TLS key exchange} \quad k_2 \quad k_2

HTTP data enc with \( k_2 \)

Attacker proxies data between user and bank.
Sees all traffic and can modify data at will.
What to do?  (many good ideas)

1. Public-key pinning (static pins)
   - Hardcode list of allowed CAs for certain sites (Gmail, facebook, ...)
   - Browser rejects certs issued by a CA not on list

2. Certificate Transparency (CT): [LL’12]
   - idea: CA’s must advertise a log of all certs. they issued
   - Browser will only use a cert if it is published on (two) log servers
     • Server attaches to certificate a signed statement from log (SCT)
     • Companies can scan logs to look for invalid issuance
CT requirements

April 30, 2018: CT required by chrome

• Required for all certificates with a path to a trusted root CA
  (not required for an installed root CA)
• Otherwise: HTTPS errors

Cert for crypto.stanford.edu published on five logs:
  cloudflare_nimbus2018
  google_argon2018, google_aviator
  google_pilot, google_rocketeer
3. Mixed Content: HTTP and HTTPS

Page loads over HTTPS, but contains content over HTTP

(e.g. \(<script\ src="http://.../script.js">\)

⇒ Active network attacker can hijack session by modifying script en-route to browser

IE7: 

Old Chrome:

https://www.google.com/calendar/

 Mostly ignored by users ...
https://badssl.com  (Chrome 124, 2024)

Mixed script:  `<script src="http://mixed-script.badssl.com/nonsecure.js"></script>`

(setting mixed-script.badssl.com) (no visible warning)

script is not loaded!  developer tools show an error.


(setting mixed-form.badssl.com)

Warning if user tries to submit data

The information you’re about to submit is not secure

Because this form is being submitted using a connection that’s not secure, your information will be visible to others.
4. Peeking through TLS: traffic analysis

• Network traffic reveals length of HTTPS packets
  – TLS supports up to 256 bytes of padding

• Some sites interact frequently with the web server
  – These interactions expose specific internal state of the page

Chen, Wang, Wang, Zhang, 2010
Peeking through TLS: an example [CWWZ’10]

Vulnerabilities in an online tax application
No easy fix. Can also be used to ID Tor traffic
THE END