Web security

HTTPS and the Lock Icon

Acknowledgments: Lecture slides are from the Computer Security course taught by Dan Boneh at Stanford University. When slides are obtained from other sources, a reference will be noted on the bottom of that slide. A full list of references is provided on the last slide.
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

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

\[
PMS = A^b
\]

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

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

\[
\text{PreMasterSecret} = g^{ab} = (g^b)^a = B^a = (g^a)^b = A^b
\]
(2) Certificates

How does Alice (browser) obtain $PK_{Bob}$?

Browser Alice

- $PK_{CA}$
- Verify cert

Server Bob

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

CA

- check proof
- $SK_{CA}$

Bob’s key is PK

Bob uses Cert for an extended period (e.g. one year)
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 Encryption (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.  cs.stanford.edu  , or
• A wildcard cert, e.g.  *.stanford.edu  or  cs*.stanford.edu

matching rules:

“*” must occur in leftmost component, does not match “.”
example:  *.a.com  matches  x.a.com  but not  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
TLS 1.3 session setup (simplified)

Client Hello: nonce\textsubscript{C}, KeyShare

Server Hello: nonce\textsubscript{S}, KeyShare, Enc[\text{cert}\textsubscript{S},...]

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

session-keys $\leftarrow$ HKDF(DHkey, nonce\textsubscript{C}, nonce\textsubscript{S})

Encrypted ApplicationData

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

ClientHello: nonce$_C$, KeyShare, $\text{Enc}[0\text{-RTT data}]$

ServerHello: nonce$_S$, KeyShare, $\text{Enc}[\text{cert}_S,...]$

CertVerify: $\text{Enc}[\text{Sig}_S(\text{data})]$ is verified

Client

Server

secret key

cert$_S$

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

Two complications

Web proxies
solution: browser sends CONNECT domain-name before client-hello

Virtual hosting: many sites hosted at same IP address
solution in TLS 1.1: SNI (June 2003)
client_hello_extension: server_name=cnn.com

SNI defeats privacy benefit of encrypted cert in TLS 1.3.
Solution: encrypted SNI, encrypted with pk in server DNS
HTTPS for all web traffic?

Old excuses:

- Crypto slows down web servers  (not true anymore)
- Some ad-networks still do not support HTTPS
  – reduced revenue for publishers

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

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 (basic) lock 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
The lock UI:  Extended Validation Certs

Harder to obtain than regular certs
  • requires human at CA to approve cert request
  • no wildcard certs  (e.g. *.stanford.edu )

Helps block “semantic attacks”:  www.bankofthevvest.com

This UI is ineffective:  removed from Chrome in 2019.
A general UI attack: picture-in-picture

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:

```html
<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

Common use pattern:
• browse site over HTTP; move to HTTPS for checkout
• connect to bank over HTTP; move to HTTPS for login

SSL_strip attack: prevent the upgrade [Moxie’08]

![Diagram showing HTTP to SSL upgrade]

- `<a href=https://...>` → `<a href=http://...>`
- `Location: https://...` → `Location: http://...` (redirect)
- `<form action=https://... >` → `<form action=http://...>`
Tricks and Details

Tricks: drop-in a clever fav icon (older browsers)

⇒ fav icon no longer presented in address bar

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

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

Check status and eligibility

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
```

```
<img src="http://site.com/img">
<img src="http://othersite.com/img">
<a href="http://site.com/img">
<a href="http://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

ServerCert (rogue)

attacker

ClientHello

ServerCert (Bank)

Bank

Badguy

SSL key exchange

k₁

HTTP data enc with k₁

SSL key exchange

k₂

HTTP data enc with k₂

(cert for Bank by a valid CA)

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
   - Now deprecated (because often incorrectly used in practice)

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 a signed statement from log (SCT) to certificate
     • 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:

 Mostly ignored by users ...
Mixed script:  

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

Mixed form:  

```html
<form action="http://http.badssl.com/resources/submit.html">
```

Form loaded, but no HTTPS indicator
4. Peeking through SSL: traffic analysis

- Network traffic reveals length of HTTPS packets
  - TLS supports up to 256 bytes of padding
- AJAX-rich pages have lots and lots of interactions with the server
- These interactions expose specific internal state of the page

BAM!

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

Vulnerabilities in an online tax application

No easy fix. Can also be used to ID Tor traffic
THE END