



Web Security

PART III: HTTPS and the Lock

Dr. Bheemarjuna Reddy Tamma

IIT HYDERABAD

Note: This is a revised version of slide deck of Prof. Dan Boneh (Stanford) with material from various Internet sources

Outline

Integrating HTTPS into the browser

- Has lots of user interface problems ...
- Attacks and defense mechanisms
- Web Security Guidelines

HTTPS: HTTP over TLS

- HTTPS: RFC2818
- Use https:// URL rather than http://
 - and port 443 rather than 80
- Connection initiation
 - TCP handshake, TLS handshake & then HTTP request(s)
- Encrypts
 - URL, document contents, form data, cookies, HTTP headers
- Connection close
 - have “***Connection: close***” in HTTP record
 - TLS level exchange ***close_notify*** alerts
 - can then close TCP connection using **FIN & ACK** messages

Other Problems with SSL/TLS

- Browser provides feedback to user about whether HTTPS is in use, but many users don't pay attention☹
- If a certificate is bad/unknown, browser issues warning dialogs
 - certificate_expired
 - certificate_revoked
 - bad_certificate
 - unknown_ca
 - bad_record_mac



This site is not secure

This might mean that someone's trying to trick you or steal any information that you send to the server. You should close this site immediately.

 [Go to your Start page](#)

Details

The hostname in the website's security certificate differs from the website you are trying to visit.

Error Code:

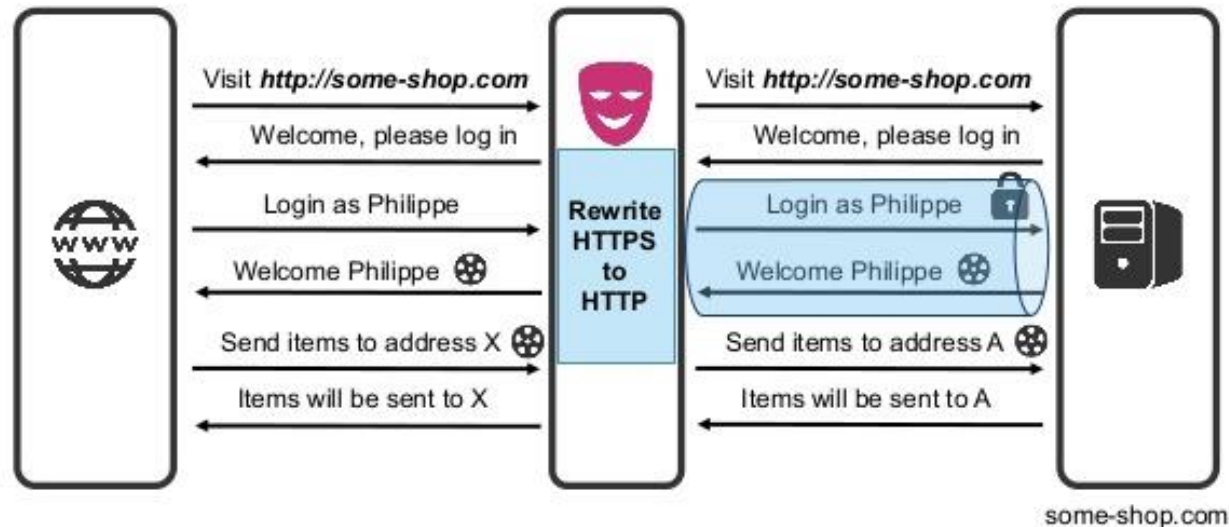
DLG_FLAGS_SEC_CERT_CN_INVALID

[Go on to the webpage](#) (Not recommended)

Other Problems with SSL/TLS

- Legacy use pattern: user browses site with HTTP, upgrades to HTTPS for checkout or login
- "Just in time" HTTPS:
 - Login page displayed with HTTP, but Form posted with HTTPS hopefully!
 - Appears secure but it may not:
 - MITM attacker can corrupt HTTP login page during transit from Web Server to Client
 - SSL stripping during form post: nothing indicates that the actual connection didn't use SSL
 - Send uid/password somewhere else during form post 😞

Stripping HTTPS from Login Forms



Solution: before returning HTML for login page, browser checks for HTTPS; or if page fetched via HTTP, redirect to the HTTPS version

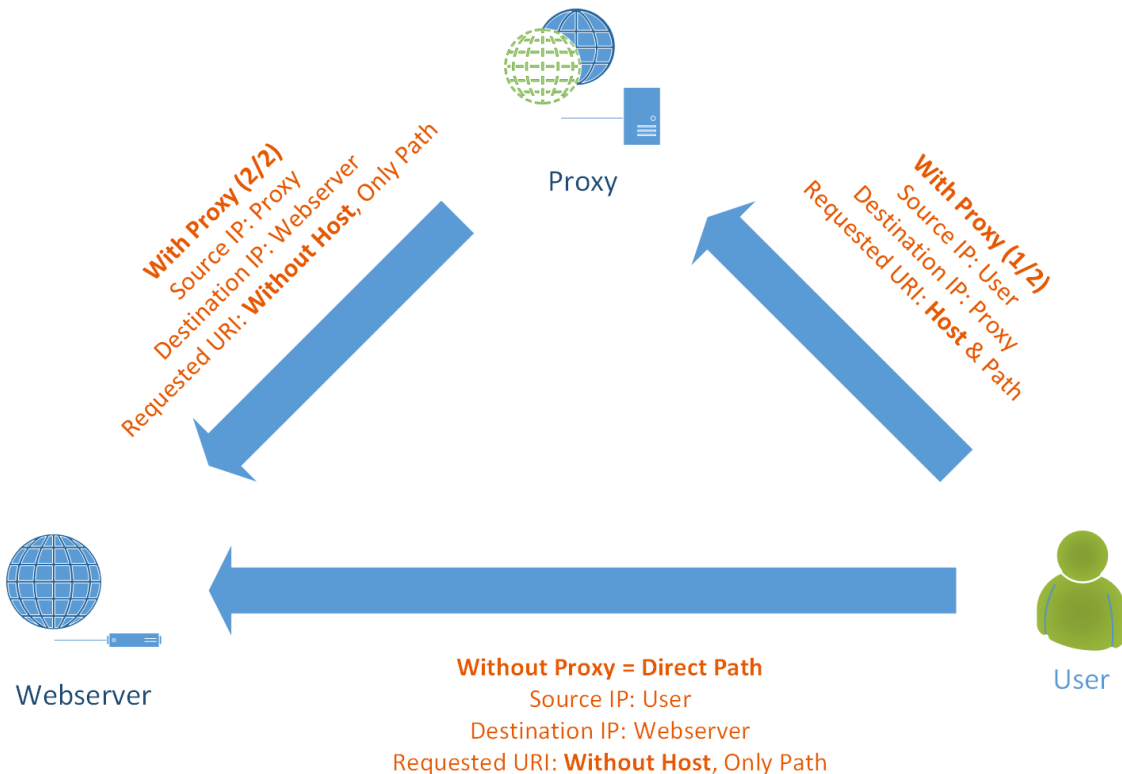
Integrating TLS with HTTP: HTTPS

Two complications

Web proxies/GWs: MITM

I. Pass-through (bypass) web proxy for HTTPS traffic

- Caches resources to reduce bandwidth and achieve speed up & hides your IP address while browsing

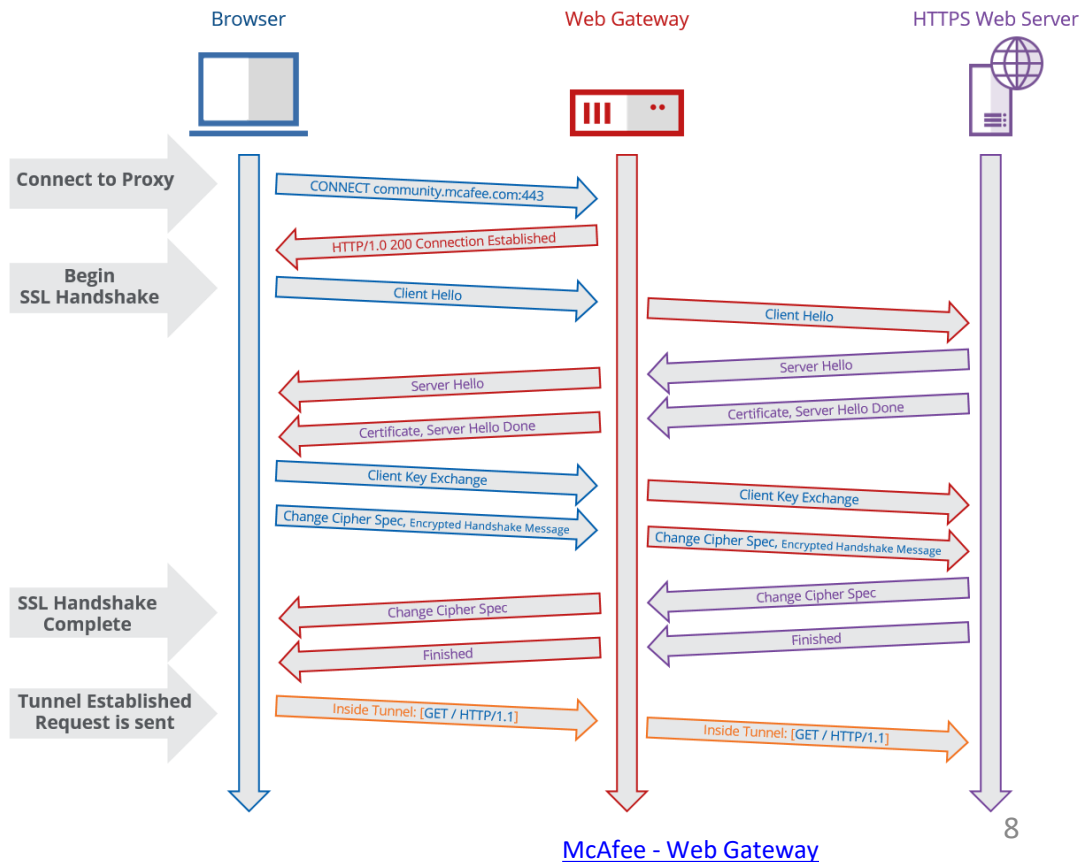


Integrating SSL/TLS with HTTP: HTTPS

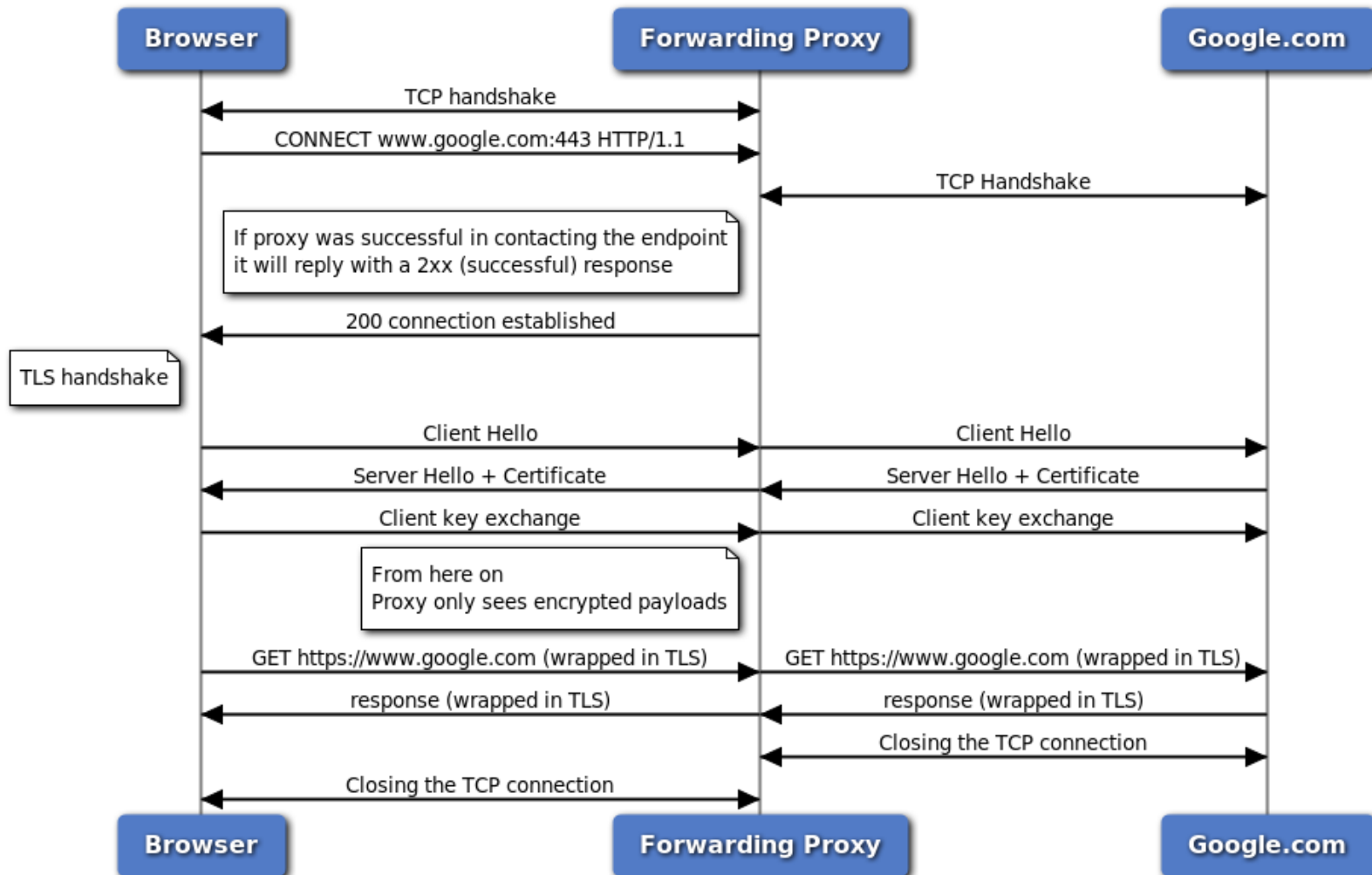
Two complications

Web proxies/GWs: MITM

- Pass-through (bypass or forwarding) web proxy for HTTPS traffic
 - HTTP **CONNECT** method before TLS client_hello to seek Proxy to create a TLS tunnel between Client and Server



GET https://www.google.com with a forwarding proxy

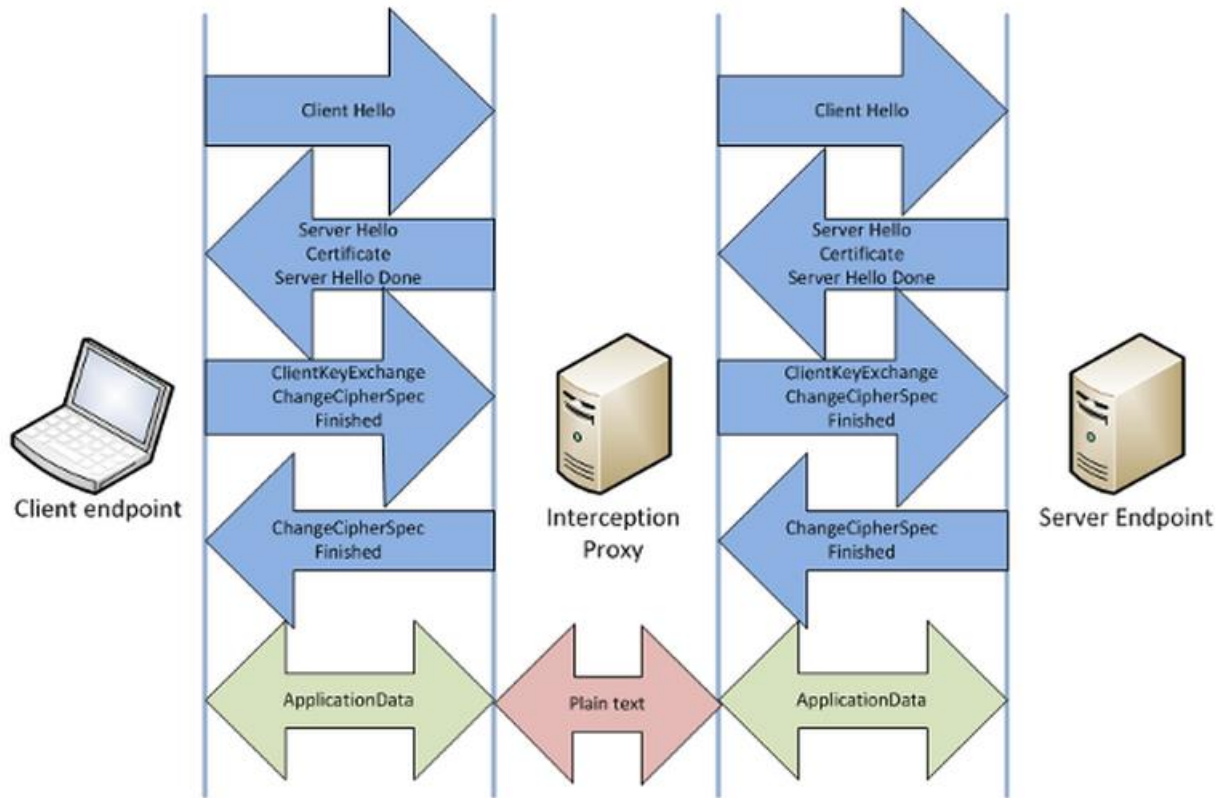


Integrating SSL/TLS with HTTP: HTTPS

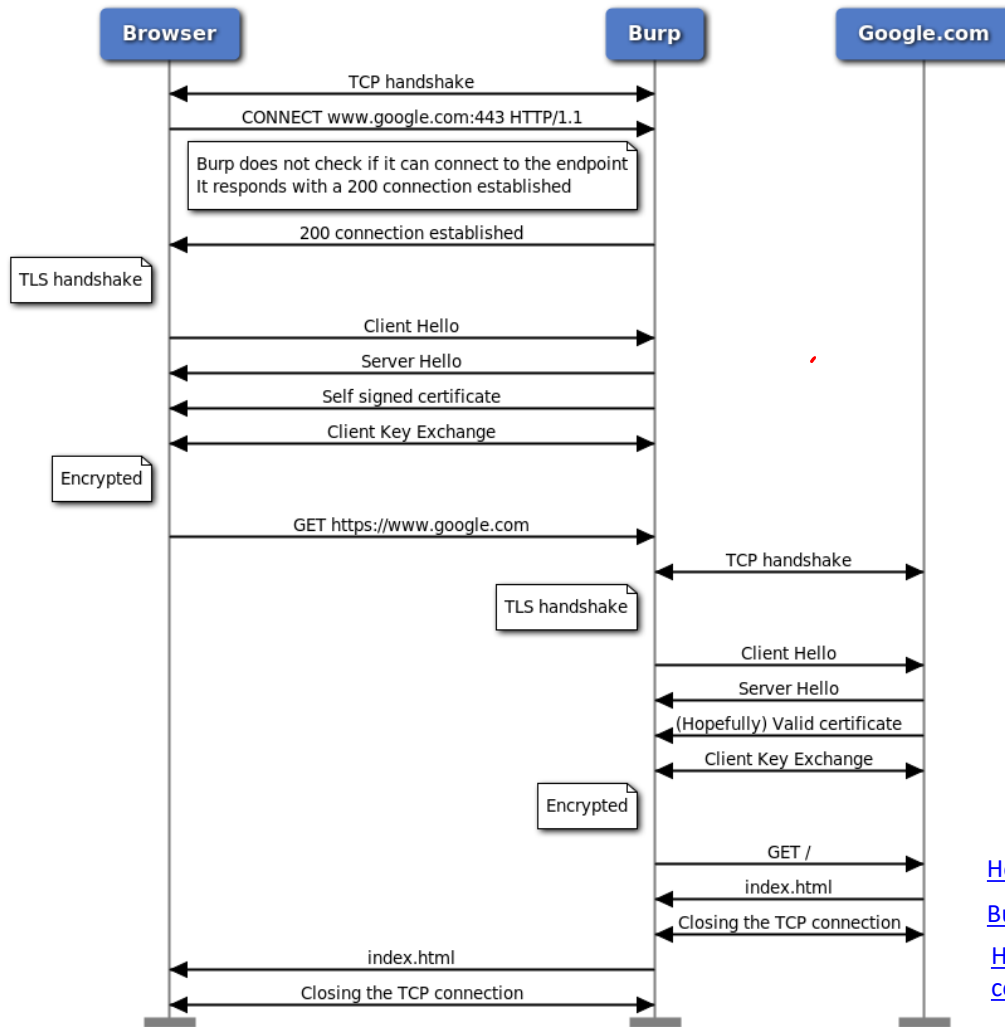
Two complications

Web proxies/GWs: MITM

- Bypass proxy for HTTPS
- Intercept HTTPS traffic
 - Squid
 - Cert of Proxy need to be stored in Trusted Root Certificate Authorities store of Client's browser



GET https://www.google.com using Burp



[How HTTP\(s\) Proxies Work \(parsiya.net\)](http://parsiya.net)

[Burp Suite Community Edition - PortSwigger](#)

[HTTPS interception dilemma: Pros and cons - Help Net Security](#)

Integrating SSL/TLS with HTTP: HTTPS

Two complications

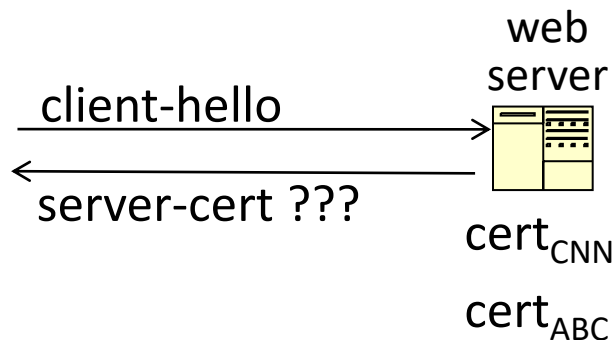
Virtual hosting:

Two sites hosted at same IP address

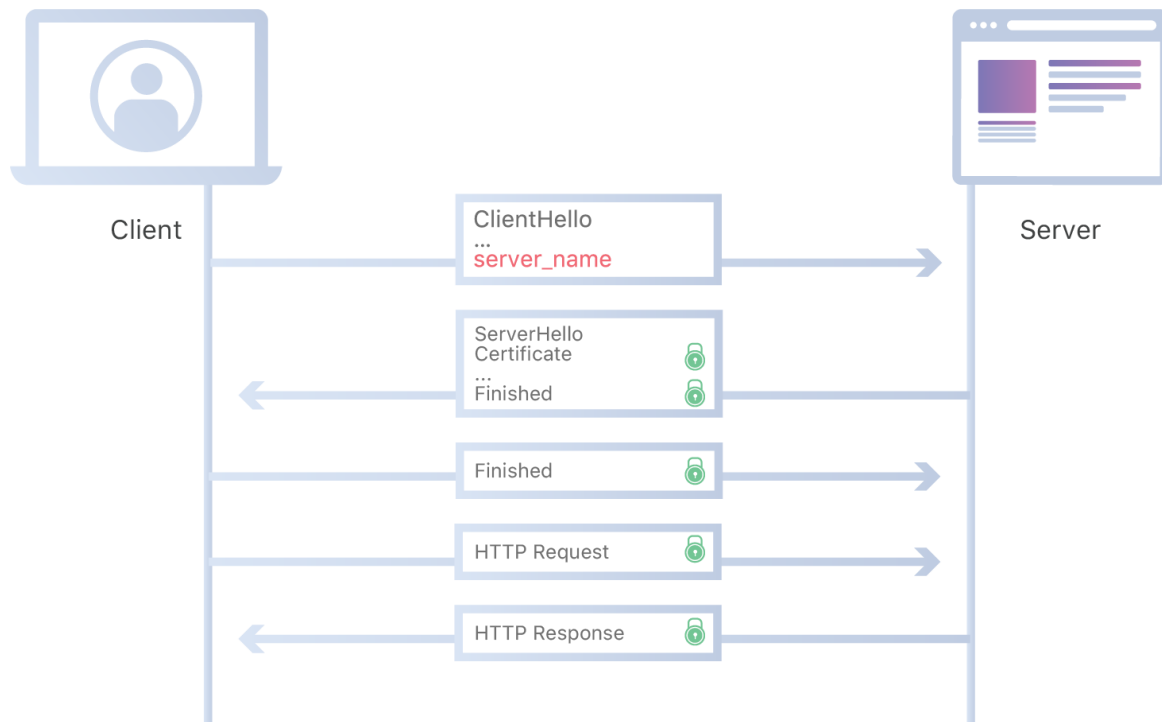
Solution in TLS 1.1: Server Name Indication (SNI)

client_hello_extension: server_name=cnn.com

Implemented since FF2 and IE7 (vista)

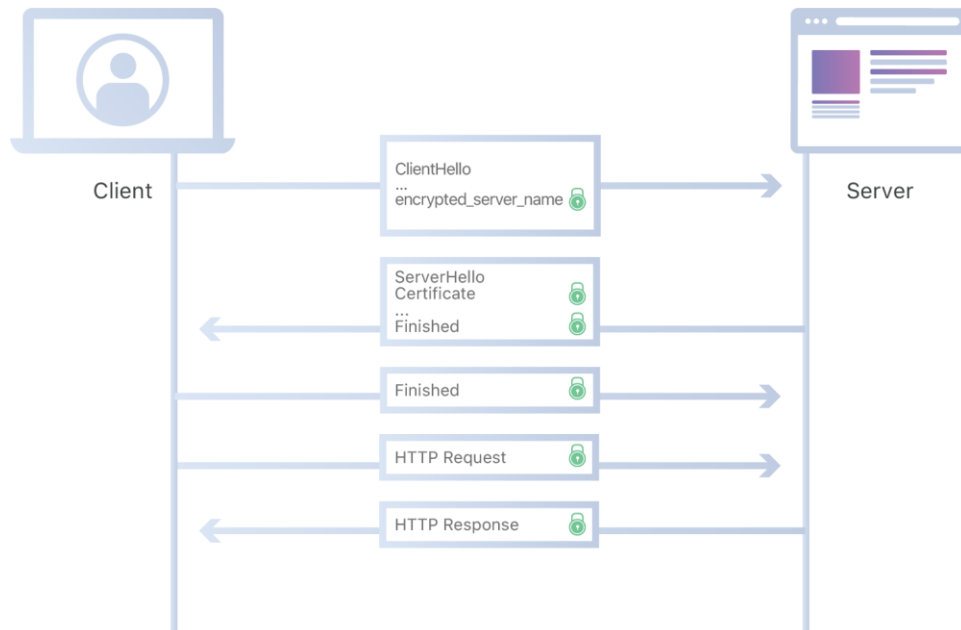


TLS with Unencrypted SNI



ISPs and firewalls track which sites a user is visiting! **CLOUDFLARE**

TLS 1.3 offers encrypted SNI



But how come the original SNI couldn't be encrypted before, but now it can in TLS 1.3?

Encrypted SNI Extension

- Where does the encryption key come from if client and web server have n't negotiated one yet?
 - DNS: Web server publishes a public key of DHE from its end on a well-known DNS record
 - Client making DNS query gets public key as well & generates a shared encryption key for encrypting SNI in client_hello
 - Web server also generates the same encryption key after receiving public key of DHE from client side
 - Note: These DHE paras are different from ones used for key exchange
- Simply using DNS (which is, by default, unencrypted) would make Encrypted SNI extension worthless
 - Solution:
 - DNS over TLS or DNS over HTTPS
 - Plus DNSSEC (offers signed DNS responses)

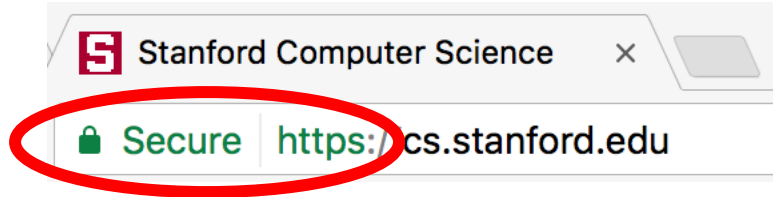
Why is HTTPS not used for all web traffic?

- Crypto slows down web servers (but not by much if done right)
- Some ad-networks still do not support HTTPS
 - Reduced revenue for publishers
- Incompatible with virtual hosting (older browsers)
 - March 2017: IE6 \approx 1-5% in China (ie6countdown.com)

Aug 2014: Google boosted ranking of sites supporting HTTPS!

HTTPS in the Browser

The lock icon: TLS indicator



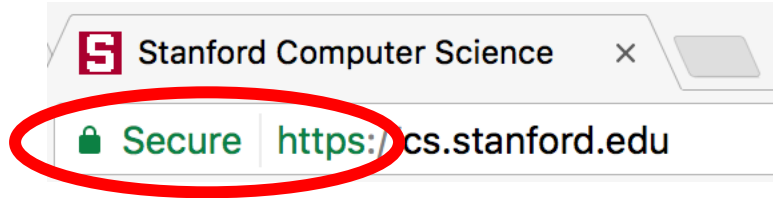
Intended goal:

- Provide user with identity of page origin
- Indicate to user that page contents were not viewed or modified by a **network attacker**



In reality: many problems (next few slides)

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

Extension	Subject Alternative Name (2.5.29.17)
Critical	NO
DNS Name	*.google.com
DNS Name	*.android.com
DNS Name	*.appengine.google.com
DNS Name	*.cloud.google.com
DNS Name	*.google-analytics.com
DNS Name	*.google.ca
DNS Name	*.google.cl
DNS Name	*.google.co.in
DNS Name	*.google.co.jp
DNS Name	*.google.co.uk
DNS Name	*.google.com.ar
DNS Name	*.google.com.au

A general UI attack

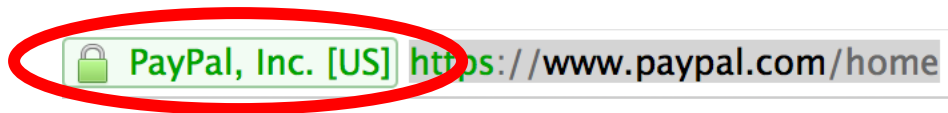


The lock UI: Extended Validation Certs

Harder to obtain than regular certs

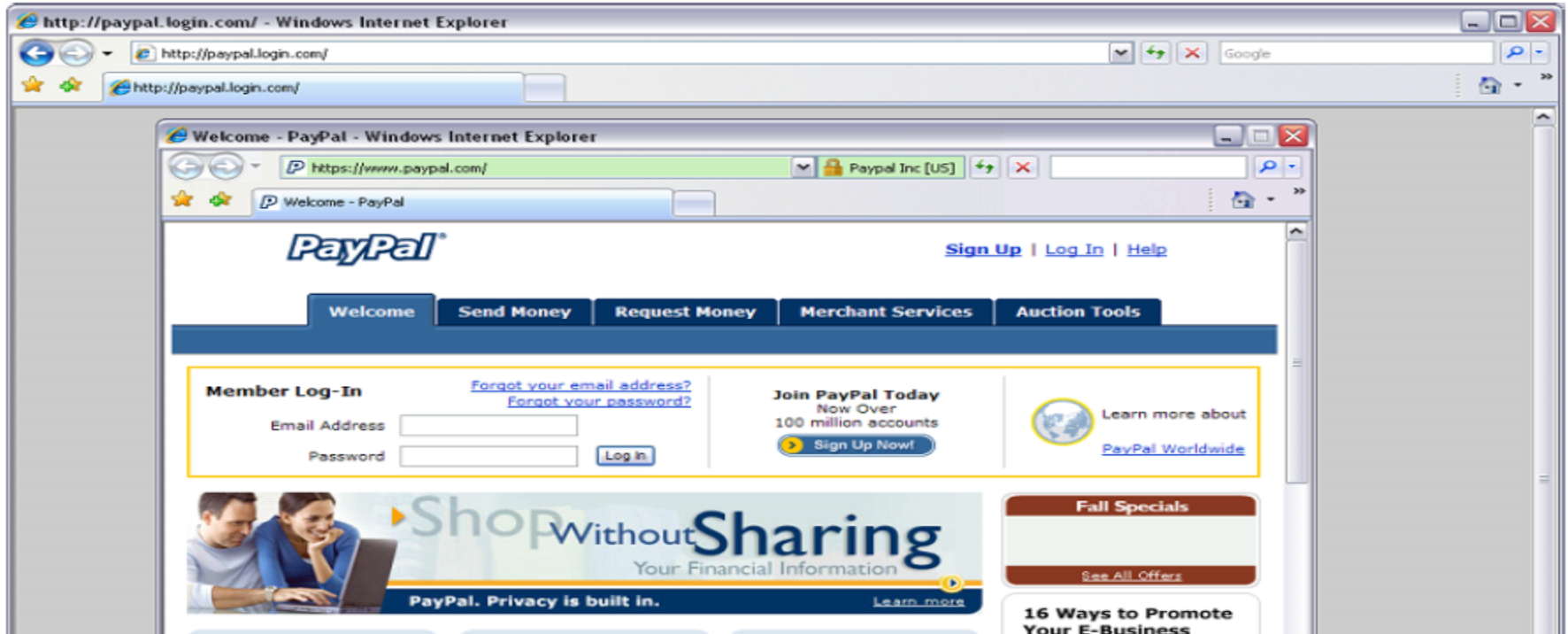
- requires human at CA to approve cert request
- no wildcard certs (e.g. *.stanford.edu)

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



note: HTTPS-EV and HTTPS are in the same origin

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

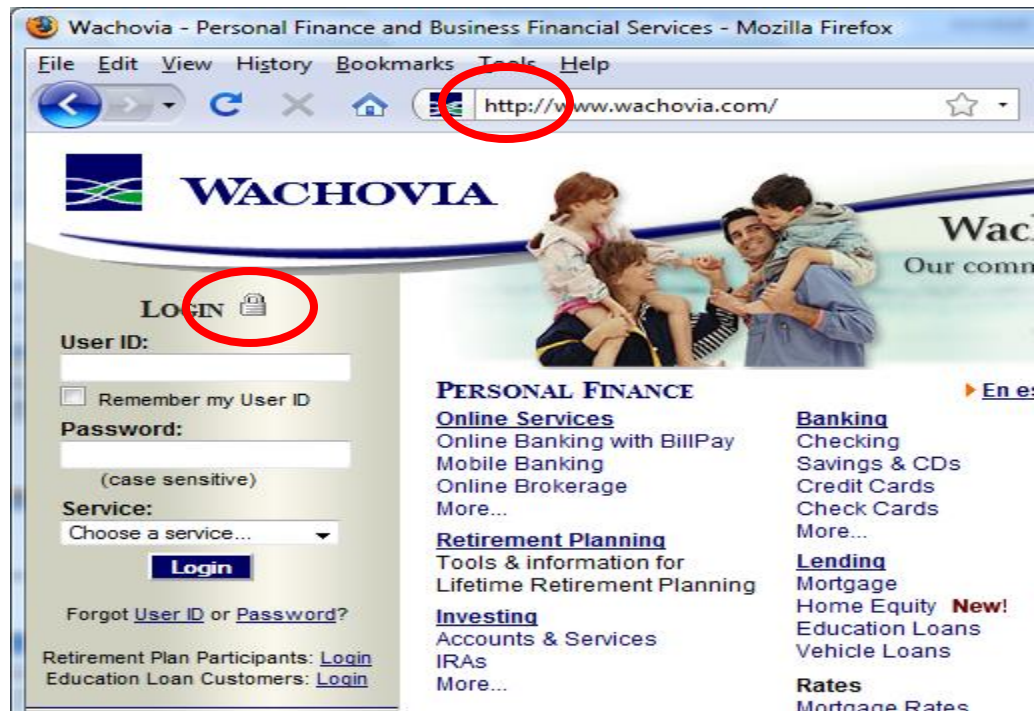
Users often land on login page over HTTP:

- Type HTTP URL into address bar
- Google links to HTTP page

View source:

```
<form method="post"
```

```
action="https://onlineservices.wachovia.com/..."
```



(old site)

HTTPS and login pages: guidelines

General guideline:

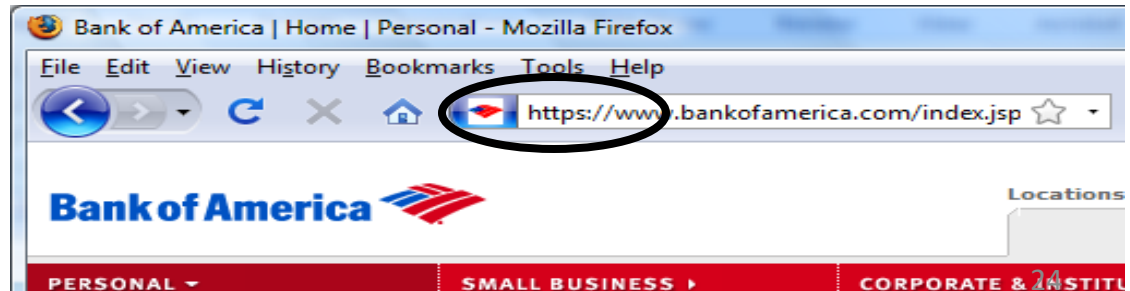
Response to

<http://login.site.com>

should be Location: <https://login.site.com>
(redirect)

Should be the response
to every HTTP request ...

→ Automatic HTTPS Rewrites



Problems with HTTPS and Defenses

Problems with HTTPS and Defenses

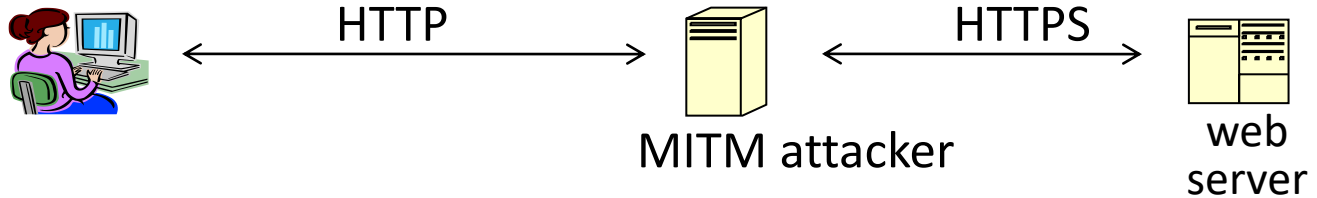
1. Stripping of upgrade from HTTP to HTTPS
2. Mixed content: HTTP and HTTPS on the same page
3. Forged certs
4. Does HTTPS hide web traffic?

1. HTTP \Rightarrow HTTPS upgrade

Legacy use pattern:

- browse site over HTTP; move/upgrade to HTTPS for checkout
- connect to bank over HTTP; upgrade to HTTPS for login

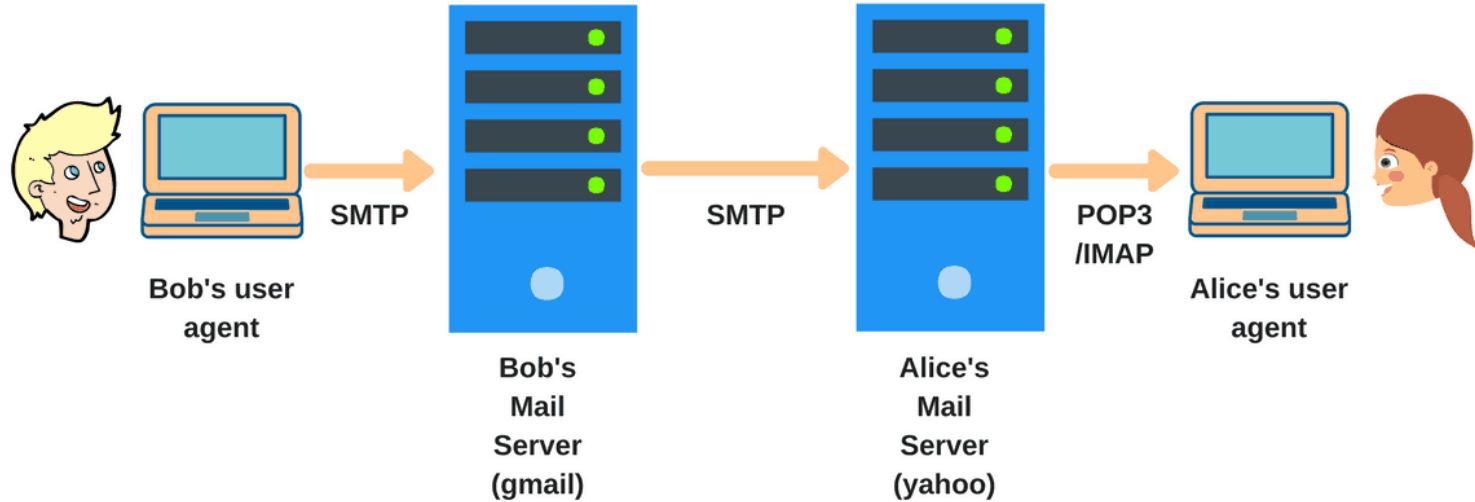
SSL_strip attack: prevents the upgrade [Moxie'08]



<code></code>	\rightarrow	<code></code> before fwding to client
Location: <code>https://...</code>	\rightarrow	Location: <code>http://...</code>
<code><form action=https://... ></code>	\rightarrow	<code><form action=http://...></code>

- Similar attack on SMTP STARTTLS by [ISP Cricket in USA in 2014](#)

Email System



SMTP Transport Example

```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.org>
C: To: "Alice Example" <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 Jan 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
{The server closes the connection}
```

SMTP Extension: STARTTLS

S: <waits for connection on TCP port 25>

C: <opens connection>

S: 220 mail.imc.org SMTP service ready

C: EHLO mail.example.com

S: 250-mail.imc.org offers a warm hug of welcome

S: 250-8BITMIME

S: 250-STARTTLS

S: 250 DSN

C: STARTTLS

S: 220 Go ahead

C: <starts TLS negotiation>

C & S: <negotiate a TLS session>

C & S: <check result of negotiation>

C: EHLO mail.example.com

S: 250-mail.imc.org touches your hand gently!

S: 250-8BITMIME

S: 250 DSN

- Opportunistic TLS
 - Upgrades the existing session on port 25 or 587
 - Effective only against passive MITM attacks
 - No end-to-end encryption (E2EE)
 - Silent fail-back to plaintext mode
- Alternative solution: Implicit TLS
 - Still no E2EE
 - Pretty Good Privacy (PGP) or Secure/Multipurpose Internet Mail Extensions (S/MIME) offer E2EE
 - digitally sign and encrypt msgs

Tricks and Details

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



⇒ fav icon no longer presented in address bar



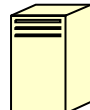
- Always-on SSL: Add 301 redirects to HTTPS URLs by Rewriting HTTP (port 80) requests to HTTPS (port 443) on your webserver

Defense: HTTP Strict Transport Security ([HSTS](#))



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

(ignored if not over HTTPS)



web
server

Header tells browser to connect over HTTPS to the server for max-age

So, subsequent visits must be over HTTPS

- Browser refuses to connect over HTTP if site presents an invalid cert
- Requires that entire site and its subdomains be served over valid HTTPS

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

Refer: <chrome://net-internals/#hsts> and <https://www.chromium.org/hsts>

Preloaded HSTS list in Browsers

Helps in enforcing HTTPS **all the time**.

<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**

HSTS is supported in Chrome, Firefox, Safari, Edge, Opere, etc.

Examples : Gmail, Paypal, Twitter, Simple, Linode, Stripe, Lastpass, ... *(but not google.com!)*

<https://cloud.google.com/docs/security/encryption-in-transit>

2. Mixed Content: HTTP and HTTPS

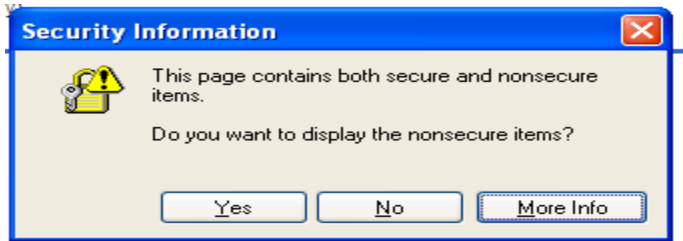
Page loads over HTTPS, but contains content over HTTP

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

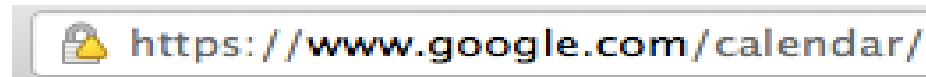
 never write this

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

IE7:









Old Chrome:

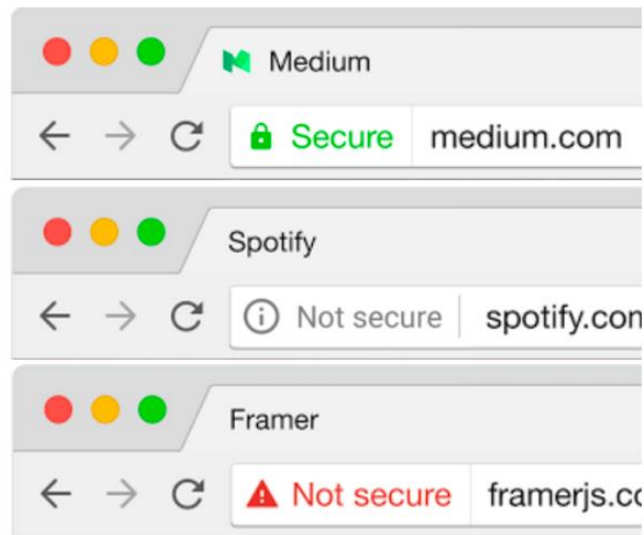


Mostly ignored by users ...

2. Mixed Content: HTTP and HTTPS

	Chrome 51	Chrome 52
Secure HTTPS	 https://www.google.com	 https://www.google.com
HTTP	 www.example.com	 www.example.com
Broken HTTPS	 https://expired.badssl.com	 https://expired.badssl.com

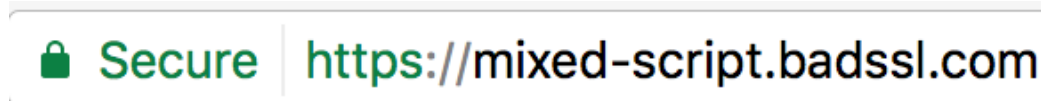
ⓘ HTTPS with minor errors uses the same indicator as HTTP.



https://badssl.com

(Chrome 58, 2017)

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



(script is blocked, click to load)

Mixed image: ``

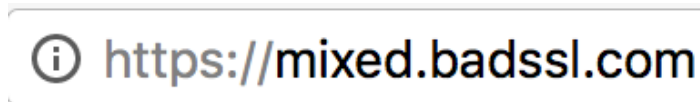


Image loaded, but no HTTPS indicator

CSP: upgrade-insecure-requests

The problem: many pages use ``

Cross Site Scripting ([XSS](#)) and data injection attacks

Solution: gradual transition using [CSP](#)

Content-Security-Policy directive in HTML Response: upgrade-insecure-requests

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



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

Always use protocol relative URLs `` & `script-src 'self'` ³⁷

3. Certificates: wrong issuance

2011: **Comodo** and **DigiNotar** CAs hacked, issued fake certs for Gmail, Yahoo!

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

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

Result: (1) India CCA revoked NIC's intermediate certificate

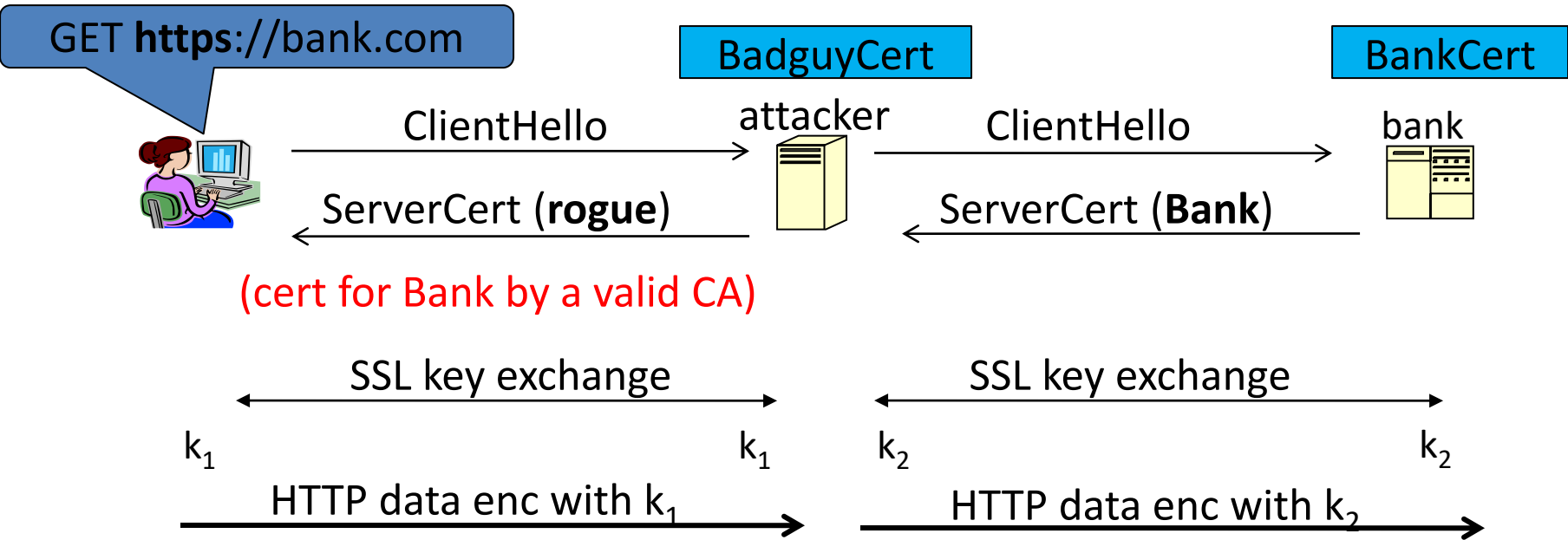
(2) Chrome restricts India CCA root to only seven Indian domains

2015: **MCS** (intermediate CA cert issued by **CNNIC**) issued certs for Google domains

Result: current CNNIC root no longer recognized by Chrome

⇒ leads to passive & active MITM attacks w/o a warning on user's session

Man in the middle attack using rogue cert



Attacker hacks CA and creates a rogue cert of Bank to launch MITM attacks.
Attacker proxies data between user and bank.
Sees all traffic and can modify data at will ☹️

What to do??

(many good ideas)

1. Dynamic HTTP public-key pinning (HPKP) (RFC 7469)
 - Let a site declare CA that can sign its cert (similar to HSTS)
 - Web server tells a client which public key belongs to it
 - TOFU: Trust on First Use
 - On subsequent HTTPS, client rejects certs issued by other CAs

2. DNS CA Authorization (CAA) Records

Certificate Authority Authorization Record

- Web server adds CAA record in its name server
- CAs will query this record before cert issuance

thesslstore.com. CAA 0 issue "digicert.com"

thesslstore.com. CAA 0 issuewild "sectigo.com"

3. Certificate Transparency: [LL'12] (RFC 6962)

- Idea: CA's must advertise a log of all certs they issued
- Browser will only use a cert if it is published on log server
 - Efficient implementation using Merkle hash trees
- Companies can scan logs to look for invalid issuance

HPKP example (HTTP header from webserver)

```
Public-Key-Pins[-Report-only]: max-age=2592000;  
    pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";  
    pin-sha256="LPJNul+wow4m6DsqqxbninhsWHlwfp0JecwQzYpOLmCQ=";  
    report-uri="https://example.net/pkp-report"
```

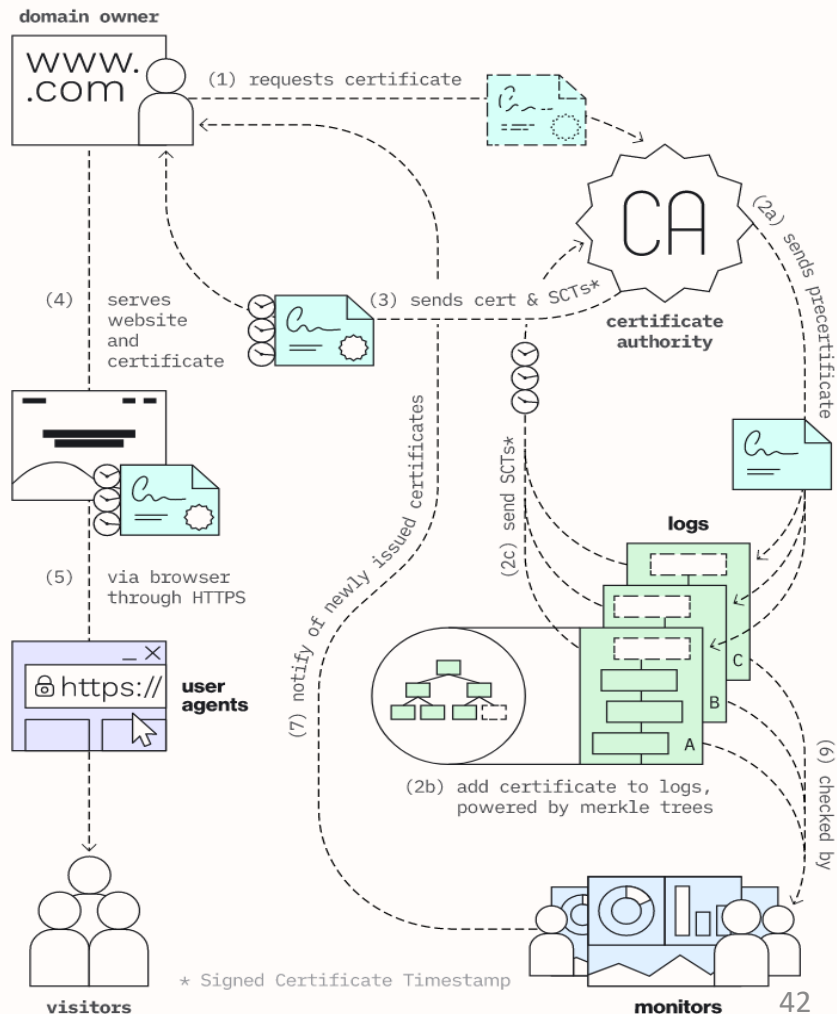
Max-age: 2,592,000 seconds is the most common max-age value used (30 days)

Examine browser's pinning DB: <chrome://net-internals/#hsts>

HPKP can be potentially very dangerous if config is done incorrectly

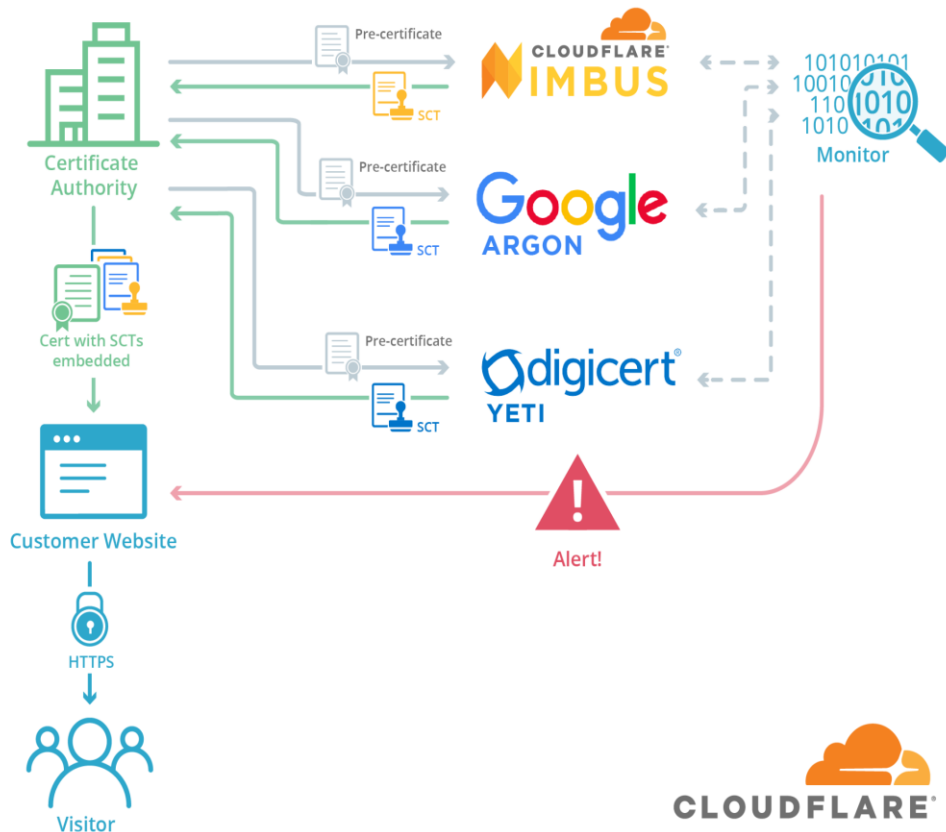
Certificate Transparency

- An ecosystem that makes the issuance of Certs transparent and verifiable
- Certificates are deposited in public, transparent log servers by CAs
 - Logs are **tamper-proof**, **append-only** and **publicly-auditable ledgers** of certificates being created/updated/expired
- CA gets Signed Certificate Timestamp (SCT) from the log server(s) and may embed into Cert issued
 - SCT: a promise to add Cert to the log within the Maximum Merge Delay (MMD)

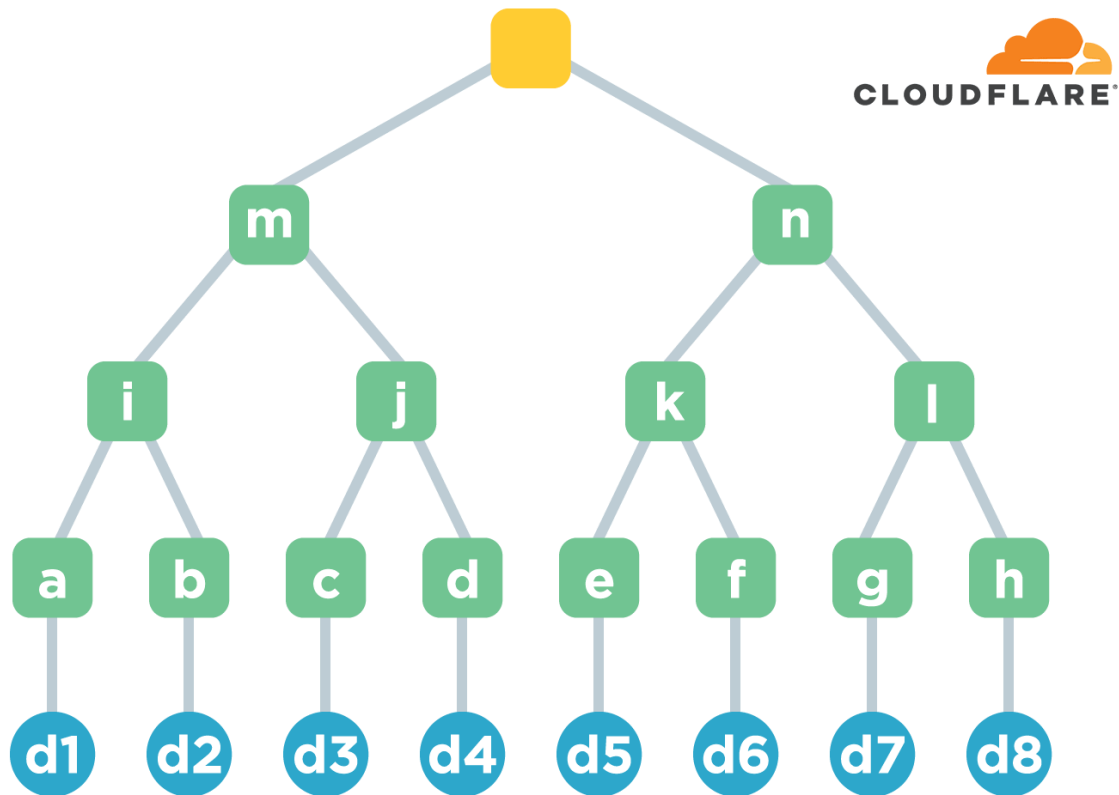


Certificate Transparency

- Browsers can check for SCTs during TLS handshake and trust only the Certs logged in one or more vetted log servers
- Logs can be cryptographically monitored by anyone (Monitors)
 - Merkle Tree Hash to prevent tampering
 - It also offers nice balance to the log operators and auditors in terms of cost to add certs and validate their consistency
- CT only protects users if all certificates are logged!



Certificate Transparency: Merkle Hash Tree



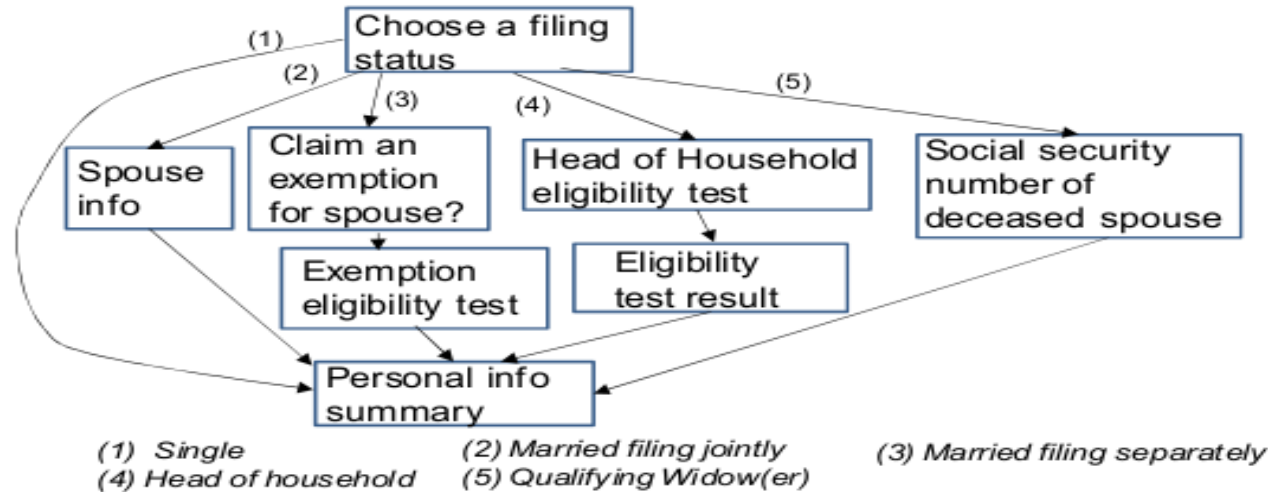
4. Peeking through SSL: traffic analysis

- 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



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 peek into Tor traffic

Online tools to test SSL/TLS security

- [Qualys SSL Labs](#)
- <https://www.digicert.com/help/>

Web Security Guidelines

- https://infosec.mozilla.org/guidelines/web_security
- [SSL and TLS Deployment Best Practices · sslabs](#)
- <https://web.dev/secure/>

References

- <https://badssl.com/dashboard/>
- <https://ct.cloudflare.com/>
- [SNI: Virtual Hosting for HTTPS - SSL.com](#)
- [How HTTP\(s\) Proxies Work \(parsiya.net\)](#)
- [Burp Suite Community Edition - PortSwigger](#)
- [HTTPS interception dilemma: Pros and cons - Help Net Security](#)
- <http://blog.scphillips.com/posts/2017/04/intercept-https/>
- [McAfee Support Community - Web Gateway: Understanding "Client Context"](#)
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>

References

- [BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf](#)
- [https://developer.mozilla.org/en-US/docs/Web/HTTP/Public_Key_Pinning](#)
- [chrome://net-internals/#hsts](#)
- [https://www.html5rocks.com/en/tutorials/security/content-security-policy/](#)
- [DNS Checker - DNS Check Propagation Tool](#)
- [What Is a CAA Record? Your Guide to Certificate Authority Authorization - Hashed Out by The SSL Store™](#)
- [https://spectrum.ieee.org/tech-talk/telecom/security/we-know-what-youre-watching-even-if-its-encrypted](#)
- [https://w3c.github.io/webappsec-upgrade-insecure-requests/](#)
- [https://w3c.github.io/webappsec-upgrade-insecure-requests/](#)
- [https://caniuse.com/?cats=Security&statuses=all](#)