Web Security

PART III: HTTPS and the Lock

Dr. Bheemarjuna Reddy Tamma

IIT HYDERABAD

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

Outline

Integrating HTTPS into the browser

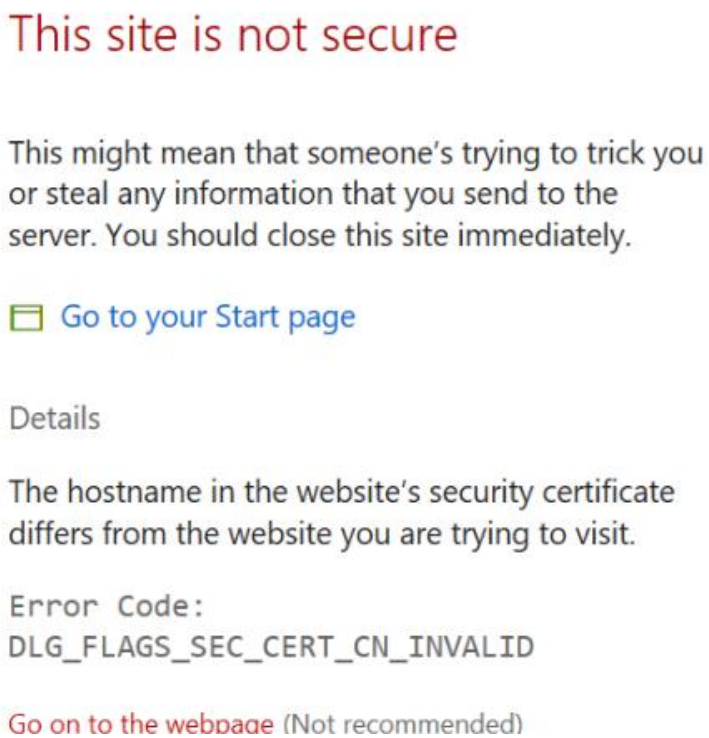
- Has lots of user interface problems …
- Attacks and defense mechanisms
- Web Security Guidelines

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

Other Problems with SSL/TLS

- Browser provides feedback to user about whether HTTPS is in use, but most 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

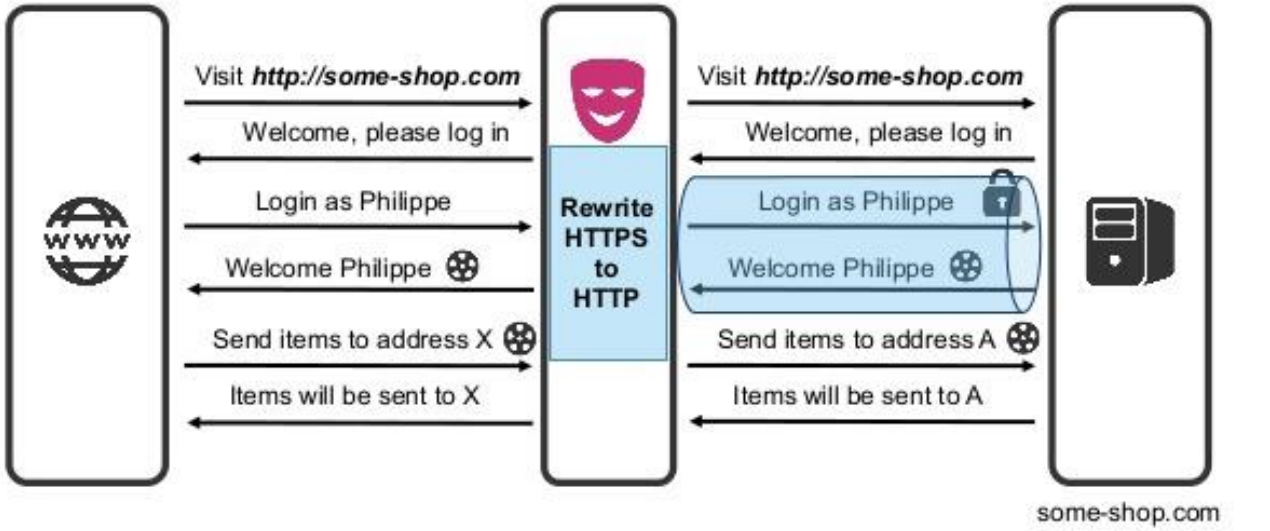


https://badssl.com/dashboard/

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’s 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, check for HTTPS; if page fetched via HTTP, redirect to the HTTPS version

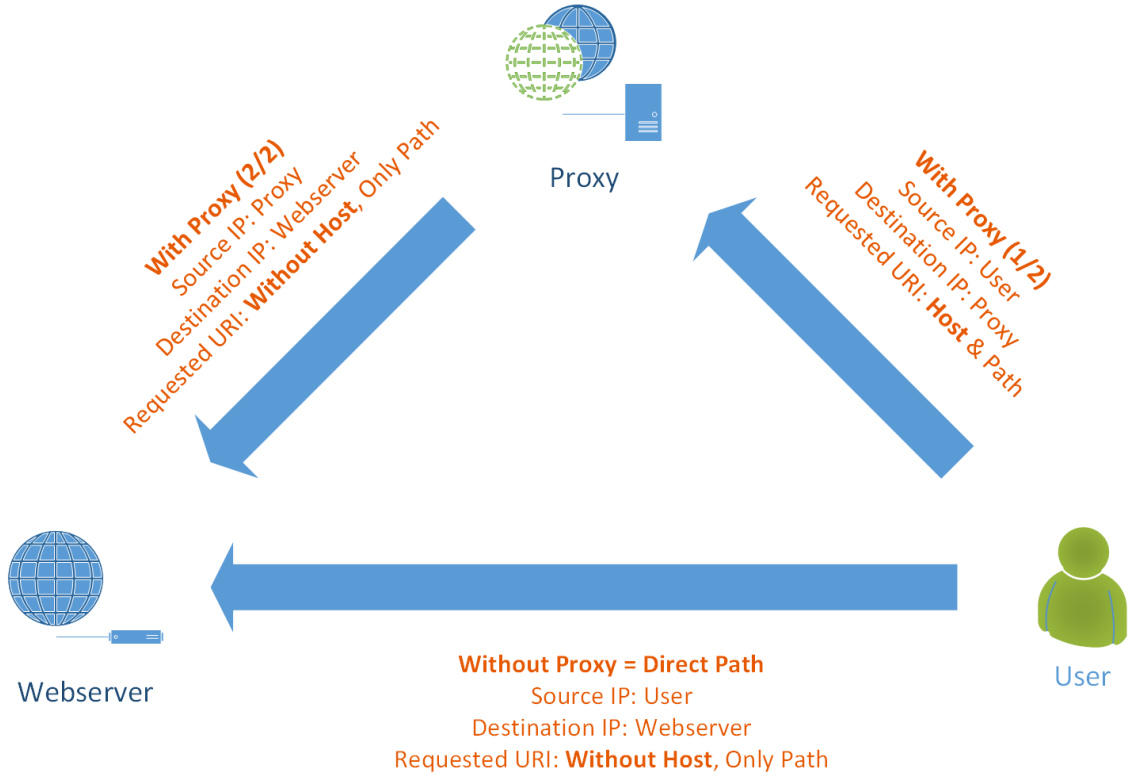
Integrating TLS with HTTP:HTTPS

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

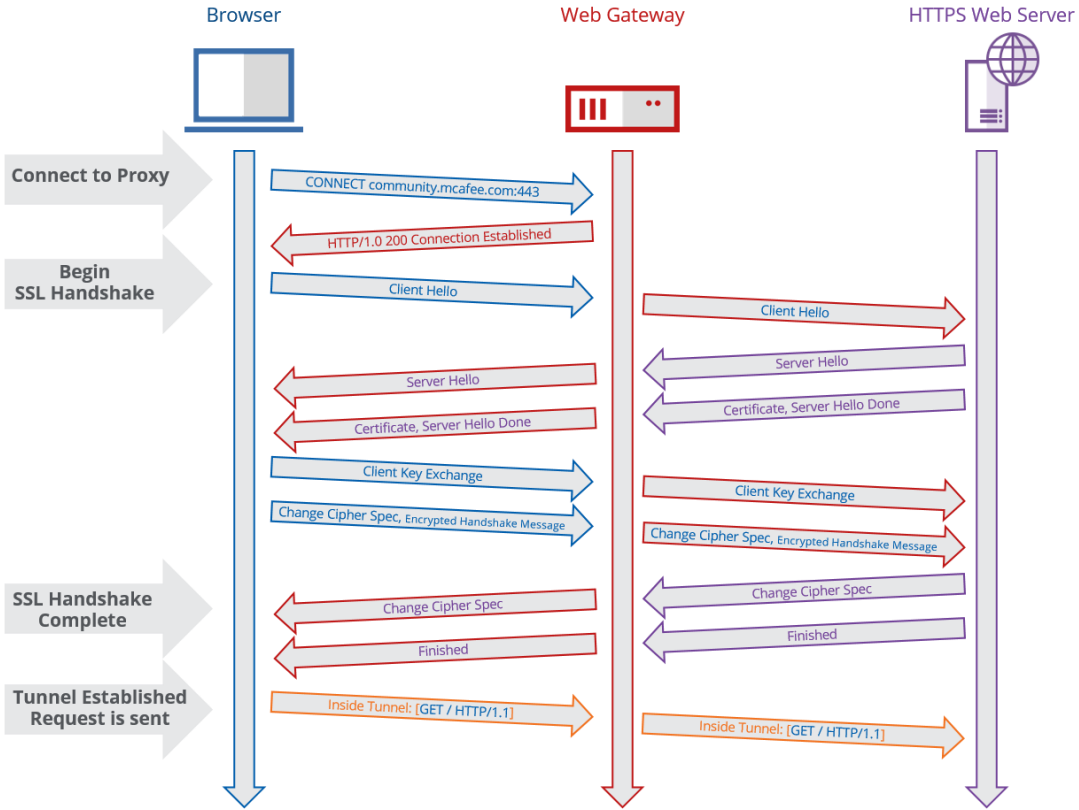


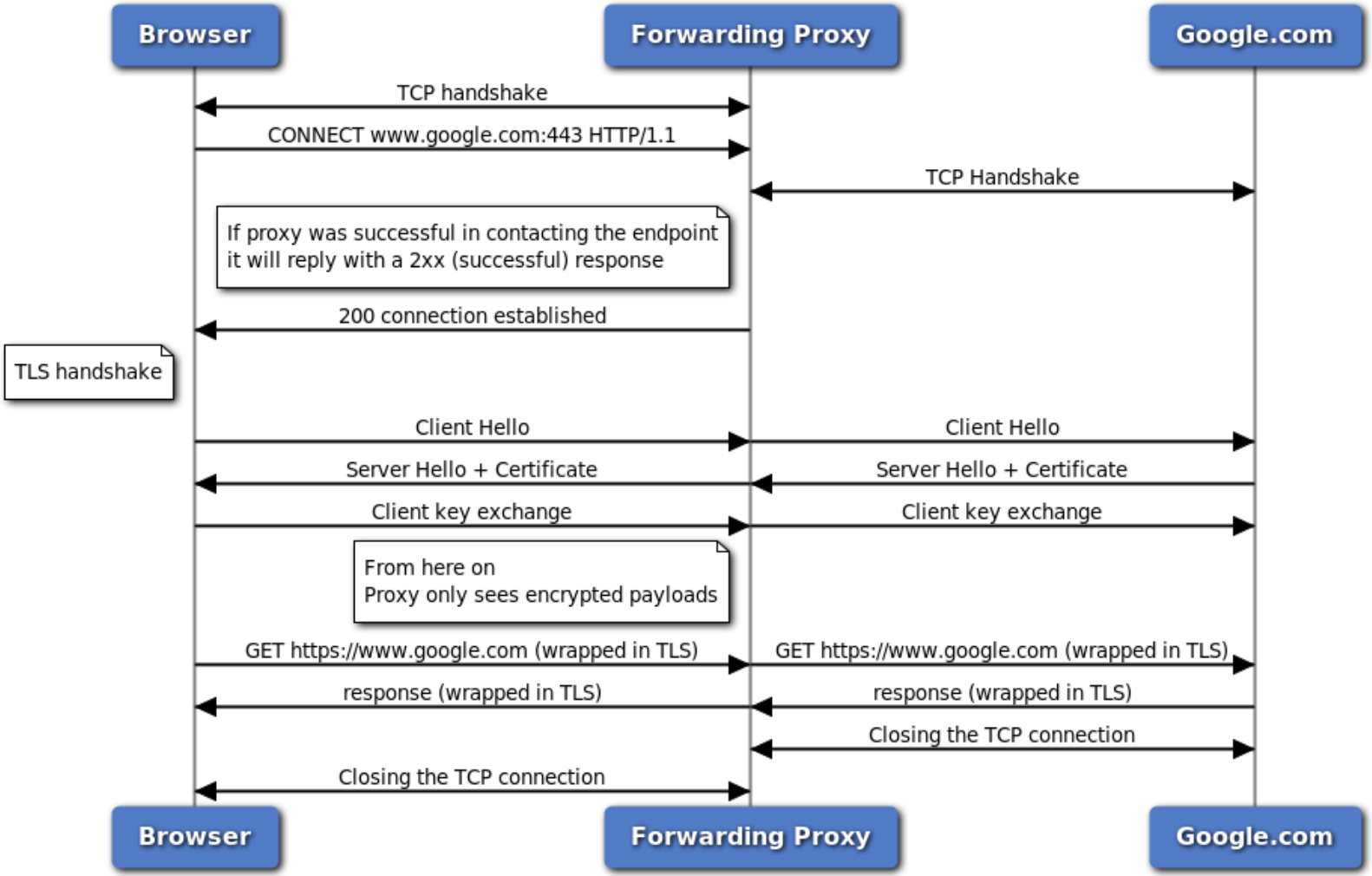
Integrating SSL/TLS with HTTP:HTTPS

Two complications

Web proxies/GWs: MITM

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



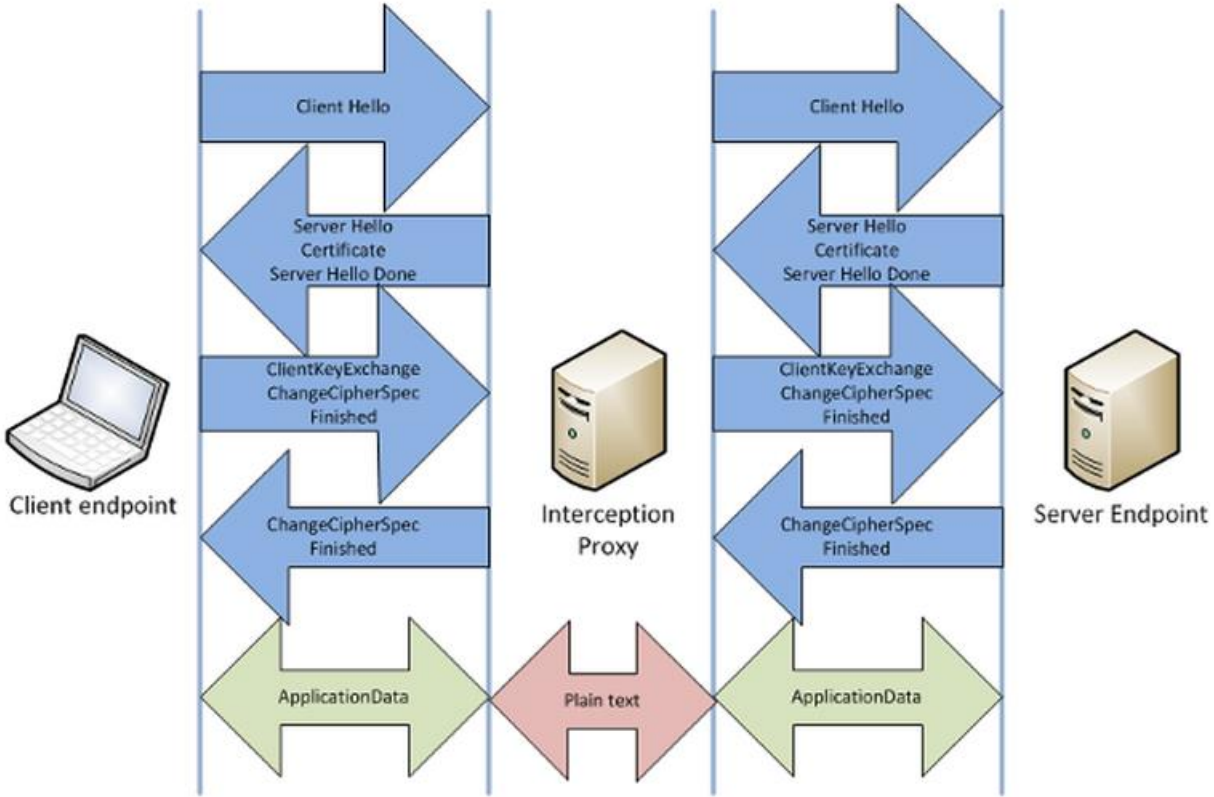

GET https://www.google.com with a forwarding proxy
Browser
Forwarding Proxy
Google.com
TCP handshake
CONNECT www.google.com:443 HTTP/1.1
TCP Handshake
If proxy was successful in contacting the endpoint
it will reply with a 2xx (successful) response
200 connection established
TLS handshake
Client Hello
Client Hello
Server Hello + Certificate
Server Hello + Certificate
Client key exchange
Client key exchange
From here on
Proxy only sees encrypted payloads
GET https://www.google.com (wrapped in TLS)
GET https://www.google.com (wrapped in TLS)
response (wrapped in TLS)
response (wrapped in TLS)
Closing the TCP connection
Closing the TCP connection
Browser
Forwarding Proxy
Google.com

Integrating SSL/TLS with HTTP:HTTPS

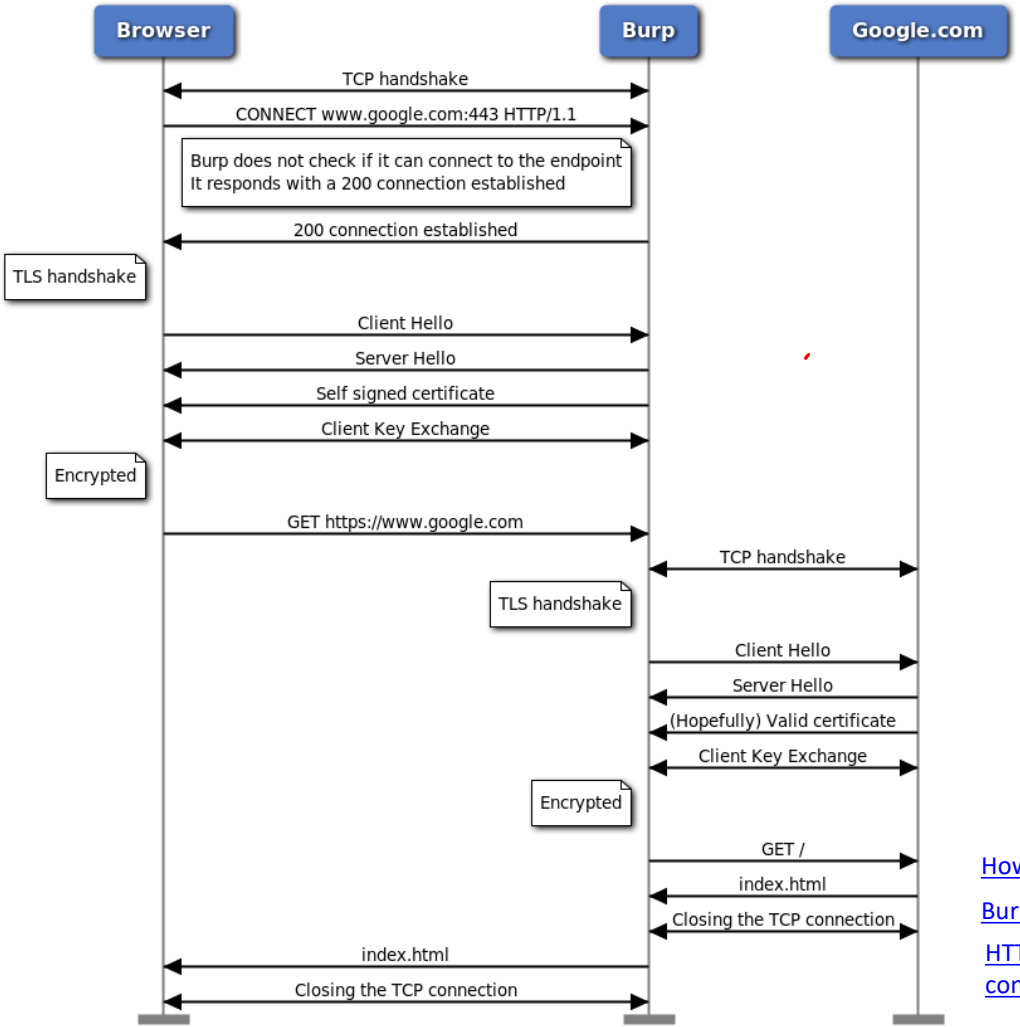
Two complications

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



http://blog.scphillips.com/posts/2017/04/intercept-https/



[How HTTP(s) Proxies Work (parsiya.net)]

[Burp Suite Community Edition - PortSwigger]

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

Integrating SSL/TLS with HTTP:HTTPS

Two complications

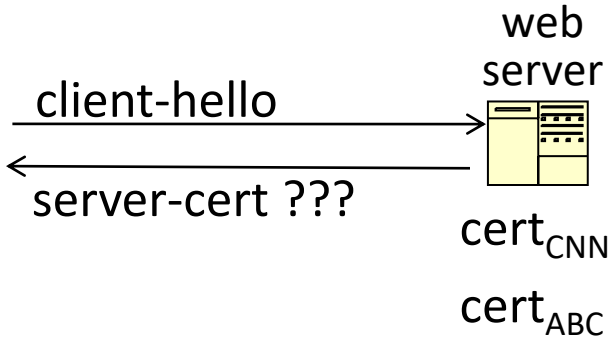
Virtual hosting:

Two sites hosted at same IP address

Solution in TLS 1.1: Server Name Indication (SNI)

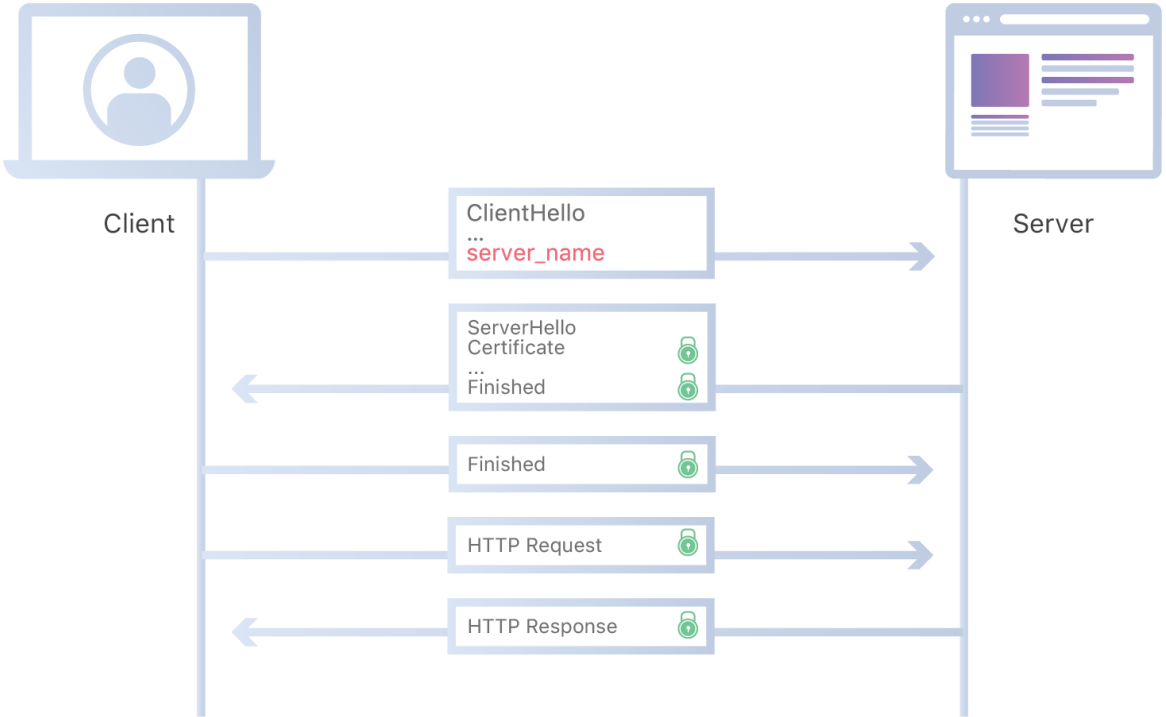
client_hello_extension: server_name=cnn.com

Implemented since FF2 and IE7 (vista)



Reference

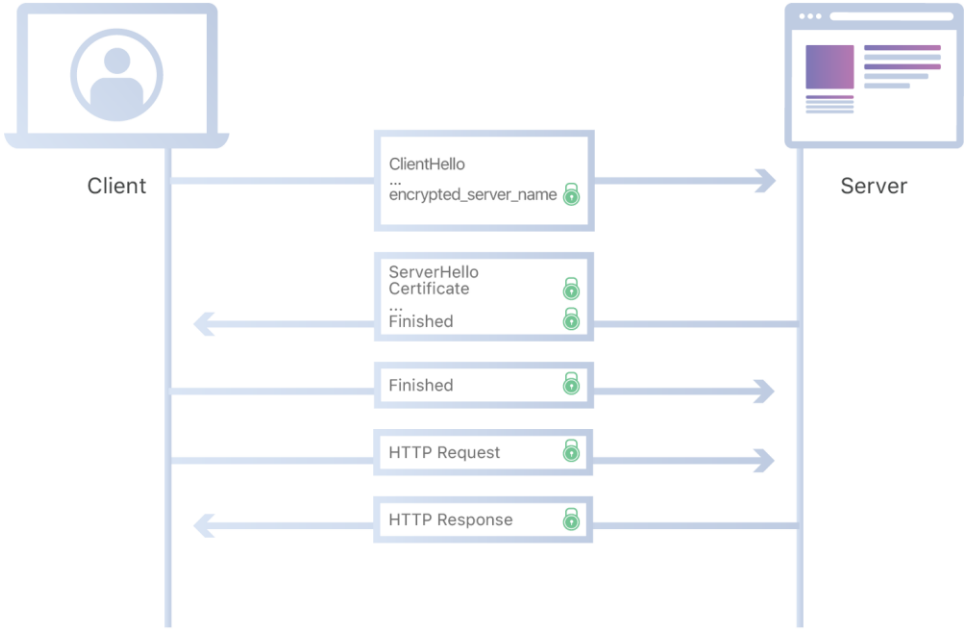
TLS with Unencrypted SNI



ISPs and firewalls track which sites a user is visiting!

CLOUDFLARE

TLS 1.3 offers encrypted SNI



CLOUDFLARE

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

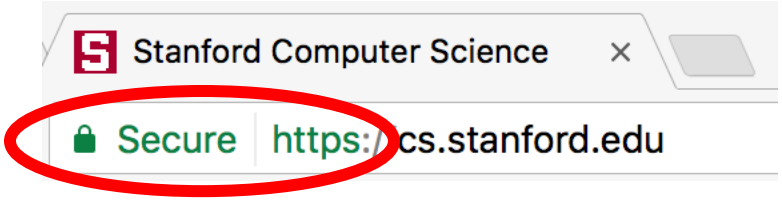
- Incompatible with virtual hosting (older browsers)

  March 2017: IE6 ≈ 1-5% in China (ie6countdown.com)

Aug 2014: Google boosted ranking of sites supporting HTTPS!

HTTPS in the Browser

The lock icon: TLS indicator

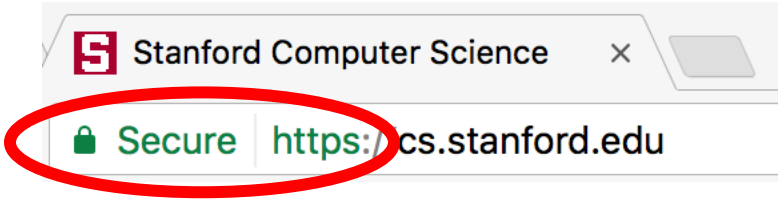


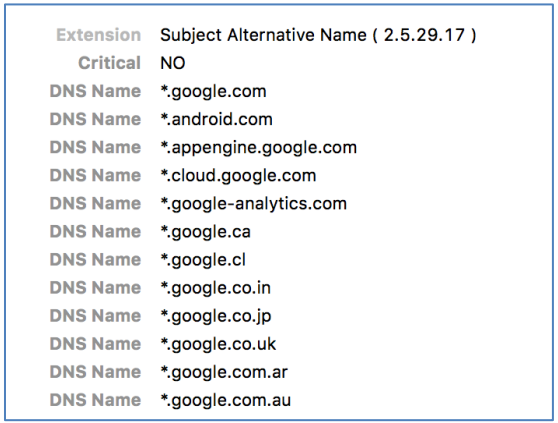
Intended goal:

- 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

A general UI attack

The lock UI: Extended Validation Certs

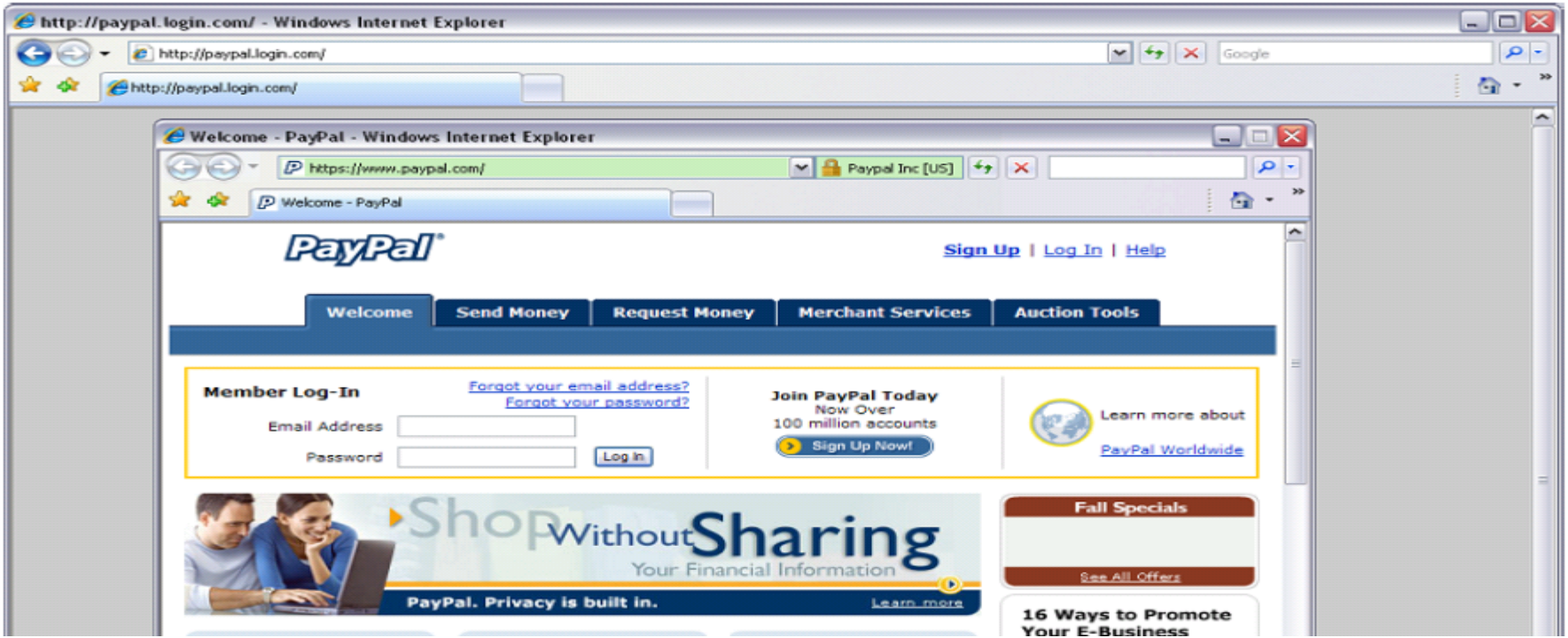
Harder to obtain than regular certs

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

Helps block “semantic attacks”: www.bankofthevvest.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

(old site)

View source:

```
<form method="post"
      action="https://onlineservices.wachovia.com/..."
```

HTTPS and login pages: guidelines

General guideline:

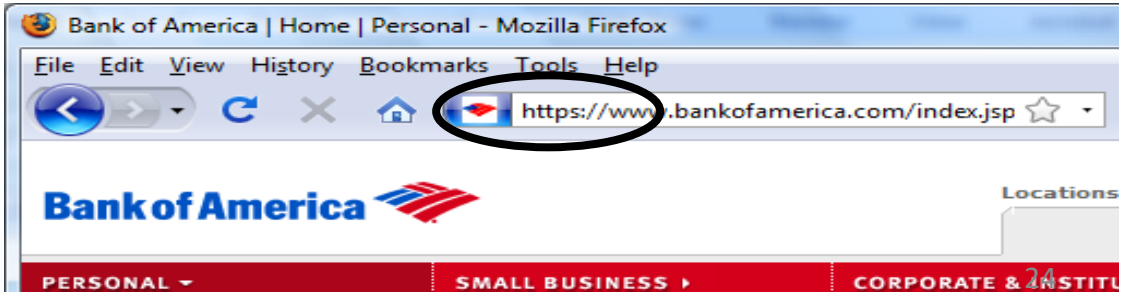
Response to http://login.site.com

should be Location: https://login.site.com

(redirect)

Should be the response to every HTTP request …

→Automatic HTTPS Rewrites



References

- https://badssl.com/dashboard/
- 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”