CSE508 Network Security

10/12/2016

TLS

Michalis Polychronakis

Stony Brook University

TLS (Transport Layer Security)

Predecessor: **SSL** (Secure Socket Layer)

Most widely used protocol for encrypted data transmission

Same basic design, different crypto algorithms

Designed to provide secure communication over the insecure internet

Authentication, confidentiality, and integrity

Used in many services and secure versions of protocols

HTTP, POP, IMAP, SMTP, OpenVPN, CalDAV, CardDAV, LDAP, NNTP, FTP, IRC, SIP, …

Separate port number: HTTPS: 443, FTPS: 990, IMAPS: 993, …

History

SSL developed at Netscape

- v1: never released
- v2 (1994): serious weaknesses
- v3 (1995): re-design, basis of what we use today

NETSCAPE®

TLS working group was formed to migrate SSL to IETF

- TLS 1.0 (1999): minor differences but incompatible with SSL 3 (different crypto algorithms)
- TLS 1.1 (2006): mostly security fixes, TLS extensions
- TLS 1.2 (2008): authenticated encryption, more flexible
- TLS 1.3 (WiP): removal of legacy/weak algorithms, lower latency

Endless cycle of vulnerabilities and improvements

- Insecure renegotiation, RC4 weaknesses, compression side channels, padding oracle attacks, buggy implementations, PKI attacks, …
- Lately with fancy names, too: *BEAST, CRIME, TIME, Lucky 13, BREACH, POODLE, FREAK, Heartbleed, DROWN, …*

Handshake protocol

Negotiate session keys and crypto algorithms to be used

Authentication (server and optionally client)

Takes 6–10 messages, depending on features used

Record Protocol

Message transport: `[header|data]` records (16K)

Encryption and integrity: after handshake completion

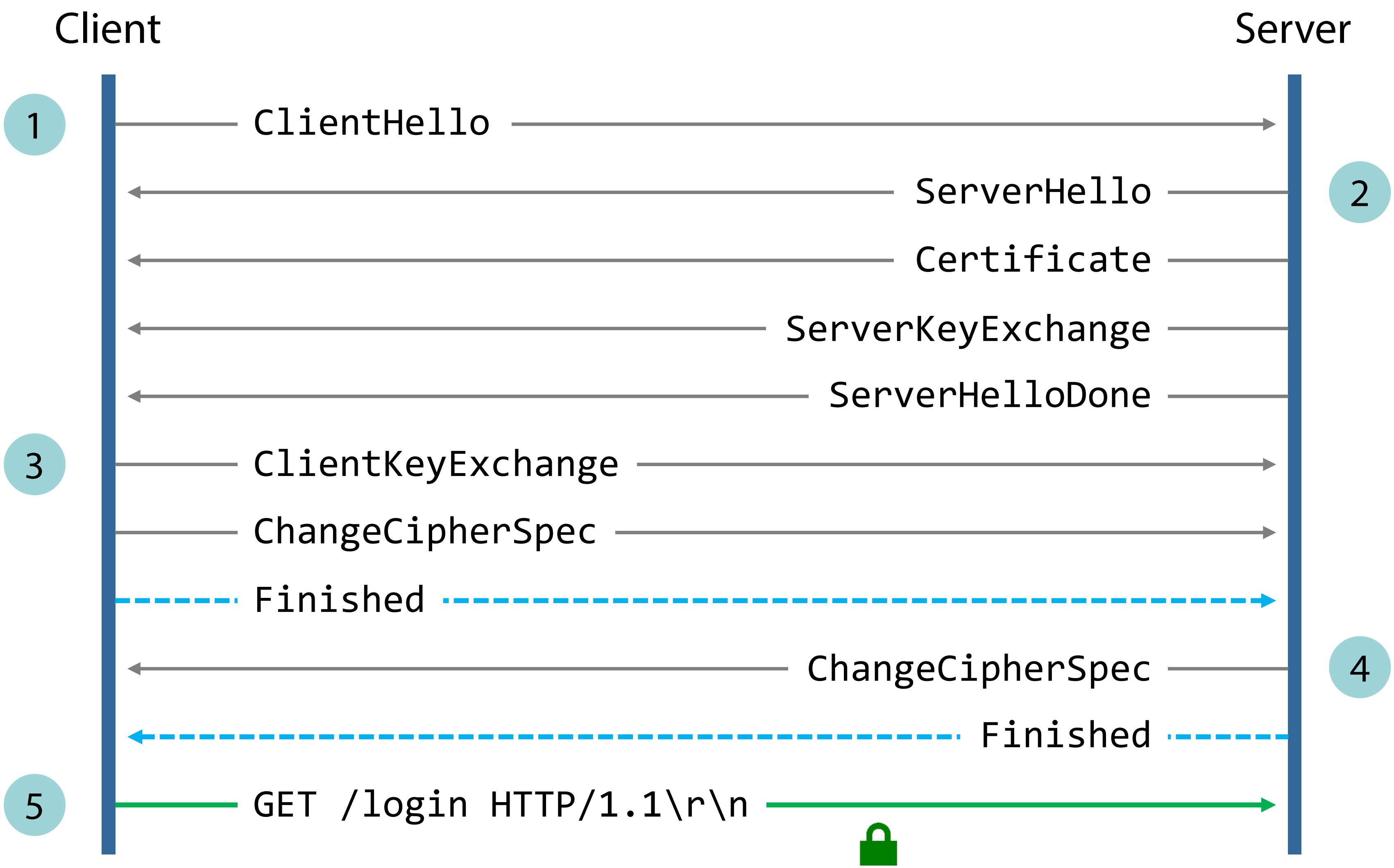
Compression: before encryption… not a good idea

- Side-channel attacks (e.g., CRIME)

Subprotocols: allow for extensibility

- TLS defines four core subprotocols:
  handshake, change cipher spec, application data, alert

TLS 1.2 Handshake (Ephemeral DH)



Cipher Suite Negotiation

ClientHello: *here are the cipher suites I support*

```
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_RC4_128_SHA
...
```

ServerHello: *let's use this one*

```
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
```

The server might not support the best of the client's suites

- Offers some other version hoping that the client will accept it

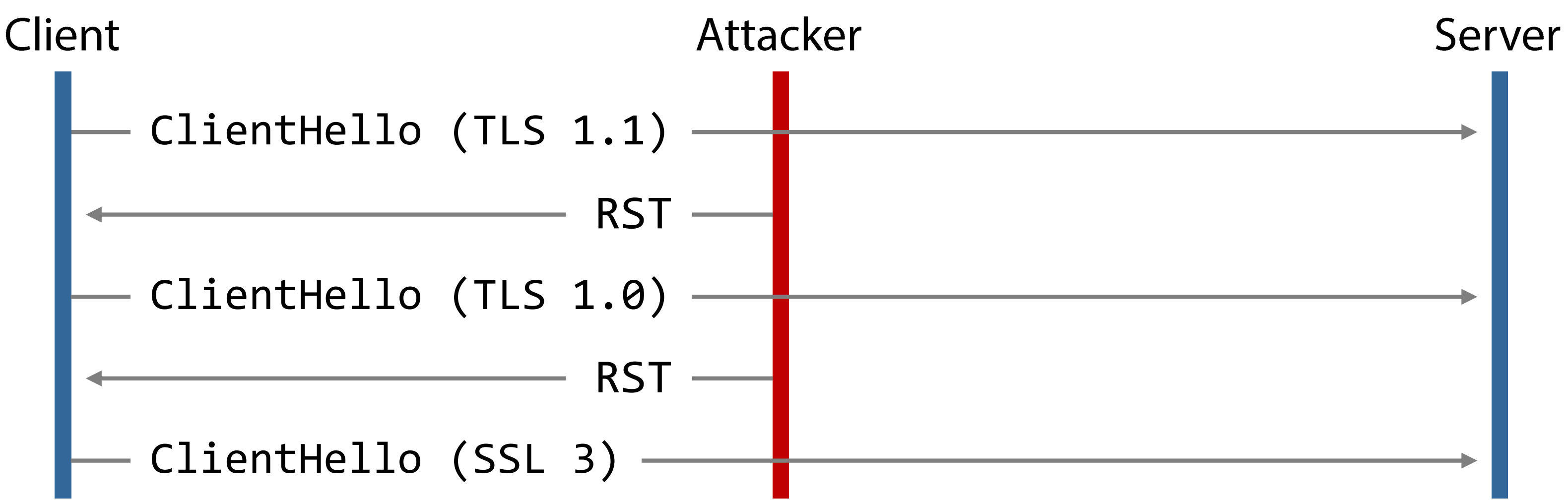
Downgrade Attacks

Force a weaker cipher suite selection through MitM

- SSL 2: no handshake integrity
- SSL 3: protocol rollback protection (still breakable)
- TLS 1.0 and on: additional protections

Due to server bugs and interoperability issues, browsers responded by voluntarily downgrading the protocol upon handshake failure

- Retrying connection with lower SSL/TLS version
- Attackers can exploit this by blocking the initial handshake attempts
- Or modify the client's list of supported clients (and include only weak ones)



SSL 3.0 is now completely removed by most browsers

TLS/SSL support history of web browsers

Browser	Version	Platforms	SSL protocols		TLS protocols				Certificate Support			Vulnerabilities fixed[n 1]						Protocol selection by user [n 2]
			SSL 2.0 (insecure)	SSL 3.0 (insecure)	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3 (proposed)	EV [n 3][48]	SHA-2 [49]	ECDSA [50]	BEAST[n 4]	CRIME[n 5]	POODLE (SSLv3)[n 6]	RC4[n 7]	FREAK[51][52]	Logjam	
Google Chrome (Chrome for Android) [n 8] [n 9]	1–9	Windows (7+) OS X (10.9+) Linux Android (4.1+) iOS (9.0+) Chrome OS	Disabled by default	Enabled by default	Yes	No	No	No	Yes (only desktop)	needs SHA-2 compatible OS[49]	needs ECC compatible OS[50]	Not affected [57]	Vulnerable (HTTPS)	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Yes[n 10]
	10–20		No[58]	Enabled by default	Yes	No	No	No	Yes (only desktop)	needs SHA-2 compatible OS[49]	needs ECC compatible OS[50]	Not affected	Vulnerable (HTTPS/SPDY)	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Yes[n 10]
	21		No	Enabled by default	Yes	No	No	No	Yes (only desktop)	needs SHA-2 compatible OS[49]	needs ECC compatible OS[50]	Not affected	Mitigated [59]	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Yes[n 10]
	22–25		No	Enabled by default	Yes	Yes[60]	No[60][61][62][63]	No	Yes (only desktop)	needs SHA-2 compatible OS[49]	needs ECC compatible OS[50]	Not affected	Mitigated	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Temporary [n 11]
	26–29		No	Enabled by default	Yes	Yes	No	No	Yes (only desktop)	needs SHA-2 compatible OS[49]	needs ECC compatible OS[50]	Not affected	Mitigated	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Temporary [n 11]
	30–32		No	Enabled by default	Yes	Yes	Yes[61][62][63]	No	Yes (only desktop)	needs SHA-2 compatible OS[49]	needs ECC compatible OS[50]	Not affected	Mitigated	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Temporary [n 11]
	33–37		No	Enabled by default	Yes	Yes	Yes	No	Yes (only desktop)	needs SHA-2 compatible OS[49]	needs ECC compatible OS[50]	Not affected	Mitigated	Partly mitigated [n 12]	Lowest priority [66][67][68]	Vulnerable (except Windows)	Vulnerable	Temporary [n 11]
	38, 39		No	Enabled by default	Yes	Yes	Yes	No	Yes (only desktop)	Yes	needs ECC compatible OS[50]	Not affected	Mitigated	Partly mitigated	Lowest priority	Vulnerable (except Windows)	Vulnerable	Temporary [n 11]
	40		No	Disabled by default [65][69]	Yes	Yes	Yes	No	Yes (only desktop)	Yes	needs ECC compatible OS[50]	Not affected	Mitigated	Mitigated [n 13]	Lowest priority	Vulnerable (except Windows)	Vulnerable	Yes[n 14]
	41, 42		No	Disabled by default	Yes	Yes	Yes	No	Yes (only desktop)	Yes	needs ECC compatible OS[50]	Not affected	Mitigated	Mitigated	Lowest priority	Mitigated	Vulnerable	Yes[n 14]
	43		No	Disabled by default	Yes	Yes	Yes	No	Yes (only desktop)	Yes	needs ECC compatible OS[50]	Not affected	Mitigated	Mitigated	Only as fallback [n 15][70]	Mitigated	Vulnerable	Yes[n 14]
	44–47		No	No[71]	Yes	Yes	Yes	No	Yes (only desktop)	Yes	needs ECC compatible OS[50]	Not affected	Mitigated	Not affected	Only as fallback [n 15]	Mitigated	Mitigated[72]	Temporary [n 11]
	48, 49		No	No	Yes	Yes	Yes	No	Yes (only desktop)	Yes	needs ECC compatible OS[50]	Not affected	Mitigated	Not affected	Disabled by default[n 16][73][74]	Mitigated	Mitigated	Temporary [n 11]
	50–53		No	No	Yes	Yes	Yes	No	Yes (only desktop)	Yes	Yes	Not affected	Mitigated	Not affected	Disabled by default[n 16][73][74]	Mitigated	Mitigated	Temporary [n 11]
	54–60 61		No	No	Yes	Yes	Yes	Disabled by default (Experimental)	Yes (only desktop)	Yes	Yes	Not affected	Mitigated	Not affected	Disabled by default[n 16][73][74]	Mitigated	Mitigated	Temporary [n 11]

https://en.wikipedia.org/wiki/Transport_Layer_Security

TLS 1.2 Session Resumption

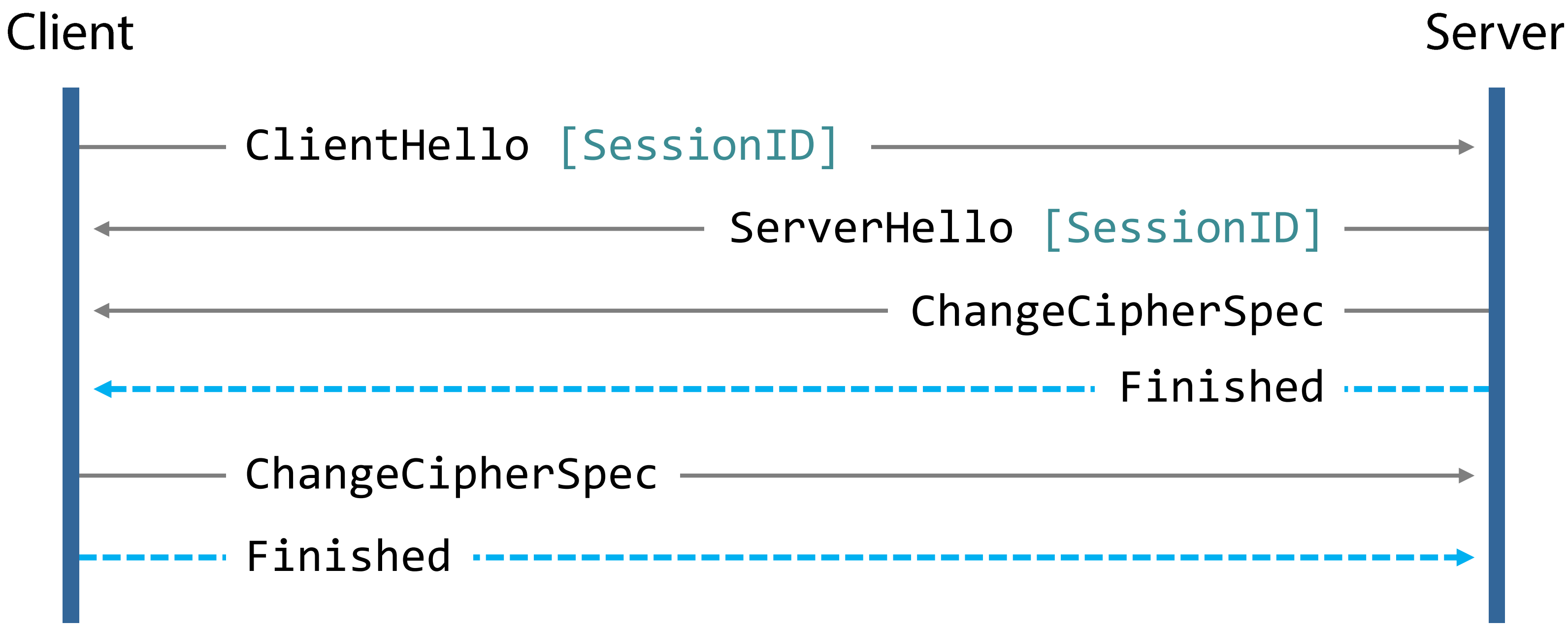
Full handshake: 6-10 messages and two network round-trips

- Along with CPU-intensive crypto operations, cert validation, …

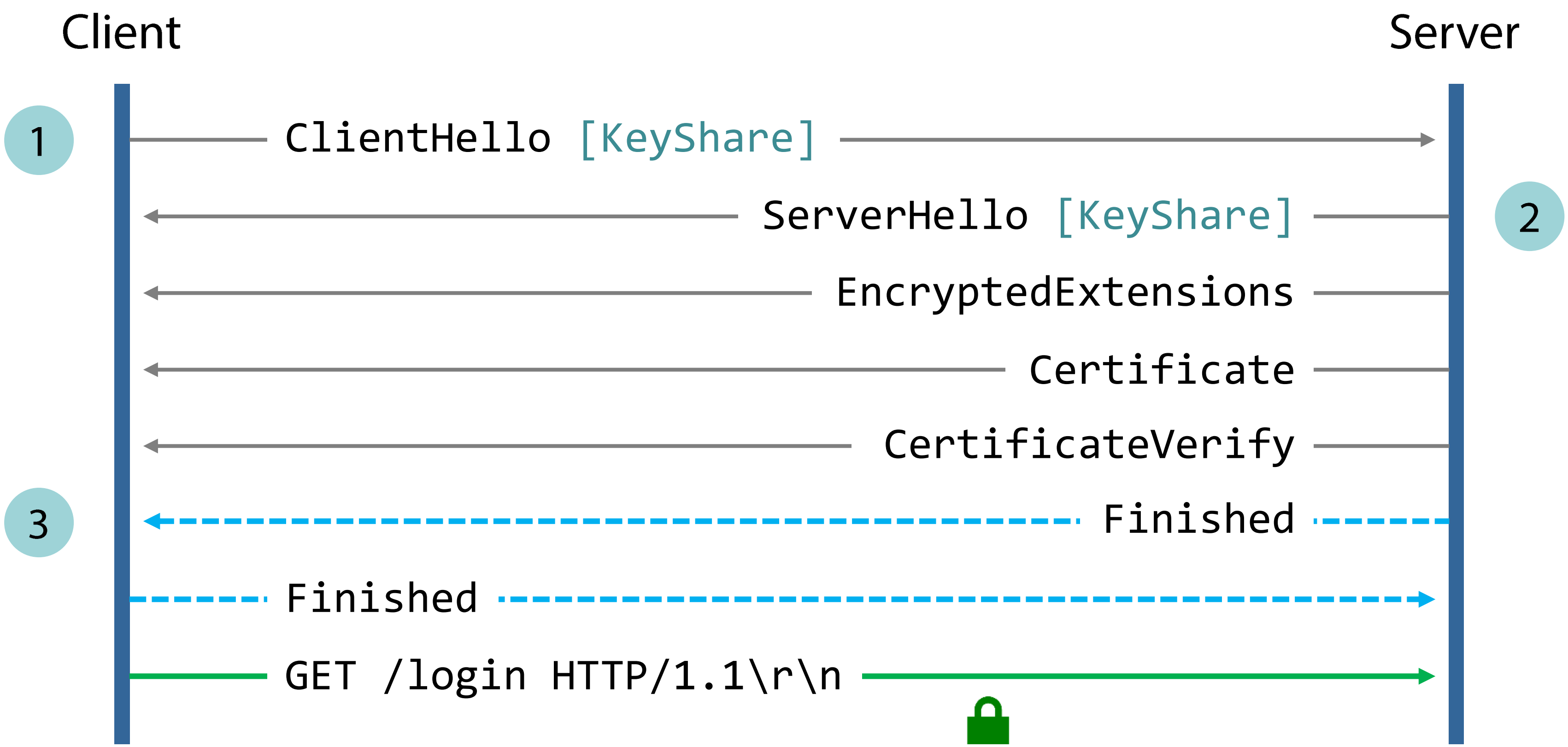
Avoid re-negotiation by remembering security parameters

- Server assigns and sends a unique session ID as part of `ServerHello`
- In future connections, the client sends the session ID to resume the session

Alternative: *session tickets* (all state is kept at client)



TLS 1.3 Handshake (Ephemeral DH)



Latest draft supports even zero-RTT handshakes

- Clients include encrypted data in the initial handshake messages
- Based on configuration identifier previously sent by the server

Server (and Client) Authentication

After handshake completion, the client knows it can "trust" the information in the server's certificate

- Assuming it trusts the issuing CA

SSL/TLS certs are based on the X.509 PKI standard

How is the certificate associated with the server?

- *Common Name (CN): server's hostname*

Same process is supported for authenticating clients

- Highly-secure web services, some VPN services, …

Not used often in practice

- Common alternative: username + password over TLS connection

Certificate Fields

Version: v1 (basic), v2 (additional fields), v3 (extensions)

Serial Number: high-entropy integer

Signature Algorithm: encryption and hash algorithm used to sign the certificate

Issuer: complex field that contains the *distinguished name (DN)* of the certificate issuer

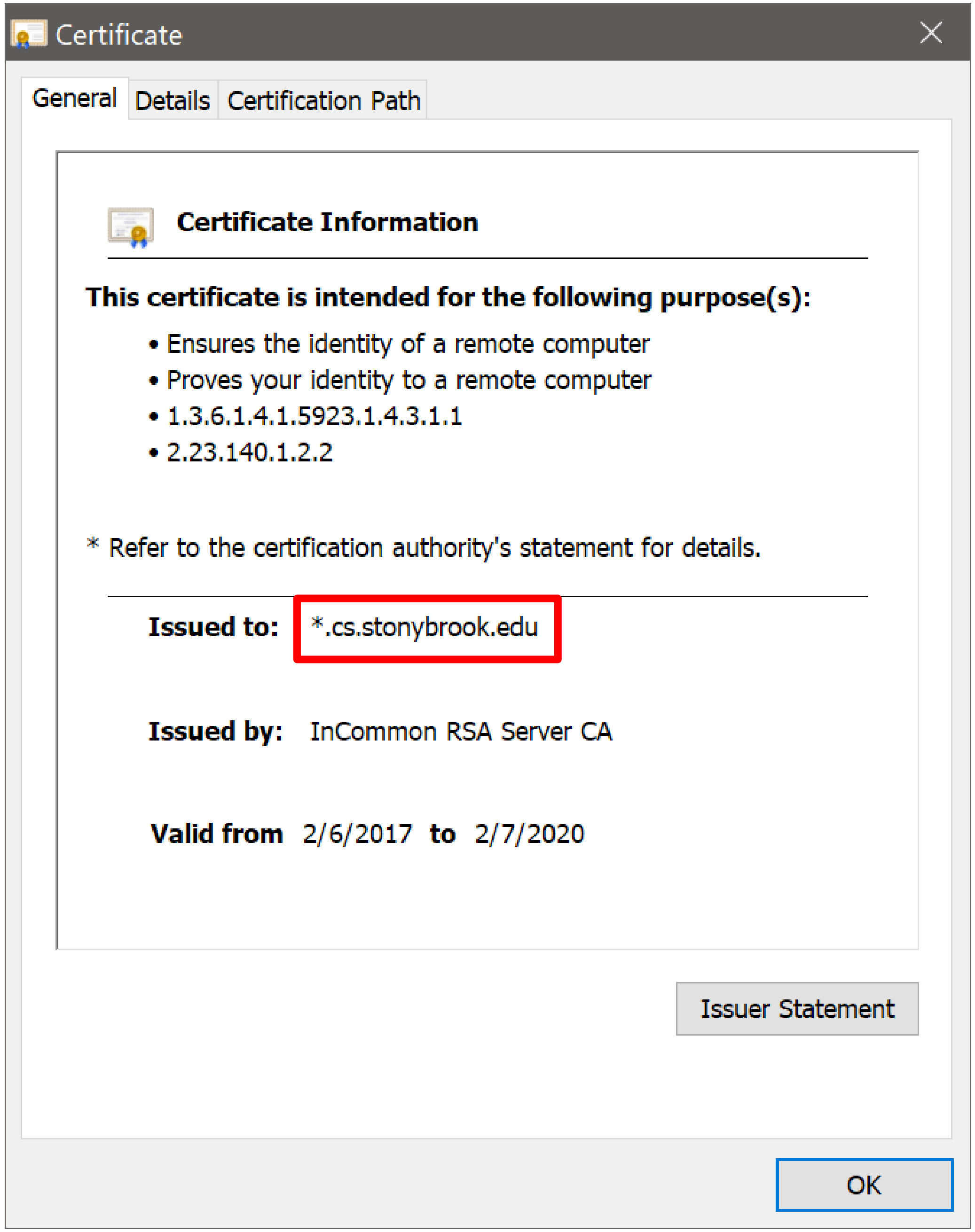
Validity: starting and ending date of validity period

Subject: DN of the entity associated with the public key for which the certificate is issued

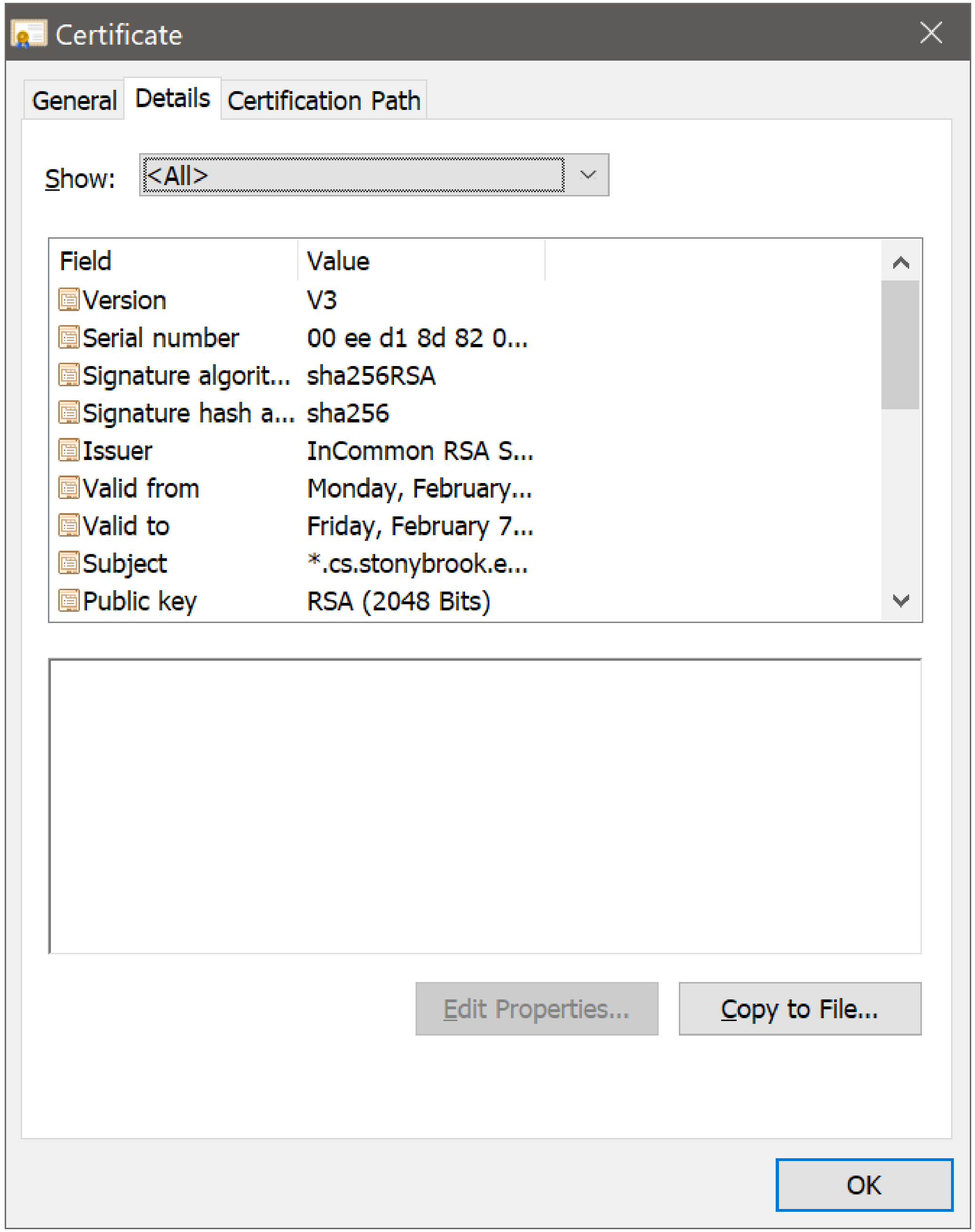
> Deprecated in favor of the Subject Alternative Name (SAN) extension: DNS name, IP address, or URI (also supports binding to multiple identities)

Public Key: The subject's public key

Signature



Certificate
General
Details
Certification Path
Certificate Information
This certificate is intended for the following purpose(s):
• Ensures the identity of a remote computer
• Proves your identity to a remote computer
• 1.3.6.1.4.1.5923.1.4.3.1.1
• 2.23.140.1.2.2
* Refer to the certification authority's statement for details.
Issued to: *.cs.stonybrook.edu
Issued by: InCommon RSA Server CA
Valid from 2/6/2017 to 2/7/2020
Issuer Statement
OK


Certificate
General
Details
Certification Path
Show:
<All>
Field
Value
Version
V3
Serial number
00 ee d1 8d 82 0...
Signature algorit...
sha256RSA
Signature hash a...
sha256
Issuer
InCommon RSA S...
Valid from
Monday, February...
Valid to
Friday, February 7...
Subject
*.cs.stonybrook.e...
Public key
RSA (2048 Bits)
Edit Properties...
Copy to File...
OK

Certificate Chains

Trust anchors: systems are pre-configured with ~200 trusted root certificates

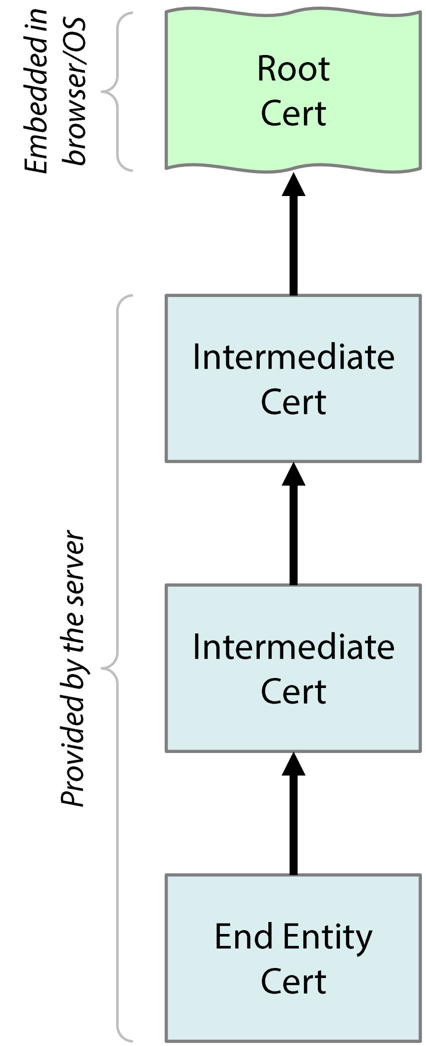
- System/public store: used by OS, browsers, …
- More can be added in the local/private cert store: vendor-specific certs, MitM certs for content inspection filters/AVs, …

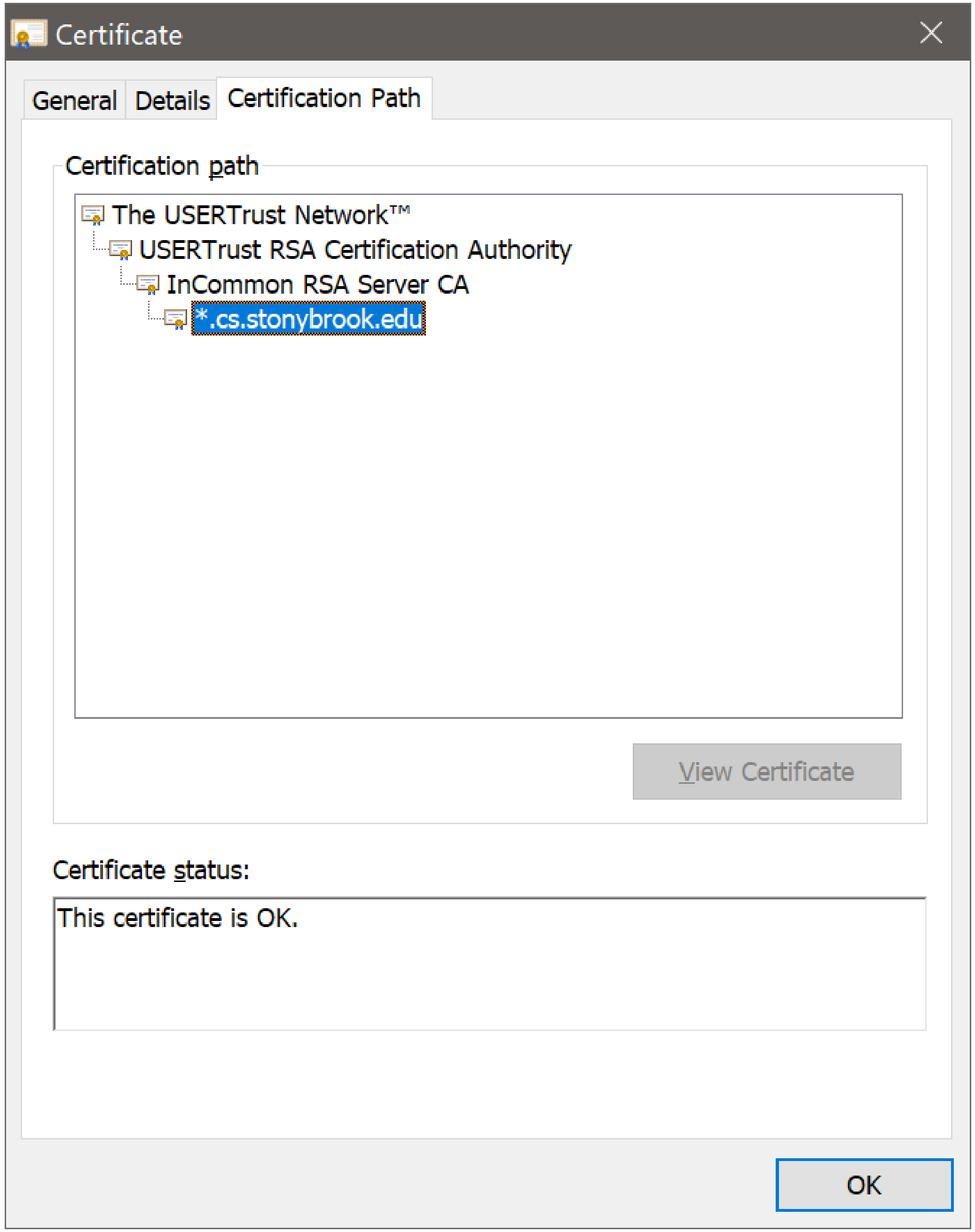
Server provides a *chain* of certificates

- A certificate from an intermediate CA is trusted if there is a valid chain of trust all the way back to a trusted root CA

Any certificate authority can issue and sign certificates for any subject

- The system is only as secure as the weakest certificate authority…




Certificate
General
Details
Certification Path
Certification path
The USERTrust Network™
USERTrust RSA Certification Authority
InCommon RSA Server CA
*.cs.stonybrook.edu
View Certificate
Certificate status:
This certificate is OK.
OK

HTTPS

Most common use of SSL/TLS

More than half of web traffic is now encrypted

Crypto is expensive, needs more CPU cycles

Not a big deal these days (native hardware support)

Mixed content: Ad networks, mashups, …

Stop using them! …easier said than done (lost revenue, increased development time)

Incentives: Google rewards HTTPS sites with higher ranking

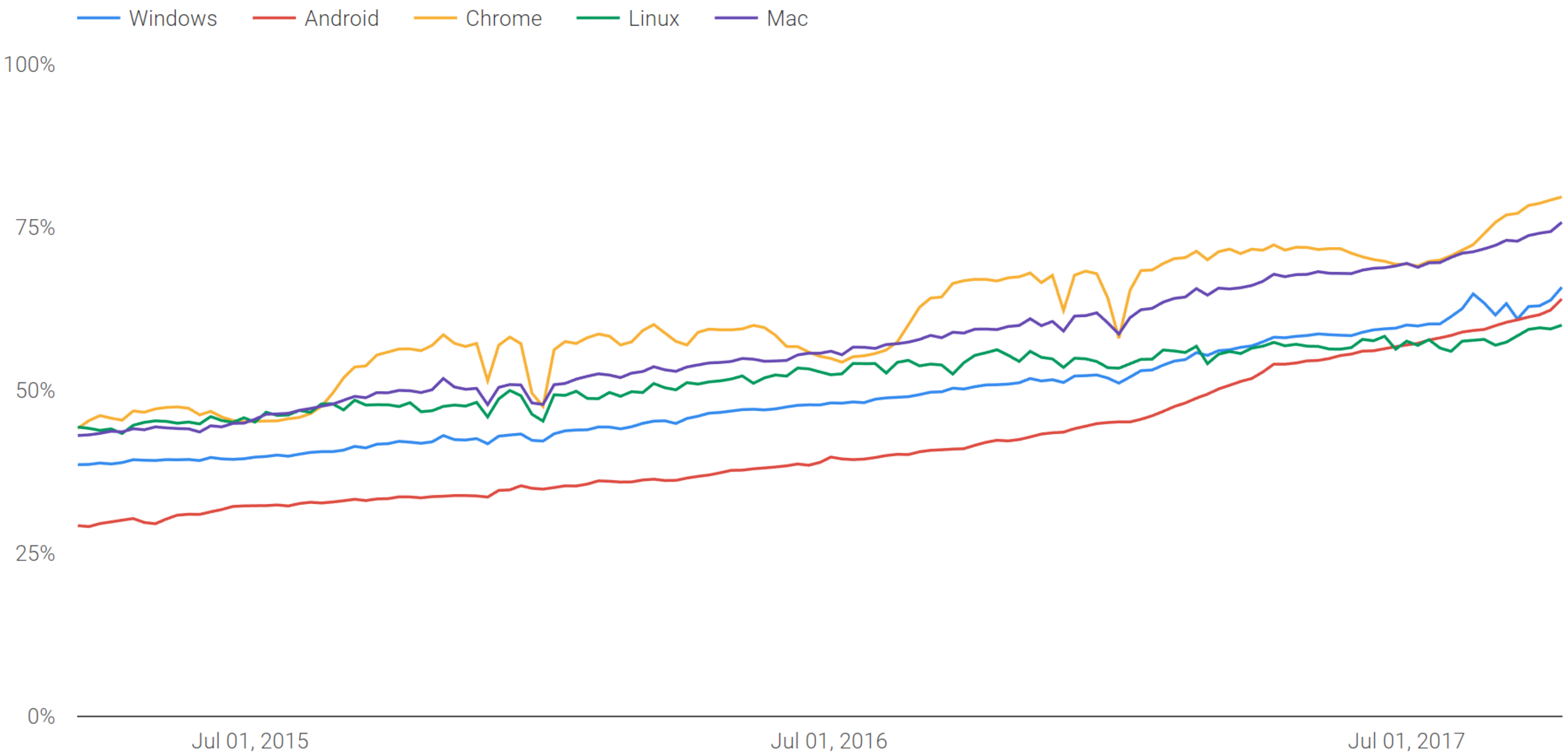
Virtual Hosting: initially incompatible

Not anymore: solved as of TLS 1.1 through the *Server Name Indication (SNI)* extension ➔ what about IE6/WinXP users?

Needs expertise and certs cost $$$$

Not anymore: letsencrypt.org

HTTPS Encryption in Chrome



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

Browser Security Indicators

Convey information about the security of a page

- Locks, shields, keys, green bars…

"This page was fetched using SSL"

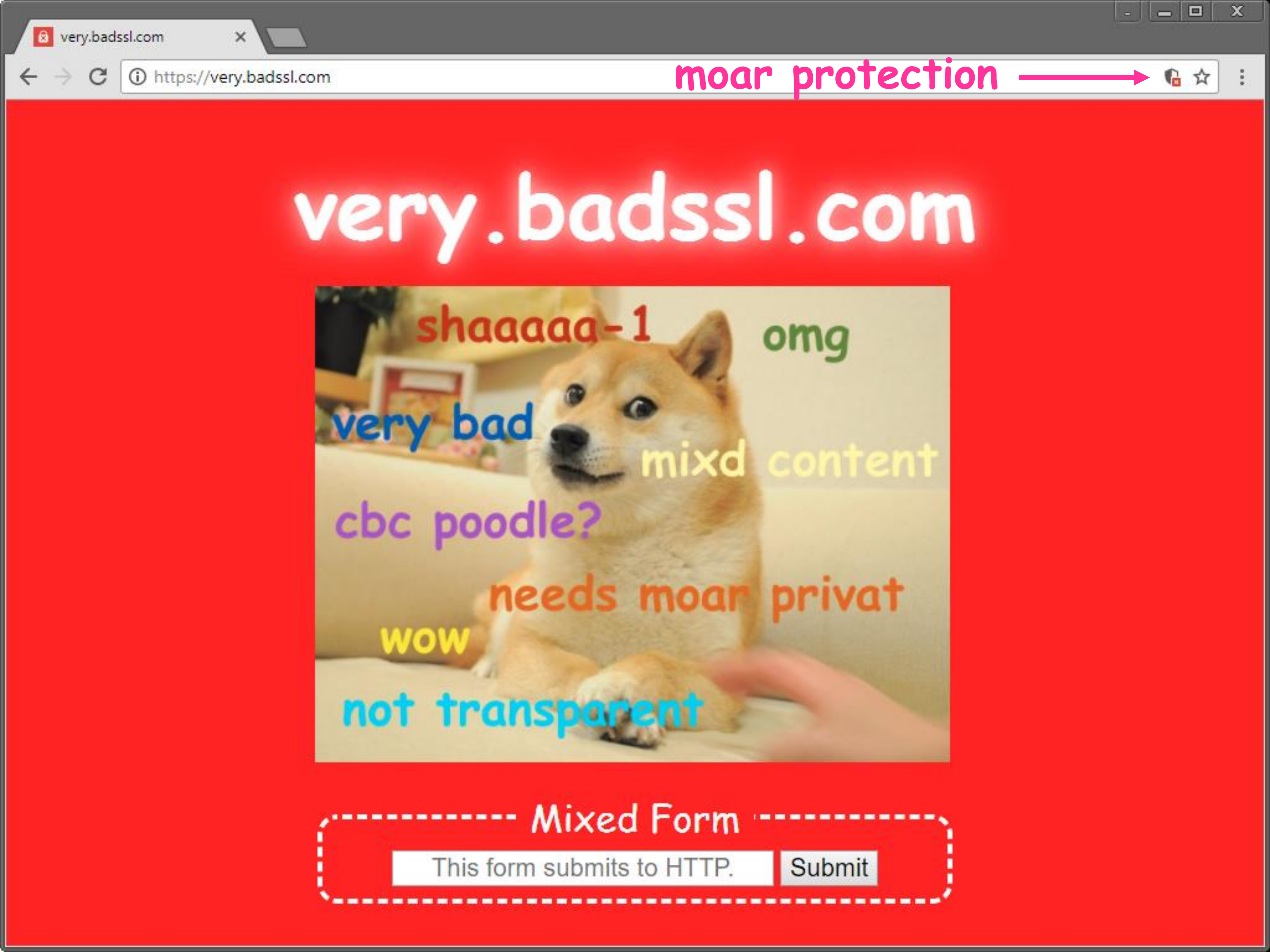
- Page content was not viewed or altered by a network adversary
- Certificate is valid (e.g. not expired), issued by a CA trusted by the browser, and the subject name matches the URL's domain

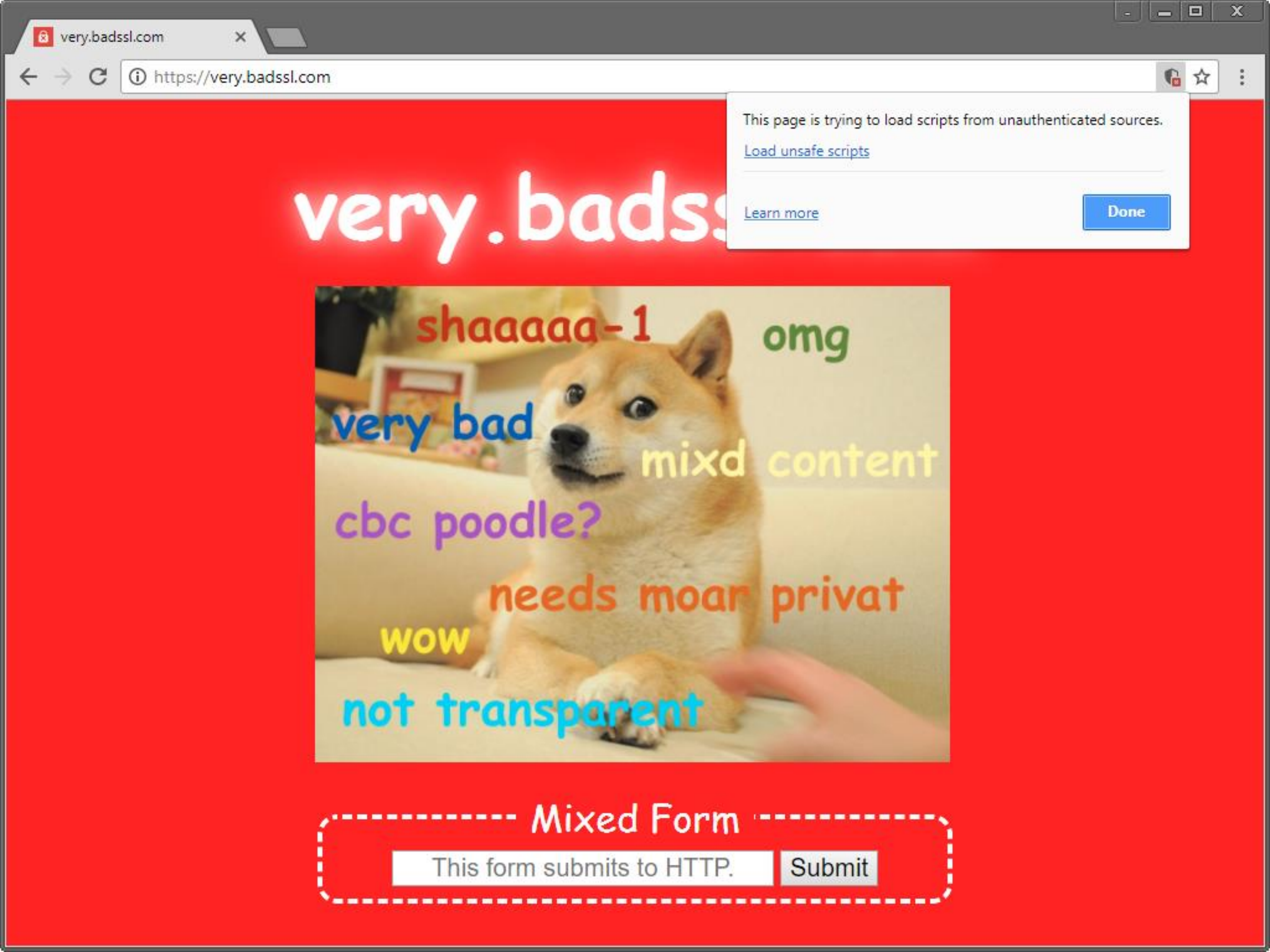
"This page uses an invalid certificate"

"Parts of the page are not encrypted"

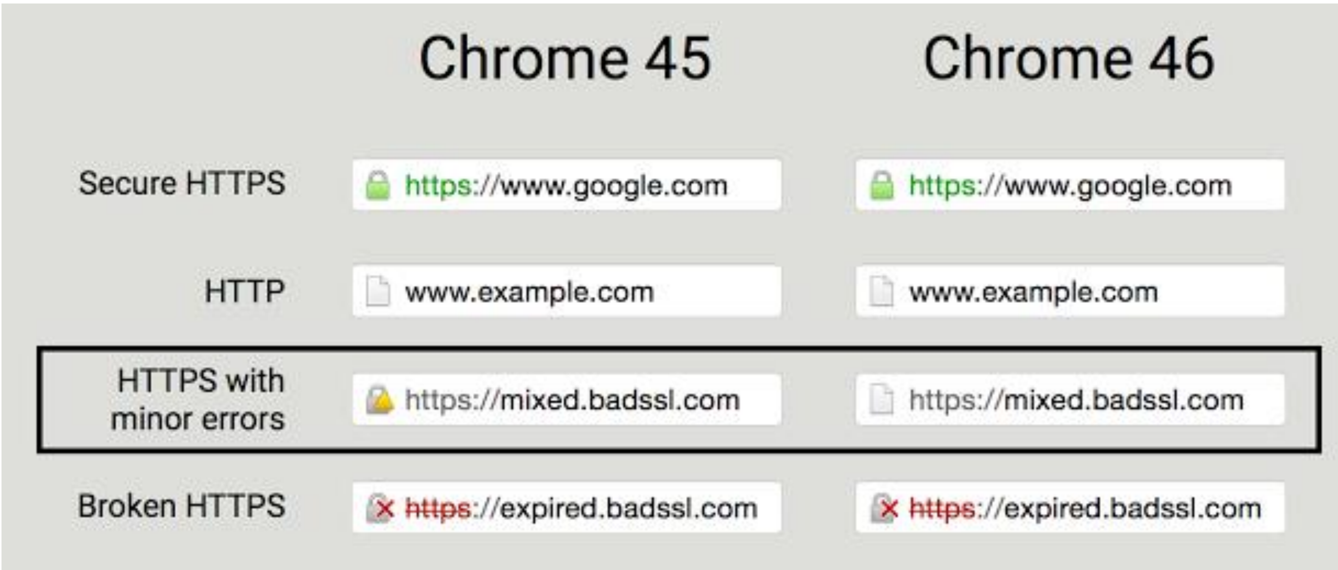
"The legal entity operating this web site is known"

- Extended Validation (EV) certificates


very.badssl.com
https://very.badssl.com
moar protection
very.badssl.com
shaaaaa-1
omg
very bad
mixd content
cbc poodle?
needs moar privat
wow
not transparent
Mixed Form
This form submits to HTTP.
Submit


very.badssl.com
https://very.badssl.com
This page is trying to load scripts from unauthenticated sources.
Load unsafe scripts
Learn more
Done
very.badss
shaaaaa-1
omg
very bad
mixd content
cbc poodle?
needs moar privat
wow
not transparent
Mixed Form
This form submits to HTTP.
Submit

Mixed Content Warning



Basically the same in terms of security

Fewer security states for users to remember

Reflects better the security state of the page

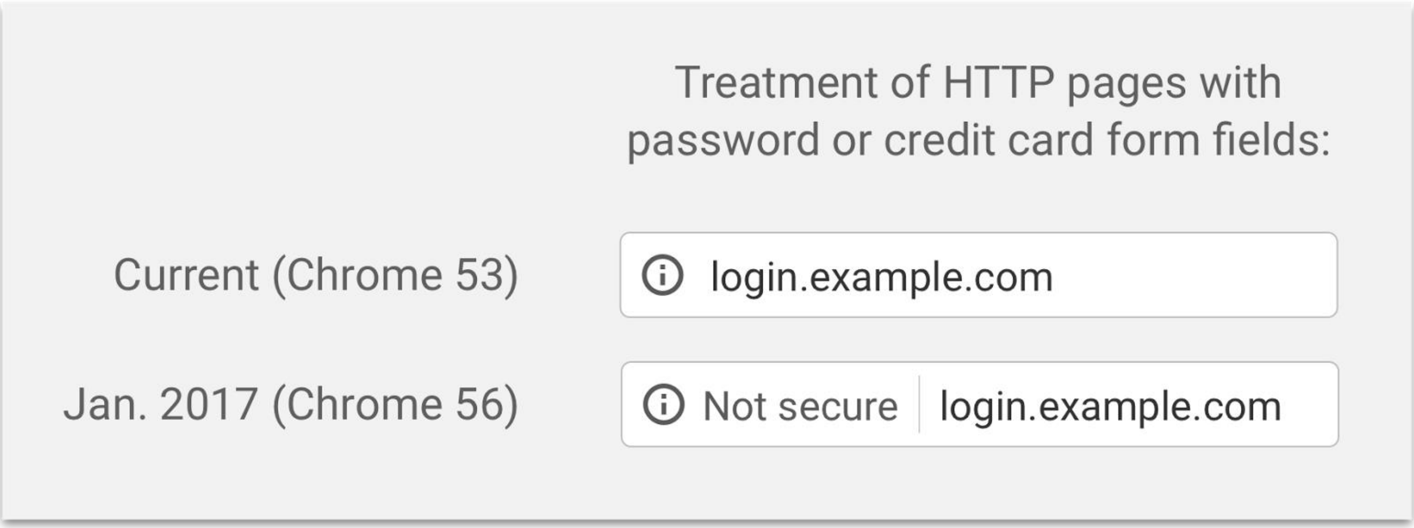
- **Non-HTTPS traffic is a vulnerability!**
- MitM/MotS attacks on the HTTP part are trivial…

https://googleonlinesecurity.blogspot.com/2015/10/simplifying-page-security-icon-in-chrome.html

Marking HTTP as Non-Secure

Phase 1: page is marked "Not Secure" when

- The page *contains a password field*
- The user *interacts with a credit card field*

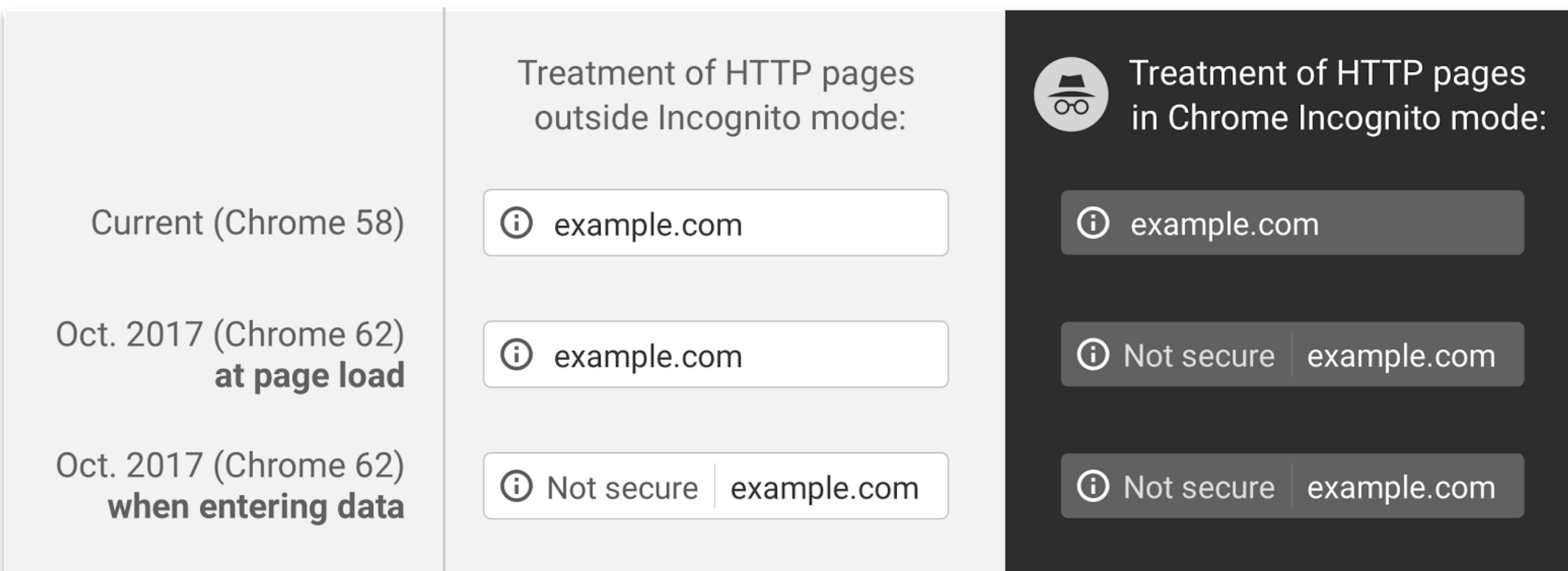


https://www.chromium.org/Home/chromium-security/marking-http-as-non-secure

Marking HTTP as Non-Secure

Phase 2: page is marked "Not Secure" when

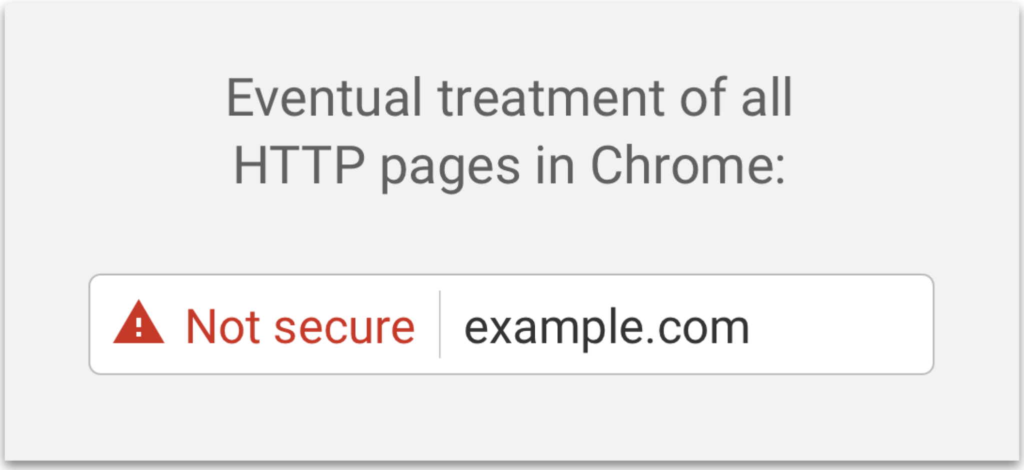
- The page *contains a password field*
- The user *interacts with any input field*
- The user is browsing in Chrome *incognito mode*



https://www.chromium.org/Home/chromium-security/marking-http-as-non-secure

Marking HTTP as Non-Secure

In the future: all HTTP pages are marked "Not Secure"



SSL stripping

Browsing sessions often start with a plain HTTP page

- Web sites switch to HTTPS only for login or checkout
- Example: Facebook in 2010 (optional full HTTPS in 2011, on by default in 2013)
- Users type addresses without specifying "https://"
- Browser connects over HTTP ➔ site may redirect to HTTPS

SSLstrip [Moxie Marlinspike, Black Hat DC 2009]

- MitM attack to prevent redirection to HTTPS
- Watch for HTTPS redirects and links, and map them to HTTP links
- …or homograph-similar *valid* HTTPS links: https://www.bank.com.attacker.com

SSL stripping


```
Location: http://…
<a href="http://…">
<form action="http://…">
```

```
Location: https://…
<a href="https://…">
<form action="https://…">
```

Missing lock icon, but who is going to notice?

HSTS (HTTP Strict Transport Security)

Defense against SSL stripping and other issues

- Convert any insecure links (http://) into secure links (https://) *before* accessing a resource
- Treat all errors (e.g., invalid certificate, mixed content) as fatal: do not allow users to access the web application

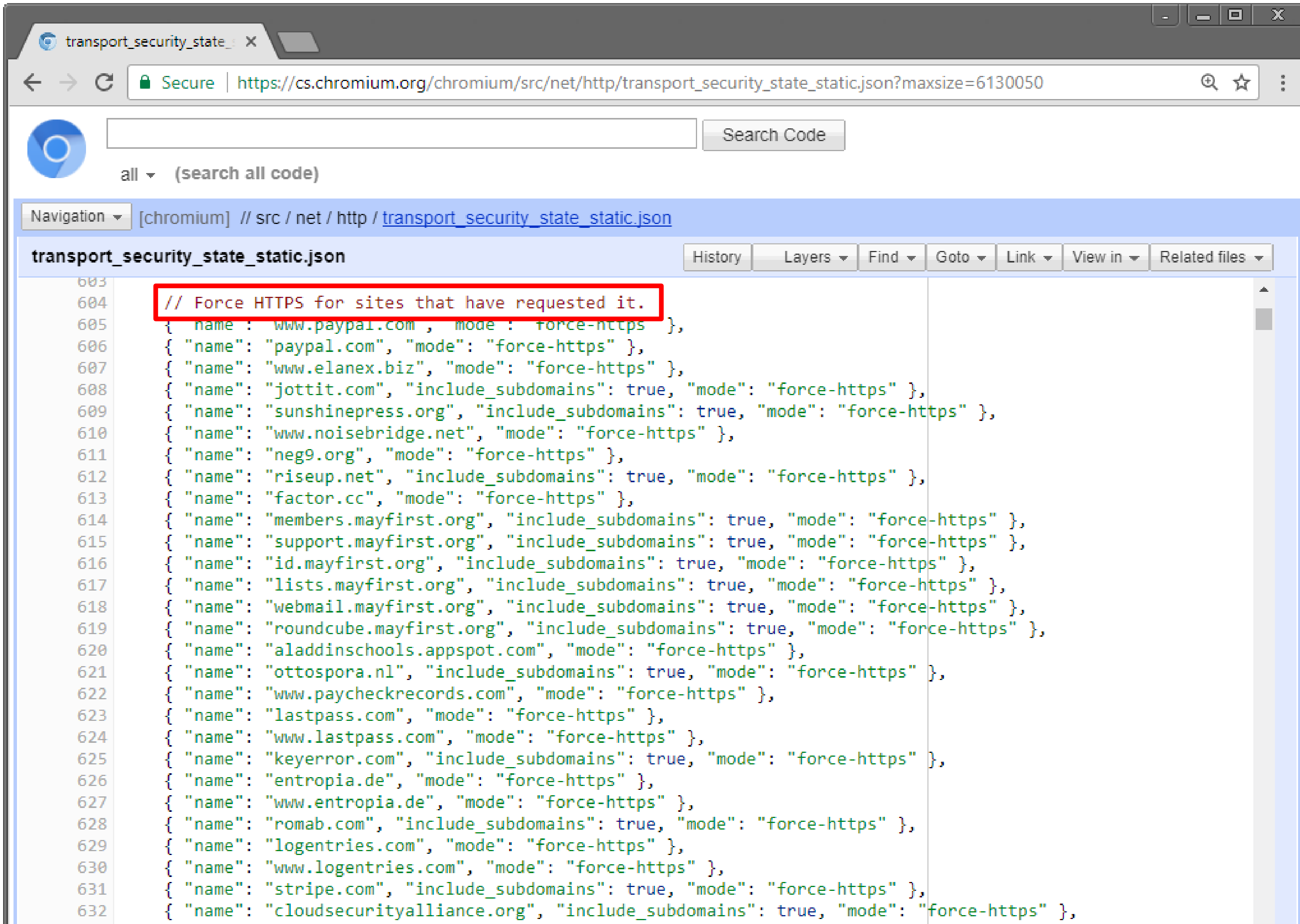
A server implements an HSTS policy by supplying an extra HTTP header

- `Strict-Transport-Security: max-age=31536000`
- *"Use only HTTPS for future requests to this domain for the next year"*

An instance of *trust on first use (TOFU)*

- The initial request *remains unprotected* if sent over HTTP
- HSTS preloading: browser comes with a list of known HSTS sites

HSTS Preloading



MitM is Still Possible…

Rogue certificates

- Most governments have a trusted root CA planted in our systems
- Attackers may break into CAs and forge certificates

Pre-planted/generated certificates

- Default static keys: Lenovo, Dell, anti-malware software, …
- Low entropy during key generation: repeated or factorable keys

Self-signed certificates

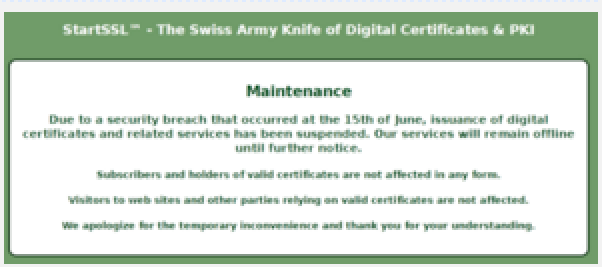
- If desperate… will trigger browser warning

Exploitation of certificate validation flaws

- Programming errors while checking date, hostname, …

StartSSL suspends services after security breach

StartSSL has suspended issuance of digital certificates and related services following a security breach on 15 June. A trademark of Eddy Nigg's StartCom, the StartSSL certificate authority is well known for offering free domain validated SSL certificates, but also sells organisation and extended validation certificates.



More than 25 thousand websites in Netcraft's SSL survey use certificates issued by StartSSL. These are recognised by Internet Explorer, Firefox, Chrome and other mainstream browsers.

StartSSL is not alone in offering free certificates. AffirmTrust recently trumped StartSSL's one-year certificates with its own offer of free three-year domain validated SSL certificates. Coincidentally, AffirmTrust announced its launch on the same day as the StartSSL security breach.

StartSSL is also not the only certificate authority to come under attack this year. In March, Comodo came under attack through three of its resellers. By compromising a GlobalTrust website, the so-called *ComodoHacker* managed to fraudulently issue several valid certificates, including ones for the login pages of Yahoo and Skype. These certificates were subsequently revoked and browser software was updated to explicitly

Most Popular

1. January 2016 Web Server Survey
2. DigitalOcean becomes the second largest hosting company in the world
3. January 2015 Web Server Survey
4. eBay scripting flaws being actively exploited by fraudsters
5. Certificate revocation: Why browsers remain affected by Heartbleed
6. September 2015 Web Server Survey
7. February 2016 Web Server Survey
8. Fraudsters modify eBay listings with JavaScript redirects and proxies
9. March 2015 Web Server Survey
10. AlphaBay darknet phishing attack impersonates .onion domain

Comodo hacker: I hacked

arstechnica.com/security/2011/09/comodo-hacker-i-hacked-diginotar-too-other-cas-breached/

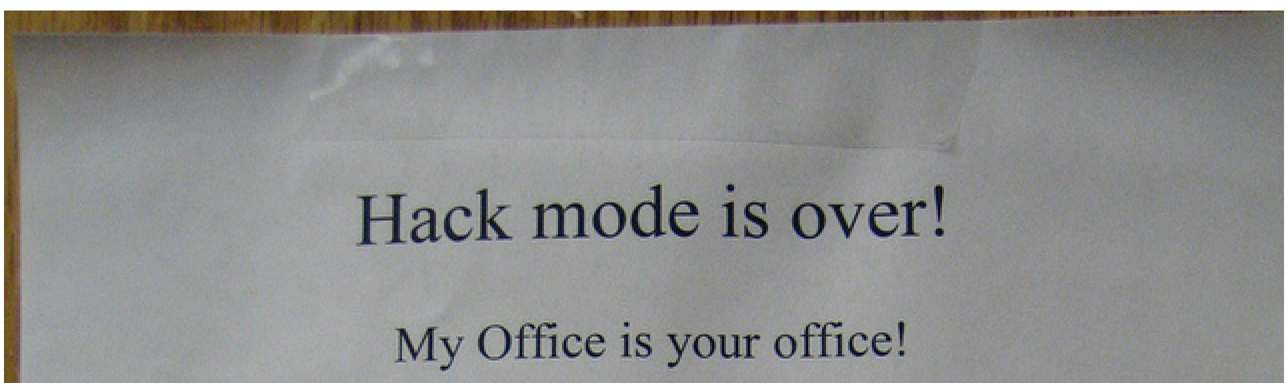
RISK ASSESSMENT / SECURITY & HACKTIVISM

Comodo hacker: I hacked DigiNotar too; other CAs breached

The hacker behind this year's Comodo hack has claimed responsibility for the …

by Peter Bright - Sep 6, 2011 5:36pm EDT

Share Tweet Email 35



Photograph by Augie Schwer

Security

Trustwave to escape 'death penalty' for SSL skeleton key

Moz likely to spare certificate-confession biz same fate as DigiNotar

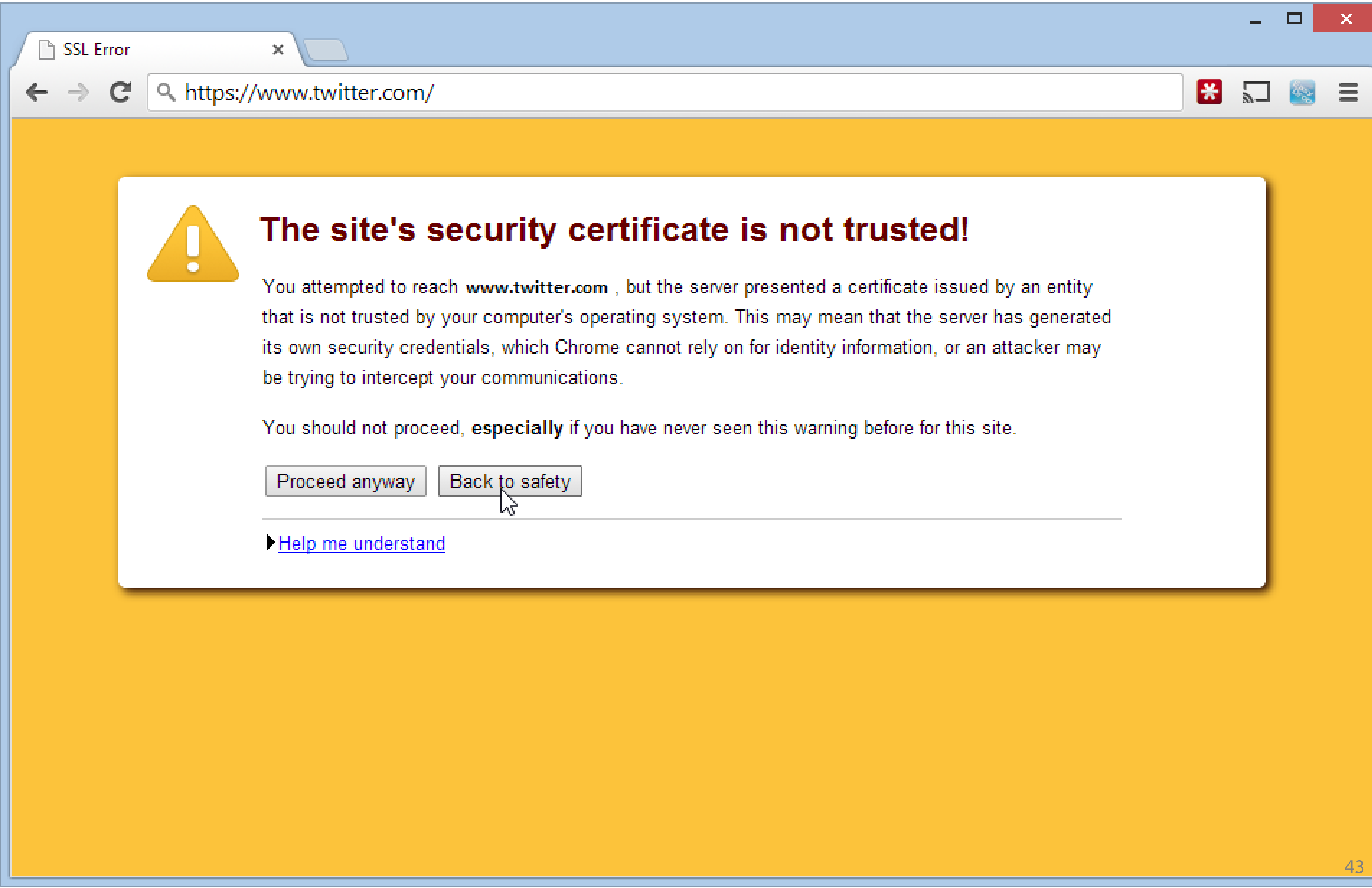
14 Feb 2012 at 09:28, John Leyden

Analysis Trustwave's admission that it issued a digital "skeleton key" that allowed an unnamed private biz to spy on SSL-encrypted connections within its corporate network has sparked a fiery debate about trust on the internet.

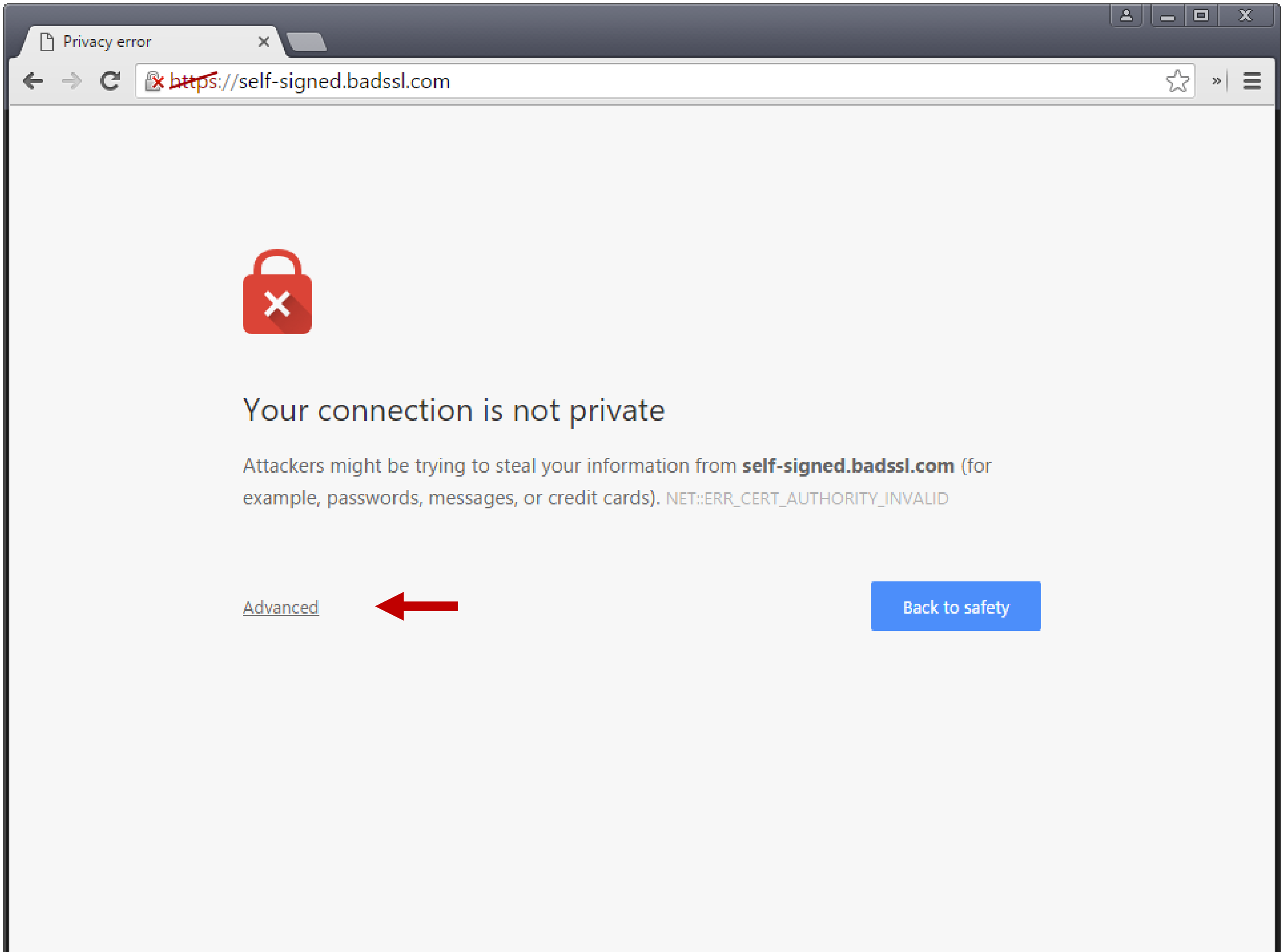
Trustwave, an SSL certificate authority, confessed to supplying a subordinate root certificate as part of an information security product that allowed a customer to monitor employees' web communications - even if the staffers relied on HTTPS. Trustwave said the man-in-the-middle (MitM) gear was designed both to be tamper-proof and to work only within its unnamed client's compound. Despite these precautions, Trustwave now admits that the whole approach was misconceived and would not be repeated. In addition, it revoked the offending certificate.

Trustwave came clean without the need for pressure beforehand. Even so its action have split security experts and prompted calls on Mozilla's Bugzilla security list to remove the Trustwave root certificate

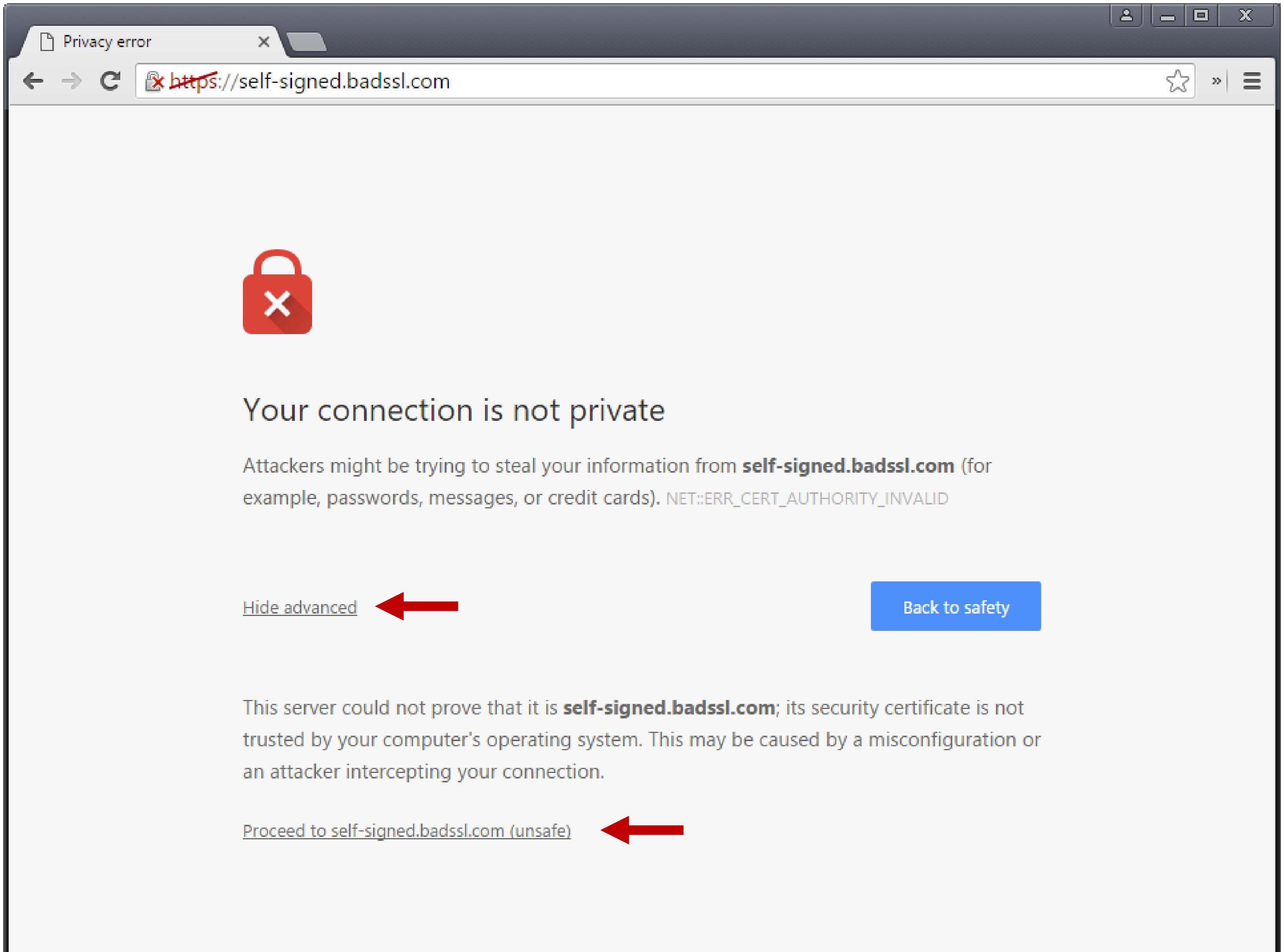
Self-signed Certificate Warning: One click away…



Self-signed Certificate Warning: Two clicks away…



Self-signed Certificate Warning: Two clicks away…



GOTO FAIL

iOS 7.0.6 signature verification error

Legitimate-looking TLS certificates with a mismatched private keys were unconditionally accepted…

```
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
            goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
            goto fail;
            goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
            goto fail;
    ...

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
```

?!!?!?!?

Check never executed

HPKP (HTTP Public Key Pinning)

Prevent *certificate forgery*: strong form of web site authentication

Browser knows the *valid* public keys of a particular website

- If a seemingly valid chain does not include at least one known pinned key, cert is rejected ➔ not issued according to the site's expectations

Doesn't apply for *private* root certificates

- Would break preconfigured proxies, anti-malware, content filters, …

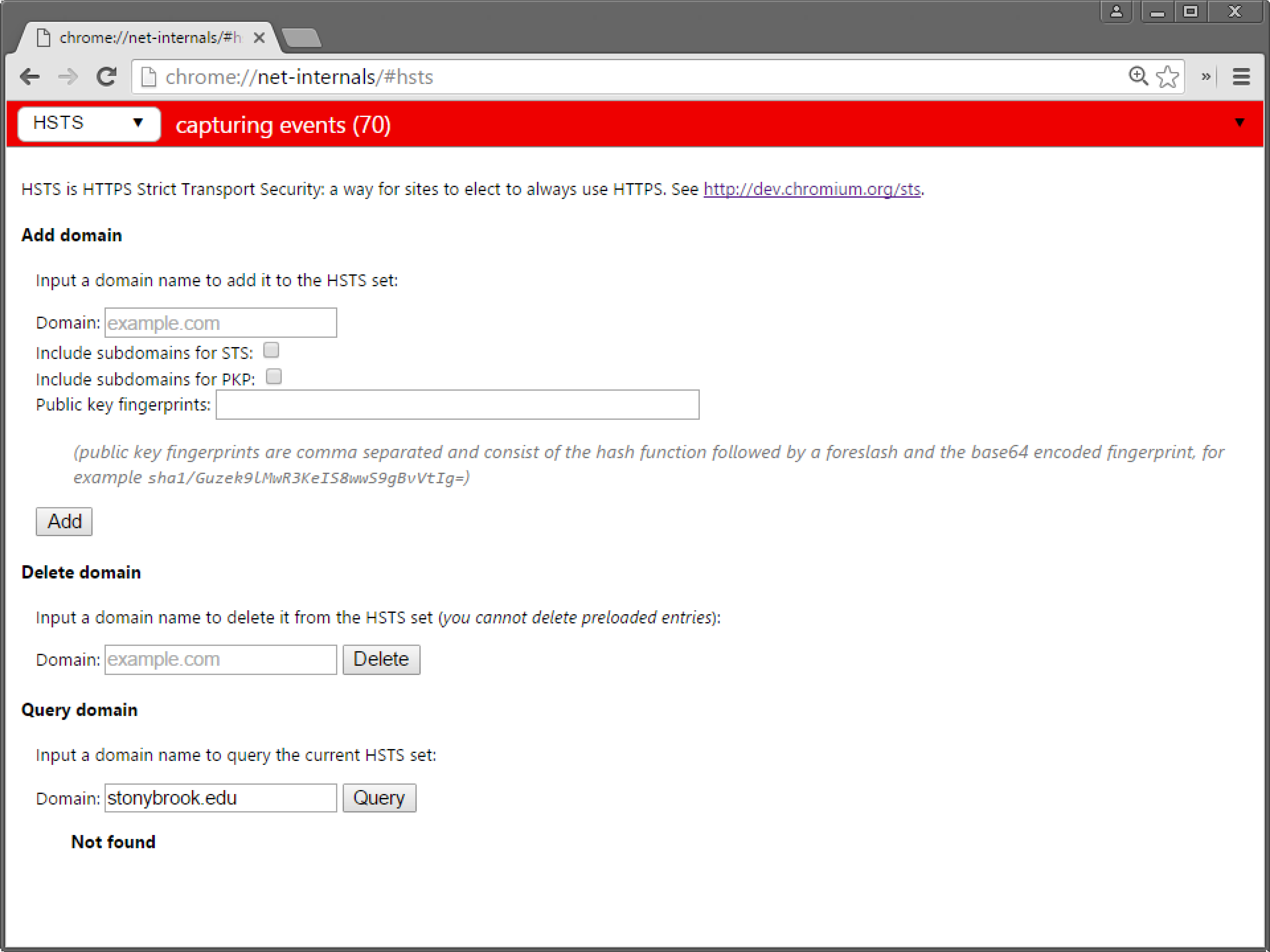
Many incidents involving rogue certificates were discovered after browsers started rolling out pinning

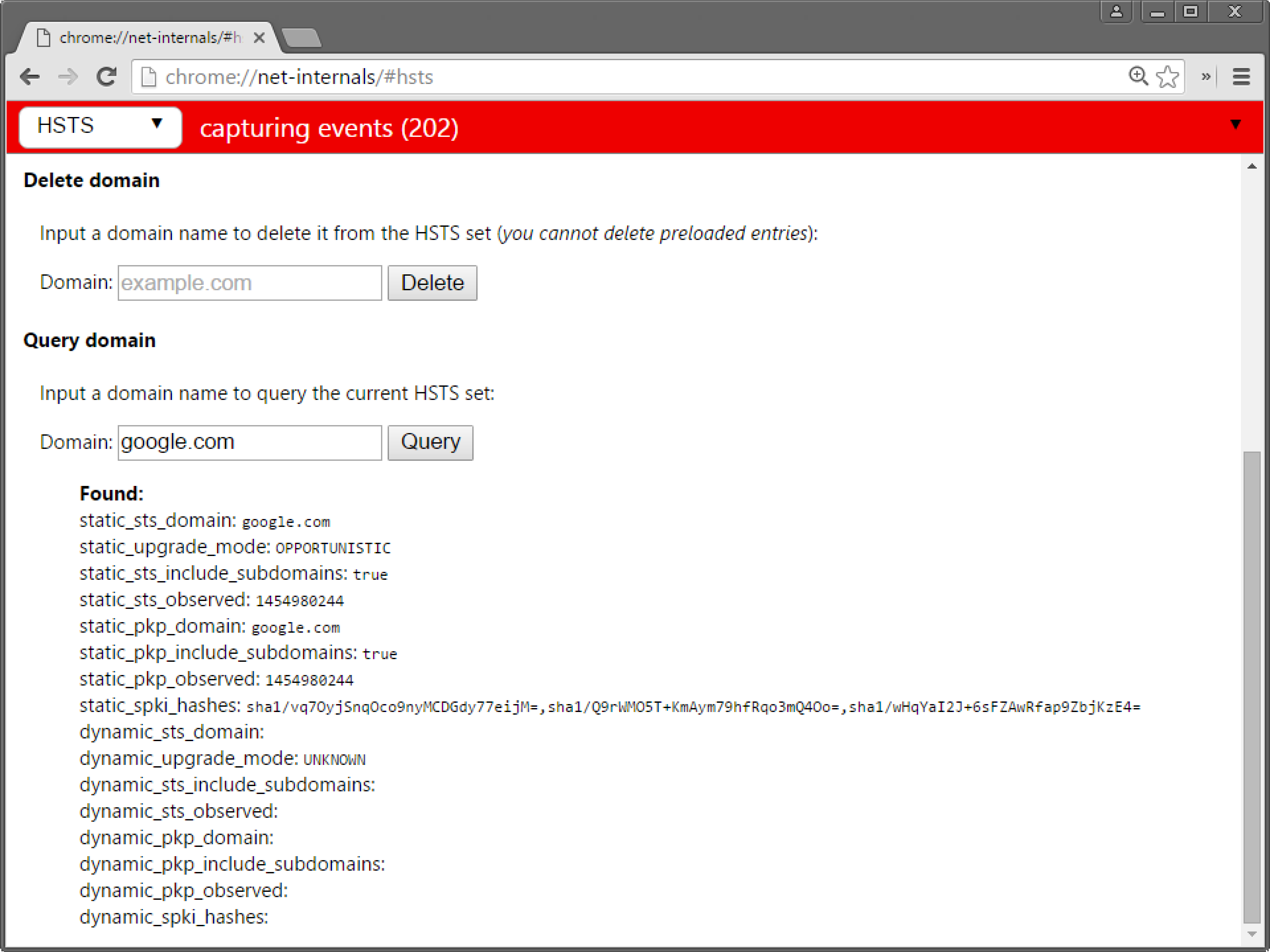
Similar deployment as HSTS

- TOFU: HTTP response header
- Built-in pins in browsers

Must be used very carefully – things can go wrong

- HPKP suicide: site can be bricked if keys are lost/stolen
- RansomPKP: compromise the server and push a malicious HPKP key


chrome://net-internals/#h
chrome://net-internals/#hsts
HSTS
capturing events (70)
HSTS is HTTPS Strict Transport Security: a way for sites to elect to always use HTTPS. See http://dev.chromium.org/sts.
Add domain
Input a domain name to add it to the HSTS set:
Domain: example.com
Include subdomains for STS:
Include subdomains for PKP:
Public key fingerprints:
(public key fingerprints are comma separated and consist of the hash function followed by a foreslash and the base64 encoded fingerprint, for example sha1/Guzek9lMwR3KeIS8wwS9gBvVtIg=)
Add
Delete domain
Input a domain name to delete it from the HSTS set (you cannot delete preloaded entries):
Domain: example.com
Delete
Query domain
Input a domain name to query the current HSTS set:
Domain: stonybrook.edu
Query
Not found


chrome://net-internals/#h
chrome://net-internals/#hsts
HSTS
capturing events (202)
Delete domain
Input a domain name to delete it from the HSTS set (*you cannot delete preloaded entries*):
Domain: example.com
Delete
Query domain
Input a domain name to query the current HSTS set:
Domain: google.com
Query
Found:
static_sts_domain: google.com
static_upgrade_mode: OPPORTUNISTIC
static_sts_include_subdomains: true
static_sts_observed: 1454980244
static_pkp_domain: google.com
static_pkp_include_subdomains: true
static_pkp_observed: 1454980244
static_spki_hashes: sha1/vq7OyjSnqOco9nyMCDGdy77eijM=,sha1/Q9rWMO5T+KmAym79hfRqo3mQ4Oo=,sha1/wHqYaI2J+6sFZAwRfap9ZbjKzE4=
dynamic_sts_domain:
dynamic_upgrade_mode: UNKNOWN
dynamic_sts_include_subdomains:
dynamic_sts_observed:
dynamic_pkp_domain:
dynamic_pkp_include_subdomains:
dynamic_pkp_observed:
dynamic_spki_hashes:

Google Online Security Bl

https://googleonlinesecurity.blogspot.com/2013/01/enhancing-digital-certificate-security.html

Google Security Blog

The latest news and insights from Google on security and safety on the Internet

Enhancing digital certificate security

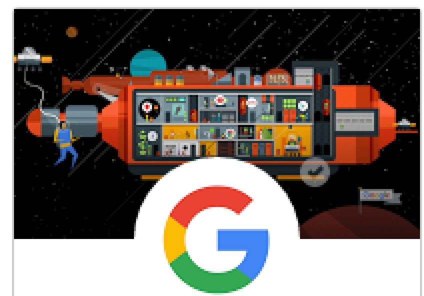
January 3, 2013

Posted by Adam Langley, Software Engineer

Late on December 24, Chrome detected and blocked an unauthorized digital certificate for the "*.google.com" domain. We investigated immediately and found the certificate was issued by an intermediate certificate authority (CA) linking back to TURKTRUST, a Turkish certificate authority. Intermediate CA certificates carry the full authority of the CA, so anyone who has one can use it to create a certificate for any website they wish to impersonate.

In response, we updated Chrome's certificate revocation metadata on

Search blog ...



Google

google.com/+google

News and updates on Google's products, technology and more

Follow +1

+ 11,007,947

Certificate Revocation

Mechanism to allow revocation of compromised or no longer needed certificates

Certificate revocation list (CRL)

- Signed list of all serial numbers belonging to revoked certificates that have not yet expired
- Main problem: lists tend to be large, making real-time lookups slow
- Can the attacker block connectivity to the status server?

Online Certificate Status Protocol (OCSP)

- Obtain the revocation status of a *single* certificate ➔ faster
- But performance and privacy issues still remain
- **OCSP stapling:** server embeds OCSP response directly into the TLS handshake

Certificate Transparency

Public monitoring and auditing of certificates

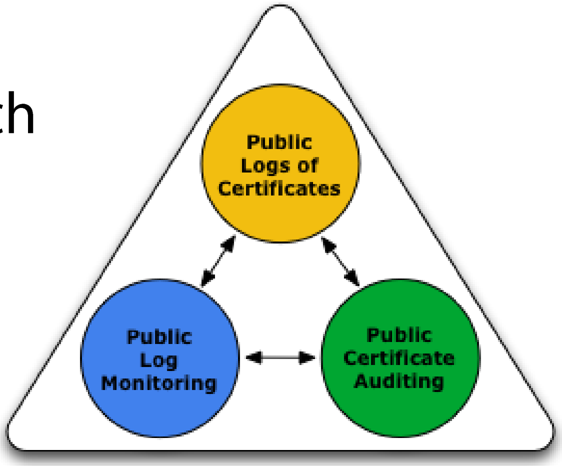
Identify mistakenly or maliciously issued certs and rogue CAs

Certificate logs

Network services maintaining cryptographically assured, publicly auditable, append-only records of certificates

Monitors

Periodically contact all log servers and watch for suspicious certificates



Auditors

Verify that logs are behaving correctly and are cryptographically consistent

Check that a particular certificate appears in a log