CSE508 Network Security

Stony Brook University

2021-03-23

# 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, DoT: 853, …

# History

NETSCAPE®

## SSL developed at Netscape

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

## 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 (2018): removal of legacy/weak algorithms, lower latency, perfect forward secrecy, …

## Endless cycle of vulnerabilities and improvements

Insecure renegotiation, RC4 weaknesses, compression side channels, padding oracle attacks, buggy implementations, PKI attacks, …

*BEAST, CRIME, TIME, Lucky 13, BREACH, POODLE, FREAK, Heartbleed, DROWN, …*

**Handshake protocol**

Negotiate public key crypto algorithms and establish shared secret keys

Authentication (server and optionally client)

Up to TLS 1.2, took 6–10 messages, depending on features used

**Record Protocol**

Uses the established secret keys to protect the transmitted data

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

*Encryption and integrity:* after handshake completion

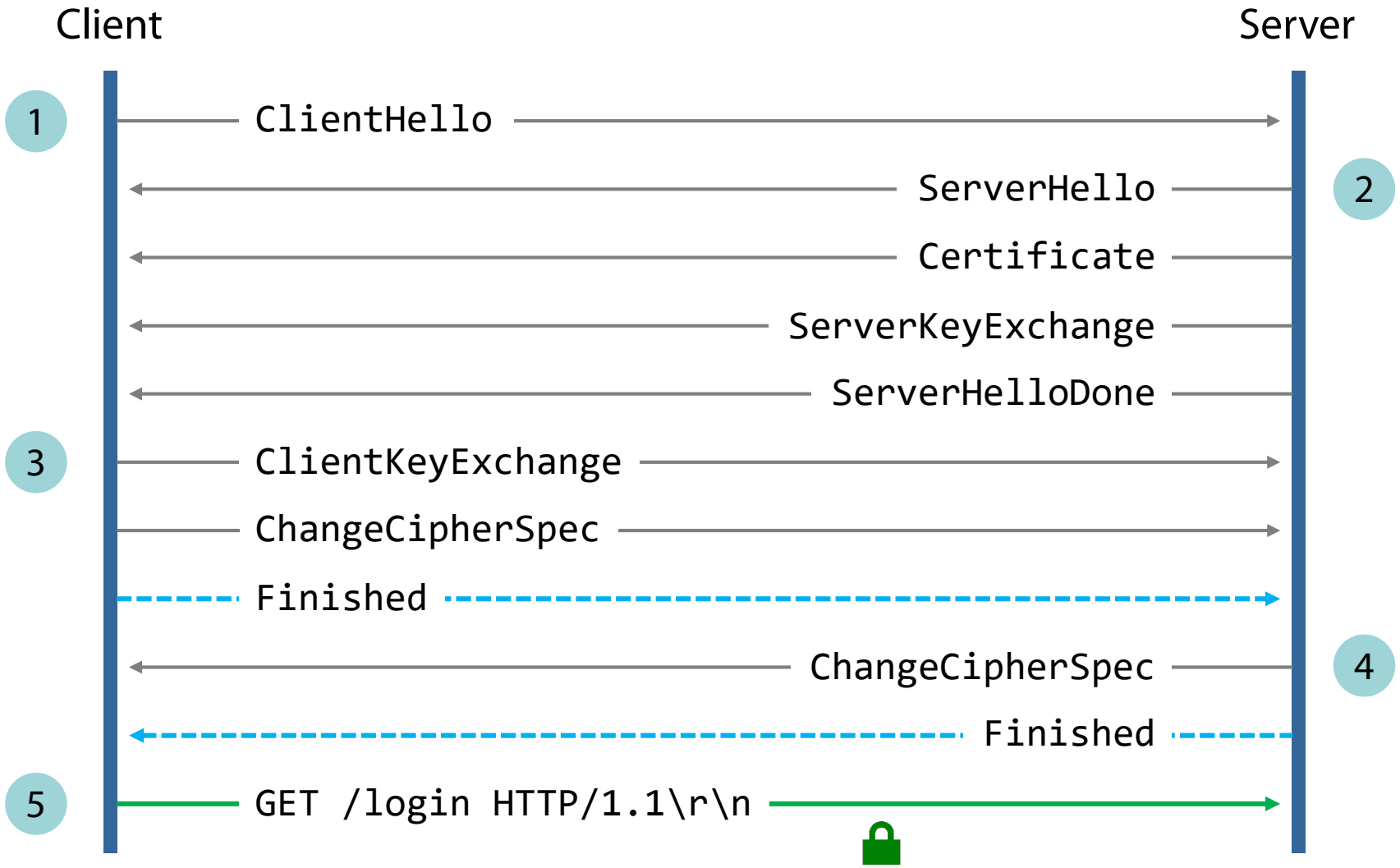
*Compression:* before encryption… not a good idea

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
...
```

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

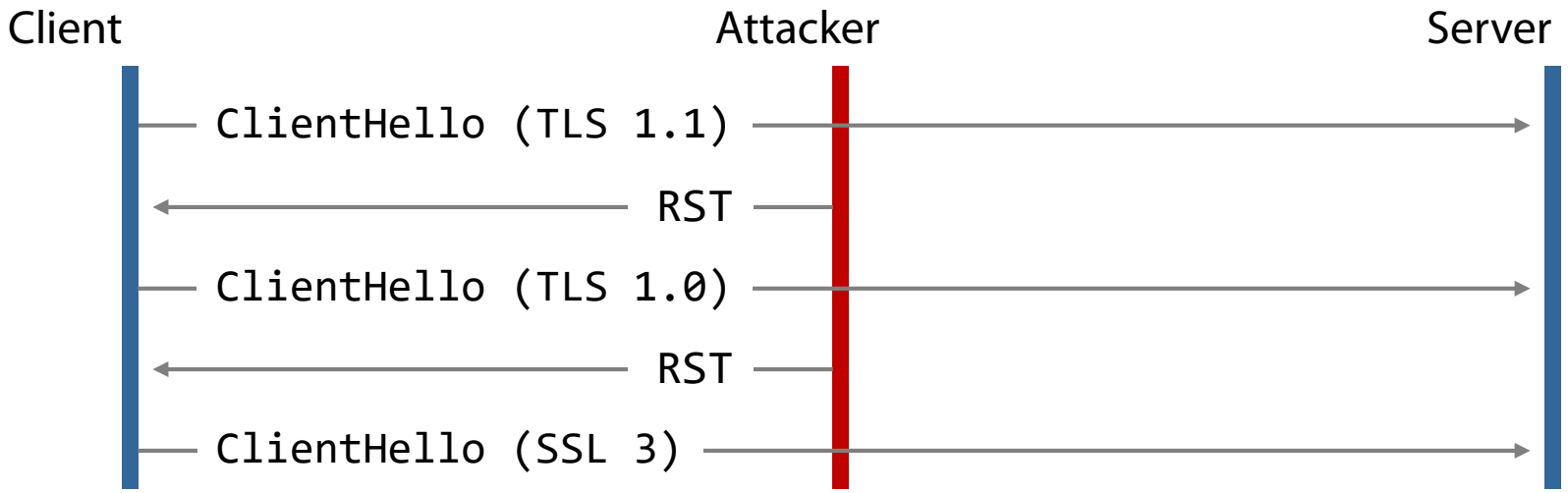
# Downgrade Attacks

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 alter the client's list of supported suites



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 (deprecated) | TLS 1.1 (deprecated) | TLS 1.2 | TLS 1.3 | EV [n 3][70] | SHA-2 [71] | ECDSA [72] | BEAST[n 4] | CRIME[n 5] | POODLE (SSLv3)[n 6] | RC4[n 7] | FREAK[73][74] | Logjam | |
| Google Chrome (Chrome for Android) [n 8] [n 9] | 1–9 | Windows (7+) macOS (10.11+) Linux Android (5.0+) iOS (12.2+) Chrome OS | Disabled by default | Enabled by default | Yes | No | No | No | Yes (only desktop) | needs SHA-2 compatible OS[71] | needs ECC compatible OS[72] | Not affected [79] | Vulnerable (HTTPS) | Vulnerable | Vulnerable | Vulnerable (except Windows) | Vulnerable | Yes[n 10] |
| | 10–20 | | No[80] | Enabled by default | Yes | No | No | No | Yes (only desktop) | needs SHA-2 compatible OS[71] | needs ECC compatible OS[72] | 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[71] | needs ECC compatible OS[72] | Not affected | Mitigated [81] | Vulnerable | Vulnerable | Vulnerable (except Windows) | Vulnerable | Yes[n 10] |
| | 22–29 | | No | Enabled by default | Yes | Yes[82] | No[82][83][84][85] | No | Yes (only desktop) | needs SHA-2 compatible OS[71] | needs ECC compatible OS[72] | Not affected | Mitigated | Vulnerable | Vulnerable | Vulnerable (except Windows) | Vulnerable | Temporary [n 11] |
| | 30–32 | | No | Enabled by default | Yes | Yes | Yes[83][84][85] | No | Yes (only desktop) | needs SHA-2 compatible OS[71] | needs ECC compatible OS[72] | 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[71] | needs ECC compatible OS[72] | Not affected | Mitigated | Partly mitigated [n 12] | Lowest priority [88][89][90] | 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[72] | Not affected | Mitigated | Partly mitigated | Lowest priority | Vulnerable (except Windows) | Vulnerable | Temporary [n 11] |
| | 40 | | No | Disabled by default [87][91] | Yes | Yes | Yes | No | Yes (only desktop) | Yes | needs ECC compatible OS[72] | 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[72] | 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[72] | Not affected | Mitigated | Mitigated | Only as fallback [n 15][92] | Mitigated | Vulnerable | Yes[n 14] |
| | 44–47 | | No | No[93] | Yes | Yes | Yes | No | Yes (only desktop) | Yes | needs ECC compatible OS[72] | Not affected | Mitigated | Not affected | Only as fallback [n 15] | Mitigated | Mitigated[94] | Temporary [n 11] |
| | 48, 49 | | No | No | Yes | Yes | Yes | No | Yes (only desktop) | Yes | needs ECC compatible OS[72] | Not affected | Mitigated | Not affected | Disabled by default[n 16][95][96] | 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][95][96] | Mitigated | Mitigated | Temporary [n 11] |
| | 54–66 | | No | No | Yes | Yes | Yes | Disabled by default (draft version) | Yes (only desktop) | Yes | Yes | Not affected | Mitigated | Not affected | Disabled by default[n 16][95][96] | Mitigated | Mitigated | Temporary [n 11] |
| | 67–69 | | No | No | Yes | Yes | Yes | Yes (draft version) | Yes (only desktop) | Yes | Yes | Not affected | Mitigated | Not affected | Disabled by default[n 16][95][96] | Mitigated | Mitigated | Temporary [n 11] |
| | 70–83 | | No | No | Yes | Yes | Yes | Yes | Yes (only desktop) | Yes | Yes | Not affected | Mitigated | Not affected | Disabled by default[n 16][95][96] | Mitigated | Mitigated | Temporary [n 11] |
| | 84–88 89 | | No | No | Warn by default | Warn by default | Yes | Yes | Yes (only desktop) | Yes | Yes | Not affected | Mitigated | Not affected | Disabled by default[n 16][95][96] | Mitigated | Mitigated | Temporary [n 11] |
| | 91[97] | | No | No | No | No | Yes | Yes | Yes (only desktop) | Yes | Yes | Not affected | Mitigated | Not affected | Disabled by default[n 16][95][96] | Mitigated | Mitigated | Temporary [n 11] |

*SSL 3.0, TLS 1.0, and TLS 1.1 are now completely removed by most browsers*

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

## TLS 1.2 Session Resumption

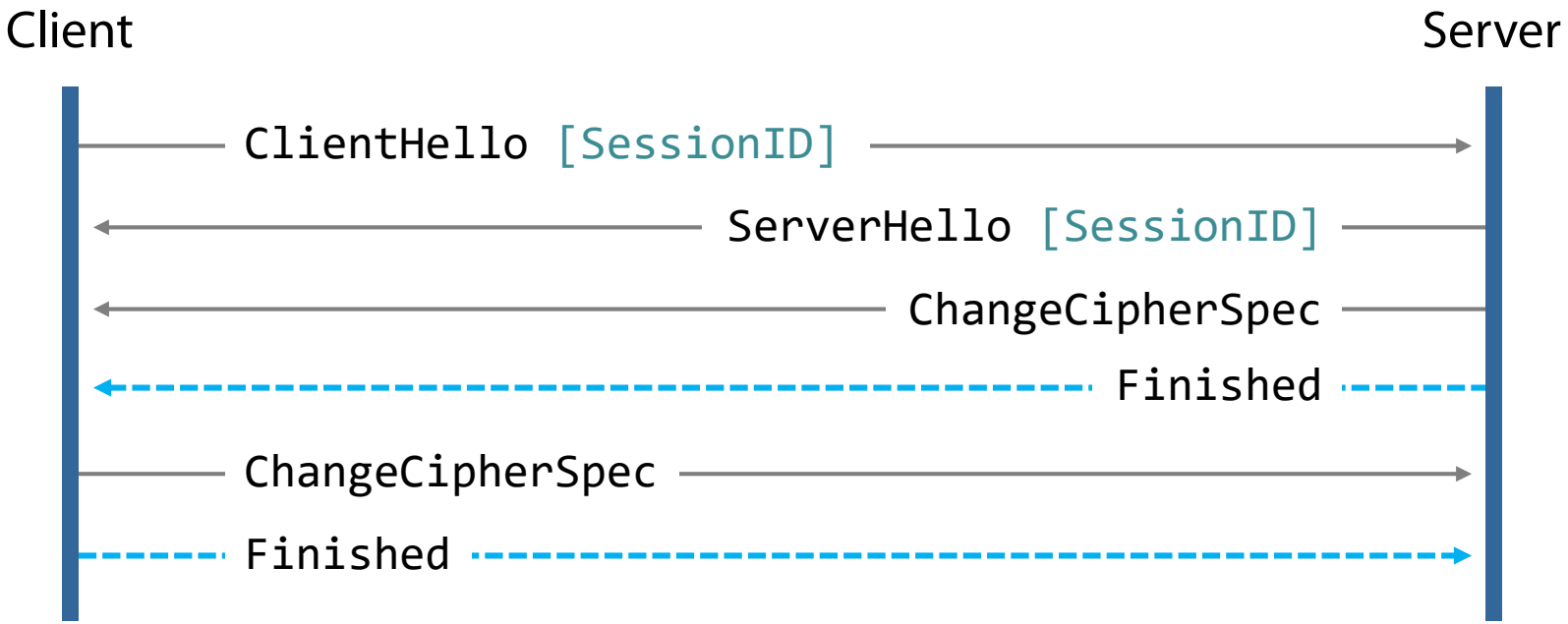
Full handshake: 6-10 messages and two network round-trips

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

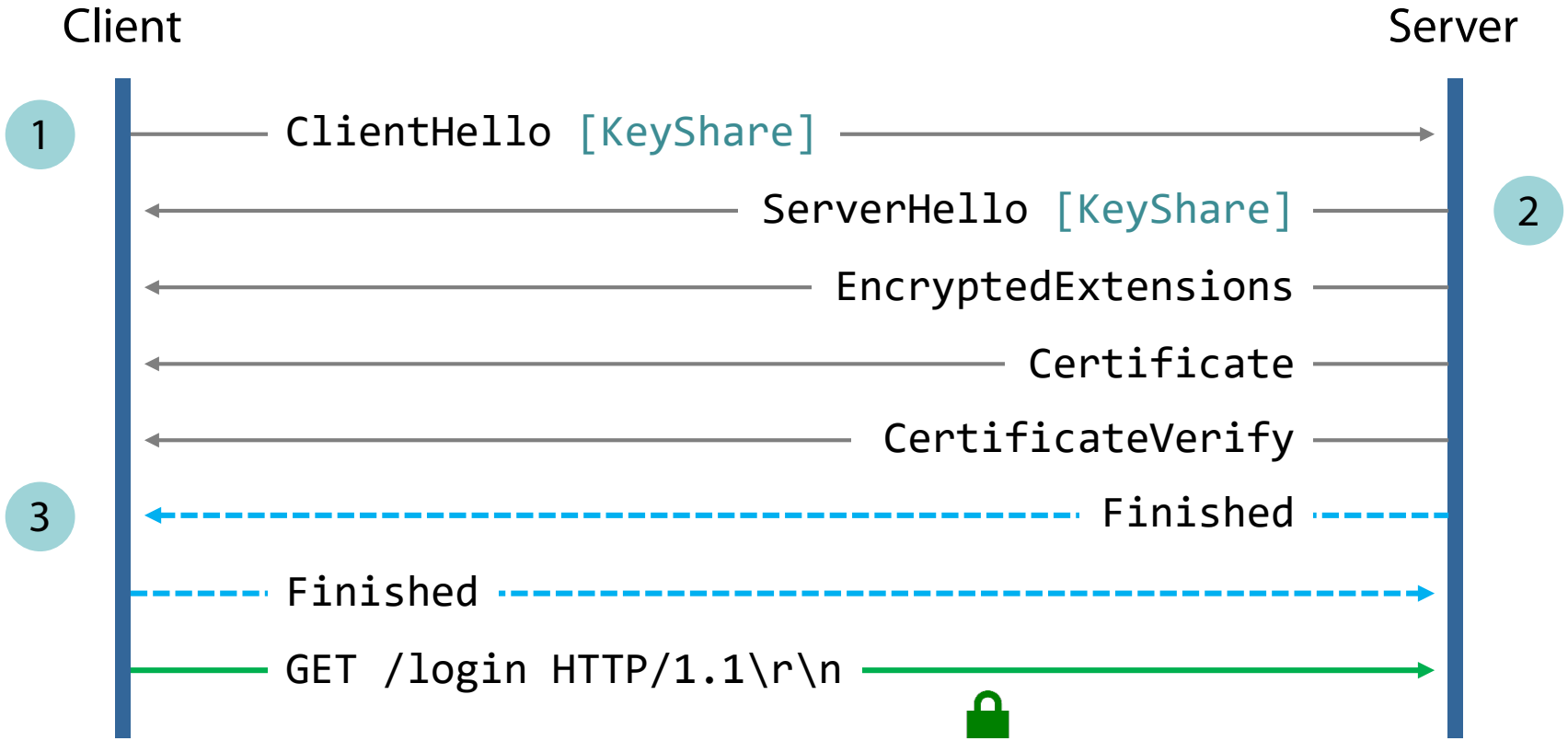
Avoid re-negotiation by remembering security parameters

- 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 messages based on config. ID previously sent by 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 certificate authority

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*

The same process is supported for authenticating clients

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

Rarely used in practice for user authentication

- 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 cert

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

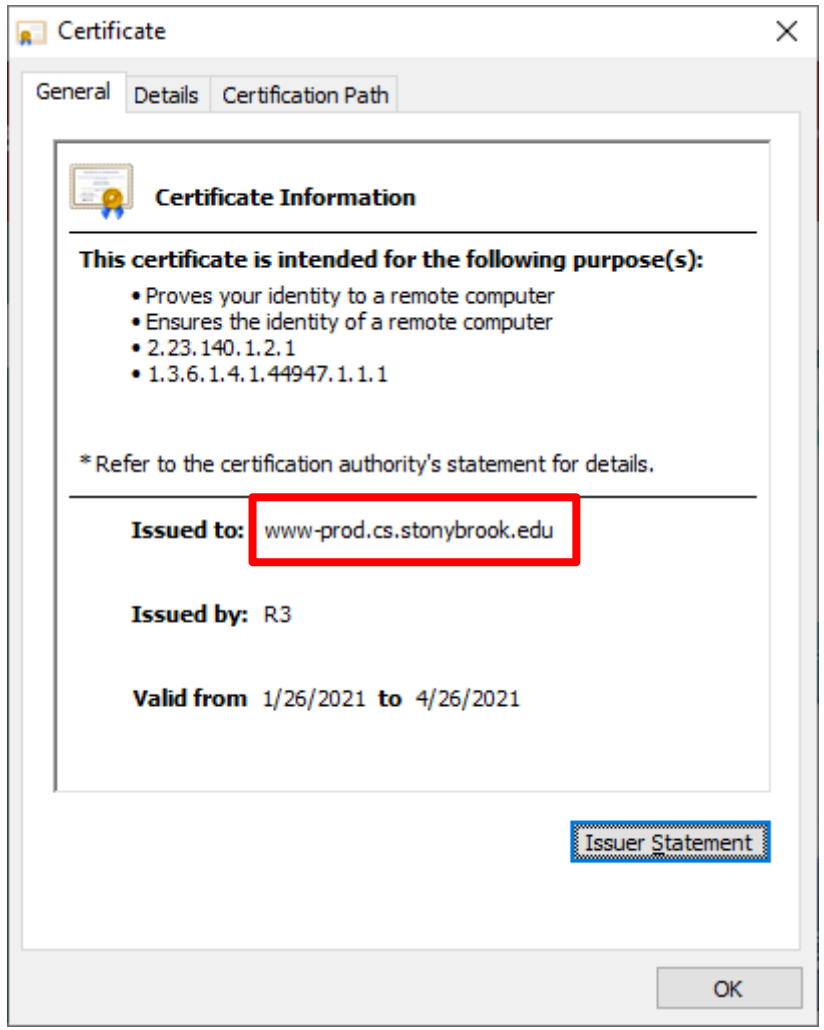
*Validity:* starting and ending date of validity period

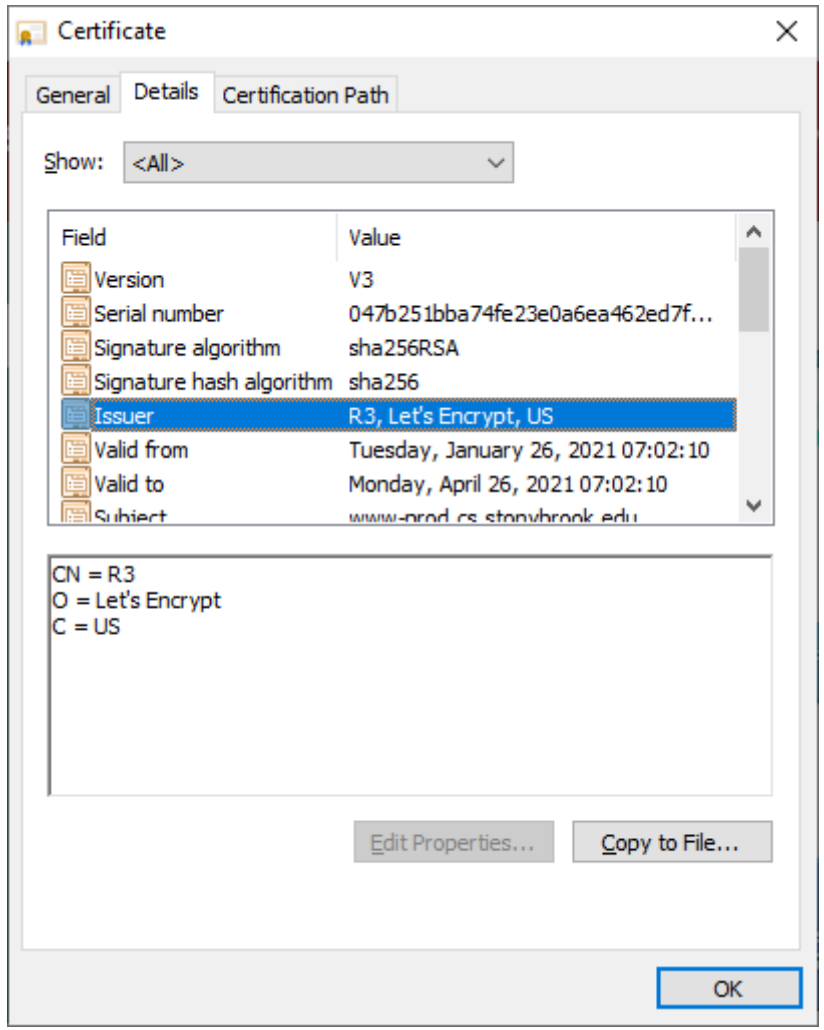
*Subject:* DN of the entity associated with the certificate's public key

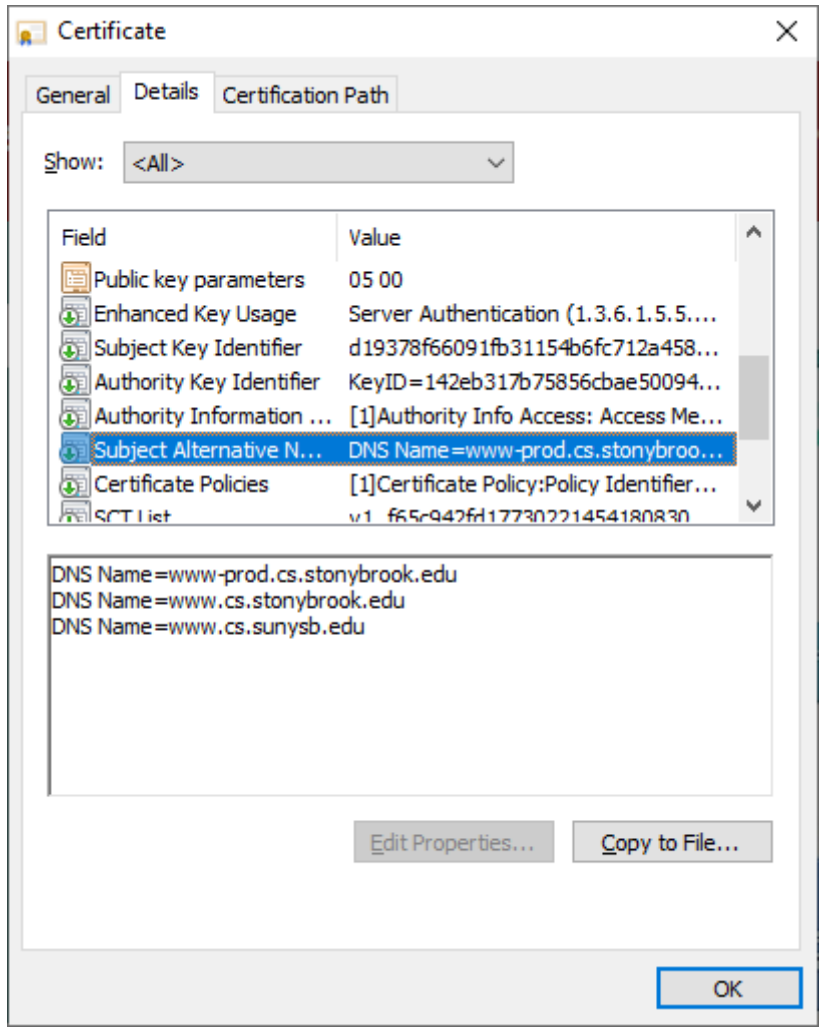
> Deprecated in favor of the Subject Alternative Name (SAN) extension: DNS name, IP address, or URI (also supports binding to multiple identities)

*Public Key:* The subject's public key

*Signature*


Certificate
General
Details
Certification Path
Certificate Information
This certificate is intended for the following purpose(s):
• Proves your identity to a remote computer
• Ensures the identity of a remote computer
• 2.23.140.1.2.1
• 1.3.6.1.4.1.44947.1.1.1
* Refer to the certification authority's statement for details.
Issued to: www-prod.cs.stonybrook.edu
Issued by: R3
Valid from 1/26/2021 to 4/26/2021
Issuer Statement
OK


Certificate
General
Details
Certification Path
Show:
<All>
Field
Value
Version
V3
Serial number
047b251bba74fe23e0a6ea462ed7f...
Signature algorithm
sha256RSA
Signature hash algorithm
sha256
Issuer
R3, Let's Encrypt, US
Valid from
Tuesday, January 26, 2021 07:02:10
Valid to
Monday, April 26, 2021 07:02:10
Subject
www-prod.cs.stonybrook.edu
CN = R3
O = Let's Encrypt
C = US
Edit Properties...
Copy to File...
OK


Certificate
General
Details
Certification Path
Show:
<All>
Field
Value
Public key parameters
05 00
Enhanced Key Usage
Server Authentication (1.3.6.1.5.5....
Subject Key Identifier
d19378f66091fb31154b6fc712a458...
Authority Key Identifier
KeyID=142eb317b75856cbae50094...
Authority Information ...
[1]Authority Info Access: Access Me...
Subject Alternative N...
DNS Name=www-prod.cs.stonybroo...
Certificate Policies
[1]Certificate Policy:Policy Identifier...
DNS Name=www-prod.cs.stonybrook.edu
DNS Name=www.cs.stonybrook.edu
DNS Name=www.cs.sunysb.edu
Edit Properties...
Copy to File...
OK

# Certificate Chains

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

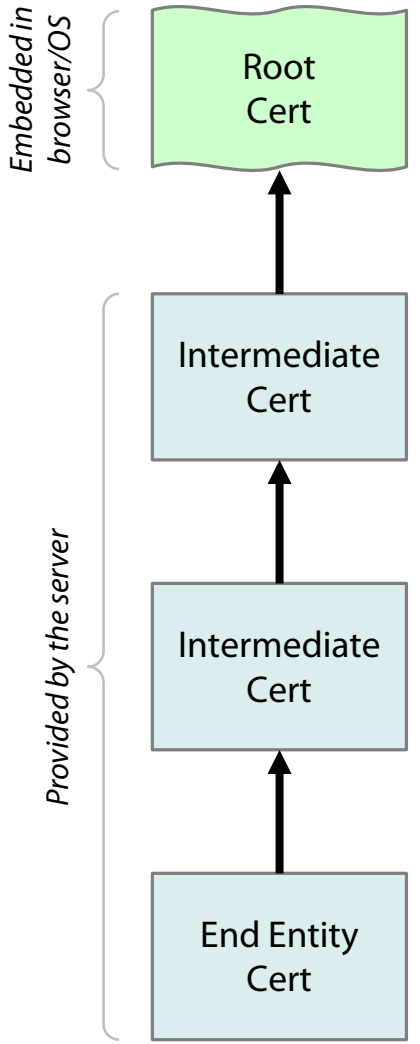
- System/public store: used by OS, browsers, …
- More can be added in the local/private cert store: vendor-specific certs, MitM certs for content inspection filters/AVs, …

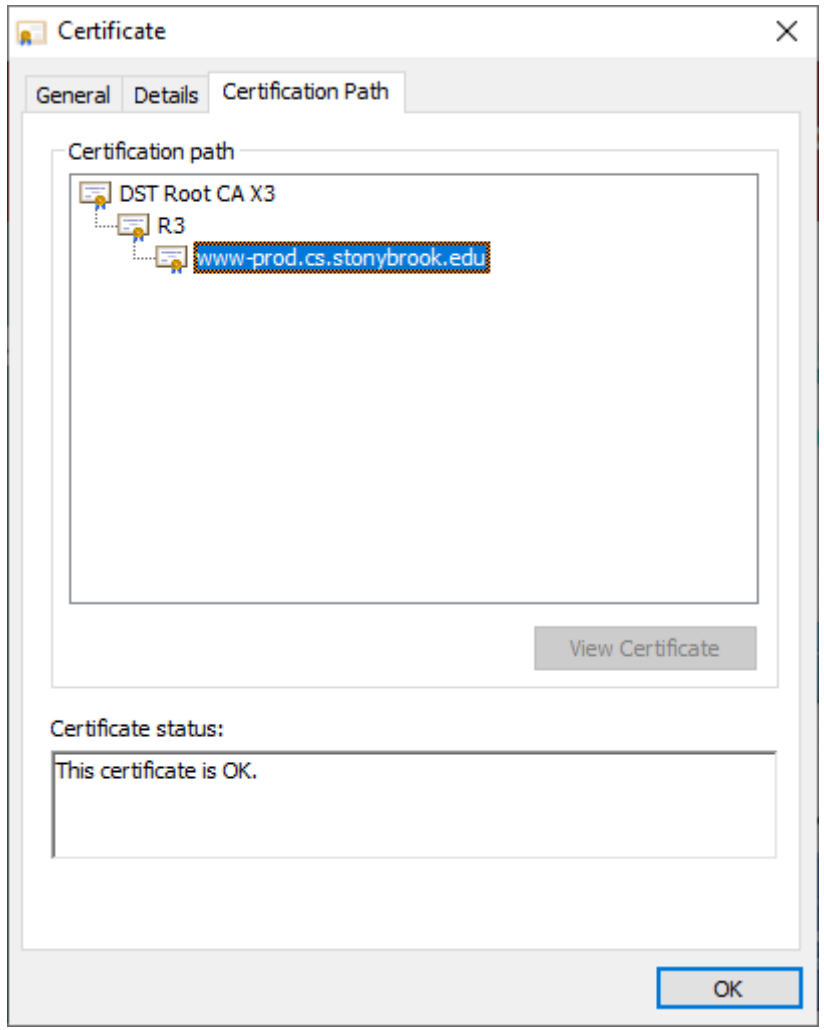
Server provides a *chain* of certificates

- 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 CA can issue and sign certificates for any subject

- The system is only as secure as the weakest certificate authority…
- Certificate Authority Authorization (CAA): can be used to restrict which CAs can issue certificates for a particular domain




Certificate
General
Details
Certification Path
Certification path
DST Root CA X3
R3
www-prod.cs.stonybrook.edu
View Certificate
Certificate status:
This certificate is OK.
OK

# Certificate Revocation

Allow revocation of compromised or no longer needed certificates

Certificate revocation list (CRL)

- Signed list of all 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 CA's server?
- CRLSets (Chrome): revocation list pushed to the browser as a *software update*

Online Certificate Status Protocol (OCSP)

- Obtain the revocation status of a *single* certificate ➔ faster
- But the latency, security, and privacy issues still remain
- OCSP stapling (Firefox): server embeds OCSP response directly into the TLS handshake (soft-fail issue remains: an adversary can suppress the OCSP response)

# HTTPS

Most common use of TLS: *most 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

Needs expertise and certs cost $$$$

- **Not** anymore: letsencrypt.org

# Firesheep In Wolves' Clothing: Extension Lets You Hack Into Twitter, Facebook Accounts Easily

**Contributor** 11:24 PM EST • October 24, 2010

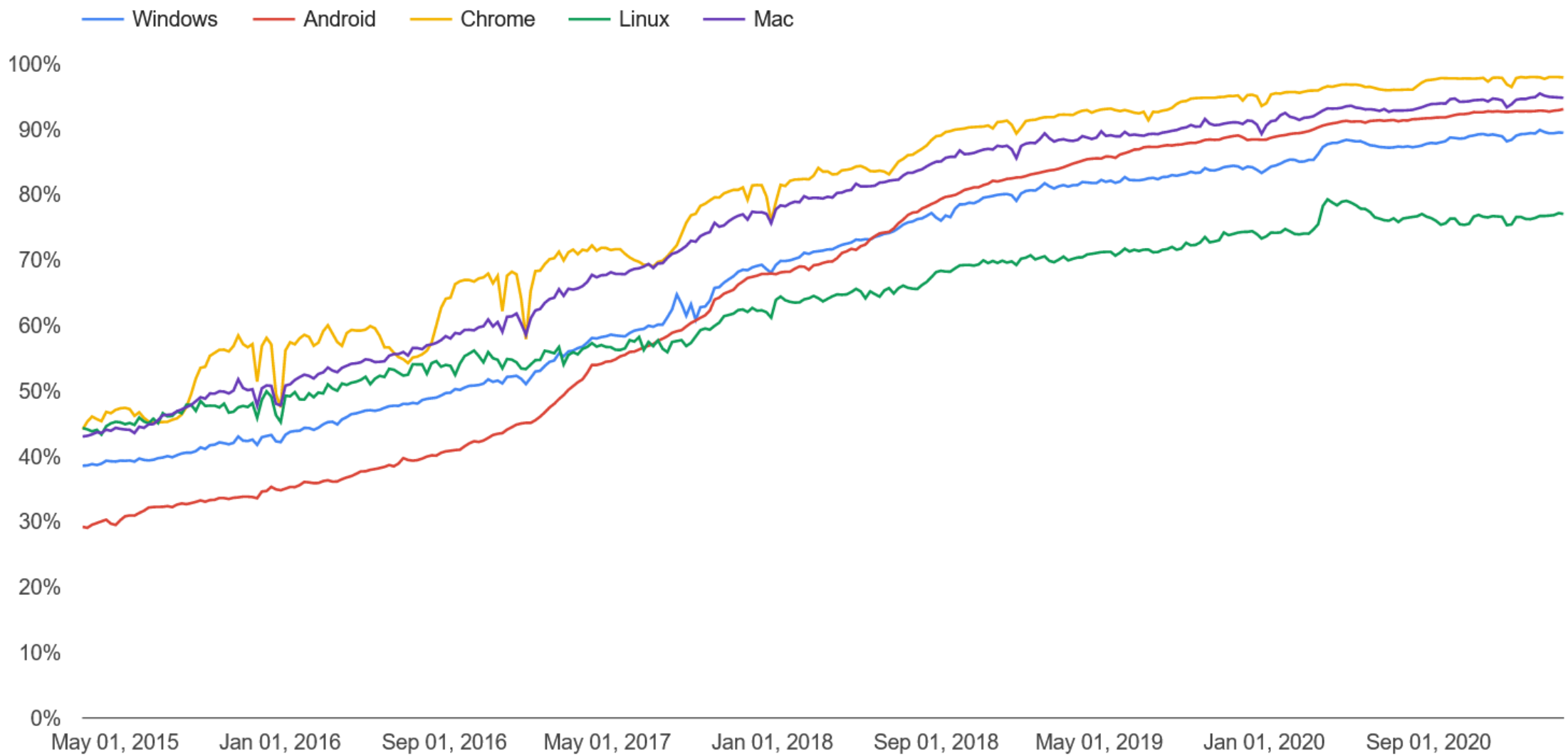
Comment


It seems like every time Facebook amends its privacy policy, the web is up in arms. The truth is, Facebook's well publicized privacy fight is nothing compared to the vulnerability of all unsecured HTTP sites — that includes Facebook, Twitter and many of the web's most popular destinations.

Developer Eric Butler has exposed the soft underbelly of the web with his new Firefox extension, Firesheep, which will let you essentially eavesdrop on any open Wi-Fi network and capture users' cookies.

As Butler explains in his post, "As soon as anyone on the network visits an insecure website known to Firesheep, their name and photo will be displayed" in the window. All you have to do is double click on

Percentage of pages loaded over HTTPS in Chrome by platform




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

# Browser Security Indicators

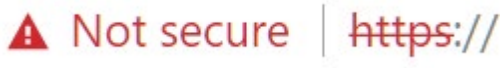
Convey information about the security of a page

- Locks, shields, key icons, green bars…

*"This page was fetched using SSL"*

- Page content was not viewable to a network adversary
- Certificate is valid (e.g. not expired, signed by a CA trusted by the browser, and the subject name matches the URL)


*"This page uses an invalid certificate"*

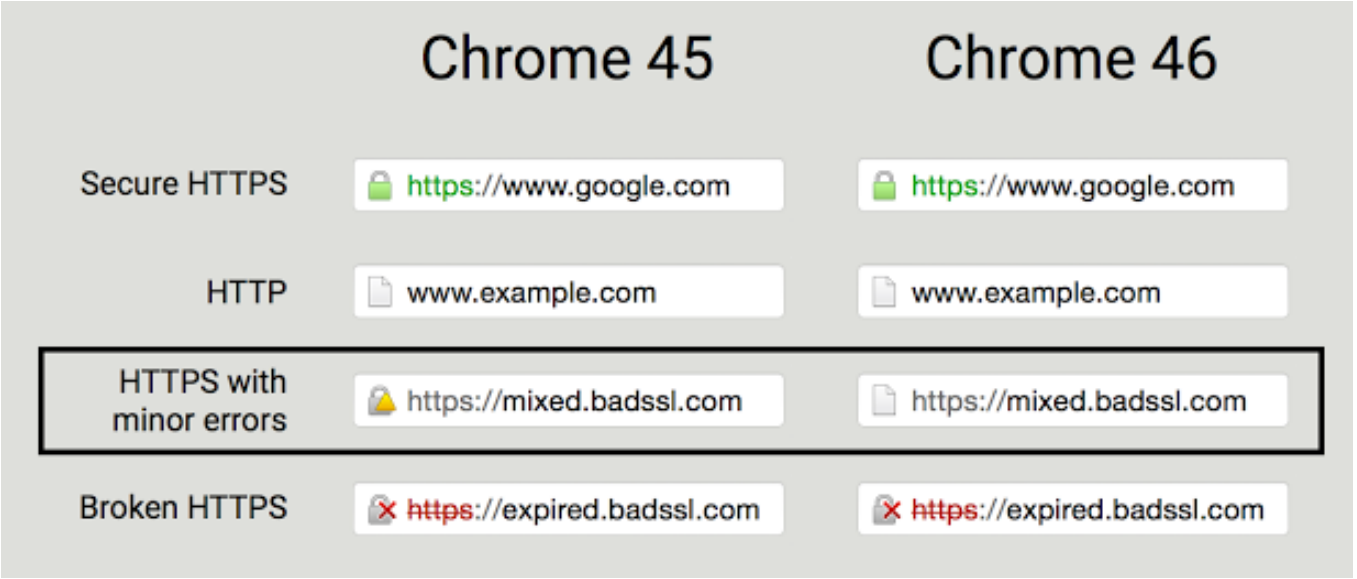



*"Parts of the page could not be encrypted"*


*"The legal entity operating this web site is known"*

- Extended Validation (EV) certificates

# Mixed Content Warning is Unnecessary




*Basically the same in terms of security*

Fewer security states for users to remember

Reflects better the security state of the page

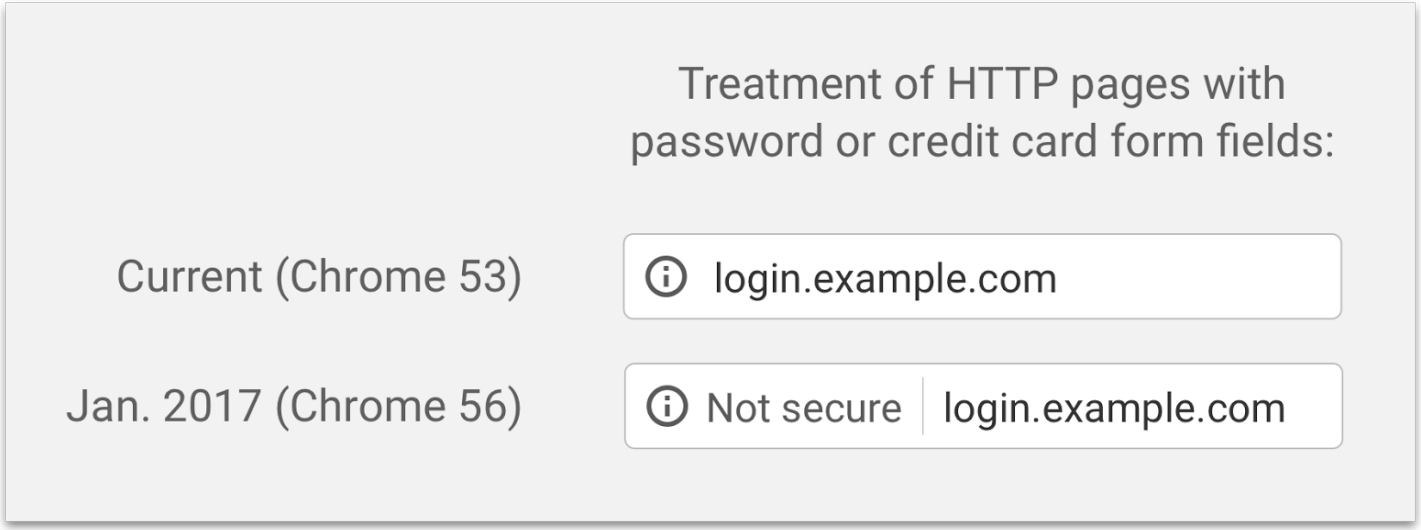
**Non-HTTPS traffic is a vulnerability!** MitM/MotS attacks on the HTTP part are trivial

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

# Marking HTTP as Not Secure

Phase 1: page is marked "Not secure" when

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

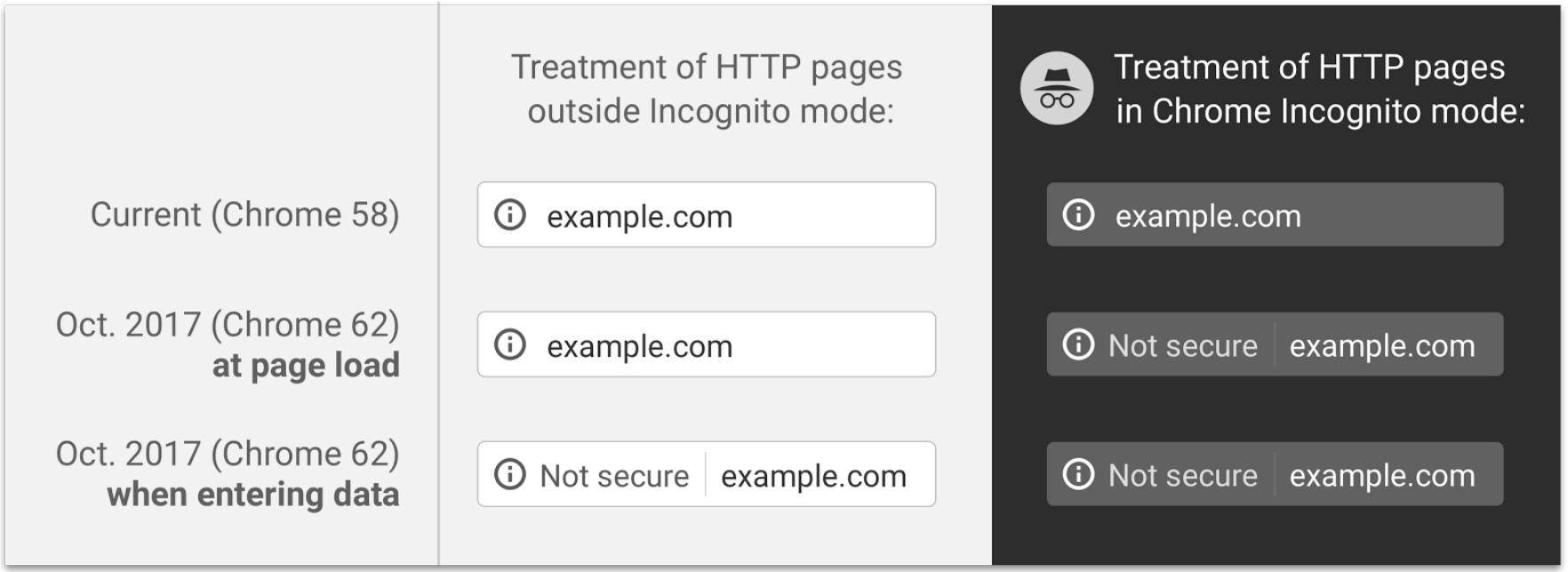



https://www.chromium.org/Home/chromium-security/marking-http-as-non-secure

# Marking HTTP as Not 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 *incognito mode*




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

# Marking HTTP as Not Secure

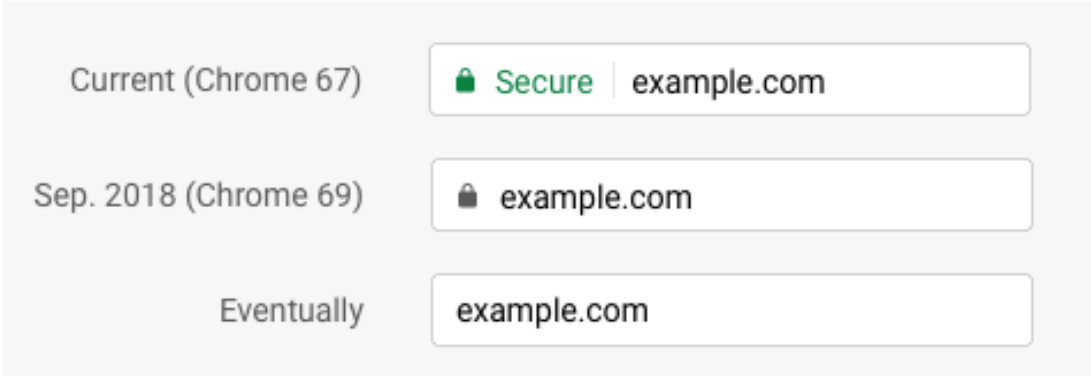
Phase 3:  all plain HTTP pages are marked “Not secure”


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

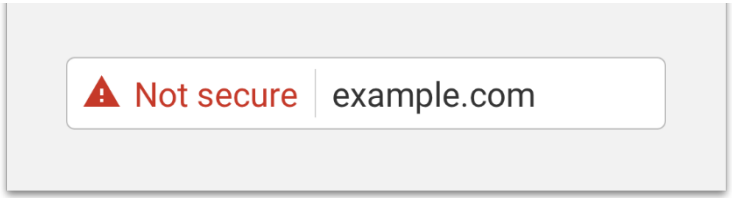
# Marking HTTP as Not Secure

Current state: HTTPS pages are marked in a more neutral way, while HTTP pages are affirmatively marked “Not secure”

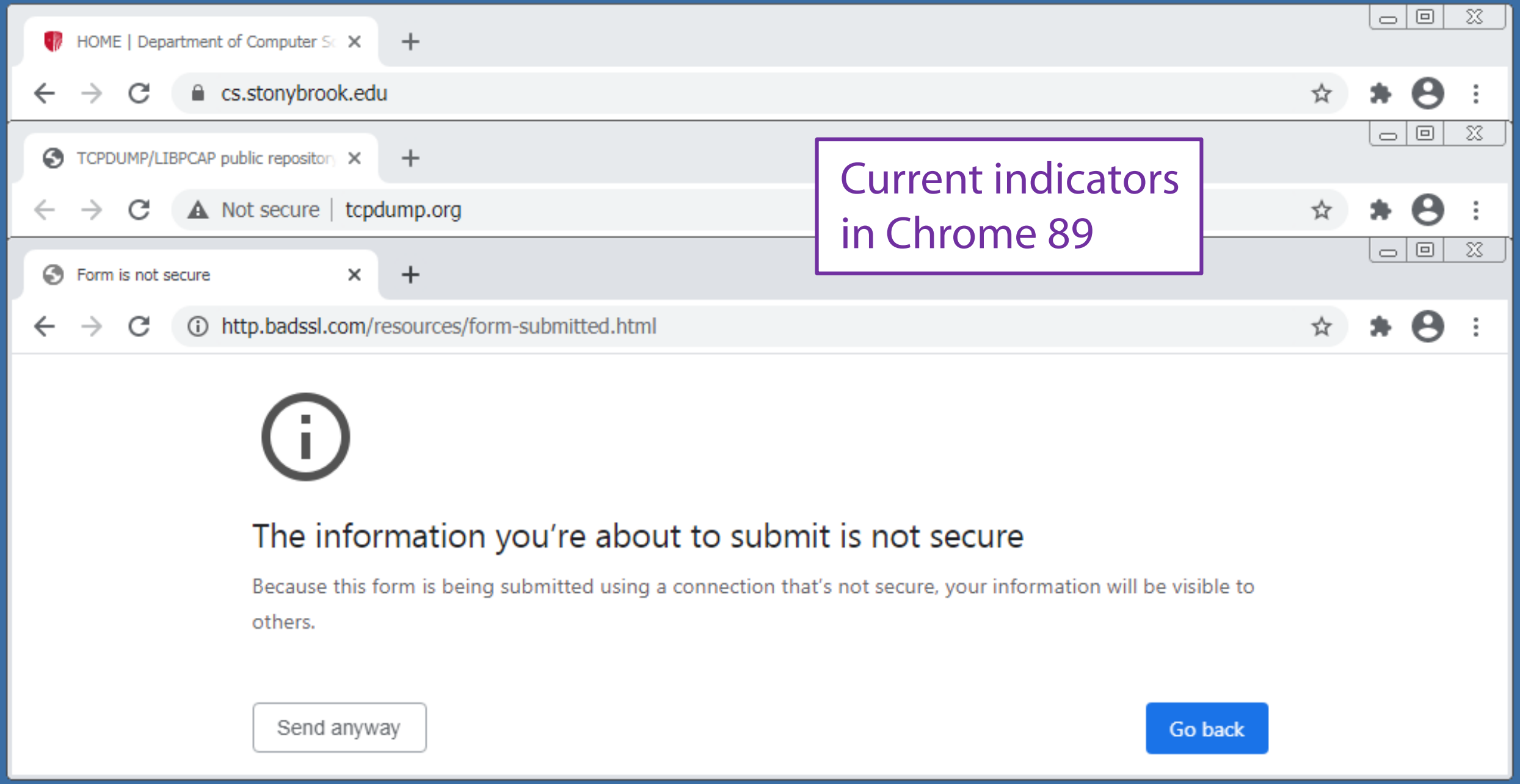
**HTTPS**




**HTTP**




https://blog.chromium.org/2018/05/evolving-chromes-security-indicators.html




Current indicators in Chrome 89

# SSL stripping

## Browsing sessions often start with a plain HTTP request

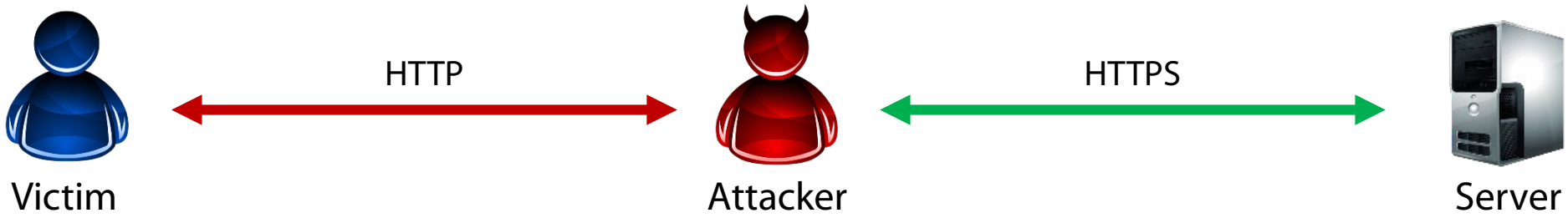
- Web sites used to switch to HTTPS only for login or checkout
- Example: Facebook in 2010 (optional full HTTPS in 2011, HTTPS by default in 2013)
- Users type addresses without specifying `https://`
- Browser connects over HTTP *by default* ➔ site may redirect to HTTPS

## 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 or different domain, but who is going to notice?*

## **HSTS** (HTTP Strict Transport Security)

Defense against SSL stripping and other similar issues

- Force the use of HTTPS instead of HTTP *before* accessing a resource
- Treat all errors (e.g., invalid certificate, mixed content, plain HTTP) as fatal: do not allow users to access the web page

Servers implement HSTS policies 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)

- Problem: the initial request *remains unprotected* because it is sent over HTTP
- **HSTS preloading:** browsers come preloaded with a list of known HSTS sites

```
780
781     // START OF LEGACY MANUAL CUSTOM ENTRIES
782     { "name": "www.paypal.com", "policy": "custom", "mode": "force-https" },
783     { "name": "paypal.com", "policy": "custom", "mode": "force-https" },
784     { "name": "www.elanex.biz", "policy": "custom", "mode": "force-https" },
785     { "name": "www.noisebridge.net", "policy": "custom", "mode": "force-https" },
786     { "name": "neg9.org", "policy": "custom", "mode": "force-https" },
787     { "name": "factor.cc", "policy": "custom", "mode": "force-https" },
788     { "name": "aladdinschools.appspot.com", "policy": "custom", "mode": "force-https" },
789     { "name": "www.paycheckrecords.com", "policy": "custom", "mode": "force-https" },
790     { "name": "lastpass.com", "policy": "custom", "mode": "force-https" },
791     { "name": "www.lastpass.com", "policy": "custom", "mode": "force-https" },
792     { "name": "entropia.de", "policy": "custom", "mode": "force-https" },
793     { "name": "www.entropia.de", "policy": "custom", "mode": "force-https" },
794     { "name": "logentries.com", "policy": "custom", "mode": "force-https" },
795     { "name": "www.logentries.com", "policy": "custom", "mode": "force-https" },
796     { "name": "dropcam.com", "policy": "custom", "mode": "force-https" },
797     { "name": "www.dropcam.com", "policy": "custom", "mode": "force-https" },
798     { "name": "epoxate.com", "policy": "custom", "mode": "force-https" },
799     { "name": "torproject.org", "policy": "custom", "mode": "force-https", "pins": "tor" },
800     { "name": "blog.torproject.org", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "tor" },
801     { "name": "check.torproject.org", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "tor" },
802     { "name": "www.torproject.org", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "tor" },
803     { "name": "dist.torproject.org", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "tor" },
804     { "name": "ledgerscope.net", "policy": "custom", "mode": "force-https" },
805     { "name": "www.ledgerscope.net", "policy": "custom", "mode": "force-https" },
806     { "name": "greplin.com", "policy": "custom", "mode": "force-https" },
807     { "name": "www.greplin.com", "policy": "custom", "mode": "force-https" },
```

# Firefox 83 introduces HTTPS-Only Mode

**Christoph Kerschbaumer, Julian Gaibler, Arthur Edelstein and Thyla van der Merwe** | **November 17, 2020**

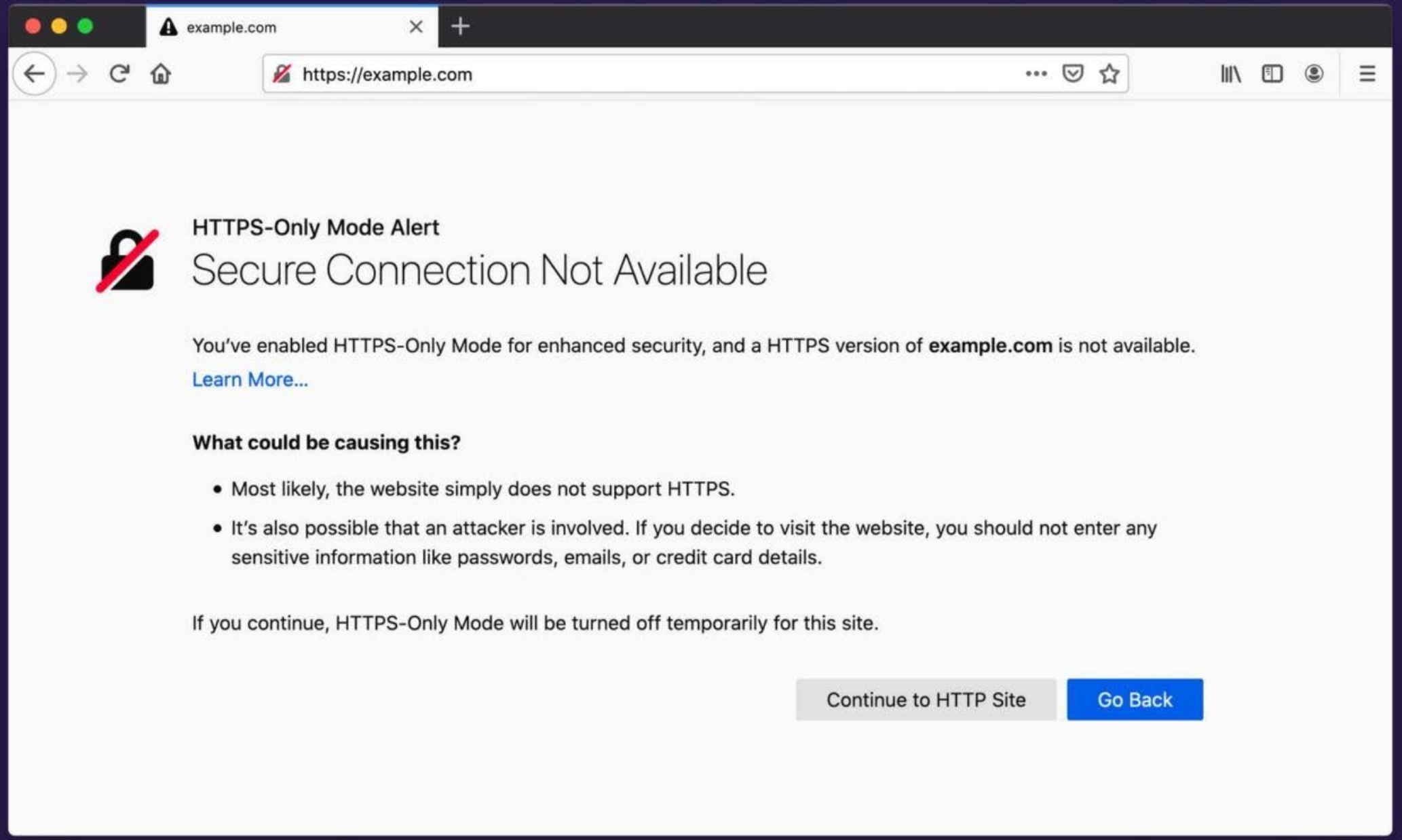
Security on the web matters. Whenever you connect to a web page and enter a password, a credit card number, or other sensitive information, you want to be sure that this information is kept secure. Whether you are writing a personal email or reading a page on a medical condition, you don't want that information leaked to eavesdroppers on the network who have no business prying into your personal communications.

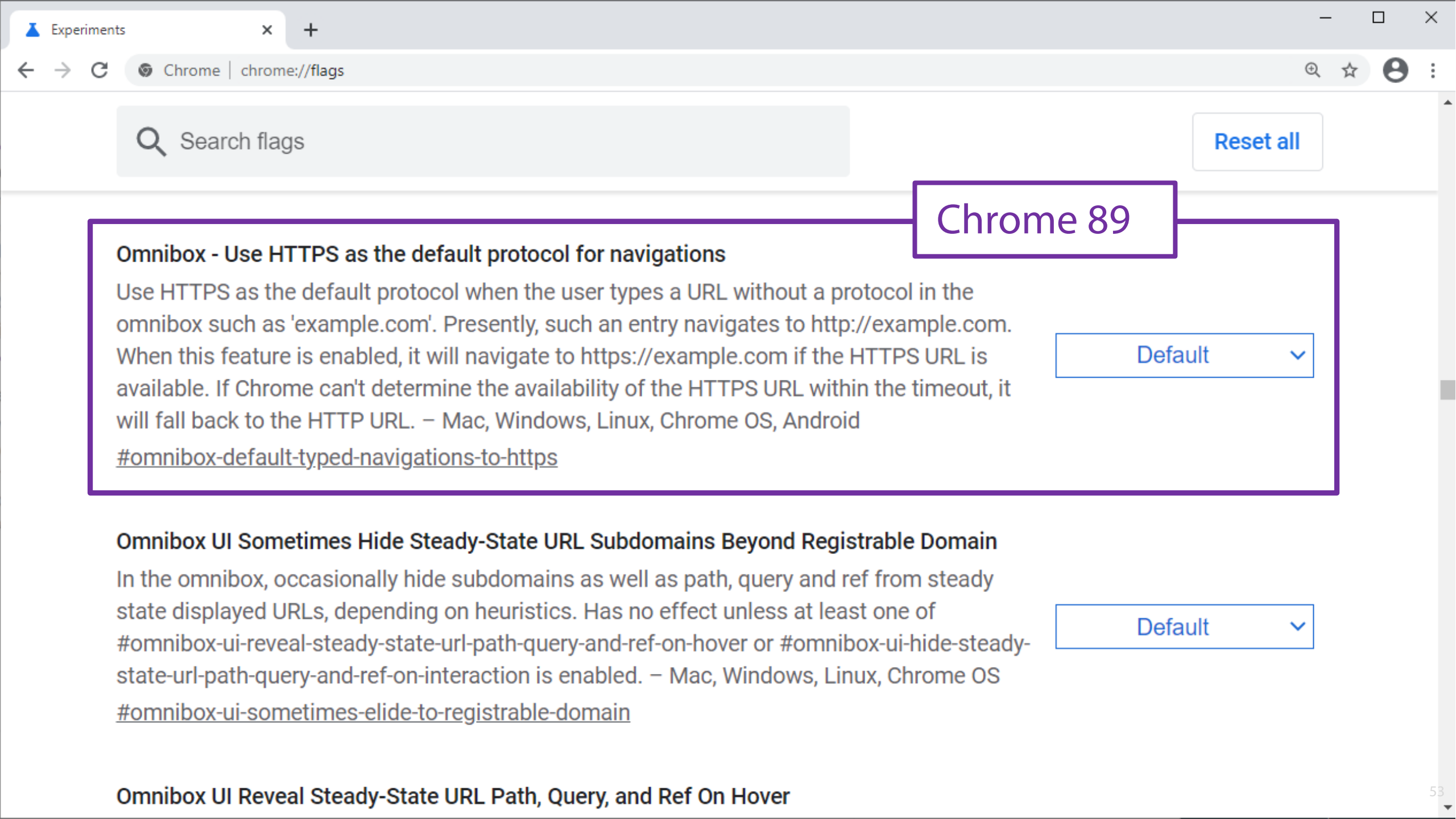
That's why Mozilla is pleased to introduce HTTPS-Only Mode, a brand-new security feature available in Firefox 83. When you enable HTTPS-Only Mode:

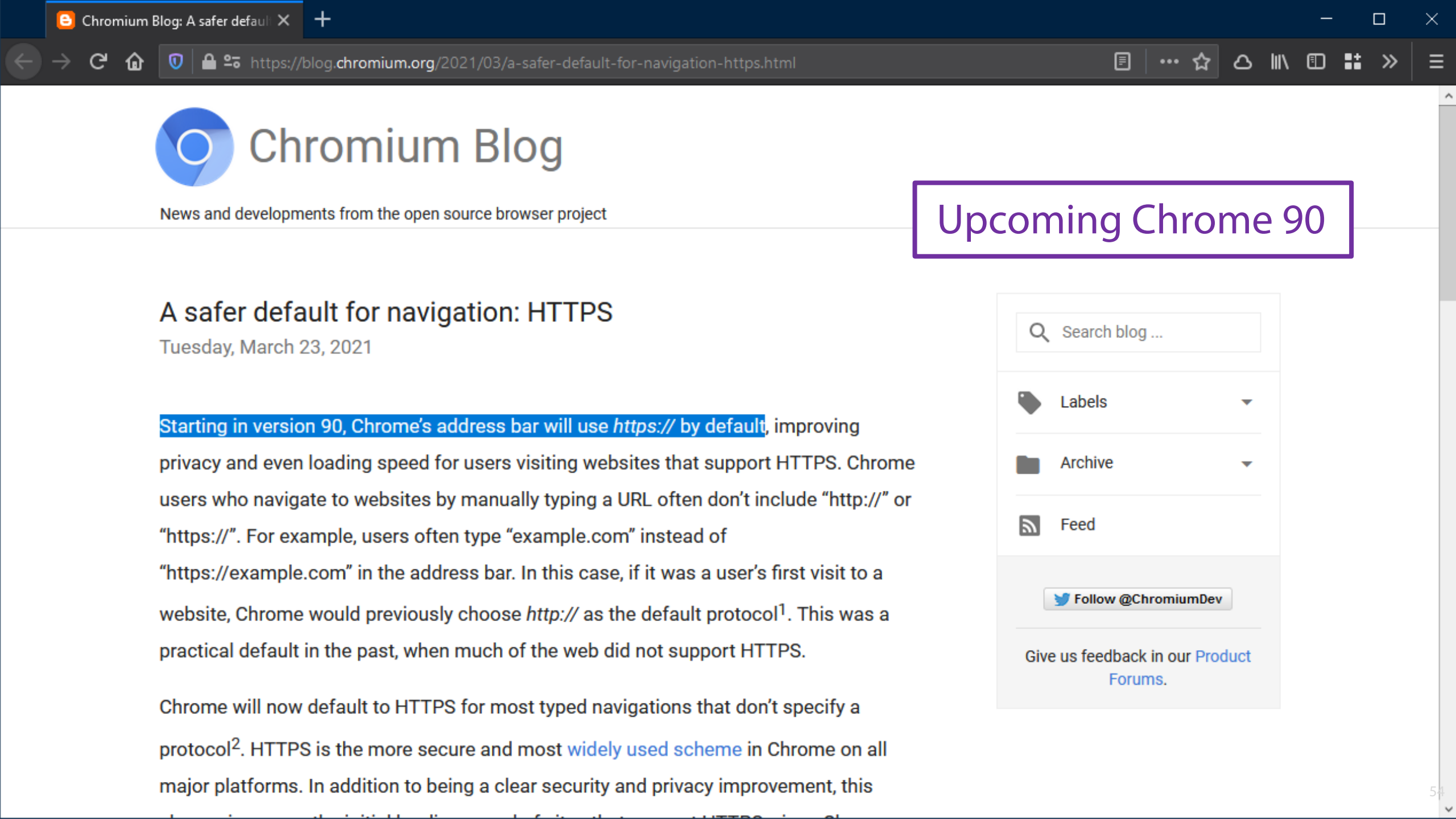
- Firefox attempts to establish fully secure connections to every website, and
- Firefox asks for your permission before connecting to a website that doesn't support secure connections.

## How HTTPS-Only Mode works

The Hypertext Transfer Protocol (HTTP) is a fundamental protocol through which web browsers and websites communicate. However, data transferred by the regular HTTP protocol is unprotected and transferred in cleartext, such that attackers are able to view, steal, or even tamper with the transmitted


example.com
https://example.com
HTTPS-Only Mode Alert
Secure Connection Not Available
You've enabled HTTPS-Only Mode for enhanced security, and a HTTPS version of example.com is not available.
Learn More...
What could be causing this?
Most likely, the website simply does not support HTTPS.
It's also possible that an attacker is involved. If you decide to visit the website, you should not enter any sensitive information like passwords, emails, or credit card details.
If you continue, HTTPS-Only Mode will be turned off temporarily for this site.
Continue to HTTP Site
Go Back


Experiments
Chrome | chrome://flags
Search flags
Reset all
Chrome 89
Omnibox - Use HTTPS as the default protocol for navigations
Use HTTPS as the default protocol when the user types a URL without a protocol in the omnibox such as 'example.com'. Presently, such an entry navigates to http://example.com. When this feature is enabled, it will navigate to https://example.com if the HTTPS URL is available. If Chrome can't determine the availability of the HTTPS URL within the timeout, it will fall back to the HTTP URL. – Mac, Windows, Linux, Chrome OS, Android
#omnibox-default-typed-navigations-to-https
Default
Omnibox UI Sometimes Hide Steady-State URL Subdomains Beyond Registrable Domain
In the omnibox, occasionally hide subdomains as well as path, query and ref from steady state displayed URLs, depending on heuristics. Has no effect unless at least one of #omnibox-ui-reveal-steady-state-url-path-query-and-ref-on-hover or #omnibox-ui-hide-steady-state-url-path-query-and-ref-on-interaction is enabled. – Mac, Windows, Linux, Chrome OS
#omnibox-ui-sometimes-elide-to-registrable-domain
Default
Omnibox UI Reveal Steady-State URL Path, Query, and Ref On Hover


Chromium Blog: A safer defaul
https://blog.chromium.org/2021/03/a-safer-default-for-navigation-https.html
Chromium Blog
News and developments from the open source browser project
Upcoming Chrome 90
A safer default for navigation: HTTPS
Tuesday, March 23, 2021
Starting in version 90, Chrome's address bar will use https:// by default, improving privacy and even loading speed for users visiting websites that support HTTPS. Chrome users who navigate to websites by manually typing a URL often don't include "http://" or "https://". For example, users often type "example.com" instead of "https://example.com" in the address bar. In this case, if it was a user's first visit to a website, Chrome would previously choose http:// as the default protocol[1]. This was a practical default in the past, when much of the web did not support HTTPS.
Chrome will now default to HTTPS for most typed navigations that don't specify a protocol[2]. HTTPS is the more secure and most widely used scheme in Chrome on all major platforms. In addition to being a clear security and privacy improvement, this
Search blog ...
Labels
Archive
Feed
Follow @ChromiumDev
Give us feedback in our Product Forums.

# 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 scary 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 certificiates. 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/

ars technica

MAIN MENU · MY STORIES: 25 · FORUMS · SUBSCRIBE · JOBS

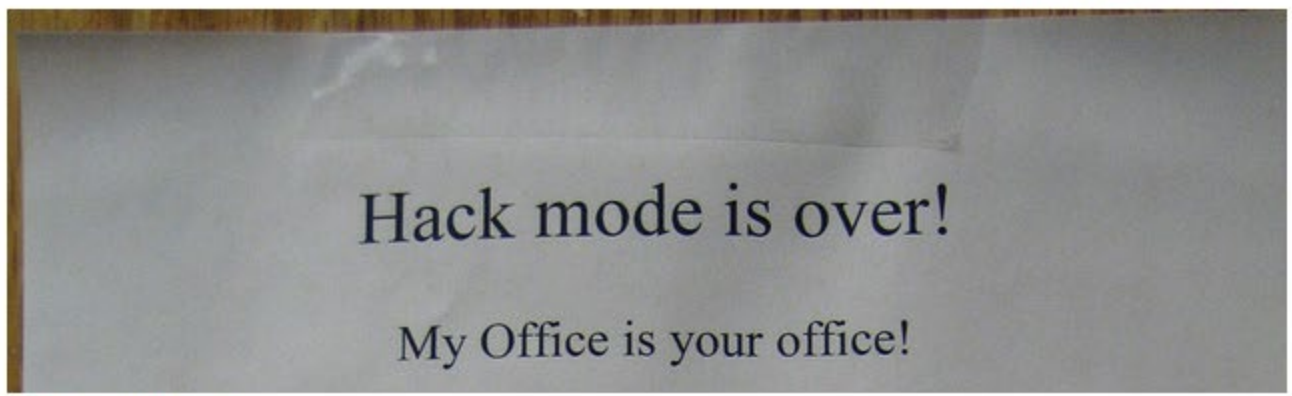
RISK ASSESSMENT / SECURITY & HACKTIVISM

# 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

LATEST FEATURE STORY

FEATURE STORY (2 PAGES)

**This could be the food future—if you can har**

It's an evening of entomology—cooki and trying to understand an insect di

WATCH ARS VIDEO

Trustwave to escape 'deatl

www.theregister.co.uk/2012/02/14/trustwave_analysis/

The Register®
Biting the hand that feeds IT

DATA CENTER SOFTWARE NETWORKS SECURITY INFRASTRUCTURE DEVOPS BUSINESS HARDWARE SCIENCE BOOTNOTES FORUMS

Security

# Trustwave to escape 'death penalty' for SSL skeleton key

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

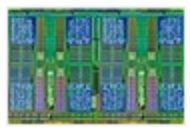
14 Feb 2012 at 09:28, John Leyden 12 10

**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

### Most read


AMD to fix slippery hypervisor-busting its CPU microcode

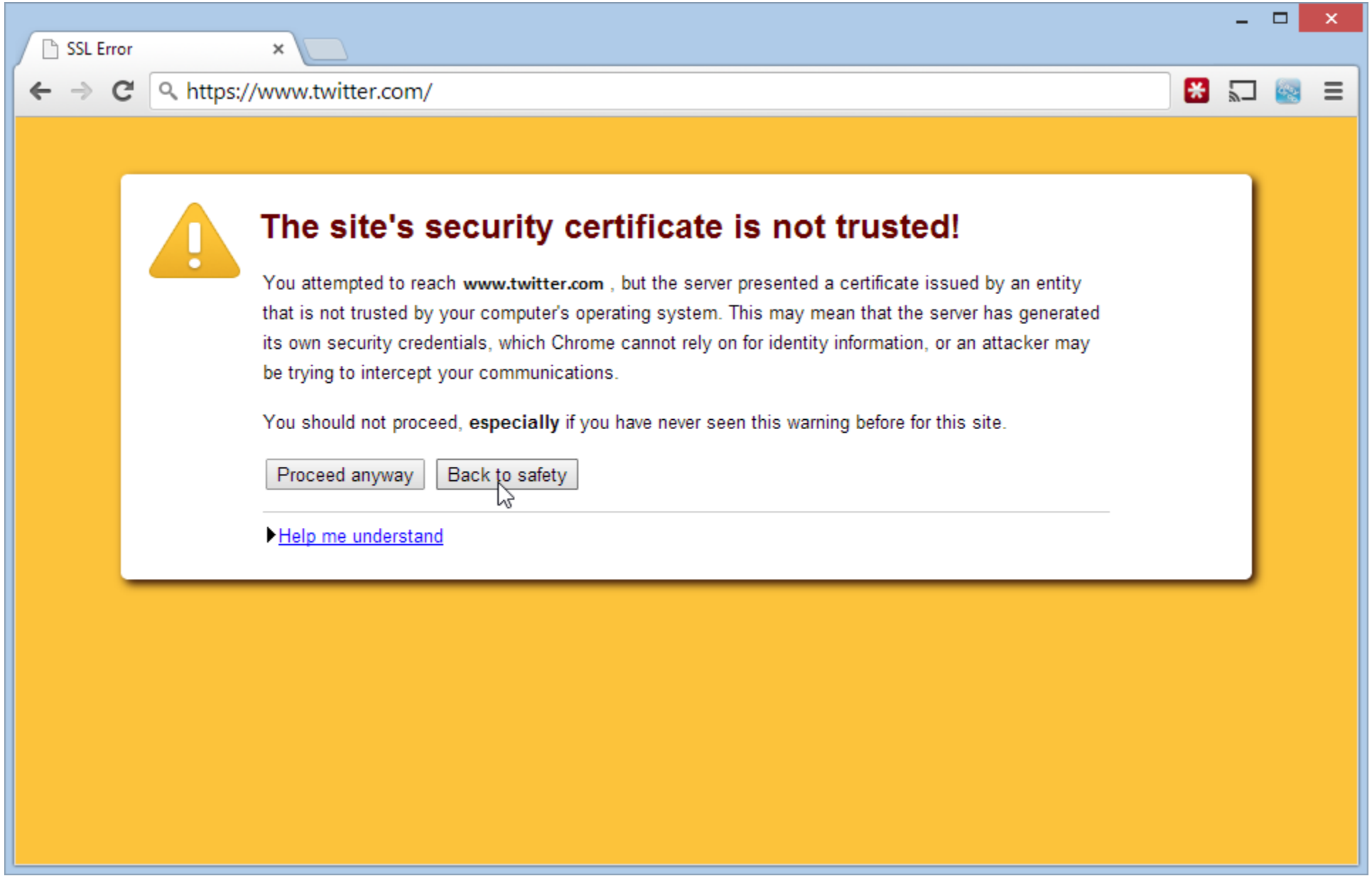

First working Apple ransomware infects Transmission BitTo app downloads


Amazon douses fla vows to restore Fire fondleslab encrypti

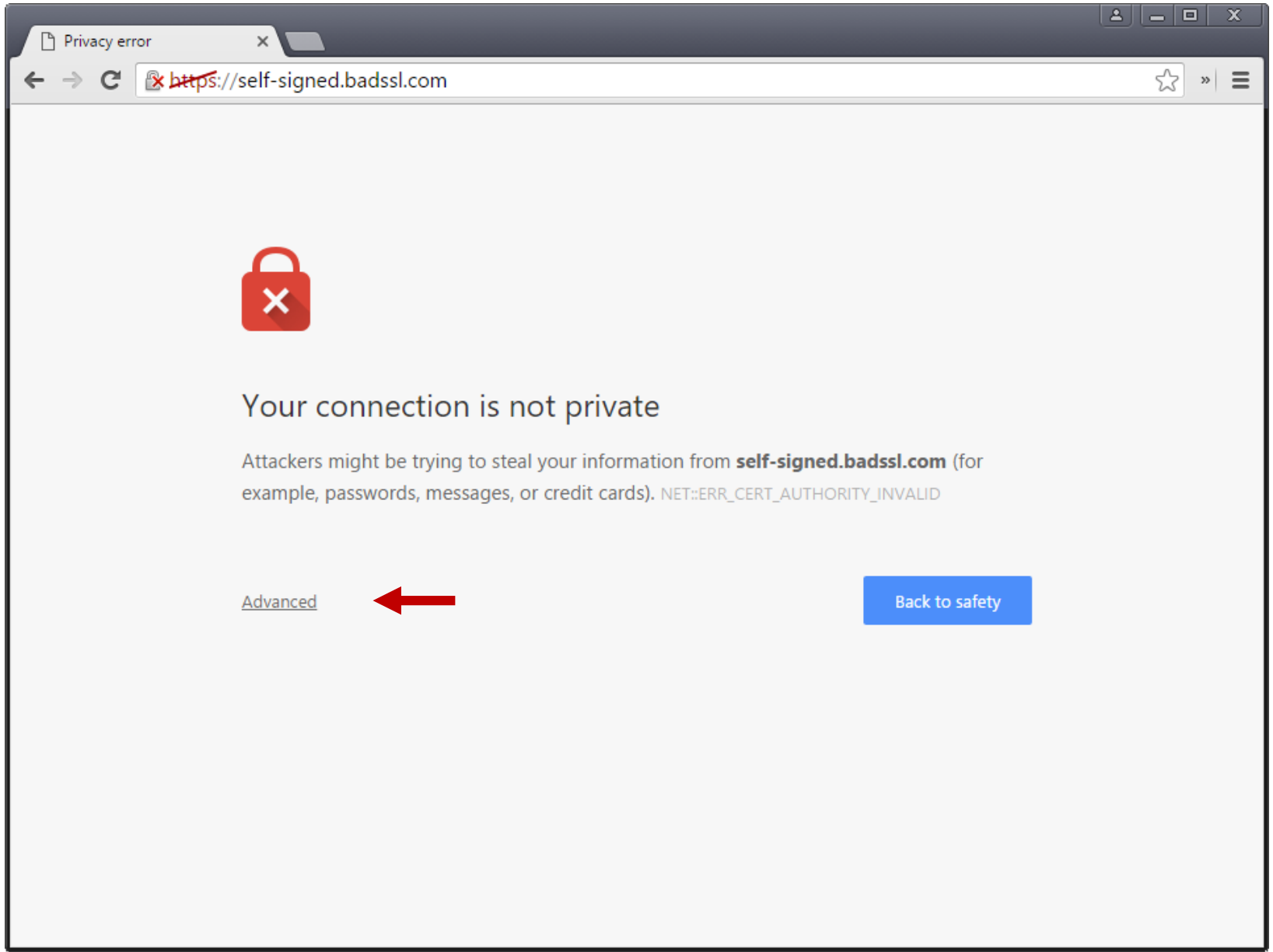

MAME goes fully F


McAfee gaffe a qui kill for enterprising

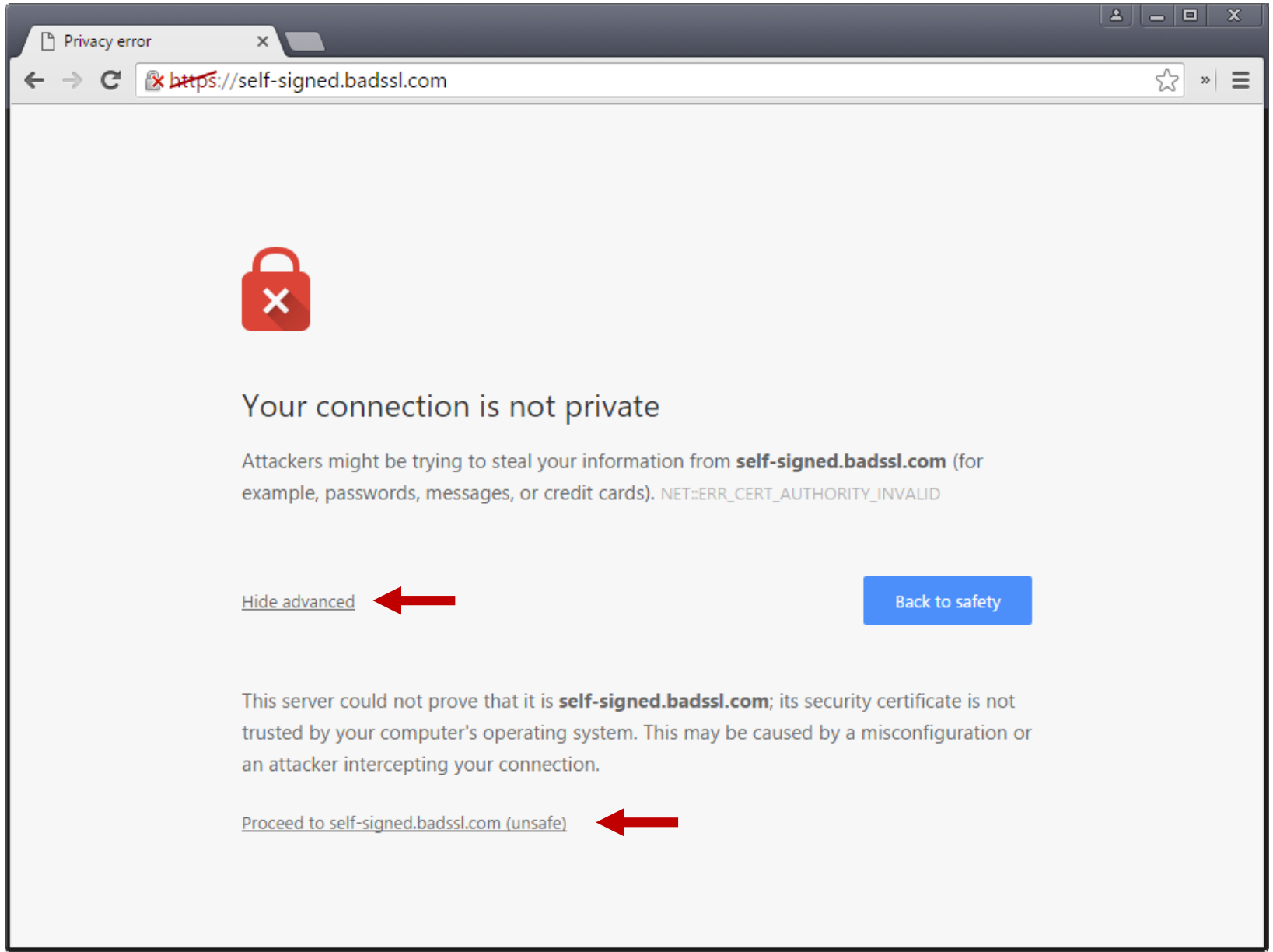
# Self-signed Certificate Warning:  One click away…



# Self-signed Certificate Warning: Two clicks away…



# Self-signed Certificate Warning:  Two clicks away…



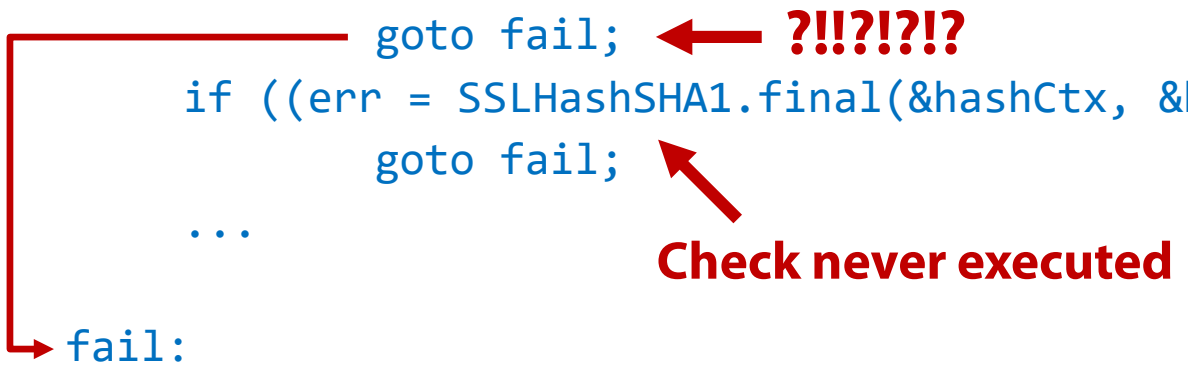
# GOTO FAIL

## 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;
```



**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, the cert is rejected
- 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

**HPKP** (HTTP Public Key Pinning)

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

Browser knows *valid* public keys of a pa website

- If a seemingly valid es not include at l known pinned key, the cert is rejected
- Doesn't apply for *priva* rtificates (w k preconfigured proxies, anti-malware, content filters, …)

Many incidents involving f ates were discovered after browsers started rolling out pinning

Similar deployment as HST

- TOFU: HTTP response he
- Built-in pins in browse

**Must be used ver ully – things ca rong**

- HPKP suicide: s e bricked if keys are lost/sto
- RansomPKP: comp omise the server and push a malici us HPKP key

*Deprecated in favor of* ***Certificate Transparency*** *and the* ***Expect-CT*** *header*

Google Online Security Blc

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 Transparency

Public monitoring and auditing of certificates

Identify mistakenly or maliciously issued certificates and rogue CAs

*Certificate logs*

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

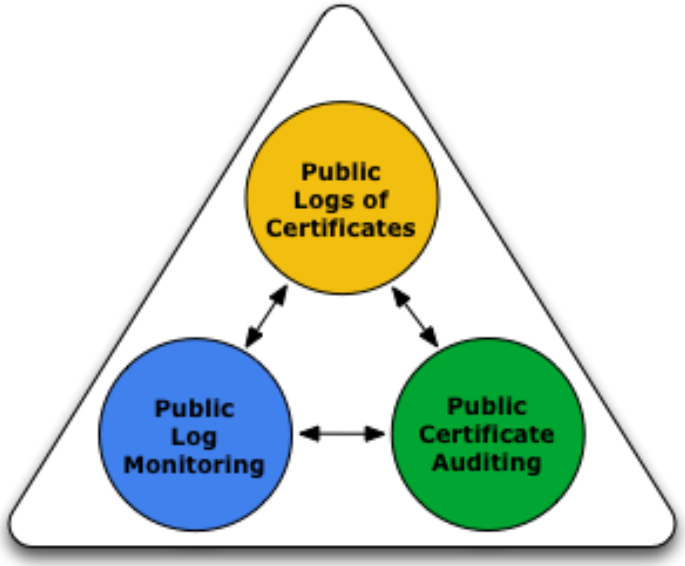
*Monitors*

Periodically contact all log servers and watch for suspicious certificates

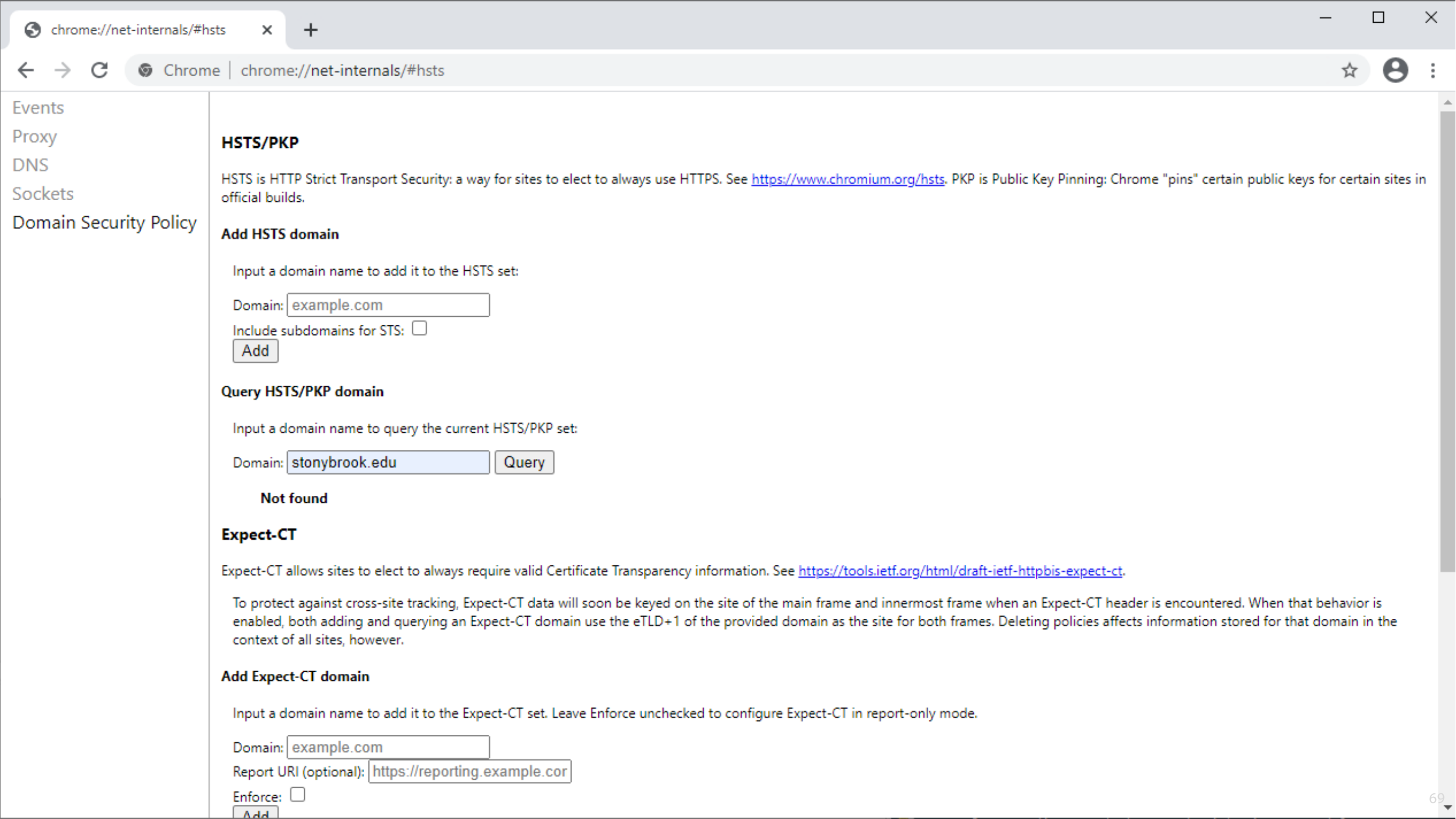
*Auditors*

Verify that logs are behaving correctly and are cryptographically consistent

Check that a particular certificate appears in a log




https://certificate.transparency.dev/


chrome://net-internals/#hsts
Chrome | chrome://net-internals/#hsts
Events
Proxy
DNS
Sockets
Domain Security Policy
HSTS/PKP
HSTS is HTTP Strict Transport Security: a way for sites to elect to always use HTTPS. See https://www.chromium.org/hsts. PKP is Public Key Pinning: Chrome "pins" certain public keys for certain sites in official builds.
Add HSTS domain
Input a domain name to add it to the HSTS set:
Domain: example.com
Include subdomains for STS:
Add
Query HSTS/PKP domain
Input a domain name to query the current HSTS/PKP set:
Domain: stonybrook.edu
Query
Not found
Expect-CT
Expect-CT allows sites to elect to always require valid Certificate Transparency information. See https://tools.ietf.org/html/draft-ietf-httpbis-expect-ct.
To protect against cross-site tracking, Expect-CT data will soon be keyed on the site of the main frame and innermost frame when an Expect-CT header is encountered. When that behavior is enabled, both adding and querying an Expect-CT domain use the eTLD+1 of the provided domain as the site for both frames. Deleting policies affects information stored for that domain in the context of all sites, however.
Add Expect-CT domain
Input a domain name to add it to the Expect-CT set. Leave Enforce unchecked to configure Expect-CT in report-only mode.
Domain: example.com
Report URI (optional): https://reporting.example.cor
Enforce:
Add

Events
Proxy
DNS
Sockets
Domain Security Policy

## HSTS/PKP

HSTS is HTTP Strict Transport Security: a way for sites to elect to always use HTTPS. See https://www.chromium.org/hsts. PKP is Public Key Pinning: Chrome "pins" certain public keys for certain sites in official builds.

### Add HSTS domain

Input a domain name to add it to the HSTS set:

Domain: example.com

Include subdomains for STS:

Add

### Query HSTS/PKP domain

Input a domain name to query the current HSTS/PKP set:

Domain: google.com Query

**Found:**

static_sts_domain:
static_upgrade_mode: DEFAULT
static_sts_include_subdomains: false
static_sts_observed: 0
static_pkp_domain: google.com
static_pkp_include_subdomains: true
static_pkp_observed: 1615499290
static_spki_hashes:
sha256/IPMbDAjLVSGntGO3WP53X/zilCVndez5YJ2+vJvhJsA=,sha256/YZPgTZ+woNCCCIW3LH2CxQeLzB/1m42QcCTBSdgayjs=,sha256/hxqRlPTu1bMS/0DITB1SSu0vd4u/8l8TjPgfaAp63Gc=,sha256/Vfd95BwDeSQo+NUYxVEEIlvkOlWY2SalKK1lPhzOx78=,sha256/QXnt2YHvdHR3tJYmQIr0Paosp6t/nggsEGD4QJZ3Q0g=,sha256/mEflZT5enoR1FuXLgYYGqnVEoZvmf9c2bVBpiOjYQ0c=,sha256/iie1VXtL7HzAMF+/PVPR9xzT80kQxdZeJ+zduCB3uj0=
dynamic_sts_domain:
dynamic_upgrade_mode: UNKNOWN
dynamic_sts_include_subdomains:
dynamic_sts_observed:
dynamic_sts_expiry:

### Expect-CT