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

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: \([\text{header} \mid \text{data}]\) 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
Serialize four core subprotocols: handshake, change cipher spec, application data, alert
TLS 1.2 Handshake (Ephemeral DH)

1. ClientHello
2. ServerHello
3. Certificate
4. ServerKeyExchange
5. ServerHelloDone

3. ClientKeyExchange
4. ChangeCipherSpec
5. Finished

4. ChangeCipherSpec
5. Finished

5. GET /login HTTP/1.1
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 cipher 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, or alter the client’s list of supported suites
SSL 3.0, TLS 1.0, and TLS 1.1 are now completely removed by most browsers.
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)
Latest draft supports even zero-RTT handshakes

Clients include encrypted data in the initial messages based on config. ID previously sent by server
TLS 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

TLS certs are based on the X.509 PKI standard

How is the certificate associated with the server?

*Common Name (CN): server’s hostname*

Certificate-based authentication is also supported for 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 Viewer: cs.stonybrook.edu

Certificate Hierarchy
- ISRG Root X1
  - R3
  - cs.stonybrook.edu

Certificate Fields
- cs.stonybrook.edu
  - Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
  - Issuer
  - Validity
    - Not Before

Field Value
- CN = R3
- O = Let's Encrypt
- C = US
Certificate Chains

Trust anchors: operating systems and browsers are pre-configured with 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
SSL Report: www.cs.stonybrook.edu (23.185.0.4)
Assessed on: Mon, 11 Mar 2024 18:40:24 UTC

Summary

This site works only in browsers with SNI support.
This server supports TLS 1.3.

Certificate #1: RSA 2048 bits (SHA256withRSA)

Subject:
• Fingerprint: SHA256: 4c2e25917d0a02424563039ed336b1b1b3013002cb84c70bae10533c5307f5
  Fingerprint: SHA256: 6b9004a431e8a39b1f105f72d2b4f7e6e88f16e80f9b79b4c9c2bab6f

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.
<table>
<thead>
<tr>
<th>Certificate Transparency</th>
<th>Yes (certificate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS CAA</td>
<td>No (more info)</td>
</tr>
<tr>
<td>Trusted</td>
<td>Yes</td>
</tr>
<tr>
<td>Additional Certificates (if supplied)</td>
<td>Certificates provided 3 (3894 bytes)</td>
</tr>
</tbody>
</table>
Certification Authority Authorization (CAA)

By default, certificate authorities (CA) are allowed to issue certificates for any domain name (after they validate control of that domain name)

- Bugs in CA’s domain validation process
- Compromised CAs

CAA: Specify which CAs are authorized to issue certificates for a domain

- Reduce the “attack surface” of CAs that could otherwise issue unauthorized certificates

Implemented as a special DNS resource record: CAA

- Specifies the allowed CAs, policy flags, …

CAs are required to check CAA records and comply with their directives

- Third parties monitoring CA behavior might check newly issued certificates against the domain’s CAA records and identify violators
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: *almost all web traffic is now encrypted*

All major roadblocks of the past are not a problem anymore

- Crypto is expensive, HTTPS needs more CPU cycles ➔ native hardware support
- Mixed HTTP+HTTPS content ➔ almost all third-party ad networks/widgets/etc. now support HTTPS
- Virtual hosting was initially incompatible ➔ solved with TLS 1.1 through the Server Name Indication (SNI) extension
- Needs expertise and certs cost $$$$ ➔ certs are now free and easy to deploy and maintain through [letsencrypt.org](http://letsencrypt.org)
Firesheep In Wolves' Clothing: Extension Lets You Hack Into Twitter, Facebook Accounts Easily

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 Facebook HTTP ➔ HTTPS

2010: HTTPS only for login
2010: Firesheep released
2011: Optional full HTTPS
2013: HTTPS on by default
**Browser Security Indicators** (state until circa 2015)

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”

“This parts of the page are not encrypted”

“The legal entity operating this web site is known”

- Extended Validation (EV) certificates
Browser Security Indicators (state until circa 2015)

Convey information about the security of a page

Locks, shields, keys, green bars…

“This page was fetched using a 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”

“This page uses an invalid certificate”

“Parts of the page are not encrypted”

“Parts of the page are not encrypted”

“The legal entity operating this web site is known”

Extended Validation (EV) certificates

Secure | https://

Not secure | https://

https://

Square, Inc. [US] | https://squ
Mixed Content Warning is Unnecessary (2015)

Non-HTTPS traffic is a vulnerability!
MitM/MotS attacks on the HTTP part are trivial
Mark HTTP as Not Secure (2018)

Majority of traffic is HTTPS ➔ negative indicator for HTTP

Before: mark HTTPS as secure

Majority of traffic was HTTP ➔ positive indicator for HTTPS

HTTPS

<table>
<thead>
<tr>
<th>Current (Chrome 67)</th>
<th>Secure</th>
<th>example.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep. 2018 (Chrome 69)</td>
<td>example.com</td>
<td></td>
</tr>
<tr>
<td>Eventually</td>
<td>example.com</td>
<td></td>
</tr>
</tbody>
</table>

HTTP

⚠️ Not secure | example.com

Lock Icon Is Gone (2023)

“Tune icon”
- Does not imply "trustworthy"
- Is more obviously clickable
- Is commonly associated with settings or other controls

https://blog.chromium.org/2023/05/an-update-on-lock-icon.html
The information you’re about to submit is not secure

Because this form is being submitted using a connection that’s not secure, your information will be visible to others.

Send anyway
The information you’re about to submit is not secure

Because this form is being submitted using a connection that’s not secure, your information will be visible to others.
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 (similar to DNS poisoning):
  https://www.bank.com.attacker.com
SSL stripping

Victim -- HTTP -- Attacker -- HTTPS -- Server

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

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

Missing lock icon “Not secure” warning 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

“From now on and for the next year, use only HTTPS for requests to this domain”
```

An instance of trust on first use (TOFU)

Problem: the initial request remains unprotected because it is sent over HTTP
Solution: **HSTS preloading**: browsers come preloaded with a list of known HSTS sites
// START OF LEGACY MANUAL CUSTOM ENTRIES

{
  "name": "www.paypal.com", "policy": "custom", "mode": "force-https" },
{
  "name": "paypal.com", "policy": "custom", "mode": "force-https" },
{
  "name": "www.elanex.biz", "policy": "custom", "mode": "force-https" },
{
  "name": "www.noisebridge.net", "policy": "custom", "mode": "force-https" },
{
  "name": "neg0.org", "policy": "custom", "mode": "force-https" },
{
  "name": "factor.cc", "policy": "custom", "mode": "force-https" },
{
  "name": "sladdinschools.appspot.com", "policy": "custom", "mode": "force-https" },
{
  "name": "www.paycheckrecords.com", "policy": "custom", "mode": "force-https" },
{
  "name": "lostpass.com", "policy": "custom", "mode": "force-https" },
{
  "name": "www.lastpass.com", "policy": "custom", "mode": "force-https" },
{
  "name": "www.entropia.de", "policy": "custom", "mode": "force-https" },
{
  "name": "www.entropia.de", "policy": "custom", "mode": "force-https" },
{
  "name": "logentries.com", "policy": "custom", "mode": "force-https" },
{
  "name": "www.logentries.com", "policy": "custom", "mode": "force-https" },
{
  "name": "dropcam.com", "policy": "custom", "mode": "force-https" },
{
  "name": "dropcam.com", "policy": "custom", "mode": "force-https" },
{
  "name": "epoxate.com", "policy": "custom", "mode": "force-https" },
{
  "name": "torproject.org", "policy": "custom", "mode": "force-https", "pins": "tor" },
{
  "name": "blog.torproject.org", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "tor" },
{
  "name": "check.torproject.org", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "tor" },
{
  "name": "torproject.org", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "tor" },
{
  "name": "dist.torproject.org", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "tor" },
{
  "name": "ledgerscope.net", "policy": "custom", "mode": "force-https" },
{
  "name": "www.ledgerscope.net", "policy": "custom", "mode": "force-https" },
{
  "name": "greplin.com", "policy": "custom", "mode": "force-https" },
{
  "name": "www.greplin.com", "policy": "custom", "mode": "force-https" },
{
  "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 data. HTTP over TLS (HTTPS) fixes this security shortcoming by creating a secure and encrypted...
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
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. 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. 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 change encourages users to keep their browser up to date and to visit and navigate to HTTPS sites.
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 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...
Comodo hacker: I hacked DigiNotar too; other CAs breached
The hacker behind this year's Comodo hack has claimed responsibility for the ...
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 in the Past: One click away…
Self-signed Certificate Warning Now: Two clicks away…
Self-signed Certificate Warning Now: Two clicks away
GOTO FAIL

iOS 7.0.6 signature verification error

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

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

*Deprecated in favor of Certificate Transparency and the Expect-CT header*
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
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/
Certificates are deposited in public, transparent logs (append-only ledgers)

Distributed and independent: anyone can query them to see what certificates have been included and when

Append-only: verifiable by Monitors

Web browsers enforce Certificate Transparency

Logs are cryptographically monitored

Monitors cryptographically check which certificates have been included in logs

Domain owners can subscribe to a CT monitor to get updates when precertificates/certificates for those domains are included in any of the logs checked by that monitor

https://certificate.transparency.dev/howctworks/
### HSTS/PKP

HSTS is HTTP Strict Transport Security; a way for sites to elect to always use HTTPS. See [https://www.chromium.org/hsts](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


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.com](https://reporting.example.com)
- Enforce: [ ]
- Add
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]