HSTS Meetup 2016
Public Notes

contact: lgarron@chromium.org

doc visibility: **PUBLIC**
last updated: 2017-01-19
shortlink: bit.ly/hsts-meetup-notes

Contents

Background

HSTS adoption has seen remarkable growth in 2016. However, this growth comes at a price to the HSTS preload list. In addition, the underlying HSTS mechanism itself has some feature gaps that have not been addressed since its creation.

This meetup was organized by Lucas Garron (Google Chrome) to gather people from several organizations who have extensive experience with HSTS on the browser side or the server side, to talk about their current challenges and priorities related to HSTS.

We met at the Google office in San Francisco on September 30, 2016.

Agenda

This document contains notes of what we discussed; see bit.ly/hsts-meetup for the original planned agenda that was sent out to participants ahead of time.

Participants

Name	Organization
Eric Mill	18F
Alex Christensen	Apple
Luke Case	Apple
Yan Zhu	Brave
Nick Sullivan	Cloudflare
Dani Grant	Cloudflare
Jacob Hoffman-Andrews	EFF
Seth Schoen	EFF
David Huang	Facebook
Lucas Garron	Google
Adam Barth	Google
Adrienne Porter Felt	Google

Emily Stark	Google
Mike Bishop	Microsoft
Kate McKinley	Mozilla
Daniel Veditz	Mozilla
Tanvi Vyas	Mozilla
Tarquin Wilton-Jones	Opera
Jeff Hodges	PayPal
Andre Pinter	Twitter

Short Talks

Adam Barth (HSTS spec author, now Google)

- 2007/2008: Collin and Adam were at a conference
- A question: what's the simplest you could use to represent security? One bit? A number?
- The deadline for the next conference submissions were due two days later; they frantically drafted the ForceHTTPS paper and implemented a prototype for Firefox.
    - How the prototype worked: Hijacked the setup process, and replaced the GUID for the URL parser with one that rewrote URLs to HTTPS.
- Audience question: When was Moxie Marlinspike's `sslstrip` talk?
    - Answer: 2009
    - Comment by Adam Langley: A lot of web attacks are known for a long time, but take a while to crystalize.

Jeff (HSTS spec author, PayPal)

- Joined PayPal to work on efforts to fix major security holes in the web.
- Goals: address CSRF, third-party content inclusion.
    - CSRF protection turned into the single-origin policy.
    - Third-party content issues were addressed by CSP.
- Moxie had talked about `sslstrip` at Blackhat 2009, Andy (at PayPal) said "can't we address that using ForceHTTPS?"
    - PayPal brought in Jeff Hodges to help turn it into a spec.
        - Worked in "community" to develop initial STS spec, then renamed HSTS and took it to IETF in the process of starting the IETF WebSec working group. WebSec participants produced HSTS, HPKP, Web

Origin, and X-Frame-Options specs (see: https://tools.ietf.org/wg/websec/)
        - Some other efforts went on concurrently:
            - Turned into RFC6265 (via restarting the HTTP-State working group (JeffH chaired, ABarth wrote spec) and RFC6125 (via an AD-sponsored effort with Peter St. Andre)
            - Also working with W3C to get the Web App Security WG started there.

Lucas (Google Chrome)

Slides

- Adam Langley ("agl@") created the Chrome preload list in 2012.
    - All major browsers now use preload lists based on Chrome.
- The preload list submission process:
    - The site sends a header with a **`preload`** directive.
    - Someone (hopefully the site operator) submits it to hstspreload.appspot.com, which checks against a list of requirements. Sites must:
        - load a "whole domain" at a time (eTLD+1).
        - Have a valid certificate.
        - Redirect from HTTP to HTTPS.
        - Serve an HSTS header.
        - `includeSubDomains`
    - Submissions are pending for up to 6 weeks
        - This used to be ≈ weekly, but Chromium currently needs to batch preload changes once per release to avoid lots of binary churn in the Chromium checkout (to fix this, the binary data needs to be generated as part of the build process).
    - Before a Chrome release goes to beta, lgarron@ re-scans all pending sites (many pending sites have already stopped meeting the requirements).
    - The JSON file in the Chromium source is updated with the new sites.
    - The accompanying header file is updated.
- Preload list growth has seen a large uptick since January 2016.
    - lgarron@ updated the requirements in early 2016 to make sure only intentional submissions get through (a lot of removals claim to be due to unintended submission, but it's unclear what the cause is).
- The growth is currently moderate, but someday the list will be too large to ship in every binary.

Jacob Hoffman-Andrews (EFF, formerly Twitter)

- Let's recognize that we're trying to take on a hard task, trying to improve HTTP, at the core of the web.
- HTTPS at Twitter
    - Inspired by Eric Butler's Firesheep demonstration.
    - Pain point: Had to rolll back preloading due to a subdomain of `api.twitter.com`
    - HSTS Carveouts: would have been helpful, but also a bad idea
        - Something they saw often: a contractor would build a site, and security would have to chase them down.
- HTTPS Everywhere (at EFF)
    - Comment by Seth on the motivation for HTTPS Everywhere: Google offered HTTPS, but didn't use it. We can fix that on the client!
    - Originally shipped with 6 sites
        - Also had to rewrite subresources
        - Sometimes rewriting directly to HTTPS didn't work; needed to regex the hostname (e.g. rewrite to point at an S3 bucket).
    - Subresource block:
        - If HSTS was allowed apply redirects before mixed content blocking, it would allow many sites to "just work," but this introduces unpleasant history-dependence to the question of whether a site "just works."
- HTTPS Everywhere is more willing than browsers to risk breaking sites for the sake of security
    - Occasionally breaks site; e.g. broke BoingBoing as they were transitioning to HTTPS.
    - No good policy yet for what to include and what not.
        - How do you prioritize importance?
- Let's Encrypt
    - Keeps a list of some domains (e.g. `google.com`, `addons.mozilla.org`) that they will never issue certs for.
        - But sometimes there are requests for exceptions.
        - Current decision: no carveouts, but more specific blocking of subdomains.

Yan (Brave)

- HSTS Supercookies
    - Can use subdomains as a bitmask to identify users.
- HSTS cross-origin history sniffing
    - Include an HTTP resource, use latency to detect if the upgraded resource is loaded/fails quickly
    - Can also use CSP.
    - Can currently use on dynamic-only HSTS using http://example.com:443

        - This is not fixed; for example, lgarron@ implemented a fix in Chrome but rolled it back)
- Question from Tanvi: should we separate HSTS between private and non-private browsing modes?
    - Daniel: Firefox uses the non-private set in private, but doesn't record new values.

Nick (Cloudflare)

- "Cloudflare, if you don't know, is basically a massive nginx".
- Around 8-10% of internet requests go through CloudFlare.
- HSTS for cloudflare.com
    - Had to understand all the managed subdomains of cloudflare.com
    - cdnjs.cloudflare.com contained ZeroClipboard, which was Flash and need to be included from the same protocol as the page that requests it.
    - Plan:
        - Enable header with 0 seconds
        - Try 6 months
        - Switch `cloudflare.com` to use `includeSubDomains`
- HSTS as a service
    - Cloudflare has a simple configuration.
    - Opt-in, with a strong warning
    - A lot of sites now send 6 months, a lot send 0, a small number send other values
    - Solves the issue of hosting providers not allowing
- Automatic HTTPS Rewrites
    - Launched last week
    - Adapted from the HTTPS Everywhere rules.
    - Adopted by 10,000 sites in the last week; no major breakages.
    - "The certificate is the easy part in 2016." This can help with the hard part (migration).
- Bonus topic: TLS Priming?
    - Some domains support TLS for resources, but not HTTPS without mixed content in pages.
    - Note: HSTS priming does not apply.

David Huang (Facebook)

- Fun fact: was Collin Jackson's PhD student
- In 2010, Facebook was mostly HTTP (apart from a login subdomain).
    - Then came Firesheep.
- 2011: Introduced an optional "Security Browsing" mode.
    - Some apps were mixed content; supported a mixed content downgraded for them but required HTTPs starting October 2011.

- 2012: Experimental HSTS rollout.
    - Rolled out to “Security Browsing” users.
    - Started with `Max-age=60`
    - Only to browsers that support HSTS, meta referrer, and `postMessage`.
    - Challenges:
        - Redirecting through an HTTP interstitial dropped referrals (fixed using meta referrer).
        - Complications with login flow for HTTP social plugins (solved using `postMessage`).
- 2013: Migration to HTTPS by default.
    - HTTPS opt-out removed in December.
- 2014: HSTS for all users
- Current:
    - HSTS on for all browsers.
    - HSTS on by default for all subdomains, with a small blacklist.
- Thoughts
    - Interested in more preload options.
    - Could HSTS help separate cookies?
        - Facebook would prefer to avoid changing cookie names.

Mike Bishop (Microsoft)

- HTTPS Everywhere vs. Good Guys Snooping
- HTTPS is growing
    - There is a push to encrypt more
    - There are more tools available.
    - There are new technologies.
- There is still a need for access restrictions.
    - Owners of computers delegate access, and need to make decisions about what users are allowed (enterprises, libraries, prisons, families…)
    - Some of those owners request: “Give us all your encrypted data!”
        - “Uh, no.”
    - “Keys?”
        - “Still no.”
- Striking a balance (in Windows):
    - Adding name-based filtering to Windows Filtering Platform (whitelist or blacklist, control access or logging).
    - Harder to implement when apps do their own DNS resolution (includes all browsers).
    - Maybe apps can be whitelisted instead.
- Other HSTS Comments
    - HSTS (and the preload list) is implemented `WinInet`.
        - Pulled from Mozilla’s filtered list.

        - Internal tools compress the list into source code.
        - Updates pushed via Windows Update.
        - Implemented in `WinInet`; requires caller to opt in to HSTS.
    - Some Microsoft domains aren't able to move.
        - Bing: mixed content (especially video)
        - `go.microsoft.com`: legacy callers don't expect HTTPS
            - The site doesn't redirect, but sends HSTS if you go to HTTPS.

Kate McKinley (Mozilla)

- HSTS Priming
    - Landed proof of concept on Tuesday (announcement)
- Quick background:
    - Sites have mixed content while upgrading; `upgrade-insecure-requests` is not always the right choice
- How it works:
    - Mixed content requests are marked as needing priming.
    - Send HEAD a request over HTTPS to check for an HSTS header.
    - Cache result (positive or negative) for 24 hours.
- All loads in Firefox should now go through the same checks (`asyncOpen2`), which makes this practical now.
- Issues:
    - Not all servers properly handle HEAD requests.
    - The first visit to a page with mixed content may load more slowly, especially if there is a timeout.
- Next steps:
    - Another browser implementation?
    - Get people to try it.

Eric Mill ("18F/TTS/GSA/USG")

- Topic: HSTS in `.gov`
- Preloading timeline:
    - Fall 2014: Emailed `agl@` to preload `18f.gsa.gov`.
    - February 2015: Got 20 `.gov` second-level domains preloaded.
    - June 8, 2015: White House policy.
        - Goal: HTTPS + HSTS for all domains by end of 2016
    - Now going after the long tail.
- A different approach to HSTS
    - There are active dashboards to measure adoption of HTTPS across government sites
        - pulse.cio.gov (Public)
        - DHS-created reports (internally distributed)
    - Using HSTS as an inventory manager

        - Require either preloading the top level, or find each subdomain and have them all send headers individually
        - Sites strongly encouraged to go for preloading.
- HSTS as a TLD/PSL migration path
    - What if we enforced HSTS through preloading?
        - Only affects browsers that support it!
    - Large TLDs
        - Automatically submit domains for preloading upon registration?
    - Small TLDs and Public Suffixes
        - e.g. `.edu`, `.gov`, `.gov.uk`, `.github.io`, `.herokuapp.com`
        - Preload the entire TLD with a few exceptions ("the bleeding has stopped"), then focus on deleting exceptions.
        - Easier to find forward-thinking owners to try this.
        - Would help test the approach to widespread preloading and builds momentum.
    - HSTS used to be for protecting a small number of high-value domains; we can get value from about trying to invert this. (Protect the long tail.)
    - A long-term goal: Invert the preload list.
    - Question from Jacob:
        - What happened to `.trust` (Jeff: `.secure`?)
        - Answer: Didn't seem to be profitable enough.
    - Nick: How do we transition from whitelist to blacklist?

Preload List Processes

At the beginning of discussion time, we went around and shared the preload list processes for all the major browsers.

- Mozilla:
    - Scan every week for sites with `max-age`>18 weeks.
    - Update Nightly/Dev every week, let it reach Beta regularly.
    - Only remove a site if there is an HTTPS response without an HSTS header.
        - Otherwise, MitM can prevent getting the info.
    - Lucas: Would 1 year be better than 18 weeks?
        - Yes, especially if Mozilla can rely on it for their freshness scans.
    - Firefox 53 is switching to fetching more frequent preload list updates from Mozilla for freshness.
    - HPKP preload list capped at 90 days.
        - Not pinned long enough for some development cycles.
    - Dynamic HPKP max is 90 days.
- Microsoft:

    - The preload list is handled by Windows Servicing and Deployment
    - Pulls the Mozilla list every quarter
    - Runs a tool that converts the list to source code
    - Included in Window update.
    - No further filtering.
    - No HPKP
    - Eric Mill: Any max `max-age`?
        - Not as far as Mike knows.
- Chrome:
    - Dynamic HSTS capped at 1 year, HPKP capped to 60 days
- Opera:
    - Have several products
    - Opera Mini:
        - Half use the legacy Presto engine, so they don't use HSTS (even when the Chromium engine is used)
    - Opera for Android doesn't use the list because it might grow pretty stale by the time it reaches users.
    - Opera Desktop uses the Chromium list (no modifications).
        - Adds a mini list to it (internal Opera services, e.g. that provide thumbnails to the browser).
    - Opera doesn't use static (preloaded) HPKP.
        - Last assessment: too risky
- Safari
    - HSTS is implemented in the platform framework.
        - Affects all clients of the API: parts of the OS, Safari, apps.
            - (Doesn't affect Chrome, Firefox, `curl`, etc.)
        - Pulls from Chrome's list.
        - Entries never expire.
        - Also handles dynamic responses.
    - Pushed with regular OS software updates (point updates/security updates)
        - There has been discussion of doing out-of-band updates.
    - No HPKP.
    - Tanvi: So it's possible that a stale entry won't work in any app?
        - Yes, unless that app uses a lower-level socket API.

Discussion

We roughly two main planned discussion topics, but most of the discussion was not directly structured.

- Accidental submissions
    - What about asking them to include an origin-specific token?

    - Jacob: Can we detect breakage before sites reach stable?
        - Why do sites not notice issues until they're preloaded?
        - Eric: `gpo.gov` had arcane subdomains that didn't break until the domain was preloaded.
- Should we tackle easy removals or only making sure sites only preload if they really know what you're doing?
    - Jeff: Can we authenticate the origin policy in the handshake?
    - Alex: Support soft-fail preloading?
- Defaulting to HTTPS:
    - Mike B:
        - If someone doesn't specify an explicit scheme (e.g. in the URL bar), assume HTTPS.
    - Dani: What about racing HTTP vs. HTTPS?
        - Non-determinism, network cost, etc.
        - Mike: anything you send over HTTP is already leaked.
        - Jacob: Trying HTTPS first is a forcing function for fixing mixed content.
        - Seth: HTTPS Everywhere has found many cases where upgrading causes compatibility issues (often subtle).
            - forbes.com is a well-known example, but there are others.
    - Tanvi: "I don't think we can prefer HTTPS in the URL bar yet." Can try to visit the HTTPS if their history supports it.
        - Lucas and Adrienne: We had a similar conclusion in a Chromium bug.
- Seth: Higher-level: How do browser developers communicate to developers to let them know what they're doing?
    - For the people who preloaded by accident: How did they find the advice that they followed?
    - HTTPS Everywhere has encountered sites that didn't know whether they could (or even should) support it.
    - Translation?
        - Chrome DevTools is not translated
            - Why not?
            - This was considered by the DevTools team, but decided against (before the Security panel was added).
        - MDN is translated, but often into only 1-2 languages.
        - Takeaways
            - Translation of important resources might go a long way.
    - Seth: We often see sites from around here (USA) that support perfect HTTPS, but e.g. Brazil most sites doesn't. What's the language/cultural barrier?
        - Adrienne: This is something we've encountered while trying to switch iconography. We can't just set a threshold [for considering HTTPS the default], because it varies a lot.

    - Lucas: Every developer has to pass the HSTS preload site on the path to full HTTPS protection. We could try to translate the site, or make sure to put relevant content there if we want people to see it.
- Jacob: Back to “how do we flip the switch?” (defaulting to HTTPS)
    - Daniel: Carrots and sticks; we’ve been trying for a long time.
        - Recent: marking passwords and forms on HTTP for Firefox Developer edition
            - Chrome is planning this for password and credit card inputs on HTTP (for all users).
    - How do we tell when a site is HTTPS-ready?
        - `alt-svc`?
        - Daniel: Any site that sends “`alt-svc`” is [advanced enough that it is] not the problem.
        - But we need some way to tell if a site is ready.
    - Dani: Could we start defaulting to HTTPS on developer versions?
        - Firefox: Could run this as an experiment.
        - Adrienne: Chrome generally doesn’t experiment with something on Canary/Dev for a long time if we don’t have the intention of shipping it to Stable.
        - Jacob: We could try building it into HTTPS Everywhere.
- Eric: To wrap up, what about out-of-band updates?
    - Lucas: What if we update the preload list more frequently? Could we all switch to more frequent out-of-band updates (for additions, removals, pruning)?
    - Apple is already considering (see above).
    - Mike: Changing cadence for dynamic updates would be a bit ticker.
    - Jacob: Tried pruning on HTTPS Everywhere.
    - Eric: Set-once-and-forget HTTP headers are democratic; anyone can set it without active maintenance.
    - General conclusion: Most browsers represented here could eventually support this.