Upgrading HTTPS in Mid-Air
An Empirical Study of Strict Transport Security and Key Pinning in the Wild
By: Michael Kranch and Joseph Bonneau

What is HSTS and Key Pinning?
- Strict Transport Security (HSTS) is a countermeasure to HTTPS stripping through which the browser learns that specific domains must only be accessed via HTTPS by a HTTP header (dynamic) or a preflight (preloaded) list.
- Key Pinning is the only currently deployed defense against a rogue certificate where the browser learns to connect to a specific HTTPS domain only if one of a designed set of keys (derived from the domain’s certificate) is present.

Measurement Setup
- We utilized the OpenWPM web-measurement utility and modified the provided Selenium backbone’s parsed DOM interface to extract all static resources (e.g., a tags, iframes, objects, etc.) from each site on the Chrome preload list.
- To extract dynamic resources (e.g., xmlhttprequest, scripts, etc.), we created a custom Firefox extension that implements the nsContentPolicyInterface in the Firefox extension API that is called prior to loading any resources.
- We used ZMAP to gather the complete header from every active HTTP and HTTPS IP address associated with the Alexa top million domains.
- Lastly, we created a custom crawl and used the XSLT library to extract the key pins from every certificate associated with a pinned site.

Deployment of HSTS and Pinning
- HSTS was initially introduced by ForceHTTPS (Jackson and Barthe) and standardized by RFC 6797 in 2012.
- HSTS is set through an HTTP header with a mandatory max-age (seconds) and an optional includeSubdomains directive.
- Google started including preloaded HSTS and pinning policies in Chrome in 2012 (see Figure 1 for growth over time).
- Firefox followed suit in 2014 by including a majority subset of Chrome’s preload list plus several additional domains.
- Google enabled automated entry (with enforcement of additional parameters) into the preload list in August 2014.
- Dynamic Pinning (HPKP) was specified via draft RFC and is just now being seen in the wild.

Figure 1: Growth of Preloaded List

Figure 2: Alexa Rank of Preloaded Domains

Figure 3: Histogram of Max-Age

Major Results of the Study

<table>
<thead>
<tr>
<th>Error</th>
<th>Prevalence</th>
<th>Vulnerability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preloaded HSTS without dynamic HSTS</td>
<td>34.6%</td>
<td>349/1,008 domains with preloaded HSTS</td>
</tr>
<tr>
<td>Erroymous dynamic HSTS configuration</td>
<td>59.5%</td>
<td>7,491/12,593 top 1M domains attempting to set HSTS</td>
</tr>
<tr>
<td>Pinned site with non-pinned active content</td>
<td>3.0%</td>
<td>8/271 base domains with preloaded pins</td>
</tr>
<tr>
<td>Pinned site with non-pinned passive content</td>
<td>4.4%</td>
<td>8/271 base domains with preloaded pins</td>
</tr>
<tr>
<td>Cookies scoped to non-pinned subdomains</td>
<td>1.8%</td>
<td>5/271 base domains with preloaded pins</td>
</tr>
<tr>
<td>Cookies scoped to non-HSTS subdomains</td>
<td>23.8%</td>
<td>182/785 base domains with preloaded HSTS</td>
</tr>
</tbody>
</table>

Table 1: Summary of Findings

Configuration Errors
- Attempts to set dynamic HSTS: 12,593 / 751
- Redirects to HTTP domain: 5,554 / 23
- Sets HTTP HSTS header only: 517 / 3
- Redirects to HTTP header only: 774 / 3
- Malformed HSTS header: 322 / 12
- Max-age = 0: 665 / 5
- 0 < max-age <= 1 day: 2,215 / 5

Table 2: Dynamic HSTS Errors

<table>
<thead>
<tr>
<th>Alexa top 1M</th>
<th>Preloaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attempts to set dynamic HSTS</td>
<td>12,593 --- 751 ---</td>
</tr>
<tr>
<td>Doesn't redirect HTTP-&gt;HTTPS</td>
<td>5,554 44.1% 23 3.1%</td>
</tr>
<tr>
<td>Sets HTTP HSTS header only</td>
<td>517 4.1% 3 0.4%</td>
</tr>
<tr>
<td>Redirects to HTTP domain</td>
<td>774 6.1% 9 3.1%</td>
</tr>
<tr>
<td>Redirects to HTTP header only</td>
<td>74 0.8% 3 0.4%</td>
</tr>
<tr>
<td>Malformed HSTS header</td>
<td>322 2.9% 12 1.6%</td>
</tr>
<tr>
<td>Max-age = 0</td>
<td>665 5.3% 0 0.0%</td>
</tr>
<tr>
<td>0 &lt; max-age &lt;= 1 day</td>
<td>2,215 17.6% 5 0.7%</td>
</tr>
</tbody>
</table>

Table 3: Types of Pinned Mixed Content Resources

Mixed Content Issues
- Traditional mixed content refers to a HTTPS page loading resources from a HTTP origin, lowering the overall security to that of the HTTP site.
- HSTS and key-pinned sites lower their overall security to that of the least secure loaded resource origin.
- Over half the non-Google pinned domains and just under a third of the preloaded HSTS domains include resources from traditional HTTPS sites.

Table 4: Vulnerable Cookies from HSTS Domains

Cookie Theft
- Many sites are vulnerable to cookie theft even when enabling HSTS. Since cookies by default apply to all subdomains, any site not setting HSTS to include subdomains is creating a security hole for cookies.

Table 5: Leaky Pinned Cookies

Conclusion
- Developers unfamiliar with these new technologies in the leading cause of errors and many developers do not seem to fully understand same-origin policy.
- We recommend establishing defaults (max-age values and includeSubdomains) and simplifying the syntax to assist new adopters.

Future Work
- Continue to monitor the affect of automation on the growth of the preloaded list.
- Evaluate the use of new tokens (e.g., include_subdomains_for_pinning_only).
- Track the deployment of new technologies (HPKP).