Measuring the Rapid Growth of HSTS and HPKP Deployments

Ivan Petrov* Denis Peskov* Gregory Coard* Taejoong Chung†
David Choffnes† Dave Levin* Bruce M. Maggs‡ Alan Mislove† Christo Wilson†

*University of Maryland †Northeastern University ‡Duke University & Akamai Technologies

ABSTRACT

A basic man-in-the-middle attack to bypass HTTPS strips the "s" off of an "https://" URL, thereby forcing the client to effectively downgrade to an insecure connection. To address such crude attacks, the HSTS (HTTP Strict Transport Security) protocol was recently introduced, which instructs clients to preemptively (or at time of first acquire) load a list of domains to whom to connect strictly via HTTPS. In a similar vein, the HPKP (HTTP Public Key Pinning) protocol has clients obtain a set of public keys; if in future visits to the website the certificate chain does not include any of those public keys, the client is supposed to reject the connection.

Both HSTS and HPKP are relatively new additions to the web's PKI that have seen a sudden surge in deployment in the last couple of years (we observe an order of magnitude greater deployment than a 2015 study of HSTS/HPKP). Using crawls across the entire IPv4 address space, we perform a large-scale, longitudinal analysis of the deployment and operation of HSTS and HPKP. Our analysis sheds light on the root cause of this recent increase in deployment, exposes instances of mismanagement of HSTS and HPKP deployments, and yields suggestions for improvements.

1. INTRODUCTION

HTTPS is the cornerstone of security in today's web, ensuring confidentiality via encryption and authenticity through digital certificates. The protocol underlying HTTPS's security, TLS (and its successor SSL) and its implementations have been studied [15, 20, 9, 6, 7, 19] and improved [17, 1, 16, 14, 13] extensively, both in research and in practice. The importance that these critical protocols operate correctly is difficult to overstate.

However, there are several classes of attacks that render even a perfect implementation of TLS/HTTPS moot. *First*, an attacker can simply direct the browser never to pursue an HTTPS connection in the first place. For instance, in an SSL stripping attack, a victim user visits an HTTP website through a man-in-the-middle (MiTM) attacker; if that website includes URLs beginning with "`https:`", the attacker simply strips out the "`s`", causing the user to request the insecure "`http:`" version of the website, thereby exposing future communication to the MiTM attacker, as well. *Second*, if an attacker is able to have a certificate created in someone else's name, the attacker can impersonate that victim domain.

Both of these attacks completely sidestep the protections that TLS seeks to provide to its users. To address these concerns, two recent additions to HTTPS have been introduced. We describe them in detail in Section 2, but at a high level:

- **HTTP Strict Transport Security (HSTS)** [10] addresses SSL stripping attacks by informing clients which domains it should connect to *strictly* over HTTPS (i.e., if presented with an `http` URL to one of these domains, they should locally upgrade the URL to `https`).
- **HTTP Public Key Pinning (HPKP)** [8] addresses rogue certificate creation by informing clients which public keys a given domain will use in the future, thereby "pinning" the public keys to that domain. If the client sees non-pinned keys, it can refuse the connection.

These defenses are relatively new: HSTS (RFC 6797) was introduced in 2012, and HPKP (RFC 7469) was introduced in 2015. However, deployment has progressed quickly: compared with a study only two years ago [12], we observe an order of magnitude more domains hosting HSTS.

In this paper, we use HTTPS scans across the Alexa top-1M most popular websites to evaluate the rapidly increasing deployment of the HSTS and HPKP defense mechanisms. Our study pursues two broad questions: (1) what has led to the over 10× increase of deployment (e.g., is it a large collection of security-conscious experimental users, or is it the decision of a few popular web hosting providers?), and (2) are they deploying these new defenses correctly?

The fact that we are measuring these protocols only a few years into their deployment, our raw numbers are inherently somewhat small: we currently see only 41,235 domains in the Alexa top-1M serving HSTS

headers, and only 598 serving HPKP headers. On the other hand, we believe that performing this study at such an early stage of deployment offers a unique insight into how new technologies are introduced to the web's PKI—moreover, it provides us an opportunity to identify issues before they become pervasive.

The rest of this paper is organized as follows. In Section 2, we describe the pertinent details of HSTS and HPKP, and we review related work. We present our study of the nascent deployments of HSTS in Section 3 and HPKP in Section 4. Finally, we conclude in Section 5.

We will make all of our code and data publicly available.

2. BACKGROUND AND RELATED WORK

2.1 HSTS

HTTP Strict Transport Security (HSTS) originated as a direct response to these SSL stripping attacks. The issue with SSL stripping is the client cannot determine if the connection with the server should be secure. To address this, HSTS in essence informs a client which domains it should upgrade any `http` URL to `https`.

There are two broad ways in which a client can learn about a website's HSTS policy. The first is via a *preload list*: this is a list that browsers push out with their updates containing a set of domains to which the browser should strictly connect via HTTPS (along with options for also visiting all subdomains via HTTPS). Website operators must manually request being added to these preload lists. The second way to disseminate HSTS data is via *HSTS headers*; HSTS-enabled websites include these headers when clients visit them, in essence telling them "for all future queries (up to an expiry time), always connect via HTTPS." In this manner, HSTS headers achieve security after time-of-first-use. Compared to preload lists, HSTS headers are often easier to deploy.[1] Preload lists often require that websites serve HSTS; part of our study is an investigation into whether websites meet this policy.

2.2 HPKP

HTTP Public Key Pinning (HPKP) allows a domain to tell the web client exactly which certificate to accept by providing a public key. Key pinning is designed to prevent spoofed certificates from being accepted by clients. This is done by limiting the number of public keys that each domain can use. HPKP headers (a time-of-first-use mechanism, like HSTS headers) allows sites to declare these pinning policies. When an HPKP-compatible browser opens a TLS connection to a server using key pinning, it checks the pins and check that any of the pins match any of the keys in the certificate chain. If the public key for a certificate has changed, the browser will not be able to validate the pin, and the connection will fail.

The HPKP header includes: the primary pin, at least one backup pin, the max-age time, and any subdomains to also pin. According to RFC 7469 [8] the backup pin key pair should be kept offline and, in case the primary private key is lost, the backup key can be immediately deployed. To this end, the primary key must be in the current certificate chain and the backup key(s) can not be in the current chain. The backup keys are used in case the primary key is revoked or expires.

2.3 Related Work

Some studies [4, 18] demonstrated the dangers and possibilities of mixed-content websites (those that serve a combination of HTTP and HTTPS content). In particular, Chen et al. [4] investigated the dangers of mixed-content websites (that serve a combination of HTTP and HTTPS pages), and showed that half of the most popular websites are susceptible to such attacks. These motivate the need for HSTS deployments.

The most closely related work to ours is a study of HSTS and HPKP deployments by Kranch and Bonneau [12]. Their study, a first of its kind, studied the implementation of HSTS and key pinning. Both security features were relatively new at the time (2015), and so implementation was very limited. The authors concluded that this could be due to the lack of understanding of both features. Now, two years later, we have observed that the deployment of these mechanisms has increased by over an order of magnitude, each. Our study is thus an updated view, with a focus on the question of what the driving factors have been towards such a rapidly increasing deployment.

In their study, Kranch and Bonneau [12] looked at mostly Chrome's preloaded list and found it contains 390 domains from 494 unique base domains (we find over 20,000 today). They found that the 390 domains on the list have an average Alexa ranking of 100,000 meaning that many smaller domains appear on the list and not enough of the big domains are included. We describe in Sections 3 and 4 how these dynamics have changed over time.

3. HSTS DEPLOYMENT

We begin by looking at the increasing deployment of HSTS over the past two years.

3.1 Datasets

Recall that there are two broad ways in which clients can learn that a domain should be accessed strictly via

[1]Let's Encrypt, for instance, offers a command-line option for reconfiguring the web browser to turn on HSTS headers, but cannot automatically request addition to the HSTS preload list.

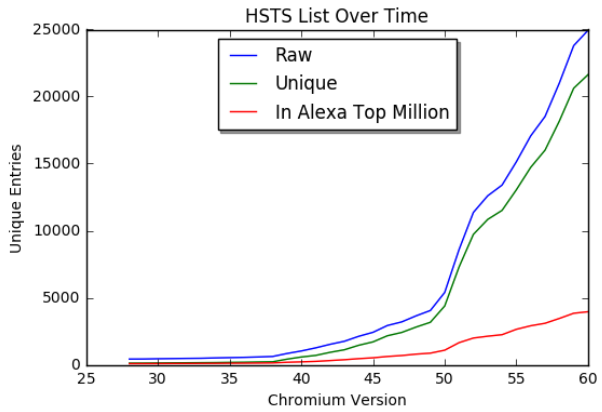

Figure 1: Growth of Chrome's HSTS preload list across versions.

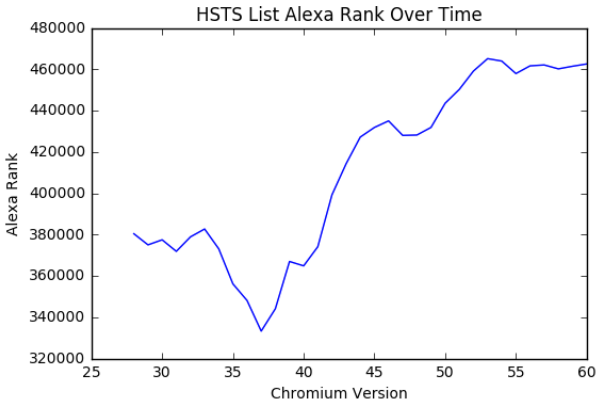

Figure 2: Average Alexa rank of domains in HSTS preload list. (The y-axis does not start at zero.)

HTTPS: at the time of first use via an HSTS header from the domain itself, or *a priori* via a "preload" list, typically made available by browsers. To study both of these delivery mechanisms, we make use of two datasets: *First*, we obtain Chrome's historic preload lists dating back to version 28, when it was first introduced to version 60 today. This is representative of browsers, as all major browsers reference this original list. *Second*, we use HTTPS transfers from the Alexa top-1M most popular websites, made available by the Censys project [5, 3]—this provides us with the IP addresses of who hosts the domains, the certificates provided as part of the TLS handshake, and it allows us to evaluate those domains that issue HSTS headers but have not been added to Chrome's preload list.

3.2 Use of the Pre-Load List

Growth of the preload list We begin by investigating who is using the HSTS preload list. Figure 1 shows the number of unique domains included in Chrome's preload list as a function of Chrome version. We make two key observations. *First*, the overall rate of adoption of HSTS preload lists is increasing extremely quickly. Kranch and Bonneau [12] performed their study of HSTS deployment in 2015, approximately when Chrome was in version 32. During that time, there were only slightly over 1,000 domains in the list, roughly a quarter of which corresponding to Google domains. Shortly thereafter, there began a steady rise in the overall number of entries; today, we see over 21,000 unique entries, over a 20× increase since the previous study.

Second, the bulk of the HSTS preload list comprises unpopular websites: approximately 80% of the entries are not within the Alexa top-1M. Although the number of entries from the Alexa top-1M most popular websites is increasing steadily, less popular websites far outpace them.

Popularity in the preload list To understand why this list is growing so quickly, we sought to determine whether it may be because a small number of organizations in ownership of many domains were simply adding many entries. To this end, we determine which organization owns the domains using techniques described by Cangialosi et al. [2], and group the domains by their owning organization—we present these as the "unique" line in Figure 1. Only 4,100 entries were found in the organization mapping list, but out of these roughly 3,700 or 90% were unique organizations. This indicates that the growth of the preload list is in fact *not* due to some small number of content providers in ownership of many domains.

Among the top-1M most popular websites, we next ask: is the increased adoption similar skewed towards the less popular sites? To answer this question, we first look at the average Alexa ranking among the domains in the HSTS preload list who are in the Alexa top-1M. Looking at this over time, Figure 2 shows that, other than an initial dip towards more popular websites early in HSTS's deployment, the overall trend is for less popular Alexa websites to have a broader HSTS deployment. Figure 3 presents the distribution of HSTS deployment over Alexa rank, looking only at the latest version of the preload list. This shows that, although the trend may be towards increasing deployment among unpopular sites, the most popular 50K domains have the largest deployment rate. We speculate, however, that this is heavily weighted towards Google's domains (who, recall, have long been part of the preload list).

Getting off the preload list Although the data spans nearly two full years and tens of thousands of domains, only 160 have ever been removed from the HSTS preload list. Anecdotal evidence suggests that being removed from this list takes months, though we do

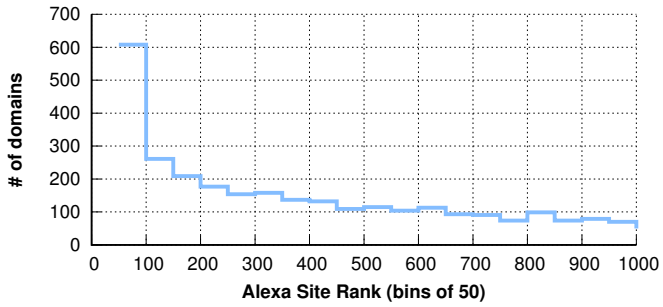

Figure 3: HSTS deployment as a function of Alexa rank; derived from the latest preload list.

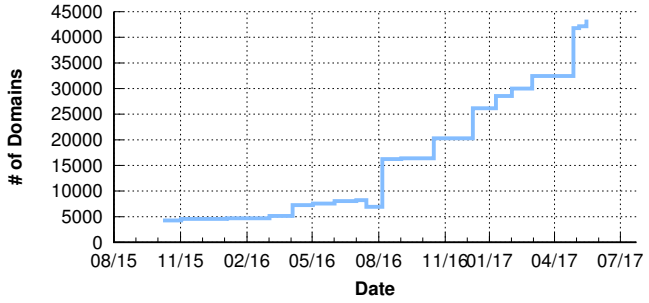

Figure 4: Number of domains in the Alexa top-1M hosting HSTS headers over time.

not envision a good security reason for being removed from the HSTS list, so we do not view this as a barrier.

3.3 Use of HSTS Headers

The second broad way of disseminating HSTS information is through time-of-first-use HSTS headers: when a client visits a website, it can send HTTP headers instructing the client whether to require HTTPS, whether to require it for all subdomains, and so on.

Growth of HSTS header deployment Figure 4 shows the number of domains in the Alexa top-1M that issue HSTS headers, since October 2015. We see a stead, accelerated increase similar to that of the preload list, but with more than 8× more domains from the Alexa top-1M than in the preload list.

Overlap with the preload list It seems natural to assume that anyone who would have gone through the manual effort of getting onto the preload list would also host HSTS headers. In fact, to get onto the HSTS preload list, websites are required to offer HSTS responses to their base domains [11]. Among all of the domains in the latest HSTS preload list, 2,669 of them are also available in our HTTPS scans dataset. Of these, 2,146 (80%) provide HSTS headers. It is not clear to us at this time why 20% of the domains on the preload list would not also serve HSTS headers—one possible explanation is that the domains wish to no longer require HSTS but have difficult getting off of the list; another possible explanation is that the website operator reasons that they need not provide HSTS headers if most major browsers are likely to have the preload list already. Whatever the cause, this shows that *HSTS preload list maintainers do not appear to be checking their own requirements*, at least not after initially being added to the list.

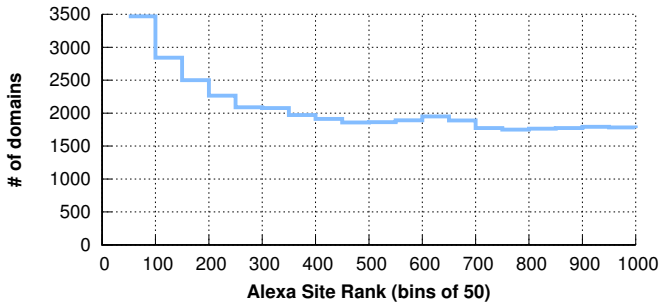

Figure 5: HSTS header deployment as a function of Alexa rank; derived from active scans.

Popularity among HSTS headers We next investigate how website popularity is reflected in HSTS header deployment. Figure 5 shows the number of domains serving HSTS headers as a function of (binned) Alexa rank. Overall, we see a similar trend to that of the HSTS preload list (Fig. 3), with more popular sites being more likely to serve HSTS headers, but with two key differences. *First*, the raw numbers differ greatly, with nearly an order of magnitude more domains offering HSTS headers than being on the preload list. This demonstrates the importance of time-of-first-use, and that for most domains, we cannot rely on the preload list to avoid MiTM attacks at first use. *Second*, the disparity between the highest and lowest ranked websites is far less in HSTS header deployment than in the preload list; this more even distribution reflects the lower barrier of entry for even unpopular websites to host HSTS headers as compared to getting onto the preload list (and knowing about it in the first place).

We have also verified that the Alexa top-1M websites are more likely to deploy HSTS headers than other, less popular websites. Only 0.6% of all web servers from Censys IPv4 scans use HSTS, significantly lower than the average for the top million. However, an adoption rate of just 7% for the Alexa top-50K shows that HSTS header use still has considerable room for growth.

Provider influence Finally, we seek to understand the root cause behind the increased deployment of HSTS. Is it due to a disparate group of motivated, security-conscious users, or is there perhaps a small set of hosting providers adding HSTS header support to all of their customers' domains?

Provider	#Domains	#HSTS
CloudFlare, Inc.	24,396	1,541 (6.3%)
Amazon.com, Inc.	14,357	1,732 (12%)
OVH SAS	11,297	571 (5.1%)
Hetzner Online AG	10,206	798 (7.8%)
Unified Layer	9,262	105 (1.1%)
GoDaddy.com, LLC	7,676	248 (3.2%)
SoftLayer Technologies	7,305	280 (3.8%)
Rackspace Hosting	7,044	526 (7.5%)
Google Inc.	5,580	138 (2.5%)
SAKURA Internet Inc.	5,322	82 (1.5%)

Table 1: Top 10 hosting providers in our dataset, and how many of their domains they serve with HSTS.

Table 1 shows the top ten providers in our (Alexa top-1M) dataset, and the number and fraction of domains they serve with HSTS. Given that many CDNs such as CloudFlare operate their customers' websites—including managing many of their certificates [2]—we hypothesized an "all or nothing" distribution of HSTS support among them, but surprisingly we find a slow, somewhat even (consistently non-zero) distribution among the top providers.

From these results, we conclude that there is indeed some provider influence driving the deployment of HSTS headers, and anticipate increased deployment as they roll out these features to more customers.

HSTS expiry time The HSTS header allows the domain to set a `max-age` parameter, which specifies for how long the browser will remember the policy. Until the time is expired, the browser will only make HTTPS requests. Figure 6 shows the distribution of these expiry times across the latest scan of Alexa top-1M domains. The results show that a majority of domains set a very long `max-age`, with most taking on values of 6 or 12 months. Conversely, 9.6% of domains set an expiry time of one day or less; such short expiry times do not seem likely to help clients who may not be able to visit the website from a trusted location (without a MiTM) on a daily (or more frequent) basis. Not shown in the figure is that 1.4% have max-age over two years, with the longest age of *two thousand years*. We believe this to be a misconfiguration wherein the website administrator may have entered *milliseconds* instead of seconds.

4. HPKP DEPLOYMENT

Like HSTS, the HTTP Public Key Pinning (HPKP) protocol is a new addition to the PKI ecosystem. When studied by Kranch and Bonneau [12], there were only a few dozen domains in the Alexa top-1M sites providing the HPKP header. This, too, has increased by an order of magnitude.

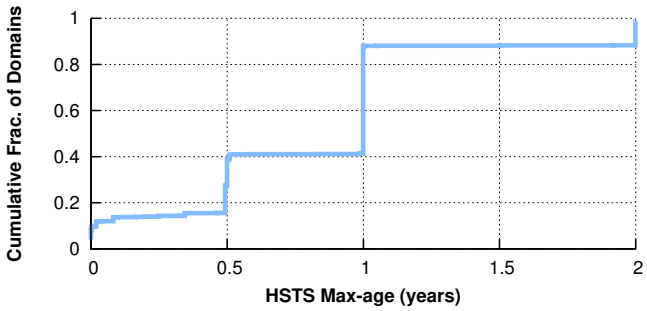

Figure 6: Distribution of the expiry time on HSTS headers.

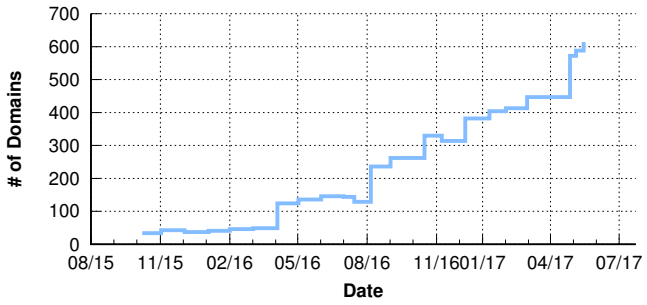

Figure 7: Number of domains in the Alexa top-1M supporting HPKP since October 2015.

4.1 Prevalence of HPKP Headers

Growth of HPKP header deployment We begin by examining how HPKP deployment has grown since the beginning of our dataset in October 2015. Figure 7 shows the number of HPKP-enabled domains over time. As with the study by Kranch and Bonneau, our initial data from October, 2015 showed that 34 of the Alexa top-1M domains had deployed the HPKP header. Since then, the deployment has been increasing at a steady, albeit modest, rate, with 588 domains as of May 2017.

One possible explanation for the lack of wide deployment at this time could be that HPKP is not widely known about or is not properly understood. Our data reveals that the highest rank domain that uses HPKP is Github (Alexa rank #63). The average rank of site using the HPKP header is 436,950 suggesting that HPKP is often used by smaller domains. It is unclear why some domains opt for implementing the HPKP header and others are on the preload list. What is clear is that there is no uniformity to who uses key pinning.

Overlap with the HSTS preload list Recall that the HSTS preload list contained 2,669 distinct domain names. Comparing the latest Censys scan's prevalence of HPKP, we are able to find only 97 (3.6%) domains also on the HSTS preload list. This fraction shows that

Provider	#HPKP
CloudFlare, Inc.	17
Hetzner Online AG	13
OVH SAS	10
SuperNetwork s.r.o	6
HLL LLC	6

Table 2: Hosting providers serving the most HPKP-enabled domains.

HSTS and HPKP are not currently being deployed in tandem, least of all among those who are on the HSTS preload list.

Provider influence We next seek to understand if providers are the main driving force behind HPKP use. Table 2 presents the hosting providers serving the most HPKP-enabled domains. Very few hosting providers serve more than two domains with HPKP.

We manually inspected some of these HPKP-enabled domains in CloudFlare. Typically, CloudFlare uses so-called "cruise-liner certificates," on which they place dozens of their customers (ostensibly to mitigate administrative overhead and to decrease the number of IP addresses they have to purchase) [2]. Of the HPKP-enabled sites we inspected, the websites were the *only* customers on the certificates. `plannedparenthood.org`, for instance, appears as the sole entry in the SAN list (with a CloudFlare domain as the common name); `atlantic.net` and `coinbase.com` each have certificates where they are the common name (and in which CloudFlare does not appear). This indicates a different class of product that CloudFlare is offering, and that HPKP is mostly used by domains without shared certificates. As with HSTS, we believe this represents a higher tier of service that CloudFlare offers its customers.

HPKP max-age Like HSTS, HPKP also has a `max-age` expiry time. We present their distribution in Figure 8. It is recommended that domains set a low `max-age` when testing out a new configuration, so that errors do not bring the website down for long. However, once a working configuration is set, the max -age value should be set higher, to avoid having to re-do the process frequently. Figure 8 shows that over 80% of all HPKP-enabled domains have a `max-age` of two months or less.

Number of pinned keys Proper use of HPKP dictates that a domain should pin at least two keys, so that if one of them has to be revoked, then the other can be brought online while maintaining at least one correct pinned key. Table 3 shows the number of pinned keys we see in Censys scans. We see that 13% do not have the requisite number of keys, demonstrating a basic misunderstanding (and lack of sanity checking) of the purpose HPKP headers are intended to serve.

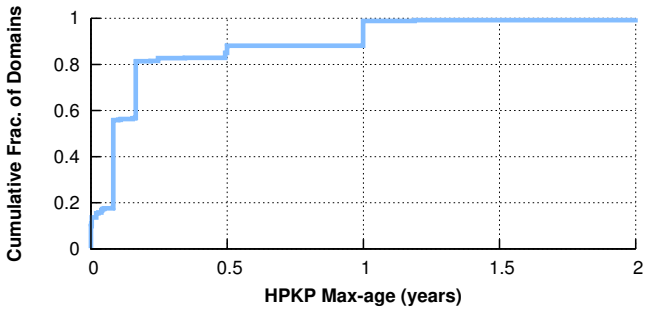

Figure 8: Cumulative fraction of HPKP expiry time.

#Pins	0	1	2	3	4	5	6	7	8	>9
#Domains	3	77	340	92	46	24	11	13	4	4

Table 3: Frequency in the number of keys pinned in an HPKP header. According to RFC 7469, there must always be at least two pins.

5. CONCLUSION

HSTS and HPKP are new additions to the HTTPS ecosystem, and serve a unique, important role at protecting against attacks that TLS's mechanisms alone are unable to. Our analysis using historical HSTS preload lists and active scans of the Alexa top-1M websites show that, since a recent study only two years ago [12], the deployments of HSTS and HPKP have grown considerably, by over an order of magnitude each. Our results indicate that hosting providers are the driving force behind a considerable fraction of these deployments. Moreover, we find instances of erroneous configurations, such as pinning fewer than two keys in HPKP, that we believe reflect a basic misunderstanding of how these defense mechanisms work. Taken together, our results indicate the importance of monitoring and supporting early deployment of new security mechanisms.

6. REFERENCES

[1] Crlsets. The Chromium Projects, 2015. `http://bit.ly/1JPsUeC`.

[2] F. Cangialosi, T. Chung, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. Measurement and analysis of private key sharing in the https ecosystem. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 628–640. ACM, 2016.

[3] Censys. `https://censys.io/`.

[4] P. Chen, N. Nikiforakis, C. Huygens, and L. Desmet. A dangerous mix: Large-scale analysis of mixed-content websites. In *Information Security*, pages 354–363. Springer, 2015.

[5] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman. A search engine backed by Internet-wide scanning. In *ACM Conference on Computer and Communications Security (CCS)*. 2015.

[6] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, and V. Paxson. The matter of heartbleed. In *ACM Internet Measurement Conference (IMC)*, 2014.

[7] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the https certificate ecosystem. In *ACM Internet Measurement Conference (IMC)*, 2013.

[8] C. Evans, C. Palmer, and R. Sleevi. Public key pinning extension for http. RFC 7469, Apr. 2015. `http://www.ietf.org/rfc/rfc7469.txt`.

[9] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In *USENIX Security Symposium*, 2012.

[10] J. Hodges, C. Jackson, and A. Barth. Http strict transport security (hsts). RFC 6797, Nov. 2012. `http://www.ietf.org/rfc/rfc6797.txt`.

[11] Hsts preload list form. `https://hstspreload.org/`.

[12] M. Kranch and J. Bonneau. Upgrading https in mid-air: An empirical study of strict transport security and key pinning. In *Network and Distributed System Security Symposium (NDSS)*, 2015.

[13] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. Crlite: A scalable system for pushing all tls revocations to all browsers. In *IEEE Symposium on Security and Privacy*, 2017.

[14] B. Laurie, A. Langley, and E. Kasper. Certificate transparency. RFC 6962, June 2013. `http://www.ietf.org/rfc/rfc6962.txt`.

[15] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, A. Schulman, and C. Wilson. An end-to-end measurement of certificate revocation in the web's pki. In *ACM Internet Measurement Conference (IMC)*, 2015.

[16] Revoking intermediate certificates: Introducing OneCRL. `http://mzl.la/1zLFp7M`.

[17] A. Schulman, D. Levin, and N. Spring. Revcast: Fast, private certificate revocation over fm radio. In *ACM Conference on Computer and Communications Security (CCS)*, 2014.

[18] T. Van Goethem, P. Chen, N. Nikiforakis, L. Desmet, and W. Joosen. Large-scale security analysis of the web: Challenges and findings. In *International Conference on Trust and Trustworthy Computing*, pages 110–126. Springer, 2014.

[19] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. A. Halderman. Towards a complete view of the certificate ecosystem. In *ACM Internet Measurement Conference (IMC)*, 2016.

[20] L. Zhang, D. Choffnes, T. Dumitraş, D. Levin, A. Mislove, A. Schulman, and C. Wilson. Analysis of ssl certificate reissues and revocations in the wake of heartbleed. In *ACM Internet Measurement Conference (IMC)*, 2014.