2018 10th International Conference on Cyber Conflict
CyCon X: Maximising Effects
T. Minárik, R. Jakschis, L. Lindström (Eds.)
2018 © NATO CCD COE Publications, Tallinn

HTTP Security Headers Analysis of Top One Million Websites

Artūrs Lavrenovs
NATO CCD COE
Tallinn, Estonia
arturs.lavrenovs@ccdcoe.org

F. Jesús Rubio Melón
Spanish Joint Cyber Defence Command
Madrid, Spain
jrubio@isdefe.es

Abstract: We present research on the security of the most popular websites, ranked according to Alexa's top one million list, based on an HTTP response headers analysis.

For each of the domains included in the list, we made four different requests: an HTTP/1.1 request to the domain itself and to its "www" subdomain and two more equivalent HTTPS requests. Redirections were always followed. A detailed discussion of the request process and main outcomes is presented, including X.509 certificate issues and comparison of results with equivalent HTTP/2 requests.

The body of the responses was discarded, and the HTTP response header fields were stored in a database. We analysed the prevalence of the most important response headers related to web security aspects. In particular, we took into account Strict-Transport-Security, Content-Security-Policy, X-XSS-Protection, X-Frame-Options, Set-Cookie (for session cookies) and X-Content-Type. We also reviewed the contents of response HTTP headers that potentially could reveal unwanted information, like Server (and related headers), Date and Referrer-Policy.

This research offers an up-to-date survey of current prevalence of web security policies implemented through HTTP response headers and concludes that most popular sites tend to implement it noticeably more often than less popular ones. Equally, HTTPS sites seem to be far more eager to implement those policies than HTTP only websites. A comparison with previous works show that web security policies based on HTTP response headers are continuously growing, but still far from satisfactory widespread adoption.

Keywords: *web security, HTTP headers, top one million websites survey, X.509 certificate, HTTP/2, HTTPS, HTTP Strict Transport Security, Content Security Policy*

1. INTRODUCTION

The main goal of this research is to assess the current adoption rate of security policies based on HTTP response headers on the most popular Internet websites. Declarative web security through HTTP response headers constitute a powerful and easy way to enhance website security, while relatively little effort is required from website operators. It has been a recurrent research topic, aided by the fact that the nature of the World Wide Web makes data publicly accessible to any interested party and that the WWW itself is continuously growing and evolving.

Besides measuring security headers adoption in popular websites, we set out to understand it in a deeper way by trying to find correlations between adoption rates and variables like HTTPS usage and popularity rank position. We want to gain insight into why and how policies based on HTTP headers are adopted. As will be shown, the most popular a website is, the more likely it will apply security through HTTP headers. Those sites also tend to be more prone to favouring HTTPS protocol over HTTP.

Regarding the structure of this paper, in the first section we present a brief literature review concerning different past security analysis and current online efforts. Next, we proceed to describe in detail the data set that served as the basis for this research. We then show our results for all analysed HTTP response headers, and we conclude with a "conclusions" section where we summarize our findings and a last section on planned future work.

2. RELATED WORK

Extensive analysis of Content Security Policy (CSP) adoption among the top one million websites is provided by Ying et al. (2015). It was found that CSP is used in less than 0.2% of the sites, and oftentimes incorrectly. They also investigated other relevant security related headers. In particular, they found that *X-XSS-Protection, X-Frame-Options* and *Strict-Transport-Security* headers were implemented, respectively, in about 4.4%, 4.1% and 1% of the websites they analysed. Despite the low adoption rate of HTTP security related headers found by Ying et al., their results show a noticeable

increase in the adoption rates observed over research done previously by Weissbacher et al. (2014). In fact, they conducted the first CSP adoption study of the Top One Million websites in 2012-2014 and found that CSP was used in less than 0.1% of sites. Other security-related HTTP headers, like *X-XSS-Protection*, *X-Frame-Options* and *Strict-Transport-Security* were seen on 4.6%, 4.1% and 0.3% of the websites, respectively. Although both of these papers primarily concentrate on CSP adoption rate and related implementation issues, they analysed other security headers as a by-product.

Chang et al. (2017) investigated the "redirection trail", which basically consists of a set of pairs, each one formed by the Location header combined with redirection HTTP status codes. Combining this redirection trail with other data readily available, like protocol and host, allowed the researchers to evaluate the security of the Top One Million websites. They found that 20.5% of them contained some configuration inconsistency related to redirection requests that could be exploited by the adversary. Sood et al.(2011) conducted a research in 2011 among the world's top 43 banks. They found that none of them implemented the HTTP security related headers available at that time.

Response HTTP header analysis from a security standpoint is also present outside academic literature. Scott Helme's (2017) website has published multiple times research on security headers prevalence and HTTPS adoption in the Alexa Top One Million websites. The latest one we know of, at the time of this writing (October 2017), is from August 2017. He has been reporting positive trends of adoption rates of most common HTTP headers. Additionally, his website (IO) provides a public tool that enables checking of security headers for any website. Based on these results, the tool assigns a given grade, from A to F, for the provided website. A similar tool is provided by Mozilla Observatory that also gathers statistics from executed checks and estimates that about 10% of the checked websites follow good practices regarding security header configuration (Mozobs). April King (2017) has conducted similar research on Alexa Top One Million websites and found similar results about positive trends.

3. THE DATA SET

A. URL Set

This research makes use of Alexa "top one million" website list (1M) as the source for domains to be analysed. For each domain contained in the list, we made four HTTP/1.1 requests: to http://domain, https://domain, http://www.domain and https://

www.domain. Timeout for connection establishment was set to 60 seconds, and response timeout was also set to 60 seconds.

B. Data collection approach

We developed a custom Python tool based on Python requests library (Pyreq). After an HTTP/1.1 response arrived, only HTTP response headers and status code were saved to a relational database. The response body was disregarded. In order to mimic real users, we set *User-Agent* and other request headers to match exactly those of the Mozilla Firefox browser (version 50.0 on Ubuntu 17.04). For all the requests we followed redirections, saved them all, and created convenient relationships between them. Finally, duplicate URL requests that arose from redirections were removed from the dataset.

Preliminary testing revealed that using Certification Authorities (CA) and Intermediaries lists bundled inside Ubuntu were not sufficient for HTTPS requests. Therefore, we made use of public CA lists Mozilla CA (Mozca) and Mozilla Intermediaries (Mozinter) (they are both internally used by the Firefox browser). Several full scans were performed during August and September 2017, and we always updated website and CA lists right before the scanning process. In this paper we will exclusively refer and analyse data gathered between September 1st and 4th, 2017. After the scan was completed, we retried those websites that had failed all of our four requests, since it just might indicate temporary network issues.

C. Data Overview

Our final dataset contained 3.135.962 recorded responses with unique URLs (either the protocol, the domain or the subdomain was different). At least one response was received from 975.729 websites (97.5% of all one million domains). Only 2.558 websites were successfully processed during the retrying process (to allow for network issues). We observed large amounts, up to 1.4 million, of duplicate URL records caused by redirection to an already visited URL. They were all removed from the database. We obtained about 27% more responses from www-subdomains than from direct domain requests.

For our current analysis we have considered only unique URL responses with HTTP response status code 200, which amounts to 1.478.750 records.

D. Data quality

The Alexa top one million list was chosen because it is a large list, but not “too large”, thus data collection can still be achieved in a few days or less. Moreover, the list contains the most popular websites, an attractive target for attackers and security researchers alike. As pointed out previously, it has been repeatedly used in various

web security surveys. Alternative lists, like the ones by Majestic (Majestic) or Cisco (Umbrella) also provide one million most popular sites, although to compare our work with previous results we have adhered to Alexa's list.

However, Alexa's list, despite its usefulness, has some caveats. It makes use of proprietary ranking and domain processing algorithms not fully disclosed and we have observed inconsistencies within the list: it contains many domains that cannot be accessed directly (typically because there is no DNS entry for them, like in *cloudfront.net*), but can be through the www-subdomain. Additionally, a significant set of entries in the list are actually subdomains (most common websites are *tumblr.com*, *blogspot.com* and *wordpress.com* with 5.698, 2.904 and 2.696 respective subdomains). Although content is different on these subdomains, these platforms usually provide little to no control for header configuration to final website authors. In fact, all of their subdomains will share the same security headers. Even some apparently unrelated domains will share the same headers configuration because they are hosted by these providers, although their domain name is totally unrelated to the hosting server.

E. Response Codes

Status code distribution observed in the responses for both HTTP and HTTPS requests are presented in Tables 1 and 2. As clearly seen from these data, most websites seem to prefer the www subdomain to the plain domain name, regardless of the protocol (47% of HTTP sites and up to 63% for HTTPS ones). As for redirections, we have observed that 45.7% of HTTP domain requests redirect to the corresponding www subdomain, and 15.5% of HTTPS domain requests point to the www subdomain. Most of the remaining requests are server-side errors either intermittent or permanent (e.g., web servers which are not properly configured to handle the domain or subdomain requests).

TABLE 1. STATUS CODES FOR HTTP REQUESTS

Domain responses		WWW subdomain responses	
status	**count**	**status**	**count**
301	46,8%	200	47.3%
200	38.9%	301	39.5%
302	12.0%	302	11.0%
403	0.7%	404	0.6%
404	0.7%	403	0.6%

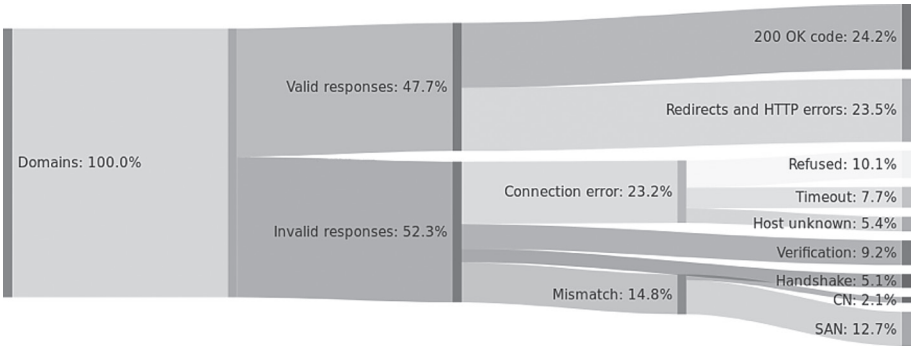
TABLE 2. STATUS CODES FOR HTTPS REQUESTS

Domain responses		WWW subdomain responses	
status	**count**	**status**	**count**
200	47.3%	200	62.7%
301	39.5%	301	26.0%
302	11.0%	302	8.4%
404	0.6%	403	0.7%
403	0.6%	404	0.6%

F. HTTPS Subset

Our data set allowed for a detailed analysis of HTTPS deployment since we stored all failed requests and their associated error messages. The result for over two million HTTPS requests are presented in the Figure 1.

FIGURE 1. HTTPS RESPONSES



Only about half of the scanned domains and www subdomains (47.7%) are properly configured for HTTPS. That number includes 24.2% of sites that are actually responding with 200 OK status code and a substantial number of redirects and HTTP errors (23.5%).

Many sites, 23.2% of all HTTPS requests, do not respond to HTTPS at all, either because of the TCP connection being refused, timeout or missing DNS records.

We found a sizeable number of cases, 29.1% of all HTTPS requests, where it was possible to establish a TCP connection on port 443, but HTTPS ultimately failed. The reason for failure is related to verification errors (mostly expired certificates, self-signed, signed by untrusted CA's or malformed), handshake errors (usually outdated

protocols) and hostname mismatching. A 29.1% rate is remarkable: it implies that a large number of web servers are already somehow configured to handle HTTPS requests, but have mostly missed the step of acquiring and installing the correct X.509 certificate (even though nowadays it is possible to quickly obtain a certificate for free, for example from the highly popular "*Let's Encrypt*" (Lets) online certification authority).

Regarding handshake errors, 5.1% of all HTTPS requests, we found that there is a small number of websites that work properly when requested by the real Mozilla Firefox browser but not by our software. We traced back that behaviour to outdated and misconfigured server sites that are still supported by the browser for backwards compatibility, but not by the OpenSSL 1.0.2g library we used in our scanning software.

Host name mismatching happens in about 14.8% of all HTTPS requests we made. That can happen because there is no Subject Alternative Name (SAN) and the requested host does not match certificate's Common Name (CN) or because, even though CN and SAN are both present, the hostname does not match either of them. This latter cause is more common (12.7%) than the former (2.1%). Name mismatching is typically found in shared environments where several websites run on a single server that oftentimes issues a "default" SSL certificate (as with the well-known shared hosting provider *Hostgator*). The most common reason for name mismatching is that CN is set to either *www.domain* or *.*domain*, and therefore certificate validation fails for the https://domain request (like the high ranked website *ups.com*).

G. HTTP/2 Analysis

Our data gathering procedure, and the subsequent response headers analysis, is entirely based on HTTP/1.1 requests. However, HTTP/2 is quickly growing in popularity and it may replace HTTP/1.1 as the main web protocol in the near future. Different protocol versions might be somehow correlated with different security settings (due, for example, to different security awareness). That raises the question whether there are different response headers, or different headers values, in HTTP/1.1 and HTTP/2 data subsets. To answer this question, we used the same Alexa top one million websites list (1M) and followed the same data gathering approach, but this time making an additional HTTP/2 request to each website's domain and www-subdomain. We made use of Python hyper library (Hyper). If an HTTP/2 request was successful, we made the equivalent HTTP/1.1 request to the same URL and compared the HTTP headers and their values for both responses. To simplify HTTP header comparison, we did not follow redirects and did not analyse response status codes. Furthermore, we did not take into account the fact that multiple backends can serve a single domain or www subdomain (in principle, those servers could have different configurations and that might produce different HTTP headers).[1]

1 Those backends could be either serving requests using single IP address or multiple IP addresses, but we chose not to manipulate to which IP addresses HTTP requests are being sent.

The resulting dataset consists of 746.758 records, totalling 211.638 unique domains that support HTTP/2 protocol (21% of all Alexa top one million websites). This percentage is a bit higher than the figure reported by the *w3techs* portal, 17%, as the HTTP/2 support rate across all the world wide web (W3tech). However, it is coherent with the fact that we are analysing most popular websites, not all existing ones. The failure rate of HTTP/1.1 requests to same domain following successful HTTP/2 requests is insignificant (0.26%).

1) Missing headers

We analysed the top 10 HTTP headers missing from HTTP/2 responses but present in HTTP/1.1 responses, and vice versa. The results, header names and their missing count in their counterpart protocol requests, are presented in Table 3. As might be expected, most significant differences are related to headers used in establishing and maintaining the HTTP/1.1 connection (those headers are unneeded in HTTP/2). Fortunately, none of these missing headers are related to any security issue.

Regarding security related headers, some may be missing in one version of the protocol, but present in the other, although the numbers are insignificant in all cases. For example, *X-XSS-Protection* response header is missing in 28 HTTP/2 requests that issue it in the equivalent HTTP/1.1 requests. Similarly, 45 HTTP/1.1 requests did not contain that header, although it was present in the equivalent HTTP/2 ones. We found that no common misconfiguration pattern is distinguishable, and most common cause could be attributed to responses coming from different backends serving the same domain name, but different protocol.

TABLE 3. RESPONSE HEADERS NAMES COMPARISON

Missing in HTTP/2	count	Missing in HTTP/1	count
connection	350152	content-length	11082
transfer-encoding	262147	link	3076
keep-alive	28559	pragma	1080
upgrade	5816	set-cookie	1058
cache-control	3014	vary	672
content-length	2983	cache-control	640
last-modified	1987	expires	614
x-nananana	1137	x-pingback	415
content-encoding	1107	accept-ranges	405
vary	1084	x-litespeed-cache-control	297

2) Different values in HTTP headers

The top 20 response headers that carry different values in equivalent HTTP/1.1 and HTTP/2 requests are presented in Table 4.

TABLE 4. RESPONSE HEADERS VALUES COMPARISON

Missing in HTTP/2		Missing in HTTP/1.1	
Header	**count**	**Header**	**count**
set-cookie	215265	last-modified	5229
cf-ray	183755	x-served-by	4921
date	181046	x-timer	4838
expires	23789	vary	4756
x-cache	14628	content-encoding	4706
server	10732	x-contextid	4434
content-length	9244	x-servedby	4384
x-varnish	6902	x-request-id	4295
via	6865	x-via	4283
x-amz-cf-id	6308	x-cache-hits	3664

Set-Cookie differences are due to different session identifiers. The differences in the response headers *Cf-Ray, X-Cache, X-Varnish, Via, X-amz-cf-id, X-Served-By, X-Timer, X-Contextid, X-Servedby, X-Request-Id, X-Via* and *X-Cache-Hits* are due to debug information, usually set by cloud providers and caching frontend servers. *Date, Expires* and *Last-Modified* response headers contain timestamps that are usually one second apart, in agreement with the fact that the requests are made consecutively. *Content-Encoding* differences are due to *Brotli*, the compression algorithm used for HTTP/2. Differences in the *Vary* header value are related to different compression algorithms. *Content-Length* variations are caused by the dynamic nature of the generated content. Nevertheless, none of these headers can be related to any security risk, and the variations are meaningless from the point of view of our security analysis.

Regarding security related headers, only *Content-Security-Policy* and *X-XSS-Protection* showed any significant count difference in 401 and 358 of the requests, respectively. Almost all of the CSP differences lie either in nonce tokens or report URI identifiers. *X-XSS-Protection* differences can be always traced back to different report URL's found in the value of the header.

Server header values show some differences between the protocol versions and in most cases it is irrelevant (variations of nginx server identified by names like

openresty, Tengine, kinsta-nginx). For example, 68.6% of HTTP/2 requests carried nginx as the Server value, but openresty for the equivalent HTTP/1.1 requests (77.2% of these cases correspond to tumblr.com subdomains). However, in about 400 cases we identified obvious attempts to try to conceal the server name by removing the header or changing it to an undescriptive one in one of the protocol version, but not in the alternative one.

In summary, regarding versions 1.1 and 2 of the HTTP protocol, no significant HTTP response headers variations were found from the security perspective. The only noticeable risk that shows some correlation with protocol version is information leakage via *Server* response header. Additionally, a potential security risk could arise due to inconsistent configuration management across sets of backend servers (which still could be useful to an attacker). However, this issue requires further investigation and lies outside the current research.

4. HTTP RESPONSE HEADERS ANALYSIS

As stated previously, the evaluation of the security of the websites is done through an analysis of the HTTP headers sent from the web server. Some HTTP headers, among all possible server-side headers, were devised to instruct the web browser to protect the web application against certain security threads. Accordingly, their analysis constitute the basis of our current research. Additionally, a few HTTP server-side headers may carry information about the web application that potentially can help an attacker to perform malicious actions. They will also be analysed as part of our research.

The headers involved in each group are the following:

Security headers:

- *Strict-Transport-Security*
- *Content-Security-Policy* (and related *Content-Security-Policy-Report-Only*)
- *X-XSS-Protection*
- *X-Frame-Options*
- *Set-Cookie*
- *X-Content-Type*

Information revealing headers:

- *Server* (and related headers)
- *Date*
- *Referrer-Policy*

We have deliberately excluded HTTP Public Key Pinning (HPKP) from our research. Standardized in IETF 7469, HPKP provides a mechanism by which the TLS protocol is protected against Certification Authority (CA) attacks and spoofed certificates. However, it is well known that its implementation poses considerable risks for website operators. It is currently supported by Chrome, Firefox and Opera. Nevertheless, Google has recently announced that it will deprecate it in Chrome in May 2018, and soon thereafter it will be completely removed Palmer 2017. Research by Scott Helme (2017) regarding his own analysis of Alexa Top One Million sites indicates that it is usually implemented wrongly by website operators. Security expert Ivan Ristic (2016) has also pointed out similar concerns about HPKP.

Subresource Integrity (SRI) has been sometimes taken into account in the context of top one million analysis (see Mozilla Observatory (Mozobs) or recent April King results (l2017). By specifying a hash token together with the URL of any given resource on a web page, a browser can check that the resulting content obtained from actually downloading the resource has not been unexpectedly altered. This technique is effective, for example, against attackers manipulating JavaScript libraries located in Content Delivery Networks. Chrome, Firefox and Opera already implement this feature. SRI is a relatively new protective mechanism, and current recommendation is from 2016 (SRI). Despite the undeniable interest in measuring its current adoption rate among the top one million websites, it entails parsing the HTML content found in the body of the HTTP responses, and it lies outside the scope of the current research, centred around HTTP response headers analysis.

A. Security Headers

1) Strict-Transport-Security Header

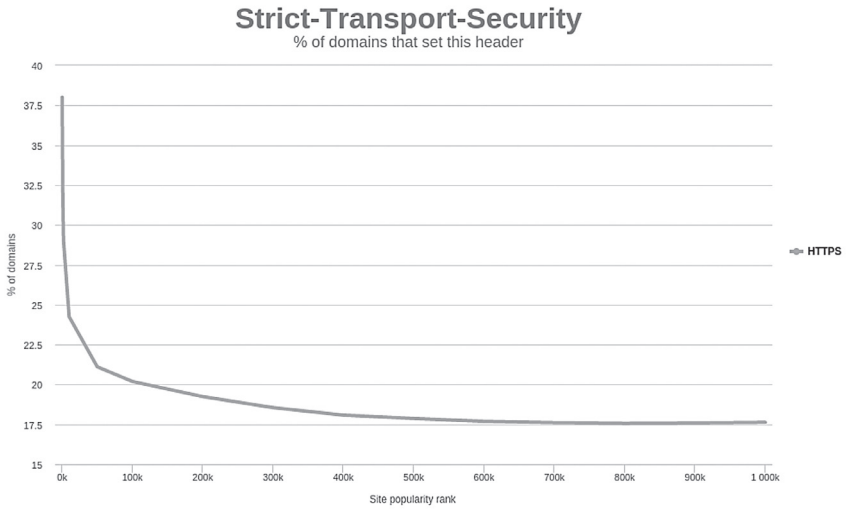
HTTP *Strict-Transport-Security* header (or HSTS, for short) allows a web server to inform the browser that all subsequent connections for all requests should be established exclusively through HTTPS, never through HTTP, using a valid certificate. It helps prevent several man-in-the-middle (MITM) attacks that may arise in different situations. Some common vulnerable situations are the following:

- A user types in a URL in the browser address bar. By default, this URL will be requested by the browser through an HTTP connection, not an HTTPS one.
- By means of social engineering techniques, a user is tricked into clicking on an HTTP link, instead of an HTTPS one, therefore initiating the HTTP request that can be captured by the MITM.
- An attacker sends a fake certificate, hoping that the user will accept it by clicking through the warnings displayed by the browser.
- Forgotten HTTP links scattered throughout the web pages.

All of these vulnerable situations can be avoided by the web application just by issuing this header. In fact, no other server header or web application configuration exists that can prevent these kind of MITM attacks (at least, regarding the first two cases). That makes HSTS a key protective server-side header. The header specification was published in 2012, (RFC 6797).

By parsing the scanning data obtained from the top one million web sites we have found that most websites do not issue any HSTS header. The aggregated results can be seen in Fig 2.

FIGURE 2. HSTS IMPLEMENTATION RATE AS A FUNCTION OF WEBSITE POPULARITY



It is readily appreciated that highly popular sites tend to implement HSTS more often than those sites that are less popular. Nearly 38% of top one thousand sites implement HSTS, while only 17.5% of top one million HTTPS websites implement it. This trend will be recurrent for all headers analysed in this research: the most popular a site is, the more security headers it will tend to implement.

Our numbers are comparable to the ones published by Helme (2017), who reports a 7.3% penetration rate. The difference is due to the fact that Helme's results are referred to the whole dataset, just not to the HTTPS sites (about 40% of all websites). Our 17.5% HSTS implementation rate becomes 7.0% when referred to the whole dataset. April (2017) reports a 4.4% adoption rate in June 2017 (also referred to the whole dataset). We believe, however, that HSTS rates should be referred to the HTTPS subset, since HSTS does make sense in HTTP only sites.

Only about 2% of HTTP websites redirect to an HTTPS site while simultaneously enforcing HSTS policy. Finally, a small number of sites (0.7%) make use of HTTP protocol and respond with a status code of 200, instead of responding with a redirection 300 code.

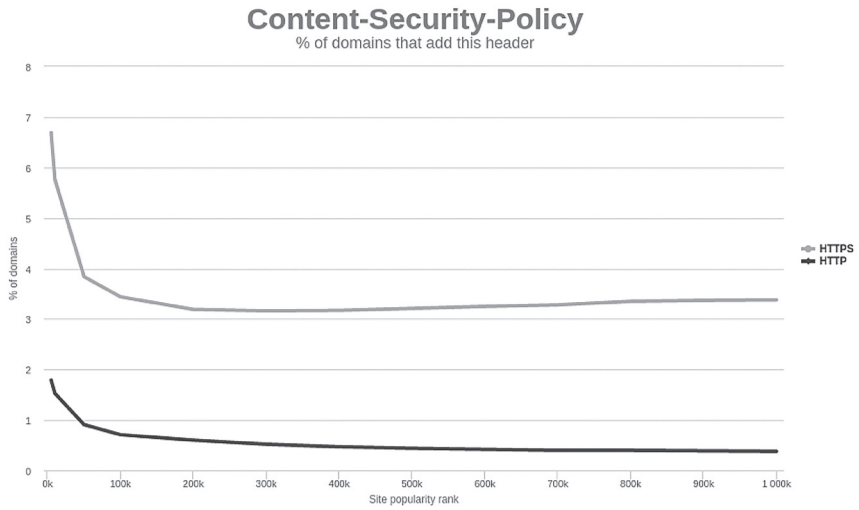
3) Content-Security-Policy Header

Content-Security-Policy (CSP) is a key response header that provides strong defence mechanisms against Cross Site Scripting (XSS) and other client-side injection attacks by whitelisting allowed sources and disabling certain insecure JavaScript features. It can also be used to prevent attacks against HTTPS, mostly those related to inadvertent HTTP links within HTTPS web pages. It has been standardized by W3C, originally in 2012 (CSP Level 1), then revised and augmented in 2015 (CSP Level 2) and currently undergoing a third revision (CSP Level 3). CSP is currently supported by all major browsers, with the exception of Microsoft Internet Explorer which uses the alternative *X-Content-Security-Policy* header.

The header directives, up to 16 in CSP2, offer the possibility of a fine-grained configuration, although at the cost of having to deal with non-trivial setup choices. In fact, due to the growing complexity of client-side scripting code and the large number of different assets handled by web applications (up to hundreds or even thousands of different resources requested from within a given page), the adoption of a CSP policy may result in unexpected glitches. Therefore, most implementation guidelines recommend starting to implement CSP by making use of the related *Content-Security-Policy-Report-Only* response header that allows web administrators to test their CSP policies before they are fully enforced without risking unwanted web application behaviour.

Our results show that CSP is scarcely implemented in HTTPS sites (3.4%) and hardly in HTTP sites (0.4%). The figures for Content-Security-Policy-Report-Only usage are even smaller (0.3% and 0.1% respectively). Globally, including both HTTP and HTTPS sites, CSP is implemented in 1.6% and CSP report only version in just 0.2% of sites. On the other hand, the implementation rate of CSP with respect to the popularity rank follows the same pattern as with other headers: more popular sites choose to issue the CSP header more often than less popular ones. These findings can be easily appreciated in Figure 3:

FIGURE 3. CSP IMPLEMENTATION RATE AS A FUNCTION OF WEBSITE POPULARITY



Our results are similar to the ones by Helme (2017), from August 2017, about 2.0% globally, while significantly higher than April's (l2017), 0.04% in June 2017.

We have also observed that there are significant differences between the directives used in HTTP and HTTPS sites. In fact, for HTTP sites, *frameAncestors* (48.27%), *scriptSrc* (35.96%) and *defaultSrc* (35.72%) are the most common directives. However, HTTPS sites typically issue different directives. The most common ones being: *upgradeInsecureRequests* (61.79%), *reportUri* (53.34%) and *defaultSrc* (20.37%).

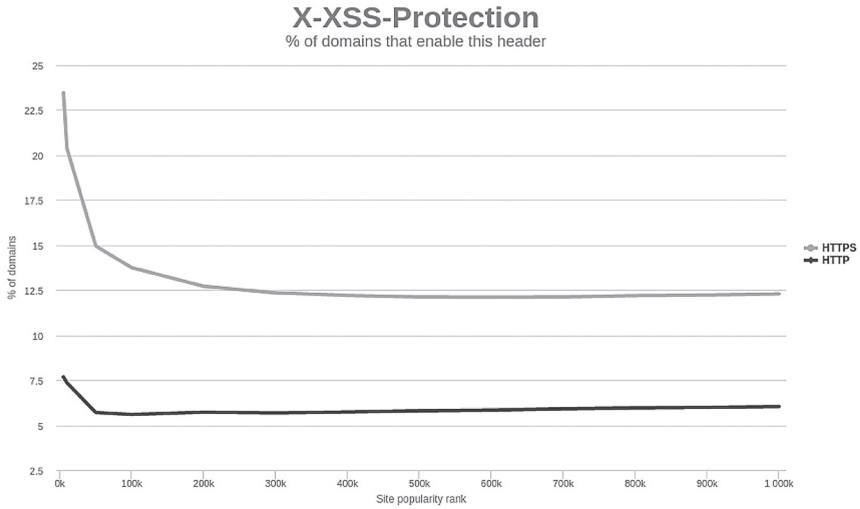
On the other hand, it is interesting to note that the CSP report only version seems to differ from fully enforcing CSP. In fact, the most common directives in HTTP sites are *reportUri* (94.68%), *blockAllMixedContent* (81.73%), *defaultSrc* (13.53%). And for HTTPS sites, most common CSP report only directives are *reportUri* (94.68%), *blockAllMixedContent* (81.73%), *defaultSrc* (13.53%).

4) X-XSS-Protection Header

This header is responsible for toggling off the XSS filter implemented by most current browsers (except, notably, Firefox). By default, the XSS filter is enabled, but website administrators can disable it by setting its value to zero (*X-XSS-Protection: 0*), possibly to prevent the browser from interfering with the desired behaviour of the web application. Web sites that issue that header, and set its value to zero, risk being vulnerable to reflected XSS attacks. Content Security Policy, and in particular CSP level 2 contains a directive, "*reflected-xss*", that completely replaces this header.

Our scanning results for the full one million set show that about 12% of HTTPS sites and 6% of HTTP sites set this header. Most of the times the header is issued so that the browser is granted the right to apply its XSS filter, but in 3% of HTTP sites, and nearly 2% of HTTPS sites, the configuration is such that the sites deny permission to apply the filter. Therefore, as expected, HTTPS sites tend to be more concerned with security. In a similar way, the more popular a site is, the more it will tend to set the header, and will mostly do it so that the browser is granted the right to enable its filter. Both trends can be appreciated in Figure 4.

FIGURE 4. X-XSS-PROTECTION IMPLEMENTATION RATE AS A FUNCTION OF WEBSITE POPULARITY



Our findings are in agreement with a global implementation rate of [Helme2017], 9.3%, and [April2017], 8.1% (none of them break up implementation rates by protocol or popularity of website).

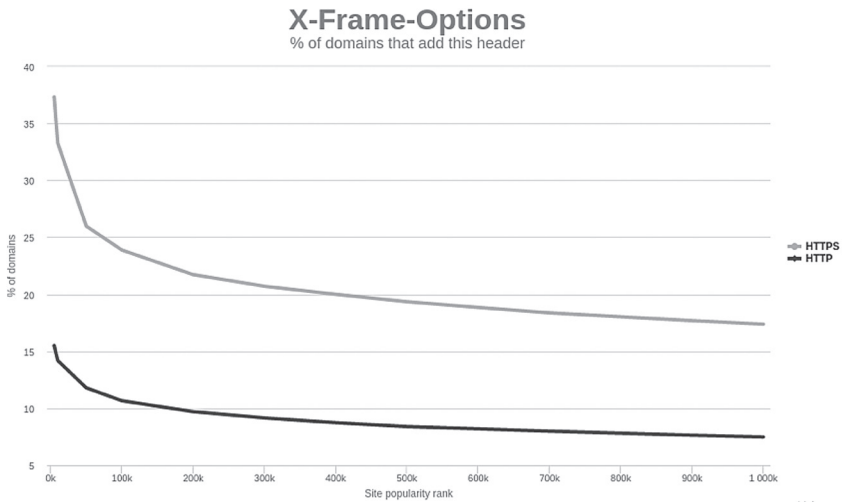
5) X-Frame-Options Header

This header, standardized in RFC 7034, is used to instruct a browser whether a given web page or resource is allowed to appear within a Frame, iFrame or Object, thereby avoiding frame based attacks, like "clickjacking" (for example, rydstedt or OWASPxfo). As with *X-XSS-Protection*, this header is superseded by CSP, which contains a directive, "*frame-ancestors*", that completely replaces *X-XSS-Protection* header. There are three "options", or directives, defined for this header: *deny, same-origin* and *allow-from*.

The results from our scanning survey show once more that HTTPS sites are prone to add this header more often than HTTP sites, 17.38% and 7.48%, respectively. For the

sites that choose to issue this header, the most common directives found are "same-origin" (86% in HTTP and 91% in HTTPS) and "deny" (12% in HTTPS and 7% in HTTP). Again, highly popular sites make use of this protection more often than less popular sites as can be readily appreciated in Figure 5:

FIGURE 5. X-FRAME-OPTIONS IMPLEMENTATION RATE AS A FUNCTION OF WEBSITE POPULARITY



These results do not essentially deviate from those of Helme (2017), 12.4% or April (2017), 11%.

6) Set-Cookie Header

This header is used by web sites to send cookies to the client side as part of the response message. Supported by all browsers, its current syntax was standardized by IETF RFC 6265. From the security perspective, the interest on this header lies on the "session cookies", i.e., those cookies that are set from the server side with the purpose of establishing a "session" between client and server (the stateless HTTP protocol was devised without any built-in session mechanism). In principle, cookies can be sent from the server to the browser without any particular security risk, unless they are session cookies. These cookies constitute a major target of many web application attacks, and therefore, we have tackled their study as part of the current research.

In order to prevent session hijacking and other web attacks that usually proceed through Cross Site Scripting or MITM attacks, it is generally agreed that session cookies should, at least, carry the directives "*HttpOnly*", for both HTTP and HTTPS sites, and "*Secure*", for HTTPS sites. *HttpOnly* offers protection against cookies being

accessed from client-side scripts (and therefore stolen under XSS attacks) and the *Secure* flag prevents the cookies from being captured through an unintended HTTP connection.

However, not all cookies need to be protected by *HttpOnly* or Secure flags, only session ones. Given the fact that session cookies need not carry any flag or follow any rules that distinguish them from non-session cookies, we have tried to tell them apart, and therefore assess the presence of the mentioned flags, by parsing the Set-Cookie value and search there for the token "sess" (case insensitively). While this is far from being a satisfactory criterion, our research shows that most of "highly probable" session cookies can be identified this way. Table 5 shows most frequent cookie names observed in the responses obtained from our data set:

TABLE 5. MOST COMMON COOKIE NAMES

Cookie name	Frequency
__cfuid	24.3%
PHPSESSID	20.3%
ASP.NET_SessionId	4.5%
JSESSIONID	2.5%

Cloud Flare ID cookie, __cfuid, cannot be considered properly as a web site session cookie (and indeed does not meet our criteria), while PHP sessions, ASP.NET sessions and Java based sessions are identified using this "sess" token technique. Taking all together, we can assume that 53.6% of all cookies received from server side are properly identified as being, or not, a session cookie. A further inspection analysing the 250 most popular cookie names proved that the "sess" token technique was enough to tell apart session cookies from ordinary cookies, up to that level of "cookie name popularity".

Our results show that, regarding HTTP sites, about 49.4% of them set a session cookie within the response to our first request, but 55.4% of those cookies do not set the *HttpOnly* flag. Regarding HTTPS sites, 42.7% do not set *HttpOnly* flag and up to 80.7% of them do not make use of the *Secure* flag. The following graphs exhibit the same pattern found in other security headers: HTTPS sites and popular sites seem to be more security concerned than HTTP or less popular sites.

FIGURE 6. SESSION COOKIES. *HTTPONLY* DIRECTIVE AS A FUNCTION OF WEBSITE POPULARITY

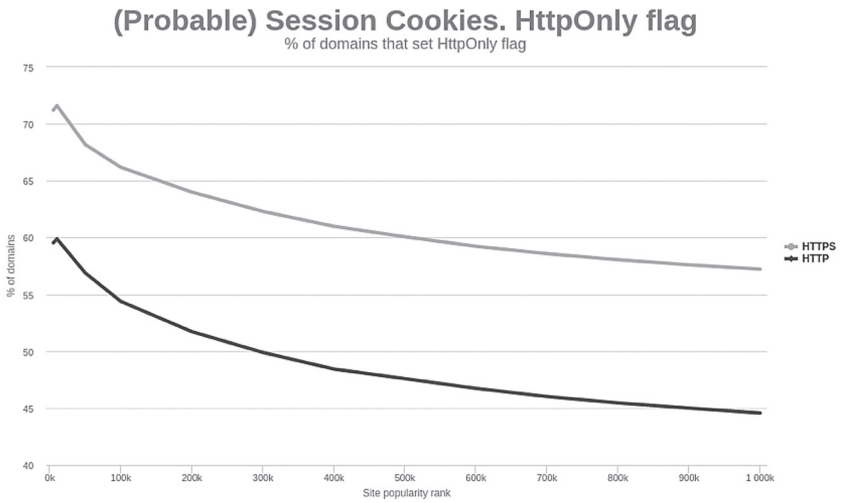
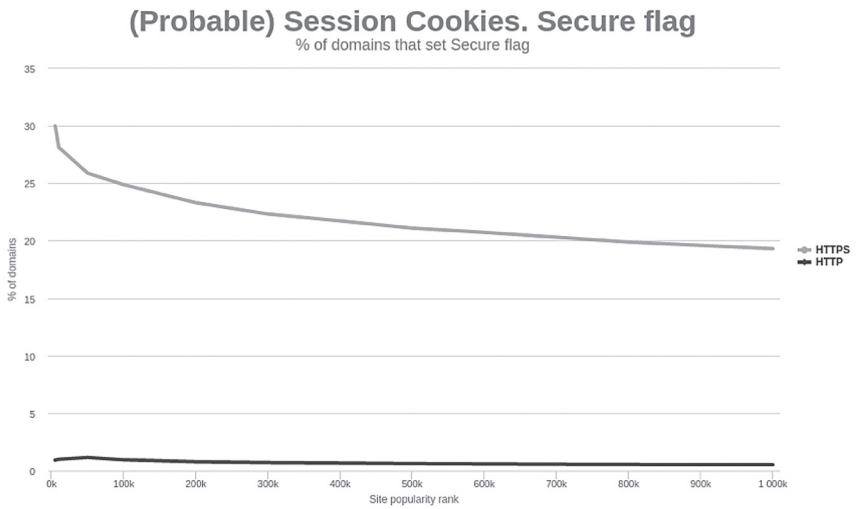


FIGURE 7. SESSION COOKIES. *SECURE* DIRECTIVE AS A FUNCTION OF WEBSITE POPULARITY



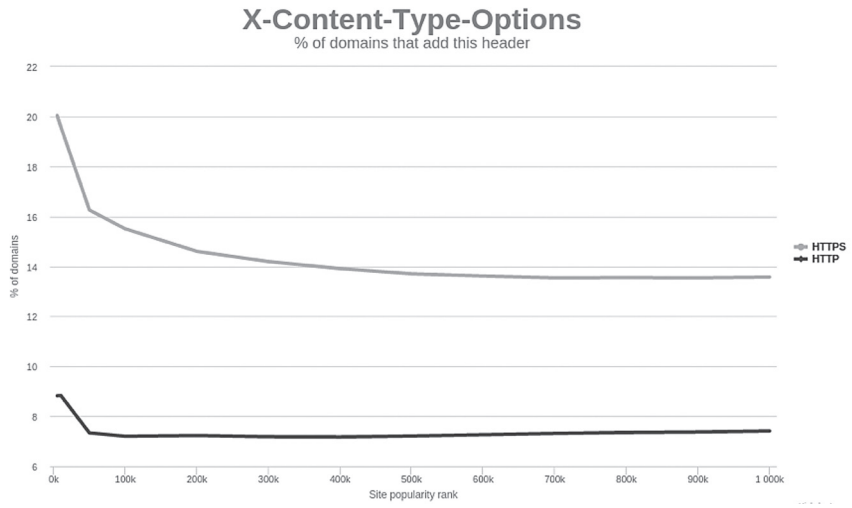
Finally, the "*SameSite*" cookie attribute recently implemented by Chrome and Opera, but still lacking in all other browsers, is an interesting flag currently defined under IETF draft (2016). It helps prevent Cross Site Request Forgery and cookie hijacking by instructing the browser not to send a cookie with that attribute to any request other than same-site requests. Although a very promising attribute, given its novelty and lack of widespread implementation, it is understandable that only 0.05% of HTTPS sites and 0.01% of HTTP sites make use of the flag.

7) X-Content-Type-Options Header

This header was defined to protect browsers from MIME sniffing vulnerabilities, by which an attacker may trick the browser into executing content that was not meant to be executed by the web application. These kinds of attacks make use of the fact that, under some circumstances, browsers do not follow the MIME type indicated in the *Content-Type* header. It is implemented by all major browsers, after Microsoft introduced it in IE8. The only allowed directive for this header is "*nosniff*".

The results follow the same pattern observed in other security headers: HTTPS sites set *X-Content-Type-Options* more often that HTTP ones (roughly, 16% vs 8%) and popular web sites do it also more often than less popular ones, as shown in the next figure:

FIGURE 8. X-CONTENT-TYPE-OPTIONS ADOPTION RATE AS A FUNCTION OF WEBSITE POPULARITY



Helme (2017) reports an adoption rate of 11.6%, whereas April (2017) finds 9.4%, global rates.

C. Information Revealing Headers

1) Server Header (and other related server-side headers)

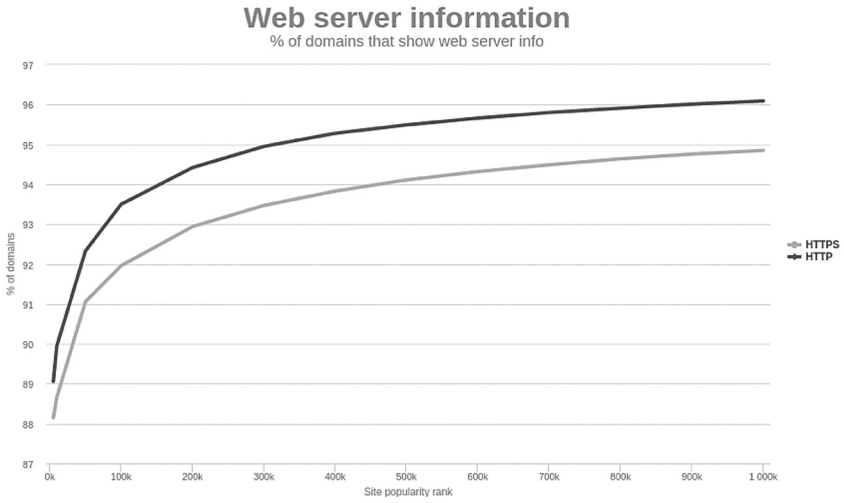
The Server header, defined as part of the RFC 7231 for HTTP/1.1 protocol, is a server-side header originally devised to inform a browser about software used in the web application. Although it is not mandatory, it is issued by most web sites (according to our scanning results, more than 90% of sites set this header). It typically contains the name and version of the web server on which the web application is running.

By itself, the presence of this header as part of an HTTP response does not pose any security thread. However, it may help an attacker to easily obtain the web server name, version and additional information, for example, the name of the CMS supporting the web application. There are currently many "fingerprinting" tools that can be used to obtain that information, regardless of the presence of the Server header. They include well known utilities like command line command nmap, dedicated tool httprint or web utilities like Netcraft that can reveal valuable information to any attacker willing to make use of known vulnerabilities and their corresponding exploits.

The interest of this header from the point of view of the current research is twofold: on the one hand, the *Server* header provides fast and valuable information to those attacks that rely on large scale Internet web site scanning to find potential victims. On the other hand, we want to study statistical correlation between this header and other web site variables, specifically domain popularity and protocol (HTTP / HTTPS) in order to help obtain a more accurate picture of the security of the sites we have analysed.

It should be taken into account that other HTTP headers, besides *Server*, can carry information regarding web server and other relevant software. In particular, we have taken into account the following additional headers: *X-Powered-By, X-AspNet-Version, X-AspNetMvc-Version* and *X-Varnish*. We have combined the information carried by these headers, if present, with the one in the *Server* header, nearly always present, in order to try to find the web server name and version. The results are shown in next figure.

FIGURE 9. WEB SERVER INFORMATION AS A FUNCTION OF WEBSITE POPULARITY



From last figure it is clearly appreciated two tendencies: a) HTTPS sites tend to be more restrictive than HTTP served sites on the information they provide, at least regarding Server and related headers, and b) the more popular a given domain is, the less information it will probably leak. The overall picture, however, shows that a huge amount of Internet web sites (at least, over 85% of them) expose their web server info through their HTTPS headers. Our statistics indicates that, within those sites with recognizable web server, most popular web server is Apache HTTP server (46% of sites) followed by nginx (38% of sites) and IIS (14%).

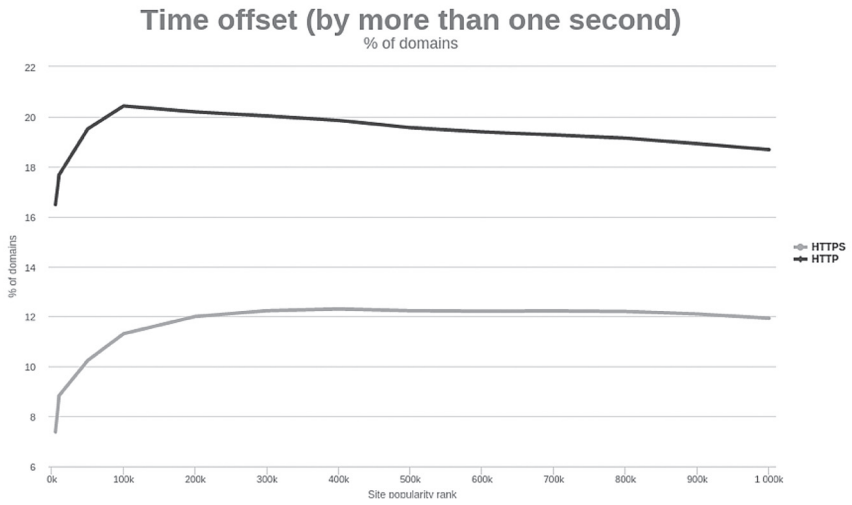
Another common header, *X-Powered-By* header, appears in 48% of responses from sites and typically (66% of cases) contains the PHP version used to develop the web site.

2) Date Header

This header, defined as part of HTTP/1.1 specification, RFC 7231, contains the date and time at which the response message was originated. In our research this header is set in over 99% of all responses. It is, in fact, the most common header seen in responses. Although it is not related to any significant security attack, server-side date and time play an important role as part of the logging information needed to analyse security incidents (see, for example, Prodromou 2016). Inaccurate timestamps will yield unreliable logging records, and therefore making then inappropriate for forensics tasks.

We see the general trends observed previously: a) HTTPS sites seem to run on more precisely configured servers than HTTP ones and b) the more popular a site is, the more secure it tends to be configured.

FIGURE 10. SITES THAT SHOW TIME OFFSET IN *DATE* RESPONSE HEADER AS A FUNCTION OF WEBSITE POPULARITY



3) Referer and Referer-Policy Headers

The *Referer* -sic- header, specified by RFC7231 allows the browser to inform the web server to which the request is made about the URI from where the user made that request. It is meant to provide information that can be processed on the server side for logging or commercial analysis, for example, getting to know where customers typically come from when reaching a given site. This header is also commonly used as a key component of web tracking technologies.

In principle, *Referer* header poses a privacy concern, not a security one, since it reveals information to a third party that a user might not want to be revealed. Sometimes, however, a URL may carry sensitive information, for example, a session token or a capability indicator [Cap2014], and under such circumstances the Referer header may pose a security risk. Both, privacy and possible security risks, have led to the proposal of a *Referrer-Policy* header (see W3C Editor's draft at 2017). This header, currently implemented by all major browsers, allows a website to control the information carried by the *Referer* header in a rather fine-grained manner. It defines up to eight different directives.

Our scanning database indicates that Referrer-Policy header is scarcely implemented. Only 0.05% of HTTP responses, and 0.33% of HTTPS responses, contain some form of a valid *Referrer-Policy*. The distribution of the different policies can be seen in Table 6.

TABLE 6. REFERER DIRECTIVES

Referrer-Policy Directives	HTTP Requests	HTTPS Requests
no-referrer	26.97%	14.86%
no-referrer-when-downgrade	23.63%	29.25%
origin	14.32%	7.06%
origin-when-cross-origin	11.46%	17.73%
same-origin	5.97%	9.18%
strict-origin	5.01%	2.87%
strict-origin-when-cross-origin	7.64%	10.77%
unsafe-URL	0%	0%

Finally, the P3P header, related to users' privacy settings, was not included as part of this study since it is not implemented by any major browser other than Microsoft IE and Edge. However, it is still being issued by 7.5% of the sites (6.9% of HTTP sites and 8.4% of HTTPS ones).

5. CONCLUSIONS

We have presented a new analysis of implementation rate in Alexa's top one million websites of web security policies based on HTTP response headers. A careful data gathering process was carried out to collect HTTP response headers from four different requests for each domain in the list: http://domain, http://www.domain, https://domain and https://www.domain. Redirections were followed. HTTPS issues were examined, finding in particular that a sizeable number of sites, 29.1% of all HTTPS requests made, exhibit some incorrect TLS configuration. They are typically X.509 certificate errors, as the leading causes for TLS misconfiguration are name mismatching and verification errors (self-signed certificates, untrusted CA's or expired certificates). We also compared HTTP response headers obtained from HTTP/1.1 and HTTP/2 equivalent requests and found that, besides some connection related headers, response headers show no significant differences.

We repeatedly showed that security policies based on HTTP response headers are always far more common in HTTPS websites that in HTTP sites. Those policies are also noticeably more commonly implemented among highly popular sites than not so popular ones. In fact, for all security headers analysed here, when implementation rates are depicted against website popularity the resulting curve follows an exponential decline pattern.

In particular, we have found that HTTP Strict Transport Security policy is implemented in about 38% among top one thousand HTTPS sites, but only 17.5% considering all top one million websites. Content Security Policy, despite its powerful prevention capability against Cross Site Scripting and other vulnerabilities, remains poorly implemented at a global 1.6% among all one million websites. HTTPS sites show a markedly larger adoption rate, 3.4%, whereas HTTP sites hardly implement this policy, only 0.4% of them. Session cookies were also analysed and we found that about 50% of sites do not set their *HttpOnly* flag (55.4 % of HTTP sites and 42.7% of HTTPS sites) and and the *Secure* directive is issued for the session cookies in about 19.3% of all HTTPS sites. Although not so relevant as these headers, other security-related response headers were analysed (*X-Frame-Options*, *X-XSS-Protection* and *X-ContentType-Options*). We also analysed information leakage from web servers through their Server and other related response headers and, again, we found that information leakage is more common among less popular and HTTP sites than in highly popular and HTTPS sites.

All in all, security policies based on HTTP headers remain low. They are slightly increasing when compared to the figures reported by previous researches during 2017 (Helme 2017, April 2017), but still well below satisfactory rates. Notably higher implementation rates observed in the most popular sites suggests that security awareness could be influenced by factors like business size. Alternatively, it may be argued that security-aware websites tend to thrive better.

6. FUTURE WORK

The authors plan to expand the current research in several ways. The survey offers a picture of certain web security policies implemented by the top one million websites at a given time (September 2017) and a periodical repetition of the scanning process will be interesting, as it will show how the adoption of these policies are evolving. Following similar initiatives like the ones by Mozilla Observatory and Scott Helme, the authors plan to assign a "global" scoring (e.g., from A to F) for each website and generate the corresponding global statistics and their correlation to HTTPS and site popularity ranking. However, our initial work on this area shows that it is far from obvious how to assign relative weights to each of the analysed HTTP headers and we firmly believe that further work is needed, taking into account, at least, web vulnerability prevalence statistics.

Additionally, we are currently exploring the possibility of considering more variables in our work, like website country and Content Distribution Network usage and how

it relates to web security policies based on HTTP response headers. Finally, we deem interesting to study Subresource Integrity current adoption resources.

7. REFERENCES

Alexa's Top One Million Websites, https://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

Li Chang, Hsu-Chun Hsiao, Wei Jeng, Tiffany Hyun-Jin Kim and Wei-Hsi Lin, "Security Implications of Redirection Trail in Popular Websites Worldwide", in *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*, 2017, pages 1491-1500. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland. DOI: https://doi.org/10.1145/3038912.3052698.

Cisco Umbrella 1 Million, https://umbrella.cisco.com/blog/2016/12/14/cisco-umbrella-1-million/.

Scott Helme, "I am giving up on HPKP", blog post, 24 August 2017, https://scotthelme.co.uk/im-giving-up-on-hpkp/.

Scott Helme, "Alexa Top 1 Million Analysis - August 2017", blog post, 29 August 2017, https://scotthelme.ghost.io/alexa-top-1-million-analysis-aug-2017/.

Scott Helme, "Daily scans of the top one million sites", https://scans.io/study/scott-top-one-million.

Httprint web server fingerprinting tool, http://www.net-square.com/httprint.html.

Hyper: HTTP/2 for Python, https://python-hyper.org/en/latest/.

Internet Engineering Task Force, "HTTP Header Field X-Frame-Options", IETF Informational, RFC 7034, October 2013, https://tools.ietf.org/html/rfc7034.

Internet Engineering Task Force, "HTTP State Management Mechanism", IETF Standard, RFC 6265, April 2011, https://tools.ietf.org/html/rfc6265.

Internet Engineering Task Force, "HTTP Strict Transport Security (HSTS)", IETF Standard, RFC 6797, November 2012, https://tools.ietf.org/html/rfc6797.

Internet Engineering Task Force, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", IETF Standard, RFC 7231, June 2014, https://tools.ietf.org/html/rfc7231.

Internet Engineering Task Force, "Public Key Pinning Extension for HTTP", IETF Standard, RFC 7469, April 2015, https://tools.ietf.org/html/rfc7469.

Internet Engineering Task Force, "Same-Site Cookies", IETF Internet-Draft Standard, 20 June 2016, https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site-00.

April King, "Analysis of the Alexa Top 1M sites (June 2017)", 13 June 2017, https://pokeinthe.io/2017/06/13/state-of-security-alexa-top-one-million-2017-06/.

Let's Encrypt - Free SSL/TLS Certificates, https://letsencrypt.org/.

Majestic Million database, https://blog.majestic.com/development/majestic-million-csv-daily/.

Mozilla HTTP Observatory Website, https://mozilla.github.io/http-observatory-website/.

Mozilla Included CA Certificate List, https://wiki.mozilla.org/CA/Included_Certificates.

Mozilla Intermediate Certificates, https://wiki.mozilla.org/CA/Intermediate_Certificates.

Netcraft Site Report, http://toolbar.netcraft.com/site_report.

Nmap, https://nmap.org/.

OWASP, Clickjacking Defence Cheat Sheet, https://www.owasp.org/index.php/Clickjacking_defence_Cheat_Sheet.

Chris Palmer, "Intent to deprecate and remove HPKP", forum post, 27 October 2017, https://groups.google.com/a/chromium.org/forum/#!msg/blink-dev/he9tr7p3rZ8/eNMwKPmUBAAJ.

Agathoklis Prodromou, "Using logs to investigate a web application attack", blog post, 11 May 2016, https://www.acunetix.com/blog/articles/using-logs-to-investigate-a-web-application-attack/.

Requests: HTTP for humans. v2.18.4 Python library, http://docs.python-requests.org/en/master/.

Ivan Ristic, "Is HTTP Public Key Pinning dead?", blog post, 6 September 2016, https://blog.qualys.com/ssllabs/2016/09/06/is-http-public-key-pinning-dead.

Gustav Rydstedt, Elie Bursztein, Dan Boneh and Collin Jackson, "Busting Frame Busting: a Study of Clickjacking Vulnerabilities at Popular Sites", in *IEEE Oakland Web 2.0 Security and Privacy (W2SP 2010)*, https://crypto.stanford.edu/~dabo/pubs/papers/framebust.pdf.

[IO] Security Headers Website, https://securityheaders.io/.

Aditya Sood and Richard Enbody, "The state of HTTP declarative security in online banking websites", in *Computer Fraud & Security*, Volume 2011, Issue 7, July 2011, pages 11-16. DOI: https://doi.org/10.1016/S1361-3723(11)70073-2.

Usage of HTTP/2 for websites, W3Techs, https://w3techs.com/technologies/details/ce-http2/all/all.

Michael Weissbacher, Tobias Lauinger and William Robertson, "Why Is CSP Failing? Trends and Challenges in CSP Adoption", in *Research in Attacks, Intrusions and Defenses*. RAID 2014. Lecture Notes in Computer Science, vol 8688. Springer, Cham. DOI: https://doi.org/10.1007/978-3-319-11379-1_11.

World Wide Web Consortium, "Content Security Policy 1.0" (CSP1), W3C Working Group Note, 19 February 2015, discontinued, https://www.w3.org/TR/CSP1/.

World Wide Web Consortium, "Content Security Policy Level 2" (CSP2), W3C Recommendation, 15 December 2016, https://www.w3.org/TR/CSP2.

World Wide Web Consortium, "Content Security Policy Level 3" (CSP3), W3C Working Draft, 13 September 2016, https://www.w3.org/TR/CSP/.

World Wide Web Consortium, "Good Practices for Capability URLs", W3C First Public Working Draft, 18 February 2014, https://www.w3.org/TR/capability-urls/.

World Wide Web Consortium, "Referrer Policy", W3C Candidate Recommendation, 26 January 2017, https://www.w3.org/TR/referrer-policy/.

World Wide Web Consortium, "Subresource Integrity", W3C Recommendation, 23 June 2016, https://www.w3.org/TR/SRI/.

Ming Ying and Shu Qin Li, "CSP adoption: current status and future prospects", in *Security and Communication Networks*, Vol 9, Issue 17, 25 November 2016, pages 4557-4573. DOI: https://doi.org/10.1002/sec.1649.