Analysis of the Adoption of Security Headers in HTTP

Article *in* IET Information Security · October 2017

DOI: 10.1049/iet-ifs.2016.0621

3 authors:

William J Buchanan
Edinburgh Napier University
571 PUBLICATIONS **1,576** CITATIONS
SEE PROFILE

Scott Helme
scotthelme.co.uk
2 PUBLICATIONS **16** CITATIONS
SEE PROFILE

Alan Woodward
University of Surrey
6 PUBLICATIONS **62** CITATIONS
SEE PROFILE

IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

Analysis of the adoption of security headers in HTTP

ISSN 1751-8709
Received on 10th March 2017
Revised 11th July 2017
Accepted on 27th September 2017
doi: 10.1049/iet-ifs.2016.0621
www.ietdl.org

William J. Buchanan[1] ✉, Scott Helme[2], Alan Woodward[3]

[1]*School of Computing, The Cyber Academy, Edinburgh Napier University, Edinburgh, UK*
[2]*Independent Contractor, scotthelme.co.uk*
[3]*Surrey Centre for Cyber Security, University of Surrey, Surrey, UK*
✉ *E-mail: w.buchanan@napier.ac.uk*

Abstract: With the increase in the number of threats within web-based systems, a more integrated approach is required to ensure the enforcement of security policies from the server to the client. These policies aim to stop man-in-the-middle attacks, code injection, and so on. This study analyses some of the newest security options used within HTTP responses, and scans the Alexa Top 1 Million sites for their implementation within HTTP responses. These options scanned for include: content security policy, public key pinning extension for HTTP, HTTP strict transport security, and HTTP header field X-frame-options, in order to understand the impact that these options have on the most popular websites. The results show that, while the implementation of the parameters is increasing, it is still not implemented on many of the top sites. Along with this, the study shows the profile of adoption of Let's Encrypt digital certificates across the one million sites, along with a way of assessing the quality of the security headers.

1 Introduction

With HTTP, we have a request, such as GET request, and the server responds with a response. These responses contain header information which includes parameters defined in a list of key:value pairs [1]. There are many standard application layer protocols that are used to exchange information, including HTTP [1], SMTP [2], FTP [3], and DNS [4]. These specifications were often written to support a simple text-based exchange of messages. Some of these are stateless, such as for DNS and HTTP, while others, such as SMTP and FTP, require a session to be created, with commands and responses. At the time, too, security was often an after-thought, and where the application layer protocols were improved for their security with the addition of the secure socket layer (SSL), such as with HTTPs [5].

While SSL and transport layer security (TLS) purely protected the contents of the message exchange, the addition of CSP – content security policy – [6] integrates a policy language that sets content restrictions on a web resource, and where the server transmits the policy to the client, for it to enforce the policy. New security extensions have also been added to prevent man-in-the-middle (MITM) attacks, such as the public key pinning extension for HTTP (HPKP) [7] and which allows a site to associate itself with specific cryptographic public keys and thus protects against forged digital certificates. Within HTTP strict transport security (HSTS) [8] a web server can require that a client (and its associated web browser) should only interact with it through a secure connection (such as with HTTPS). In 2013, RFC 7034 defined HTTP header field X-frame-options (XFO) [9], which protects web applications against clickjacking, and against the integration of code from non-trusted sources. Again the focus is on the server informing the client of its policy and configuration options.

This paper analyses the usage of these new security features within the Alexa Top 1 Million sites, in order to identify the impact of the roll-out of the security response headers. It can be seen, in Fig. 1, that the response headers include: CSP, content-security-policy-report-only (CSPRO), public-key-pins (PKP), public-key-pins-report-only (PKPRO), X-content-type-only, XFO, and X-Xss-protection (XXP). Along with this, the adoption of Let's Encrypt [10] shows an interesting move towards the usage of free digital certificates. The analysis will thus also look at its adoption, and see if the top million sites are using the Let's Encrypt certificate authority (CA).

2 Background

While SSL/TLS purely protects the content of the data exchange, the addition of CSP [1] provided a policy language that could set content restrictions on the web resource, and where the server transmits the policy to the client, for it to enforce the policy. New security extensions have also been added to prevent MITM attacks, such as the HPKP [11] which allows a site to associate itself with specific cryptographic public keys and protects against forged digital certificates. With HSTS [12] a web server can require that a client (and its associated web browser) should only interact with it through a secure connection (such as with HTTPS).

RFC 7034 added HTTP header field XFO [13], which protects web applications against clickjacking, and within the integration of content from other web pages. Again the focus is on the server telling the client its policy and configuration options. Response headers include: CSP, CSPRO, PKP, PKPRO, X-content-type-only, XFO, and XXP.

Overall the web focuses on a same-origin policy [14], where the script contained in one origin is only permitted to access data within that origin, and thus each origin is isolated from others. Unfortunately, this overly restricts developers, along with attackers using clear tricks to inject malicious code from other domains. Many media sites often, too, use content and scripts from other sites and would struggle to support content which restricted them to their own site. The integration of code from other sites can lead to the problem of cross-site scripting (XSS) attacks, as the code within the web page is often full trusted. Within CSP we thus have a number of methods that protect against XSS. With this CSP supports multiple policies for a resource, either in a `Content-Security-Policy` header or within the <meta> element, such as [14]:

```
Content-Security-Policy: default-src https:
<meta  http-equiv="Content-Security-Policy'
content="default-src https:">
```

One of the most powerful features of CSP is to define the whitelist for the client to use. A common problem is thus that a browser will trust all of the code in a page, so that content from

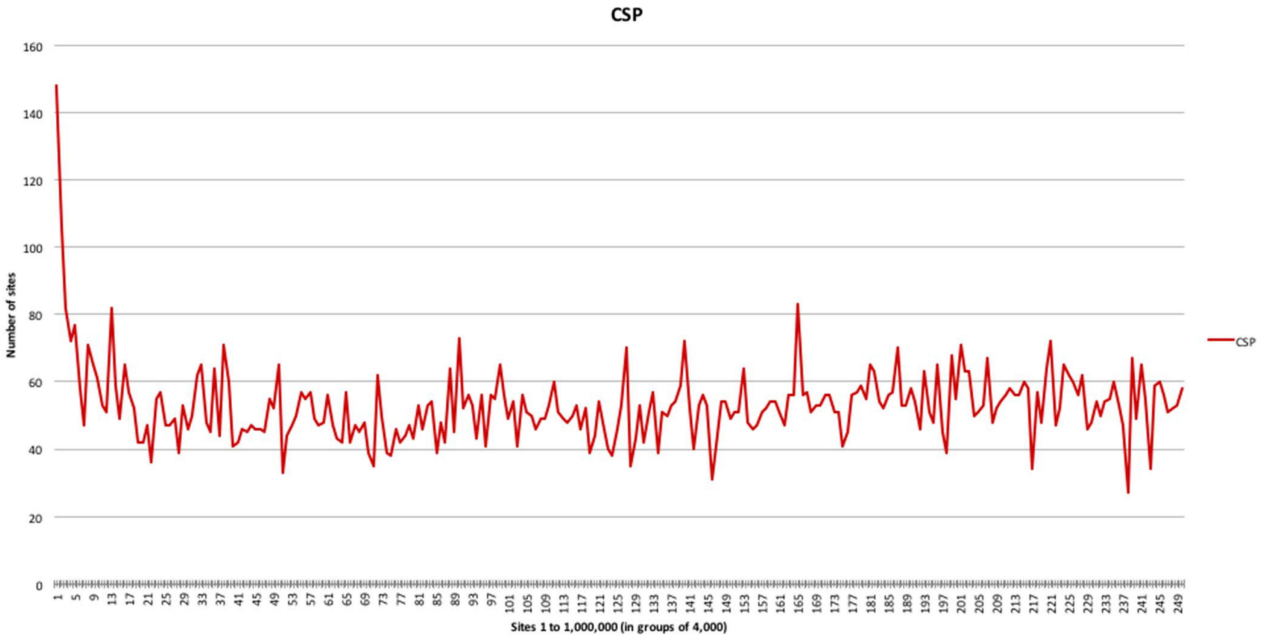

Fig. 1 *CSP (May 2017)*

other sites are trusted, so, within CSP, the `Content-Security-Policy` HTTP header is used to define the trusted sources, and where no other sources of content can be used. For example, if we only wanted code from https://asecuritysite.com we could use:

```
Content-Security-Policy: script-src 'self' https://asecuritysite.com
```

In this case, we are trusting our own source code ('self') and any scripts from https://asecuritysite.com. When code is injected into the page from another source, an error will appear. For content, we can also restrict the location that images, media, plug-ins, style sheets, and other objects can be installed from using the directives of:

- `img-src`. Defines the sources for images.
- `media-src`. Defines the sources for audio and video.
- `object-src`. Defines the sources for plug-ins, such as Adobe Flash.
- `style-src`. Defines the sources for stylesheets.
- `font-src`. Defines the sources for fonts.
- `script-src`. Defines the sources for all scripts.

By default, all of the directives are open, so if a directive is not set it will allow all for that part of the policy. If certain plug-ins could compromise the page, we can limit them with: `plugin-types`: and which defines the plug-in types that can be trusted by the browser. For example, we can allow PDF and Flash plug-ins with:

```
Content-Security-Policy: plugin-types application/pdf
Content-Security-Policy: plugin-types application/pdf application/x-shockwave-flash
```

HTTPS sourced content is now often preferred to HTTP, as the traffic can be protected against an attacker spying on the contents of data packets, along with a secure tunnel being created between the client and the server. Thus, CSP supports a restriction on requests so that HTTP requests are automatically rewritten to HTTPS one. This is defined as part of the CSP policy with:

Content-Security-Policy: upgrade-insecure-requests

and which will rewrite this code:

```
<img src="http://asecuritysite.com/good.png">
```

to:

```
<img src="https://asecuritysite.com/good.png">
```

A powerful feature of CSP is that the client should not execute on a policy violation, but will report it back to the server. For this we have a reporting URI (`report-uri:`) to define the location to post the report back to. We can also restrict the URLs on a page using: `base-uri`: and which restricts embedded frame contents to the `<base>` element; and `child-src:`. For example:

```
child-src https://youtube.com
```

enables the embedding of videos from YouTube, but not from other origins. For example, we may restrict iframes to https://youtube.com with:

```
Content-Security-Policy: child-src https://youtube.com/
```

Where the following would be blocked:

```
<iframe src="https://fake-youtube.com"><iframe>
```

An example attack is to redirect the parameters posted from a form. With CSP we can thus restrict the end-points with: `form-action`: and this defines the endpoints for the `<form>` tag. While whitelisting helps in detecting code being run from non-trusted sources, one of the greatest threats within XSS is related to inline script injection [15], such as for:

```
<script>someEvilCode();<script>
```

CSP overcomes this problem by banning any form of inline script tags, along with inline event handlers and in the form of:

javascript: URL

3 Literature review

3.1 Content security policy

The risk around web security increases by the day, with OWASP defining their top 10 risks [16] and include: broken authentication and session management, XSS, insecure direct object references, security misconfiguration, sensitive data exposure, missing function level access control, and cross-site request forgery (CSRF). Along with this different methods are defined in order to define a risk score [17]. While many companies now concentrate on the OWASP top 10, other risks exist, such as where the HTTP headers can be used as a source of hiding information [18].

The increasing focus of web application vulnerabilities, especially for XSS and CSRF, has led to a call for the creation of applications which are free from vulnerabilities [19]. In the real world, this is likely to be difficult, especially where there is a loose coupling between the front-end, middleware and back-end web

infrastructure. CSP thus integrates into a layered approach to security and uses content restrictions, and a content restrictions enforcement scheme.

The researchers within [20] created a CSP implementation to successfully mitigate a wide range of XSS attack against in four popular browsers. They define that an XSS attack is either:

- Persistent XSS. This type stores the script on the server, such as in a database, and will thus run each time the associated page is accessed.
- Non-persistent XSS. This type hides some malicious script and tricks the user in running the harmful script, in order to steal authentication cookies or data [21].
- DOM-based XSS. This type of attack modified the DOM structure of a web page, in order that it runs the code in a comprised way.

Using CSP, they used a collection of XSS attacks from [21] and 50 unique ones, for the different attack types, from Chrome, Firefox, Safari, and Opera. Their results showed 37 successful attacks against Firefox, 19 against Chrome, Safari, and Opera, while using CSP reduced the success rates of all of the attacks to zero.

As with many advancements within web development, the adoption of CSP has been slow and where problems still exist, such as with insecure server-side JavaScript generation. In [22], the authors propose PreparedJS an extension to CSP. It does this using a safe script templating mechanism and a light-weight script checksum scheme. In [23], Google analysed CSP identify flaws that can be bypassed in 94.72% of all distinct policies, and use a search engine corpus of around 100 billion pages from over 1 billion hostnames. This 1,680,867 hosts with CSP deployed and 26,011 unique CSP policies. In their paper, they identify three common methods for bypassing CSP. One of the key factors they identify as a weakness is that 75.81% of distinct policies use a whitelist that is too lax and can be used to bypass the CSP.

In [24], Van Goethem *et al.* analysed over 22,000 websites within 28 EU countries and found the presence of common vulnerabilities and weaknesses, and that the adoption of improved defence mechanisms, tended to be focused on the popular sites. Also Chen [25] provided an analysis of 18,000 European websites over a 2-year period, and analysed the adoption of client-side security methods, and found that the finance and education sectors were the most successful in their adoption.

3.2 Scanning

The scanning of the Alexa sites is a well-defined method of understanding the changing nature of the request, such as in the ever-changing behaviour of HTTP web pages, and, for example, where [26] found a trend towards large pages including multimedia content, and in dynamic content creation. In [11], the authors have outlined the usage patterns HTTP request messages including and found that they vary greatly in the number, field names, and field values. Other protocols too, such as with SMTP have been shown to leak information [12] and where the metadata used in sending emails can be used to determine system data which could lead to a compromise. For [13], the researchers analysed the HTTPS certificates from Censys, and certificate transparency (CT) logs. In this work, they surveyed the top one million Alexa sites, and they found that aggregated CT logs and Censys snapshots cover 99% of all certificates found.

Recent work has involved large-scale vulnerability analysis of web infrastructures, such as in China [27] using 57,112 web vulnerability incidents, and creating new threat models which understand the modern vulnerabilities [28]. Unfortunately, the usage of security headers still seems to be taking a while to make a significant impact on the most popular websites.

With the increasing threat around XSS attacks, Ying and Li [29] analysed the adoption rate of CSP for the 100 most popular sites, and found that only 8% of them used in January and June 2015. On a larger scan they found an adoption of 0.066 and 0.133% used in January and June 2015, showing a 73% increase.

3.3 Let's Encrypt

There are many risks around digital certificates, including man-in-the-middle attacks and fake certificates [30]. One of the major problems with them is that they are often expensive to purchase, thus the Let's Encrypt initiative looks to create a CA ecosystem that is free to use and automates certificate signing [31]. It is focused on CT [10]. In [32], the authors analyse data from CT logs of over 18 million certificates. This uses sources such as Censys, Alexa's historic records, Geolocation databases, and VirusTotal. In their results that only 54% of domains use the certificates which they have acquired and that there were many occurrences of misconfigured servers, along with evidence of use of Let's Encrypt certificates used in malware-laden sites.

3.4 HTTP public-key pinning

Back in 2011, a Dutch CA called DigiNotar was hacked [33]. After gaining access to their systems, the hackers managed to make their way to the CA servers and issued over 500 rogue certificates to themselves. Included in those certificates was one for the google.com domain. This certificate was used to launch a MITM attack against 300,000 Iranian users of Google services, including GMail. From the end user perspective, the attack was undetectable. The browser received a certificate for the domain, it was valid and the chain of trust was intact. They had all the indicators of a secure connection but it was compromised. There was a lot of speculation about who was responsible for the hack, but from the host's perspective, their users were compromised without any way for them knowing.

In 2008, and again in 2011, the StartCom CA was breached [34]. The 2008 breach, carried out by a security researcher, resulted in rogue certificates being issued for the paypal.com and verisign.com domains. The 2011 breach, carried out by an unknown attacker, came very close to the crown jewel, the StartCom root key. With access to the StartCom root key, the attacker could have generated a certificate for any domain they like, but not only that, it would have led to the necessary revocation and reissue of every single certificate issued by StartCom.

Presently, there are hundreds of CAs who are able to issue digital certificates, and which can lead to fraudulent certificates. PKP, initially stated by Google, enables site owners to define the certificates which are valid for their site, and was defined in RFC 7469 [7]. In [35], the author believes that it is difficult and dangerous to use, as it can be easy to block much of the access to websites. A problem is the memory effect, where a pin once set will then remain valid for a period of time.

Within [36] the researchers outline the implementation of HPKP, and where web servers inform their clients that they must remember (or ‘pin’) their public keys. They outline that HSTS and HPKP are new features to enforce HTTPS connections and allow certificate pinning over HTTP, but that the adoption has been poor, and where the implementation is also weak in terms of security. For this they have already found possible attacks against HSTS and HPKP. In 2015, Kranch and Bonneau [37] analysed the adoption of HSTS and KPKP and found evidence that the adoption of the features was not well understood by developers, and many sites had problems such as loading non-pinned resources which could be used to hijack the page, and where pinned domains could leak cookie values.

4 Methodology

Previous scans of security headers, such as for CSP have been achieved on a fairly limited scan [29], and while [23] has achieved a wide-spread scan of CSP policies, its focus has been on breaching CSP. This paper outlines two main scans of the Alexa Top 1 Million in August 2015 and May 2017. For CSP, the scanner checked for the following headers: CSP, CSPRO. This response header can be used by developers to understand the effects of policy enforcement, and where a JSON document is sent back as an HTTP POST to a defined URI; X-Webkit-CSP (XWC); and XCSP.

Table 1 A capture from the first few headers for a scan in May 2017

Site from	Site to	Skipped	CSP	CSPRO
1	4000	591	148	45
4001	8000	551	106	18
8001	12,000	622	82	19

```
// The value to award headers.
$values = array("strict-transport-security" => 25,
                "content-security-policy" => 25,
                "public-key-pins" => 30,
                "x-frame-options" => 20,
                "x-xss-protection" => 20,
                "x-content-type-options" => 20);
```

Fig. 2 *Value to award headers*

Table 2 % of the total possible score used as a criteria for each grade awarded

% of total achieved	Grade awarded
0–13	F
14–28	E
29–49	D
50–59	C
60–74	B
75–95	A
95–100	A +

In [20], the authors found that the CSP header is supported by Firefox Version 23, Chrome Version 25, Safari Version 7, and iOS Safari Version 7.1, and later versions, while the XCSP and X-WebKit-CSP headers were used within earlier browser versions. Often, to ensure compatibility, administrators configure both the XCSP and CSP headers. For other header scans, the following are detected: PKP, PKPRO, strict-transport-security (STS), X-content-type-options (XCTO), XFO, XXP, X-download-options (XDO), and X-permitted-cross-domain-policies (XPCDP).

Alongside this the scanner recorded the redirection of HTTP to HTTPS, such as is the case for google.com or facebook.com, where the crawler will default to http:// (domain name) and follow redirects until they are complete. Other metrics are also captured: *access-control-allow-origin*: this header can be used to define that only content from the origin site can be allowed; *Let's Encrypt*: this defines if the certificates are generated from Let's Encrypt; and *securityheaders.io grade*: this providers a score for the security headers.

A key objective of the work is to understand the trends in using the security headers as we move through the one million sites. Thus, we group into 4000 servers and count the headers for each range. Table 1 shows a capture from the first few headers for a scan in May 2017.

The scoring for the headers is based on the security benefit that they offer and the ease of their deployment (https://securityheaders.io/). As the security benefit or the difficulty increases, the score increases. These are the values awarded (Fig. 2).

By far the most difficult header to deploy and the one that offers the most protection is HPKP, which scores the highest value. It is not possible to score an A + without deploying HPKP due to its high value. Next are HSTS and CSP which both score equal value due to have some deployment considerations and offering a significant amount of protection. The 'X' headers all score equally as they are simple enough to deploy and offer a worthwhile level of protection for their low complexity. As for the % grading bands, these are designed such that sites can easily elevate themselves from the lower grades to encourage further improvements but a distinct effort is required to achieve an A, where all headers except HPKP are needed, and an A + which does require all headers be present and properly deployed.

This gives a total score for HTTPS sites of 140 whilst HTTP sites have a total score of 85 as HPKP and HSTS are ignored by the browsers if delivered over an insecure connection. Each header that the site returns is tested against the headers that are awarded a score for a case-insensitive match and if one is found the header value is then checked for correct syntax. If these checks pass, then the site is awarded the score associated with that header. Sites cannot be awarded the score for the same header a second time if the header is duplicated. The grade awarded to the site depends on the % of the total possible score that they achieved and uses the criteria shown in Table 2.

In the work reported here, each crawl of the Alexa Top 1 Million was completed over a period of ∼12 h using a single server. The server breaks the Top 1 Million list into 250 smaller lists of 4000 sites and then runs 250 crawler threads concurrently to dramatically reduce the time taken to crawl all 1 million sites. The crawler is written in PHP and uses the cURL library to issue requests, the results are stored in a MySQL database. This was executed on a virtual server emulating an Intel x64 system hosted by Digital Ocean running Ubuntu 14.04. The Alexa data is provided as a comma separated file in the format rank, hostname, e.g. 1, google.com and 2, youtube.com, and so on.

The crawler defaults to communicating with the site over HTTP by simply concatenating the 'http://' string with the hostname provided in the Alexa data 'http://hostname'. The crawler will issue this initial GET request and then follow all redirects until completion with a timeout on requests set to 10 s, and it simulates a modern browser by setting a user-agent string with the value:

```
'Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/52.0.2743.82 Safari/537.36'.
```

Once the crawler has received a final response it parses the response and searches for strings matching the headers being sought. If a response contains one of the headers being sought, this is marked on the appropriate entry in the database.

Importantly, a site is only marked as supporting HTTPS if it redirects the client request from HTTP to HTTPS as only such sites are truly capitalising on HTTPS: few users will enter HTTPS as part a URL being sought so unless HTTP defaults to HTTPS the authors feel HTTPS is nugatory, and hence should not be counted. In the reporting-only versions, such as CSPRO and PKPRO, the response header can be used by developers to understand the effects of policy enforcement, and where a JSON document is sent back as an HTTP POST to a defined URI. This work does log such headers for completeness, but we do not attach a great significance to them as indicating the level of security of a site if they are used without other appropriate headers.

5 Results

A scan of Alexa Top One Million sites took <12 h in May 2017. Table 3 and Fig. 3 outline the results. Fig. 3 outlines the results from the scan, within groups of 4000. Overall we can see a spike in adoption at the top end of the ranking, followed by a sharp decline, and then a steady tail off as you move down the rankings. In May 2017, CSPRO and PKP both saw weak results, with an adoption rate of <1%. The results show a large increase in the number of sites that have a CSP deployed and have a healthy boost in the number of sites with a PKP policy deployed too. Another improvement is a large rise in the number of sites deploying the

Table 3 Results (August 2015 and May 2017)

	August 2015	August 2015, %	May 2017	May 2017, %	% change
CSP	1365	0.1476	13,253	1.5736	870.92
CSPRO	211	0.0228	1028	0.1221	387.20
XWCSP	183	0.0198	362	0.0430	97.81
XCSP	304	0.0329	921	0.1094	202.96
PKP	148	0.0160	6624	0.7865	4375.68
PKPRO	21	0.0023	87	0.0103	314.29
STS	11,308	1.2231	45,527	5.4057	302.61
XCTO	44,315	4.7933	89,053	10.5739	100.95
XFO	55,042	5.9536	93,601	11.1139	70.05
XXSSP	41,948	4.5373	70,032	8.3154	66.95
XDO	192	0.0208	7134	0.8471	3615.63
XPCDP	346	0.0374	6993	0.8303	1921.10
HTTPS	62,043	6.7108	208,710	24.7815	236.40

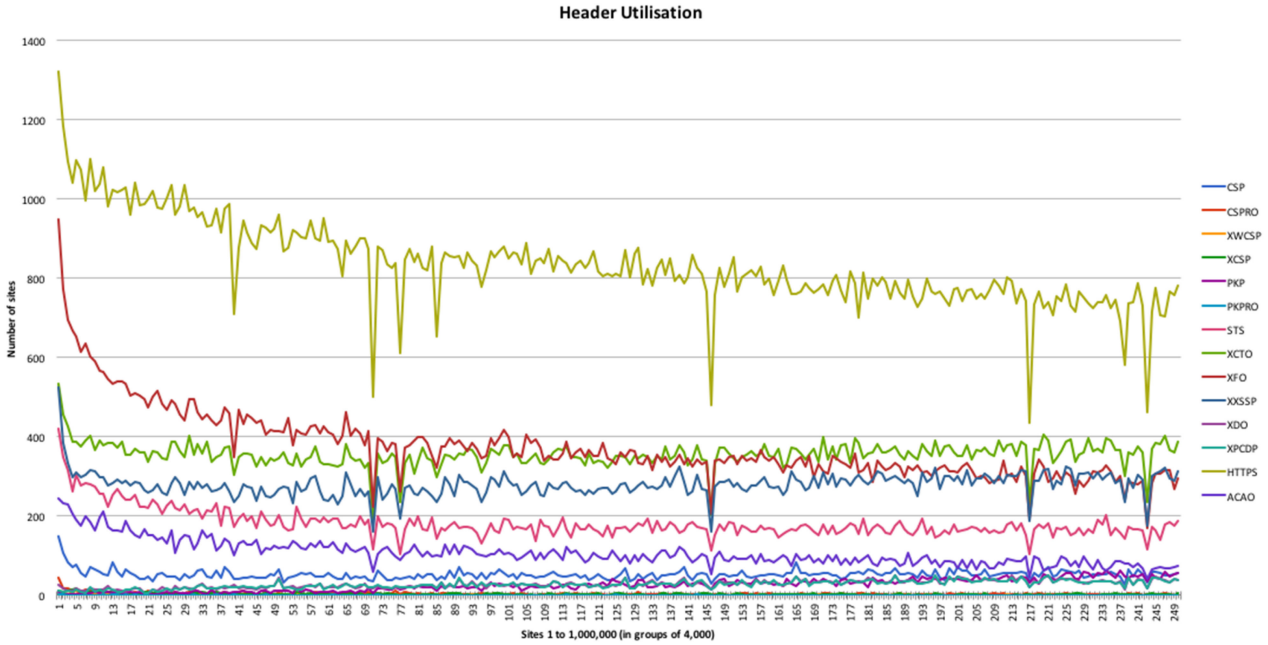

Fig. 3 *Header utilisation (May 2017)*

report-only version of these policies: CSPRO and PKPRO. A 387.2% increase in the number of sites testing out a CSPRO policy shows perhaps that many organisations are now looking to deploy a full version of CSP. Along with this there is a 302% increase in the number of sites issuing an STS policy and with an increase the number of sites redirecting to HTTPS by over 236%.

Fig. 1 shows the CSP graph for the May 2017 scan. The trend is simulator across the results with the only difference being the scale on the *Y*-axis has increased. The same comparison can be drawn for all of the other headers, and even for the deployment of HTTPS.

HTTP public key pinning (HPKP) was only found on 6624 out of 1,000,000 sites (Fig. 4), and is one of the most under-utilised security based HTTP response header (0.79%). The increasing adoption for PKP for lower ranked sites is thought to relate to the adoption of PKP within the Tumblr sites.

The most widely used of the non 'X' headers is strict transport security, and is used on 45,527 of the sites making for a 5.41% usage rate (Fig. 5). This is still fairly low for the one million most visited. Interestingly, of the 1 million sites, 208,710 of them (24.78%) were actively redirecting to HTTPS on their domain, which leaves 163,183 domains that are redirecting to HTTPS, but not enforcing it with STS.

For the more common 'X' headers, there is a clear-cut distinction between these and the remaining headers in terms of the number of sites that use them (Fig. 6). The XFO header is by far the most prevalent security based header in use at 93,601 sites (11.11%) and shows the same downward trend as the rest. Interestingly, though, the XCTO and XXSSP headers both show similarly high usage at 89,053 (10.57%) and 70,032 (8.31%), respectively, but, after their initial dip they actually show an upward trend as you move down the list, which goes against the trend of all other headers.

One of the other things that the crawler was looking for was how many of the domains would redirect to HTTPS if you loaded them over HTTP on the first request (Fig. 7). As mentioned previously, a total of 208,710 sites redirected to HTTPS (24.78%), and we can see the same downward trend as we move down the list of domains.

The vast majority of sites scored an F grade on the securityheaders.io scan. This means they either do not issue any security-based HTTP headers, or they do, and they are not configured properly. Table 4 highlights just how wide the gap is. Unfortunately, the sheer number of sites getting an F pretty much drowns out the rest, but you can still see trends. The further down the ranking you go, the more likely you are to get an F grade, with the spikes in the better grades just visible at the bottom.

As expected the usage of Let's Encrypt certificates increases as you go down the rankings (Fig. 8). There is also the factor that Let's Encrypt does not issue EV certificates, so probably cannot cater for a few of the sites either. Reference [32] scanned the Alexa sites from 29 July to 29 August 2016, and found 8% of the million sites containing Let's Encrypt sites, compared with 30% for Comodo and 17% for GeoTrust. In their Common Crawl method,

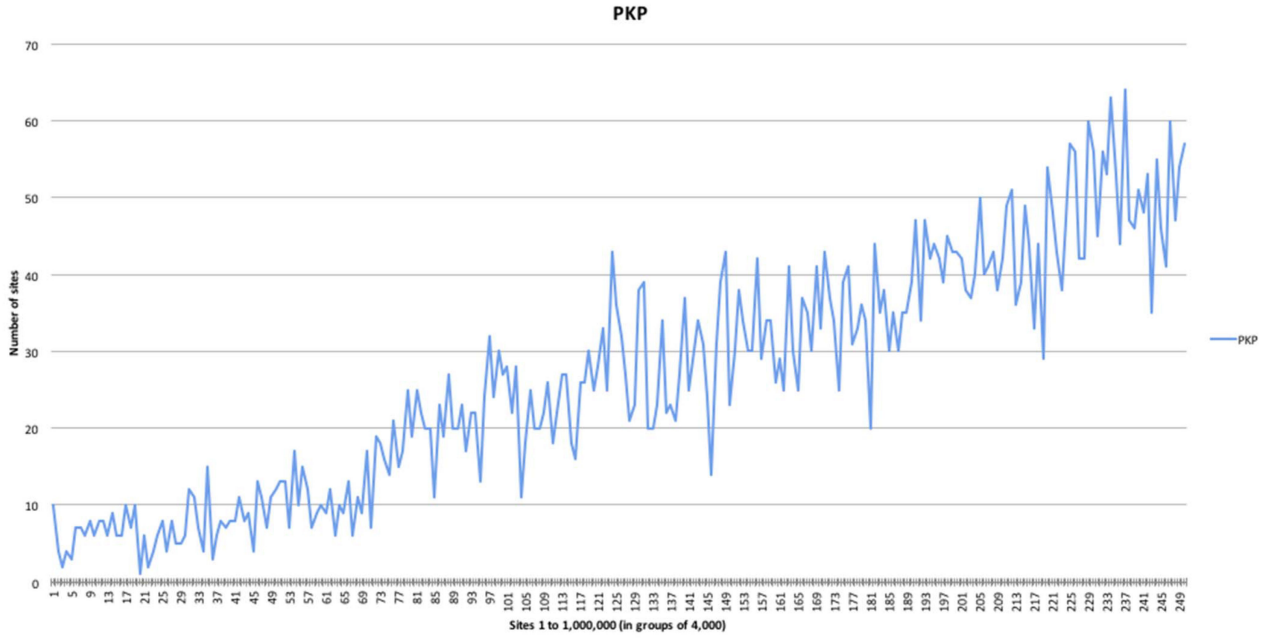

Fig. 4 *PKP*

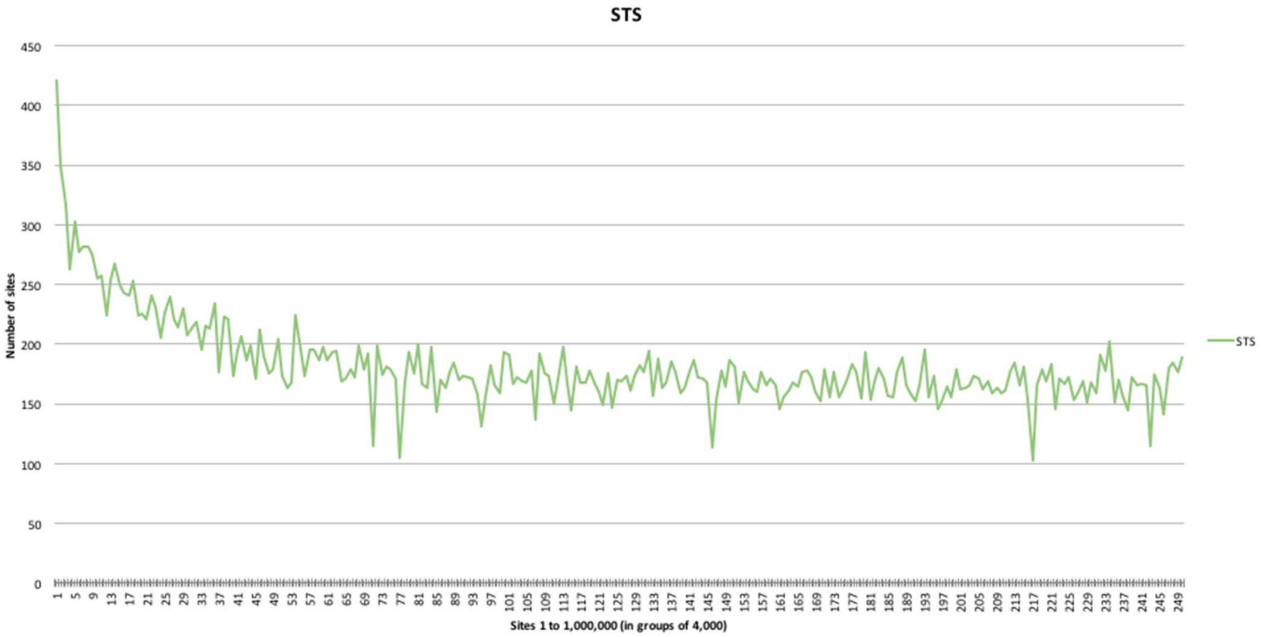

Fig. 5 *STS*

they found that Let's Encrypt was used in 24% of the sites found, with Comodo at 21%. They found that 1.2% of the Alexa top million, for May 2017, have transitioned from paid CAs to Let's Encrypt, while we found 5.29%. In Fig. 8, we see that the adoption of Let's Encrypt certificates is weaker for the top sites. In Table 5, we see that the adoption of the Let's Encrypt in the top sites is fairly weak, and only has an adoption of 0.6%, while the sites near the end of the top one million sites have and adoption rate of between 4.5 and 5.8%.

6 Conclusions

All-in-all the metrics are promising, with an increase in usage of the security headers. It is likely that we will see increases in the use of CSP and CSPRO. Overall we see the increasing in the reporting of security headers, many companies could now be investigating the usage of the enhanced security methods, and there could be a considerable increase in future scans. It is, perhaps, problems around the deployment of CSP that its impact is not quite as significant as it could be [35].

The increase in sites redirecting to HTTPS is gathering pace fast. The securityheaders.io scores were a little poor, but there is plenty of opportunity for easy improvements to be made with the simpler X-based headers. For Let's Encrypt certificates, we see the great impact away from the top site, with the Top 4000 sites only have an adoption rate of 0.6%, while we go up to 5.3% for the bottom 4000 sites in the top million. The results tally with [32], but also contribute in the trend of the general increase in adoption as we move down the ranking, which ends up between 4.5 and 5.8%.

The future work continues to enhance the scoring system, and also to access more than one page from the websites. At present the method only accesses the home page of the site, and does not sample other pages.

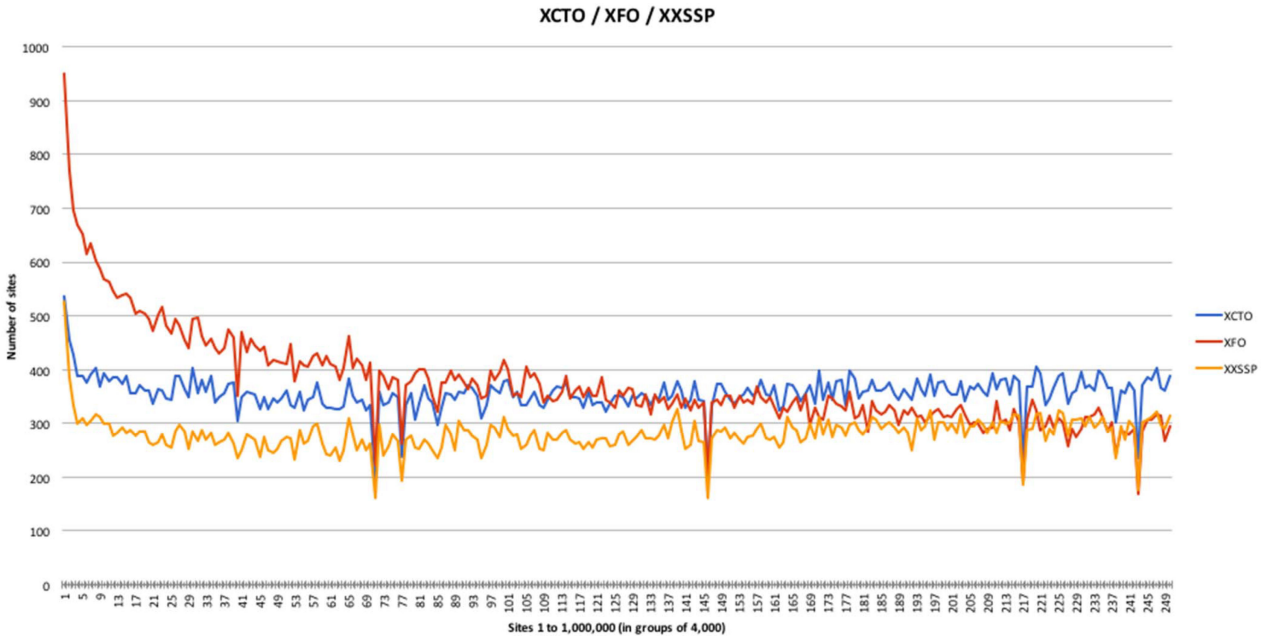

Fig. 6 *X headers*

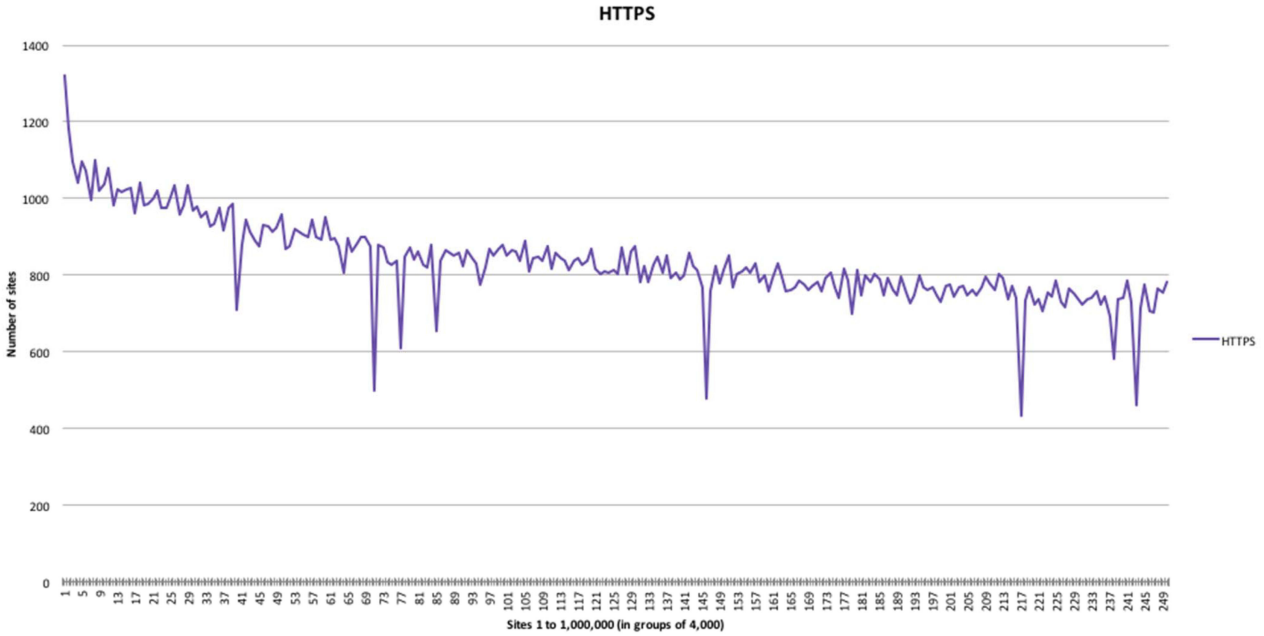

Fig. 7 *HTTPS redirection*

Table 4 Security header scores

Security headers	%
A +	0.0071
A	0.1003
B	0.2318
C	3.2153
D	5.6908
E	8.0063
F	82.7401

Table 5 Let's Encrypt certificate distribution

Site position	Number	Percentage	Site position	Number	Percentage
1–4000	25	0.6	900,001–904,000	197	4.9
4001–8000	62	1.6	904,001–908,000	197	4.9
8001–12,000	77	1.9	908,001–912,000	199	5
12,001–16,000	87	2.2	912,001–916,000	195	4.9
16,001–20,000	93	2.3	916,001–920,000	180	4.5
20,001–24,000	103	2.6	920,001–924,000	199	5
24,001–28,000	77	1.9	924,001–928,000	207	5.2
28,001–32,000	103	2.6	928,001–932,000	208	5.2
32,001–36,000	105	2.6	932,001–936,000	210	5.3
36,001–40,000	105	2.6	936,001–940,000	194	4.9
40,001–44,000	125	3.1	940,001–944,000	206	5.2
44,001–48,000	119	3	944,001–948,000	191	4.8
48,001–52,000	124	3.1	948,001–952,000	144	3.6
52,001–56,000	127	3.2	952,001–956,000	232	5.8
56,001–60,000	122	3.1	956,001–960,000	191	4.8
60,001–64,000	109	2.7	960,001–964,000	233	5.8
64,001–68,000	117	2.9	964,001–968,000	189	4.7
68,001–72,000	144	3.6	968,001–972,000	113	2.8
72,001–76,000	152	3.8	972,001–976,000	203	5.1
76,001–80,000	109	2.7	976,001–980,000	224	5.6
80,001–84,000	133	3.3	980,001–984,000	182	4.6
84,001–88,000	110	2.8	984,001–988,000	205	5.1
88,001–92,000	129	3.2	988,001–992,000	211	5.3
92,001–96,000	124	3.1	992,001–996,000	207	5.2
96,001–100,000	138	3.5	996,001–1,000,000	213	5.3

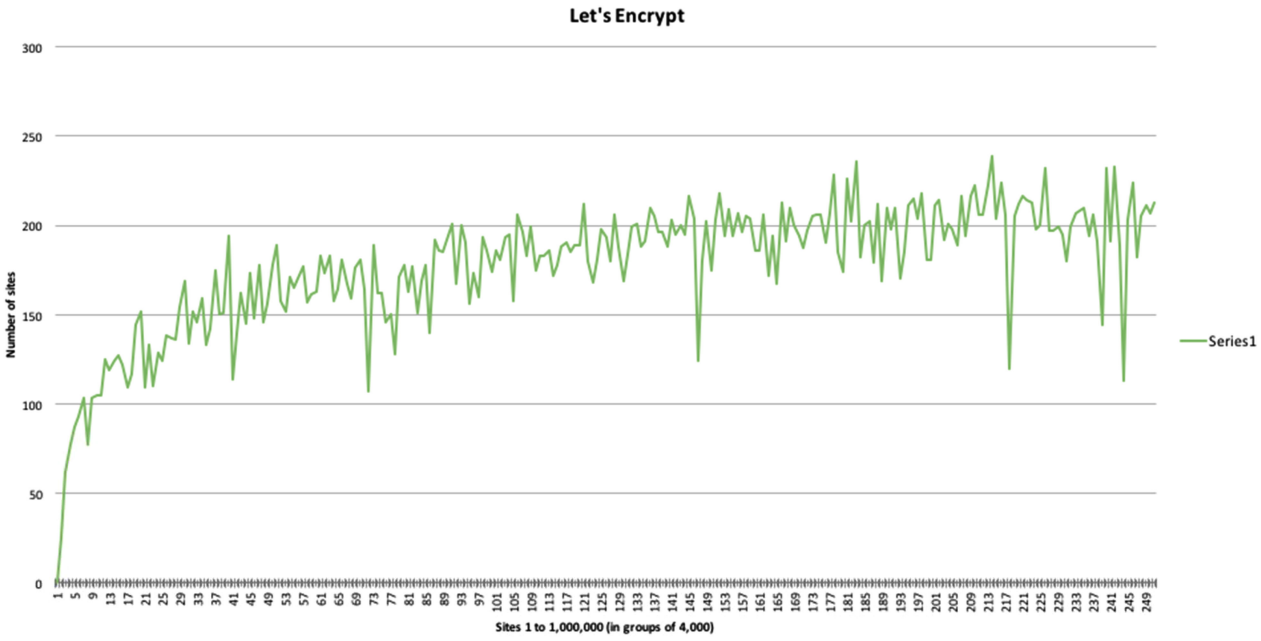

Fig. 8 *Let's Encrypt*

7 References

[1] Fielding, R., Gettys, J., Mogul, J., *et al.*: 'RFC 2616 – hypertext transfer protocol – HTTP/1.1'. Society [Internet], 1999, no. 2616, pp. 1–114. Available at: http://www.ietf.org/rfc/rfc2616.txt

[2] Klensin, J.: 'RFC 5321 – simple mail transfer protocol'. IETF RFC, 2008

[3] Postel, J., Reynolds, J.: 'RFC 959 – file transfer protocol'. Rfc 959 [Internet], 1985, pp. 1–69. Available at: https://www.ietf.org/rfc/rfc959.txt

[4] Mockapetris, P.: 'Domain names – implementation and specification [Internet]'. Request for Comments, 1987, pp. 1–55. Available at: https://www.ietf.org/rfc/rfc1035.txt

[5] Rescorla, E.: 'RFC 2818 – HTTP over TLS'. Network Working Group, IETF, 2000. p. pp. 1–8

[6] Sterne, B., Barth, A.: 'Content security policy 1.0 [Internet]. W3C. 2012'. Available at http://www.w3.org/TR/CSP/

[7] Bash, E.: 'RFC7469 public key pinning extension for HTTP'. PhD Propos, 2015, vol. **1**, pp. 1–28

[8] Hodges, J., Jackson, C., Barth, A.: 'HTTP strict transport security'. Available at http//tools.ietf.org/html/rfc6797. 2012

[9] Gondrom, T.: 'HTTP header field X-frame-options', IETF Standard, 2013. Available at: https://tools.ietf.org/html/rfc7034

[10] Hodson, H.: 'A little privacy, please'. New Sci [Internet], 2014, vol. **224, no**. 2997, p. 24. Available at: http://www.sciencedirect.com/science/article/pii/S0262407914622843

[11] Calzarossa, M.C., Massari, L.: 'Analysis of header usage patterns of HTTP request messages'. Proc. – 16th IEEE Int. Conf. on High Performance Computing and Communications, HPCC 2014, 11th IEEE Int. Conf. on Embedded Software and Systems, ICESS 2014 and 6th Int. Symp. on Cyberspace Safety and Security, 2014, pp. 847–853

[12] Nurse, J.R.C., Erola, A., Goldsmith, M., *et al.*: 'Investigating the leakage of sensitive personal and organisational information in email headers', *J. Internet Serv. Inf. Secur. [Internet]*, 2015, **1**, (February), pp. 70–84. Available at: https://www.cs.ox.ac.uk/files/7181/jisis2015_nurse_et_al.pdf

[13] VanderSloot, B., Amann, J., Bernhard, M., *et al.*: 'Towards a complete view of the certificate ecosystem'. Proc. of the 2016 ACM on Internet Measurement Conf., 2016, pp. 543–549
[14] Mozilla: 'Content-security-policy – HTTP|MDN [Internet]'. Available at https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy
[15] Fogie, S., Grossman, J., Hansen, R., *et al.*: 'XSS attacks: cross site scripting exploits and defense [internet]'. Management, 2007, p. 482. Available at http://portal.acm.org/citation.cfm?id=1534243
[16] Owasp: 'OWASP top 10 – 2013 [Internet]'. OWASP Top 10. 2013. Available at http://owasptop10.googlecode.com/files/OWASPTop10-2013.pdf
[17] Owasp: 'OWASP risk rating methodology [Internet]'. Owasp. 2013, pp. 1–5. Available at https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology
[18] Dhobale, D.D., Ghorpade, V.R., Patil, B.S., *et al.*: 'Steganography by hiding data in TCP/IP headers'. ICACTE 2010 – 2010 3rd Int. Conf. on Advanced Computer Theory and Engineering, Proc., 2010
[19] Stamm, S., Sterne, B., Markham, G.: 'Reining in the web with content security policy', Proc. 19th Int. Conf. World Wide Web WWW 10 [Internet], 2010, no. 2, p. 921. Available at http://portal.acm.org/citation.cfm?doid=1772690.1772784
[20] Yusof, I., Pathan, A.S.K.: 'Mitigating cross-site scripting attacks with a content security policy', *Computer (Long Beach Calif.)*, 2016, **49**, (3), pp. 56–63
[21] Yusof, I., Pathan, A.S.K: 'Preventing persistent cross-Site scripting (XSS) attack by applying pattern filtering approach'. 2014 the 5th Int. Conf. on Information and Communication Technology for the Muslim World, ICT4M 2014, 2014
[22] Johns, M.: 'Script-templates for the content security policy', *J. Inf. Secur. Appl.*, 2014, **19**, (3), pp. 209–223
[23] Weichselbaum, L., Spagnuolo, M., Lekies, S., *et al.*: 'CSP is dead, long live CSP! on the insecurity of whitelists and the future of content security policy'. Proc. 23rd ACM Conf. on Computer and Communications Security, Vienna, Austria, 2016
[24] Van Goethem, T., Chen, P., Nikiforakis, N., *et al.*: 'Large-scale security analysis of the web: challenges and findings'. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2014, pp. 110–126
[25] Chen, P.: 'Longitudinal study of the use of client-side security mechanisms on the European Web'. [cited 2017 May 26]. Available at http://www2016.net/proceedings/companion/p457.pdf
[26] Pries, R., Magyari, Z., Tran-Gia, P.: 'An HTTP web traffic model based on the top one million visited web pages'. 8th EURO-NF Conf. on Next Generation Internet, NGI 2012 – Proc., 2012, pp. 133–139
[27] Huang, C., Liu, J., Fang, Y., *et al.*: 'A study on Web security incidents in China by analyzing vulnerability disclosure platforms', *Comput. Secur.*, 2016, **58**, pp. 47–62
[28] Devi, G., Bal, R.S., Priyadarsini Sahoo, P.: 'Threats indentification in web application', *J. Netw. Commun. Emerg. Technol.*, 2016, **6**, (6), pp. 4557–4573
[29] Ying, M., Li, S.Q.: 'CSP adoption: current status and future prospects. secur commun networks [Internet]'. 2016 Oct 20 [cited 2016 Nov 15]. Available at http://doi.wiley.com/10.1002/sec.1649
[30] Bradbury, D.: 'Digital certificates: worth the paper they're written on?', *Comput. Fraud Secur.*, 2012, **2012**, (10), pp. 12–16
[31] Schuster, S., van den Berg, M., Larrucea, X., *et al.*: 'Mass surveillance and technological policy options: improving security of private communications', *Comput. Stand. Interfaces*, 2017, **50**, pp. 76–82
[32] Manousis, A., Ragsdale, R., Draffin, B., *et al.*: 'Shedding light on the adoption of let's encrypt'. arXiv Prepr arXiv161100469, 2016
[33] Leyden, J.: 'Inside 'Operation black tulip': digiNotar hack analysed [Internet]'. The Register. 2011. Available at http://www.theregister.co.uk/2011/09/06/diginotar_audit_damning_fail/
[34] SecurityWeek: 'StartSSL flaw allowed attackers to obtain SSL cert for any domain|SecurityWeek.Com [Internet]'. Available at http://www.securityweek.com/startssl-flaw-allowed-attackers-obtain-ssl-cert-any-domain
[35] Ristic, I.: 'Is HTTP public key pinning dead? – network security blog|Qualys, Inc'. [Internet]. Available at https://blog.qualys.com/ssllabs/2016/09/06/is-http-public-key-pinning-dead
[36] de los Santos, S., Torrano, C., Rubio, Y., *et al.*: 'Implementation state of HSTS and HPKP in both browsers and servers'. Int. Conf. on Cryptology and Network Security, 2016, pp. 192–207
[37] Kranch, M., Bonneau, J.: 'Upgrading HTTPS in Mid-Air: an empirical study of strict transport security and Key pinning'. [cited November 2017]. Available at http://www.jbonneau.com/doc/KB15-NDSS-hsts_pinning_survey.pdf HTTPS in Mid-Air- An Empirical Study of Strict Transport Security and Key Pinning.pdf