KNOWLEDGE – International Journal
Vol.30.3
March, 2019

HTTP SECURITY HEADERS

Lilyana Petkova
University of Library and Information technology (UNIBIT), Sofia, Bulgaria,
lilyanapetkova92@gmail.com

Abstract: Nowadays security becomes more important than the content and the SEO of a web application. Due to a lack of protection, the number of attacked websites augments in the past few years. In most of the cases, developers are either uninformed or unqualified to implement security during the application development, which causes a huge amount of data flaws.
Supporting the developers and easily managing the workflow, some organizations have developed different kind of guidelines for security integration. Such guide helps handling the security from the outset of the development process, which influence over the protection of the entire application. The one used in this article is a project developed by Open Web Application Security Project (OWASP) Foundation named OWASP Secure Headers Project. Its aim is to show the developers the balance between usability and security implemented through http headers. By giving general data and examples of HTTP response headers usability it is a platform which help increasing the security of the application.
In this article, we explain the necessity of HTTP Security Headers and how they can help in preventing a cyber invasion in our web application! We will give details on the most important HTTP headers and will retrieve a basic information for some with a lower need. We will give examples for their implementation in one ASP.NET web application to provide more descriptive perspective of their use!
In the recent years, browsers have integrated certain security header controls to support the web application security. Those headers give instructions to the browser how to behave when handling sensitive content and data of the application. If developers enable them in the application, browser will prevent attacks automatically. But not all browsers support them, which brings a compatibility question: what are the alternatives in a case of deprecated header on a specific browser.
As a part of the research we will provide an analyze of the use of the HTTP headers in some of the most common sites used in Bulgaria with the help of ALEXA Top 1 Million sites. There have already been developed a lot of applications to show if a certain website has HTTP security headers implemented. Most of them are freely to use and gives detailed information on what was done and what should be done in case that specific layer of security is missing from the web application.
The need of security in the web applications become more and more necessary. Along with other security implementations on a programming and on a server level the ones described in the article bring another layer of security management that mitigates certain types of cyberattacks and vulnerabilities.

Keywords: web security, HTTP headers, cyberattacks, vulnerability.

1. INTRODUCTION

Working with data should makes us paying attention to the way of manage it! The development life cycle does not start with applying all requirement to the application nor end with providing the client with a working application! It should start and end with a fully secure and corresponding to the requirements software which may resist to cyberattacks at any time! [19] There are a lot known and just as unknown attacks but if we may protect our application to just the known ones we have done our jobs according to client's desire or more generally said to Internet fast evolution!
With this article we want to introduce only a general information on how enabling HTTP security response headers on the application can help in preventing some of the most serious security attacks! In our further researches we will investigate some of them separately by providing more description and information on vulnerabilities it should protect from!
We have mentioned several free online tools which help in the development process but will be reviewed and demonstrated in further researches!

2. HTTP SECURITY HEADERS

2.1. Response Headers

By definition HTTP Security Headers represent a collection of metadata (*such as status error codes, cache-control, content-encoding, etc*) as a response from user activity on a specific page of the website. This data defined in a key-

value pair[3] instructs the browser how to behave when it handles the site content![12] In the recent years browser vendors integrated HTTP response headers into their browser [3] and in order to be activated into the application web developers need to enable them by following a few steps presented in the current research!

With the following set of data we are going to present the most important information for 10 HTTP headers and will pay attention to some of the most important HTTP headers according several experts! As a conclusion to this list we will show a compatibility table for browsers as Internet Explorer, Microsoft Edge, Chrome, Safari, Opera and Android!

According the Security Headers researcher Scott Helme 7 out of 10 are the most valuable headers which should be included in a site: X-Frame-Options; Strict-Transport-Security; Content-Security-Policy; X-XSS-Protection; X-Content-Type-Options; Referrer-Policy and Feature- Policy! That's why the comparison made during the research is based on those specific headers!

2.1.1. Strict Transport Security (HSTS)

Strict-Transport-Security header or HSTS restricts the application to be accessed only over HTTPS. Every non-encrypted instance run over HTTP will be eliminated entirely [9] which prevents Man-in-the-Middle(MITM) attacks![6] Enabling this header adds another layer of protections against protocol downgrade attacks [11] and provides supercookies to the application.[12]

There is a list of sites configured to use HSTS which eliminates the risk of load over HTTP which is due to the fact that browser is not aware of this policy in a first visitor's interaction. [6] This is part of the directive defined in the configuration of the STS header called **preload**. Here is some description on the 3 directive used by HSTS header:

Parameter Value	**Meaning**
max-age = SECONDS	Duration (in seconds) to tell a browser that requests are available only over HTTPS
includeSubDomains	Configuration is valid for subdomain as well
preload	Use if you would like your domain to be included in the HSTS preload list

Some of the requirements which makes the best of the preload configuration are: the availability of Certificate Authorities/CAs (a trusted set of data files that issues digital certificates[4]) approved certificate; full redirect to HTTPS; includeSubDomains directive specified; max-age at least 18 weeks (10886400 seconds); all redirects still have the HSTS header enabled![2]

2.1.2. Public Key Pinning(HPKP)

Public Key Pinning reduces the risk of Man-in-the-Middle (MITM) attacks by instructing the web browser to associate a specific cryptographic public key or certificate with the application host. This process is called Certificate pinning. It associates the application to a hashed public key part of a list of keys or certificates on the web server during a connection between the browser and the host![10] In a first load the browser saves the pinned certificate/key and checks it in future loads. That protects users in case of compromised certificate authority and helps the browser in differentiate the certificates to trust for a specific duration of time.[7]

It is a risky header because of a lost key/certificate or a wrong association, which can lock the user outside of the site.

Currently the HPKP supports SHA hash algorithm.[5]

An alternative is a Public-Key-Pins-Report-Only header, which reports problems but doesn't lock users out. [12]

Parameter Value	**Meaning**
report-uri="url"	Report to the specified URL if pin validation fails. This is optional.
pin-sha256="sha356key"	Specify the pins here
max-age=	Browser to remember the time in seconds that site is accessible only using one of the pinned keys.
IncludeSubDomains	This is applicable on a subdomain as well.

2.1.3. X-Frame-Options

The X-Frame-Options header helps in preventing clickjacking attacks. It instructs the browser if it is allowed to render the site in HTML iframe element. In this attack the user is redirected to a hacker's application which is a mirror of the requested site. Any action performed on it can be manipulated to act respectively on the original application which can harm the application. [12][13]

This header has also been deprecated and can be replaced by the Frame-Options directive of CSP header which is still in development. [6]

Parameter Value	Meaning
SAMEORIGIN	Frame/iframe of content is only allowed from the same site origin.
DENY	Prevent any domain to embed your content using frame/iframe.
ALLOW-FROM	Allow framing the content only on particular URI.

2.1.4. X-XSS-Protection

This header prevents cross-site scripting (XSS) attacks! It has been deprecated, but can be replaced by a directive in the Content-Security-Policy header called reflected-xss which is still in development! However if enabled this brings another level of security to the application and is important for older browser which do not support CSP![6]

Parameter Value	Meaning
0	XSS filter disabled
1	XSS filter enabled and sanitized the page if attack detected
1;mode=block	XSS filter enabled and prevented rendering the page if attack detected
1;report=http://example.com/report_URL	XSS filter enabled and reported the violation if attack detected

2.1.5. X-Content-Type-Options

The X-Content-Type-Options header protects against browser's attempts of content types detection called MIME, which may cause an execution of malicious code by attackers. [13] Enabling this header instructs the browser to allow all defined content file types and to disallow content sniffing. [12]

There is only one directive called "nosniff", which disables this sniffing.

2.1.6. Content-Security-Policy

Content Security Policy (CSP) gives the developers/administrators control on the resources (*such as script, styles, and so on*) used on the site![2][3] This policy protects the application against Cross Site Scripting (XSS) and other injection attacks.[9][16]

It consists of a set of directives which controls different type of resources.[2] Some of the directives to load or execute that we can protect by enabling this header are: default-src (for all resources type), script-src, object-src (for plugins), style-src, img-src, media-src (for video and audio), frame-src (for embed frames), font-src, connect-src(Define which URIs the protected resource can load using script interfaces[16]), report-uri(Specifies a URI to which the user agent sends reports about policy violation[16]).

Google provides with a free set of tools and extensions to help developers with the evaluation of the Content Security Policy! The online application can be found at this URL https://csp-evaluator.withgoogle.com/! It provides examples for safe and unsafe use and also the checks are for all versions of CSP.

XSS flaws that allows too much freedom from user's perspective are quite widespread and become more difficult to protect against. For that reason we are going to make a further deeper research for that specific policy!

2.1.7. X-Permitted-Cross-Domain-Policies

This header is used to avoid resource abuse by restricting the access of the application's assets such as PDF, DOC/x files, etc. [2][5]

Parameter Value	Meaning
none	no policy is allowed
master-only	allow only the master policy
all	everything is allowed
by-content-only	Allow only a certain type of content. Example – XML
by-ftp-only	applicable only for an FTP server

2.1.8. Referrer and Referrer-Policy

The Referrer HTTP header reveals information processed from the client side which is considered more as a privacy matter, than a security one! However, the mix of security and privacy flaws are considered in the Referrer-Policy header. [1]That header takes control over the information processed by Referrer which makes it a great for analytics[12]. Referrer-Policy consists of eight different directives:

Parameter Value	Meaning
no-referrer	Referrer information will not be sent with the request.
no-referrer-when-downgrade	the default setting where referrer is sent to the same protocol like HTTP to HTTP, HTTPS to HTTPS
unsafe-url	full URL will be sent with the request.
same-origin	referrer will be sent only for same origin site.
strict-origin	send only when a protocol is HTTPS
strict-origin-when-cross-origin	the full URL will be sent over a strict protocol like HTTPS
origin	send the origin URL in all the requests
origin-when-cross-origin	send FULL URL on the same origin. However, send only origin URL in other cases

2.1.9. Expect-CT

Expect-CT is relatively new HTTP header which indicates the need of the evaluation of the connection to the web servers emitting Certificate Transparency/CT. [8] It is considered to be safer than the HPKP and also as a substitute of HPKP due to announced plans of Google Chrome deprecation. The reasons are the flexibility in recovering from any configuration errors and the built-in support offered by a number of CAs. [15]

Parameter Value	Meaning
max-age	In seconds, for how long browser should cache the policy.
enforce	An optional directive to enforce the policy.
report-uri	Browser to send a report to the specified URL when valid certificate transparency not received.

2.1.10. Feature-Policy

This is a new security header introduced on July 2018 which is still in development. It allows the site owners to restrict certain features on their own site and those they embed on their own. As we specify this header is still new and the list of its values is still not final. The known so far are: geolocation; midi; notifications; push; sync-xhr; microphone; camera; magnetometer; gyroscope; speaker; vibrate; fullscreen; payment (PaymentRequest). [16]
The different options for site owners to work with are:

Option	Meaning
*****	This origin will allow the current page and any nested browsing contexts (i.e. iFrames) to use the feature
self	This limits use of the feature to the current page, as well as any other nested browsing contexts, provided they are on the same-origin
none	This completely disables the feature on both the page and on any other nested browsing contexts
<origin(s)>	Only the specified origins will be allowed to use this feature.

2.2. Configuration possibilities

There are 2 ways of enabling the HTTP security response headers for an ASP.NET application: through the code or through Microsoft Internet Information Services (IIS) Manager. If the header to apply has more than one possible values, they follow one another by separating them with "**;**". Here are the possibilities for HTTP security response headers configuration:

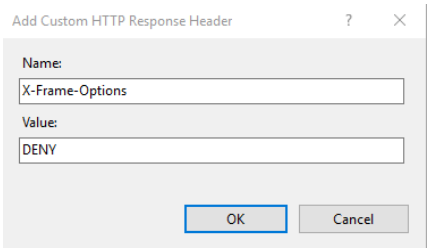
2.2.1. Through the code

One of the options to enable a HTTP header for an ASP.NET application is by adding a few lines of code to the configuration. Just open the web.config file and between **system.webServer>httpProtocol>customHeaders** settings append the necessary configuration . The best way of doing it is first to remove the relevant key from the responses and then add it again. That way there won't be any misleads:

```
<remove name="X-Frame-Options" />
<add name="X-Frame-Options" value="DENY" />
```

2.2.2. Microsoft Internet Information Services (IIS) Manager

Open the Microsoft IIS Manager and either select a site, or select the entire server. From the Features View open the HTTP Response Headers feature by double clicking on the item or by selecting it and on the right section of the window click Open Feature! On the current window all enabled (*if there are*) HTTP headers will



be displayed. You can edit an available one by clicking on the it or adding a new one by clicking the Add option under Actions section on the right. Enter the name of the header and its value.
The final step is to restart the IIS in order to apply those changes to the site.

2.3. Browser Compatibility

The following table shows the browser compatibility according the OWASP Secure Headers Project with last updates from January 2019.
What we need to take attention to is the HPKP response header! According the OWASP SHP is not supported from IE, Edge and Safari, but according to Google, after version 68 it has been deprecated since May 2018.[15] In its place according to the Google team the best replacement is the use of Certificate Transparency and the relatively new Expect-CT header.[15]

	IE	Edge	Firefox	Chrome	Safari	Opera	Android
HTTP Strict Transport Security (HSTS)	**11**	**13**	**47**	**49**	**9.1**	**39**	**4.4**
Public Key Pinning Extension for HTTP (HPKP)	**NS**	**NS**	**47**	**49**	**NS**	**39**	**51**
X-Frame-Options	**8**	**13**	**47**	**49**	**9.1**	**39**	**4.4**
X-XSS-Protection	**8**		**NS**	**4+**			
X-Content-Type-Options	**8**		**51**	**1.0**	**NS**	**13**	
Content-Security-Policy	**11**	**13**	**47**	**49**	**9.1**	**39**	**4.4**
X-Permitted-Cross-Domain-Policies							
Referrer-Policy	**NS**	**NS**	**50**	**56**	**NS**	**43**	
Expect-CT				**61**		**48**	
Feature-Policy							

3. ALEXA 6 WEB SITES SECURITY HEADERS ANALYSES

ALEXA is an amazon.com company for analytical insight. It provides a traffic estimates based on a proprietary calculating methodology which combines an average of a daily basis visits and its numbers of pageviews over the past 3 months. [14]
By using those analyses we randomly select 6 websites from an excerpt of 50 sites based on a selected Country category - Bulgaria. According to ALEXA this categorization is measurement of how a website ranks in a particular country relative to other sites over the past month. [14]
The sites selected for this articles were selected on a personal opinion based on the most advertised and the most commonly used website. We are not going to present the name of sites due to law and policies restrictions!
As we have already mentioned there are a number of free tools which helps us check if a particular website uses HTTP security headers. In the guide of the OWASP Secure Headers Project the experts provide a list of such scanners which according to them "can help in achieving more secure and trustworthy web systems"[8].
The one used in this research can be found on the following Internet address: https://securityheaders.io/. This tool shows the integrated HTTP security headers and their values on a specific web application and the missing ones with more details on them.
As we see from the table below 3 out of 6 websites are poor secure and only one is with higher security level. Which makes us think about the level of HTTP security response headers integration!

HTTP header	site A	site B	site C	site D	site E	site F
X-Frame-Options	-	+	-	+	+	-
Strict-Transport-Security	+	+	-	?	-	+
Content-Security-Policy	+	-	-	-	-	-
X-XSS-Protection	+	-	-	+	-	-
X-Content-Type-Options	+	-	-	-	-	-
Referrer-Policy	+	-	-	+	-	-
Feature-Policy	-	-	-	-	-	-
Assessment	**A**	**D**	**F**	**D**	**F**	**D**

And outside of the provided statistics for those websites in our personal experience the number of clients familiar with this security functionality is lower than expected! Which brings the question about the side responsible for bringing it on the surface of thoughts!

In the other hand, all of those checks provided from https://securityheaders.io can be programmed with only 3 lines of code:

```
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(site_url);//or just take the HttpContext.Current
HttpWebResponse response = (HttpWebResponse)request.GetResponse();
if (response.Headers.Count>0){}
```

Those 2 lines provide with the HTTP headers information of an already deployed application. But if you replace the first line with HttpContext.Current query, you will be aware if the site in the development processes is having that security layer enabled.

4. CONCLUSION

The lack of security provides a huge omission in the application or software development life cycle. And the only obstacle of implementing it is time! But as shown above with enabling HTTP security headers time is no longer an impediment! According to our own experience adding them into the configuration of an ASP.NET application takes less than an hour! So enabling them we go up with one more step on the climb to assure the confidentiality and the protection of our application.

REFERENCES

[1] Artūrs Lavrenovs, F. Jesús Rubio Melón, HTTP Security Headers Analysis of Top One Million Websites, 2018
[2] Koray Emre KISA, Dr. Emin İslam TATLI, Analysis of HTTP Security Headers in Turkey, 9th International Information Security and Cryptography Conference (ISCTurkey16), October 2016
[3] William J. Buchanan1 , Scott Helme2, Alan Woodward, Analysis of the adoption of security headers in HTTP, October 2017
[4] Certificate Authorities & Trust Hierarchies
[5] CHANDAN KUMAR, How to Implement Security HTTP Headers to Prevent Vulnerabilities?, December 2018
[6] Chris Reeves, Top HTTP Security Headers and How to Deploy Them, July 2018
[7] Brian Jackson, Hardening Your HTTP Security Headers, October 2018
[8] OWASP Secure Headers Project
[9] Jay Thakkar, HTTP Security Headers: 5 Headers You Must Implement on Your Sit, April 2018
[10] Johne Jacob, 7 Security Response Headers Every Security Tester Should Know, June 2018
[11] Madhusudhan Rajappa, An Introduction to HTTP Response Headers for Security, March 2018
[12] Luka Labrovic, The 8 HTTP Security Headers Best Practices, July 2017
[13] Bc. Klara Drhova, Authentication, authorization, and session management in the HTTP protocol, Master's thesis, April 2018
[14] ALEXA
[15] Liam Tung, Google: Chrome is backing away from public key pinning, and here's why, October 2017
[16] Scott Helme, A new security header: Feature Policy, July 2018
[17] ICT Security Trends, Willian Dimitrov, Sofia, 2017, Avangard, ISBN 978-619-160-766-2
[18] Software testing, Willian Dimitrov, Sofia, 2017, Avangard, ISBN 978-619-160-765-5
[19] ICT Security Model, Willian Dimitrov, Sofia, 2018, Avangard, ISBN 978-619-160-950-5