Tweet

Hello All,

I have been working as a software developer over 2 years on different technologies such as PHP, javascript, HTML, Python, nodeJS, ShellScript etc. So, Web Application Security is one of my interested fields. The reason for writing this post is, as a developer, we always focus on complete the functionality of the project.

However, writing the secure code is important as well. So here I would like to share some Important configuration details that Every Developer should aware of that. This blog will help you to secure your web application from different attacks OR Fixing the application vulnerable to attacks such as XSS, ClickJacking, Code Injection, Man In The Middle Attack etc.

This attacks can be done because of some security misconfiguration in server or code is not sanitized, its very easy for an attacker to detect security flaws in an application by using the browser console attacker can check for security headers configuration. if headers not properly configured on any HTTP request, its become entry point for an attacker.

Well, we will see in detail all and configuration as well.

Index

4. **X-Content-Type-Options**
5. **HTTP Public Key Pinning**
6. **Content Security Policy**
7. **X-Permitted-Cross-Domain-Policies**
8. **Referrer Policy**
9. **Expect-CT**
10. **Clear-Site-Data**

What is HTTP Request?

the client sends HTTP request to the server for the resource. HTTP Request made up of following.

Request line

this contains the HTTP method i.e. GET, HEAD, POST, etc. followed by the request URL.

```
e.g. GET  /xyz.com/test/  HTTP/1.1
```

Request Headers

HTTP Request headers contain 0 or more request headers in the request section, followed by request line to pass additional information with request or response, headers are the important part of Request or Response mainly intended for communication between server or client.

Message Body

The message body part is optional for an HTTP message but if it is available, then it is used to carry the entity-body associated with the request or response. If entity body is associated, then usually **Content-Type** and **Content-Length** headers lines specify the nature of the body associated.

Types of HTTP Headers

1. **Generate headers**: this header field has general applicability for request or response.
2. **client request headers**: this header field has general applicability for request message.
3. **server response headers**: this header field has general applicability for response message.
4. **entity headers**: it defines meta information about entity body.

Why Headers are important from a Security Perspective?

There are many ways to do this while writing the secure code, but Its good to start with exploring your HTTP Security Headers.

When a user visit website on Browser, the server responds with HTTP Response Headers with Response Body. These headers tell the browser how to behave during communication with the site. These headers mainly include **metadata**, **content-**

encoding, **cache-control**, **status**, **error codes**, etc.

In this article, we are focusing on implementation of **Security Headers** using Configuring Server and Doing by Coding.

- Apache Server Configuration file.
- .htaccess Rules
- PHP Programming.

Headers

Let's Talk about each Header in depth.

X-XSS-Protection

XSS protection header can prevent some level of XSS attack.This header doesn't let the page load when it detects a cross-site scripting attack. There are four possible parameters to configure this header.

1. **X-XSS-Protection: 0**
   - xss filter disabled.
2. **X-XSS-Protection: 1**
   - xss filter enebled.
3. **X-XSS-Protection: 1; mode=block**
   - xss filter enebled and & prevented rendering the page if attack is detected.
4. **X-XSS-Protection: 1; report=<reporting-uri>**
   - xss filter enebled & reported violation if attack is detected.

Apache Server Configuration file (Add following entry in httpd.conf file)

```
Header set X-XSS-Protection "1; mode=block"
```

Set header in .htaccess file

```
<IfModule mod_headers.c>
Header set X-XSS-Protection "1; mode=block"
```

```
</IfModule>
```

PHP Programming(Add in PHP file)

```
<?php header("X-XSS-Protection: 1; mode=block"); ?>
```

HTTP Strict Transport Security

HTTP Strict transport Security header is used to ensure that all communication between client and server is sent over HTTPS, this header prevent HTTP click through prompts and redirect the HTTP request to https.

***Note**: Before implementing this header you must ensure all pages of your website are accessible over https, else this pages will be blocked.*

This header includes 3 parameters

- **max-age**
  - duration in seconds to tell browser that request 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.

Apache Server Configuration file (Add following entry in httpd.conf file)

```
Header set Strict-Transport-Security "max-age=31536000;
```

```
includeSubDomains; preload"
```

Set header in .htaccess file

```
<IfModule mod_headers.c>
Header set Strict-Transport-Security "max-age=31536000;
includeSubDomains; preload" env=HTTPS
```

```
</IfModule>
```

PHP Programming(Add in PHP file)

```
<?php header("strict-transport-security: max-age=600"); ?>
```

X-Frame-Options

X-Frame-Option header is used to prevent **clickjacking** on your website, by implementing this header you instruct the browser not to enable your webpage in an iframe.

X-Frame-Options help to protect against these kinds of attacks. This is done by disabling the iframes present on the site. In other words, it doesn’t let others embed your content.

This header has **3 parameters**.

1. **DENY**
   - Prevent any domain to embed your content using an iframe.
2. **SAMEORIGIN**
   - The content of an iframe is only allowed from same-origin.
3. **ALLOW-FROM**
   - Allowing framing content only on particular URL.

Apache Server Configuration file (Add following entry in httpd.conf file)

```
Header always set X-Frame-Options "SAMEORIGIN"
```

Set header in .htaccess file.

```
<IfModule mod_headers.c>
Header always append X-Frame-Options SAMEORIGIN
```

```
</IfModule>
```

PHP Programming(Add in PHP file)

```
<?php header('X-Frame-Options: SAMEORIGIN'); ?>
```

X-Content-Type-Options

This header is used to Prevent MIME types security risk by adding this header to your web page's HTTP response. Having this header instruct browser to consider files types as defined and disallow content sniffing.

There is only one parameter you got to add **"nosniff".**

Apache Server Configuration file (Add following entry in httpd.conf file)

```
Header set X-Content-Type-Options nosniff
```

Set header in .htaccess file.

```
<IfModule mod_headers.c>
Header set X-Content-Type-Options nosniff
```

```
</IfModule>
```

PHP Programming(Add in PHP file)

```
<?php header('X-Content-Type-Options: nosniff'); ?>
```

HTTP Public Key Pinning

HPKP header is used to reduce the risk of Man-In-The-Middle attack, by pinning certificates, this is possible with HPKP header. you can pin root certificate public key or immidiate certificate at the time of writing.
This header includes 4 parameters.

- **report-url=”url”**
  - report to specific url if pin validation fails.
- **pin-sha 256 = “sha256 key”**
  - here we can specify the pins.
- **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.

Aadd following Apache Server Configuration file (Add following entry in httpd.conf file)entry in httpd.conf file.

```
public-key-pins-report-only:max-age=500; pin-
sha256="WoiWRyIOVNa9ihaBciRSC7XHjliYS9VwUGOIud4PB18="; pin-
sha256="r/mIkG3eEpVdm+u/ko/cwxzOMo1bk4TyHIlByibiA5E="; pin-
sha256="q4PO2G2cbkZhZ82+JgmRUyGMoAeozA+BSXVXQWB8XWQ="; report-
```

```
uri=http://xyz.com/hpkp/
```

Set header in .htaccess file.

```
<IfModule mod_headers.c>
Header set Public-Key-Pins "pin-
sha256=\"d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM=\"; pin-
sha256=\"E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=\"; max-
```

```
age=604800; report-uri=\"https://example.net/pkp-report\""</IfModule>
```

PHP Programming(Add in PHP file)

```
<?php
header('Public-Key-Pins: pin-
sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM="; pin-
sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g="; max-age=604800;
includeSubDomains; report-uri="https://example.net/pkp-report"');
```

```
?>
```

Content Security Policy

This header policy is used to specify which content should be loaded in a web application, also used to prevent clickjacking, code injection attack by implementing the content security policy header.

This header includes 2 parameters.

- **default-src**
  - Load everything from a defined source.
- **script-src**
  - Load only scripts from defined source.

Apache Server Configuration file (Add following entry in httpd.conf file)

```
Header set Content-Security-Policy "default-src 'self';"
```

Set header in .htaccess file.

```
<IfModule mod_headers.c>
Header set Content-Security-Policy "default-src 'none'; script-src
'self'; connect-src 'self'; img-src 'self'; style-src 'self';"
```

```
</IfModule>
```

PHP Programming(Add in PHP file)

```
<?php
header("Content-Security-Policy: default-src 'none'; script-src
'self'; connect-src 'self'; img-src 'self'; style-src 'self';");
?>
```

X-Permitted-Cross-Domain-Policies

this header prevents Adobe Flash and Adobe Acrobat from loading content on your site. this header instruct the browser how to handle request over cross domain by implementing this header, you restrict loading your site assets from other domain to avoide resource misuse.

This header includes following parameter.

- **Value**
  - Description
- **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. (e.g.-XML)
- **by-ftp-only**
  - Applicable only for an FTP server

Apache Server Configuration file (Add following entry in httpd.conf file)

```
Header set X-Permitted-Cross-Domain-Policies “master-only”
```

Set header in .htaccess file.

```
<IfModule mod_headers.c>
```

```
Header set X-Permitted-Cross-Domain-Policies “master-only” </IfModule>
```

PHP Programming(Add in PHP file)

```
<?php
Header ("X-Permitted-Cross-Domain-Policies “master-only | none”");
```

```
?>
```

Referrer Policy

Referrer-Policy header used to prevent cross-domain Referer leakage. Referral policy deals with what information related to the url the browser sends to a server to retrieve an external resource.

Referrer-Policy supports the following syntax.

- **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.

Apache Server Configuration file (Add following entry in httpd.conf file)

Header set Referrer-Policy “no-referrer”

Set header in .htaccess file

```
<IfModule mod_headers.c>
Header set Referrer-Policy "origin-when-cross-origin"
```

```
</IfModule>
```

PHP Programming(Add in PHP file)

```
<?php
header("Referrer-Policy: origin-when-cross-origin");
```

```
?>
```

Expect-CT

Expect-CT allows web host operators to discover misconfigurations in their Certificate Transparency deployments, also hosting operaters can use Expect-CT to ensure that, if a user agent which supports Expect-CT accepts a misissued certificate, that certificate will be discoverable in Certificate Transparency logs. header still in experimental status is to instruct the browser to validate the connection with web servers for certificate transparency.

This project by Google aims to fix some of the flaws in the SSL/TLS certificate system.

Expect-CT has the following parameters.

- **max-age**
  - Browser should cache the policy as the time specified in seconds.
- **enforce**
  - It is an optional directive to enforce the policy.
- **report-uri**
  - Browser to send a report to the specified URL when valid certificate transparency not received.

Apache Server Configuration file (Add following entry in httpd.conf file)

```
Header set Expect-CT 'enforce, max-age=43200, report-
```

```
uri="https://any_domain.com/report"'
```

Set header in .htaccess file.

```
<IfModule mod_headers.c>
Header set Expect-CT "max-age=7776000, enforce"
```

```
</IfModule>
```

PHP Programming(Add in PHP file)

```
<?php header("Expect-CT: max-age=7776000, enforce"); ?>
```

Clear-Site-Data

prompts the user agent to clear browsing data associated with the requesting website. A sensitive website can trigger local data deletion after the user signs out. A website dealing with a persistent XSS attack can use this to ‘reset’ itself to a clean state.

This header has following parameters

- **“*” (wildcard)**
  - clear all types of data.
- **“cache”**
  - clears browser cache
- **“cookies”**
  - clear all browser cookies on entire domain, including subdomains.
- **“storage”**
  - clear all DOM storage, including localStorage, sessionStorage, IndexedDB, AppCache, WebSQL, Server Workers.

Apache Server Configuration file (Add following entry in httpd.conf file)

```
Clear-Site-Data: "cache", "cookies", "storage", "executionContexts".
```

Set header in .htaccess file.

```
<IfModule mod_headers.c>
Header set Cache-Control "no-cache, no-store, must-revalidate" Header
set Pragma "no-cache" Header set Expires 0
```

```
</IfModule>
```

PHP Programming(Add in PHP file)

```
<?php
header("Expires: Tue, 01 Jan 2000 00:00:00 GMT"); header("Last-
Modified: " . gmdate("D, d M Y H:i:s") . " GMT"); header("Cache-
Control: no-store, no-cache, must-revalidate, max-age=0");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
```

```
?>
```

Hope You Enjoying Reading and Learnt something from this post. Feel Free to ask anything.

Thank You.

Sachin Sase

I am Full Stack Developer working in Pune. I have great experienced in various language such as **SQL, Web Technologies, PHP, ASP.NET, Angular JS, Node JS, HTML, Bootstrap, CSS, JavaScript** and **SEO** as well. I have built many **Information Security** related **tools** in different languages. I love to share my knowledge with others.