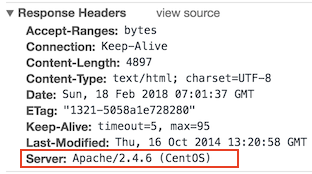
**Apache Security**

1. **Remove Server Version Banner**

I would say this is one of the first things to consider, as you don’t want to expose what web server version you are using. Exposing version means you are helping hacker to speedy the reconnaissance process.

The default configuration will expose Apache Version and OS type as shown below.



* Go to $Web\_Server/conf folder
* Modify httpd.conf by using the vi editor
* Add the following directive and save the httpd.conf

ServerTokens Prod

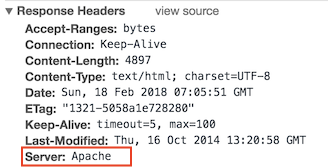
ServerSignature Off

* Restart apache

ServerSignature will remove the version information from the page generated by Apache.

ServerTokens will change Header to production only, i.e., Apache

As you can see below, version & OS information is gone.



1. **Disable directory browser listing**

Disable directory listing in a browser, so the visitor doesn’t see what all file and folders you have under root or subdirectory.

Let’s test how does it look like in default settings.

* Go to $Web\_Server/htdocs directory
* Create a folder and few files inside that

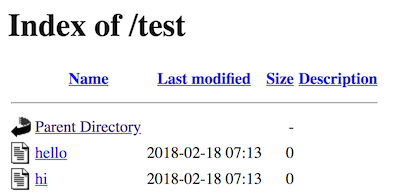
# mkdir test

# touch hi

# touch hello

Copy

Now, let’s try to access Apache by <http://localhost/test>



As you could see it reveals what all file/folders you have and I am sure you don’t want to expose that.

* Go to $Web\_Server/conf directory
* Open httpd.conf using vi
* Search for Directory and change Options directive to None or –Indexes

<Directory /opt/apache/htdocs>

Options -Indexes

</Directory>

Copy

(or)

<Directory /opt/apache/htdocs>

Options None

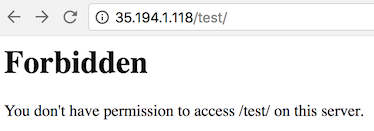
</Directory>

Copy

* Restart Apache

Note: if you have multiple Directory directives in your environment, you should consider doing the same for all.

Now, let’s try to access Apache by <http://localhost/test>



As you could see, it displays a forbidden error instead of showing test folder listing.

**Etag**

It allows remote attackers to obtain sensitive information like inode number, multipart MIME boundary, and child process through Etag header.

To prevent this vulnerability, let’s implement it as below. This is required to fix for PCI compliance.

* Go to $Web\_Server/conf directory
* Add the following directive and save the httpd.conf

FileETag None

Copy

* Restart apache

1. **Protect binary and configuration directory permission**

By default, permission for binary and configuration is 755 that means any user on a server can view the configuration. You can disallow another user to get into conf and bin folder.

* Go to $Web\_Server directory
* Change permission of bin and conf folder

# chmod –R 750 bin conf

Copy

1. **System Settings Protection**

In a default installation, users can override apache configuration using .htaccess. If you want to stop users from changing your apache server settings, you can add AllowOverride to None as shown below.

This must be done at the root level.

* Go to $Web\_Server/conf directory
* Open httpd.conf using vi
* Search for Directory at a root level

<Directory />

Options -Indexes

AllowOverride None

</Directory>

Copy

* Save the httpd.conf
* Restart Apache

1. **HTTP Request Methods**

HTTP 1.1 protocol support many request methods which may not be required and some of them are having potential risk.

Typically you may need GET, HEAD, POST request methods in a web application, which can be configured in the respective Directory directive.

Default configuration support OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT method in HTTP 1.1 protocol.

* Go to $Web\_Server/conf directory
* Open httpd.conf using vi
* Search for Directory and add the following

<LimitExcept GET POST HEAD>

deny from all

</LimitExcept>

Copy

* Restart Apache

1. **Disable Trace HTTP Request**

By default Trace method is enabled in Apache web server.

Having this enabled can allow Cross Site Tracing attack and potentially giving an option to a hacker to steal cookie information. Let’s see how it looks like in default configuration.

* Do a telnet web server IP with listening port
* Make a TRACE request as shown below

#telnet localhost 80

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

TRACE / HTTP/1.1 Host: test

HTTP/1.1 200 OK

Date: Sat, 31 Aug 2013 02:13:24 GMT

Server: Apache

Transfer-Encoding: chunked

Content-Type: message/http 20

TRACE / HTTP/1.1

Host: test

0

Connection closed by foreign host.

#

Copy

As you could see in above TRACE request, it has responded my query. Let’s disable it and test it.

* Go to $Web\_Server/conf directory
* Add the following directive and save the httpd.conf

TraceEnable off

Copy

* Restart apache

Do a telnet web server IP with listen port and make a TRACE request as shown below

#telnet localhost 80

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

TRACE / HTTP/1.1 Host: test

HTTP/1.1 405 Method Not Allowed

Date: Sat, 31 Aug 2013 02:18:27 GMT

Server: Apache Allow:Content-Length: 223Content-Type: text/html; charset=iso-8859-1 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"> <html><head>

<title>405 Method Not Allowed</title> </head><body>

<h1>Method Not Allowed</h1>

<p>The requested method TRACE is not allowed for the URL /.</p> </body></html>

Connection closed by foreign host.

#

Copy

As you could see in above TRACE request, it has blocked my request with HTTP 405 Method Not Allowed.

Now, this web server doesn’t allow TRACE request and help in blocking Cross Site Tracing attack.

1. **Set cookie with HttpOnly and Secure flag**

You can mitigate most of the common Cross Site Scripting attack using HttpOnly and Secure flag in a cookie. Without having HttpOnly and Secure, it is possible to steal or manipulate web application session and cookies, and it’s dangerous.

* Ensure mod\_headers.so is enabled in your httpd.conf
* Go to $Web\_Server/conf directory
* Add the following directive and save the httpd.conf

Header edit Set-Cookie ^(.\*)$ $1;HttpOnly;Secure

Copy

* Restart apache

1. **Clickjacking Attack**

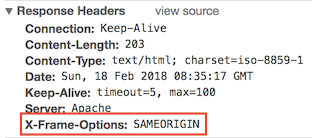
Clickjacking is a well-known web application vulnerabilities.

* Ensure mod\_headers.so is enabled in your httpd.conf
* Go to $Web\_Server/conf directory
* Add the following directive and save the httpd.conf

Header always append X-Frame-Options SAMEORIGIN

Copy

* Restart apache



X-Frame-Options also support two more options which I explained [here](https://geekflare.com/secure-apache-from-clickjacking-with-x-frame-options/).

1. **Server Side Include**

Server Side Include (SSI) has a risk of increasing the load on the server. If you have shared the environment and heavy traffic web applications you should consider disabling SSI by adding Includes in Options directive.

SSI attack allows the exploitation of a web application by injecting scripts in HTML pages or executing codes remotely.

* Go to $Web\_Server/conf directory
* Open httpd.conf using vi
* Search for Directory and add Includes in Options directive

<Directory /opt/apache/htdocs>

Options –Indexes -Includes

Order allow,denyAllow from all

</Directory>

Copy

* Restart Apache

Note: if you have multiple Directory directives in your environment, you should consider doing the same for all.

1. **X-XSS Protection**

Cross Site Scripting (XSS) protection can be bypassed in many browsers. You could apply this protection for a web application if it was disabled by the user. This is used by a majority of giant web companies like Facebook, Twitter, Google, etc.

* Go to $Web\_Server/conf directory
* Open httpd.conf using vi and add following Header directive

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

Copy

* Restart Apache

As you can see, XSS-Protection is the injected in the response header.

