CHECKLIST

APACHE

Remove Server Version Banner

Implementation:

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

```
ServerTokens Prod

ServerSignature Off
```

- Restart apache

Disable directory browser listing

- Go to **$Web_Server/htdocs** directory
- Create a folder and few files inside that

```
# mkdir test

# touch hi

# touch hello
```

Implementation:

- 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 None

Order allow,deny

Allow from all

</Directory>
```

(or)

```
<Directory /opt/apache/htdocs>
Options -Indexes
Order allow,deny
Allow from all
</Directory>
```

- **Restart Apache**

<u>Etag</u>

Implementation:

- Go to **$Web_Server/conf** directory
- Add the following directive and save the **httpd.conf**

```
FileETag None
```

- Restart apache

<u>Run Apache from non-privileged account</u>

Implementation:

- Create a user and group called apache

```
#groupadd apache
# useradd –G apache apache
```

- Change apache installation directory ownership to newly created non-privileged user

```
# chown –R apache:apache /opt/apache
```

- Go to **$Web_Server/conf**
- Modify **httpd.conf** using vi
- Search for User & Group Directive and change as non-privileged account apache

```
User apache
Group apache
```

- Save the **httpd.conf**
- Restart Apache

Verification:

grep for running http process and ensure it's running with apache user

```
# ps –ef |grep http
```

Note: You could see one process is running with root. That's because Apache is listening on port 80 and it has to be started with root. We will talk about how to change port number later in this course.

Protect binary and configuration directory permission

Implementation:

- Go to **$Web_Server** directory
- Change permission of **bin** and **conf** folder

```
# chmod –R 750 bin conf
```

System Settings Protection

Implementation:

- Go to **$Web_Server/conf** directory
- Open **httpd.conf** using vi
- Search for Directory at root level

```
<Directory />
Options -Indexes
AllowOverride None
</Directory>
```

- Save the httpd.conf
- Restart Apache

HTTP Request Methods

Implementation:

- Go to **$Web_Server/conf** directory
- Open **httpd.conf** using vi
- Search for Directory and add following

```
<LimitExcept GET POST HEAD>
deny from all
</LimitExcept>
```

Disable Trace HTTP Request

```
#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.
#
```

Implementation:

- Go to **$Web_Server/conf** directory
- Add the following directive and save the **httpd.conf**

```
TraceEnable off
```

- Restart apache

Verification:

- 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: 223

Content-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.

#
```

Set cookie with HttpOnly and Secure flag

Implementation:

- 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
```

- Restart apache

Clickjacking Attack

Implementation:

- 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
```

- Restart apache

Server Side Include

Implementation:

- 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,deny
Allow from all
</Directory>
```

- Restart Apache

X-XSS Protection

Implementation:

- Go to **$Web_Server/conf** directory
- Open **httpd.conf** using vi and add following Header directive

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

Disable HTTP 1.0 Protocol

Implementation:

- Ensure to load **mod_rewrite** module in **httpd.conf** file
- Enable **RewriteEngine** directive as following and add Rewrite condition to allow only HTTP 1.1

```
RewriteEngine On
RewriteCond %{THE_REQUEST} !HTTP/1.1$
RewriteRule .* - [F]
```

Timeout value configuration

Implementation:

- Go to **$Web_Server/conf** directory
- Open **httpd.conf** using vi
- Add following in **httpd.conf**

```
Timeout 60
```

SSL Key

Implementation:

- You can use **openssl** to generate CSR with 2048 bit as below.
- Generate self-signed certificate

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout localhost.key -out localhost.crt
```

- Generate new CSR and private key

```
openssl req -out localhost.csr -new -newkey rsa:2048 -nodes -keyout localhost.key
```

- Add Personal Cert, Signer Cert and Key file in **httpd-ssl.conf** file under below directive

```
SSLCertificateFile # Personal Certificate

SSLCertificateKeyFile # Key File

SSLCACertificateFile # Signer Cert file
```

Verification:

Execute **sslscan** utility with the following parameter. Change localhost to your actual domain name.

```
sslscan localhost | grep –i key
```

- As you can see current SSL key is 2048 bit, which is stronger.

SSL Cipher

```
sslscan –no-failed localhost
```

Implementation:

- Go to **$Web_Server/conf/**extra folder
- Modify **SSLCipherSuite** directive in **httpd-ssl.conf** as below to reject RC4

```
SSLCipherSuite HIGH:!MEDIUM:!aNULL:!MD5:!RC4
```

- Save the configuration file and restart apache server

Note: if you have many weak ciphers in your SSL auditing report, you can easily reject them adding ! at beginning.

For ex – to reject **RC4: !RC4 Verification:** Again, we will use **sslscan** utility to validate as below command. Change localhost to your actual domain name.

sslscan –no-failed localhost

So now we don't see RC4 anymore as accepted Cipher. It's good to reject any low, medium, null or vulnerable cipher to keep yourself tension free from getting attacked. You can also scan your domain against Qualys SSL Labs to check if you have weak or vulnerable cipher in your environment.

Disable SSL v2 & v3

Implementation:

- Go to **$Web_Server/conf/**extra folder
- Modify **SSLProtocol** directive in **httpd-ssl.conf** as below to accept only TLS 1.0+

SSLProtocol –ALL +TLSv1 +TLSv1.1 +TLSv1.2

Verification:

Let's use **sslscan** utility to validate as below command. Change localhost to your actual domain name.

sslscan –no-failed localhost

Alternatively, you may check your website with online SSL/TLS Certificate tool.

Mod Security (Open Source web Application firewall)

Download & Installation of Mod Security:

http://www.modsecurity.org/download/

- Extract modsecurity-apache_2.7.5.tar.gz

# gunzip –c modsecurity-apache_2.7.5.tar.gz | tar xvf –

- Go to extracted folder modsecurity-apache_2.7.5

# cd modsecurity-apache_2.7.5

- Run the configure script including apxs path to existing Apache

# ./configure –with-apxs=/opt/apache/bin/apxs

- Compile & install with make script

# make

#make install

Configuring mod security:

- Add following a line to load module for Mod Security in **httpd.conf** and save the configuration file

LoadModule unique_id_module modules/mod_unique_id.so

LoadModule security2_module modules/mod_security2.so

- Restart apache web server

Mod Security is now installed! Next thing you have to do is to install Mod Security core rule to take a full advantage of its feature. Latest Core Rule can be downloaded from following a link, which is free.https://github.com/SpiderLabs/owasp-modsecurity-crs/zipball/master

- Copy downloaded core rule zip to **/opt/apache/conf** folder
- Unzip core rule file, you should see the extracted folder as shown below
- You may wish to rename the folder to something short and easy to remember. In this example, I will rename to crs.

Go to crs folder and rename modsecurity_crs10_setup.conf.example to modsecurity_crs10_setup.conf

Now, let's enable these rules to get it working with Apache web server.

Add following in httpd.conf

<IfModule security2_module>

Include conf/crs/modsecurity_crs_10_setup.conf

Include conf/crs/base_rules/*.conf

</IfModule>

In above configuration, we are loading Mod Security main configuration file modsecurity_crs_10_setup.conf and base rules **base_rules/*.conf** provided by Mod Security Core Rules to protect web applications.

- **Restart apache web server**

Lets get it started with some of the important configuration in Mod Security to harden & secure web applications. In this section, we will do all configuration modification in **/opt/apache/conf/crs/modsecurity_crs_10_setup.conf** We will refer **/opt/apache/conf/crs/modsecurity_crs_10_setup.conf** as **setup.conf** in this section for example purpose. It's important to understand what the OWASP rules are provided in free are. There are two types of rules provided by OWASP.

Base Rules – these rules are heavily tested and probably false alarm ratio is less**.**

Experimental Rules – these rules are for an experimental purpose and you may have the high false alarm. It's important to configure, test and implement in UAT before using these in a production environment.

Optional Rules – these optional rules may not be suitable for the entire environment. Based on your requirement you may use them.

If you are looking for CSRF, User tracking, Session hijacking, etc. protection then you may consider using optional rules. We have the base, optional and experimental rules after extracting the downloaded crs zip file from OWASP download page.

These rules configuration file is available in **crs/base_rules, crs/optional_rules and crs/experimental_rules folder.**

Let's get familiar with some of the base rules.

modsecurity_crs_20_protocol_violations.conf: This rule is protecting from Protocol vulnerabilities like response splitting, request smuggling, using non-allowed protocol (HTTP 1.0).

modsecurity_crs_21_protocol_anomalies.conf: This is to protect from a request, which is missing with Host, Accept, User-Agent in the header.

modsecurity_crs_23_request_limits.conf: This rule has the dependency on application specific like request size, upload size, a length of a parameter, etc.

modsecurity_crs_30_http_policy.conf: This is to configure and protect allowed or disallowed method like CONNECT, TRACE, PUT, DELETE, etc.

modsecurity_crs_35_bad_robots.conf: Detect malicious robots

modsecurity_crs_40_generic_attacks.conf: This is to protect from OS command injection, remote file inclusion, etc.

modsecurity_crs_41_sql_injection_attacks.conf: This rule to protect SQL and blind SQL inject request.

modsecurity_crs_41_xss_attacks.conf: Protection from Cross Site Scripting request.

modsecurity_crs_42_tight_security.conf: Directory traversal detection and protection**.**

modsecurity_crs_45_trojans.conf: This rule to detect generic file management output, uploading of http backdoor page, known signature.

modsecurity_crs_47_common_exceptions.conf: This is used as an exception mechanism to remove common false positives that may be encountered suck as Apache internal dummy connection, SSL pinger, etc.

Logging

Logging is one of the first things to configure so you can have logs created for what Mod Security is doing. There are two types of logging available; Debug & Audit log.

Debug Log: this is to duplicate the Apache error, warning and notice messages from the error log.

Audit Log: this is to write the transaction logs that are marked by Mod Security rule Mod Security gives you the flexibility to configure Audit, Debug or both logging. By default configuration will write both logs. However, you can change based on your requirement**.**

The log is controlled in **SecDefaultAction** directive.

Let's look at default logging configuration in **setup.conf**

SecDefaultAction "phase:1,deny,log"

To **Debug, Audit** log – use **"log"**

To log only **audit log** – use "**nolog,auditlog**"

To log only **debug log** – use "**log,noauditlog**"

You can specify the Audit Log location to be stored which is controlled by **SecAuditLog** directive.

Let's write audit log into **/opt/apache/logs/modsec_audit.log** by adding as shown below**.**

Implementation:

Add **SecAuditLog** directive in **setup.conf** and restart Apache Web Server

SecAuditLog /opt/apache/logs/modsec_audit.log

Enable Rule Engine

Implementation:

- Add **SecRuleEngine** directive in **setup.conf** and restart Apache Web Server

SecRuleEngine On

Common Attack Type Protection

XSS Attack:-

- Open Firefox and access your application and put <script> tag at the end or URL as shown below
- Monitor the modsec_audit.log in apache/logs folder

As you can see Mod Security blocks request as it contains <script> tag which is the root of XSS attack.

Directory Traversal Attack:-

- Open Firefox and access your application with directory traversal
- Monitor the **modsec_audit.log** in **apache/logs** folder

http://localhost/?../.../boot

- As you can see Mod Security blocks request as it contains directory traversal.

Change Server Banner

Implementation:

- Add **SecServerSignature** directive with your desired server name in **setup.conf** and restart Apache Web Server

SecServerSignature YourServerName

Ex:

```
[/opt/apache/conf/crs] #grep SecServer modsecurity_crs_10_setup.conf

SecServerSignature chandank.com

[/opt/apache/conf/crs] #
```

Verification:

- Open Firefox and access your application
- Check HTTP response headers in firebug, you should see Server banner is changed now as shown below.

Configure Listen

When you have multiple interface and IP's on a single server, it's recommended to have Listen directive configured with absolute IP and Port number. When you leave apache configuration to listen on all IP's with some port number, it may create the problem in forwarding HTTP request to some other web server. This is quite common in the shared environment.

Implementation:

- Configure Listen directive in **httpd.conf** with absolute IP and port as shown example below

```
Listen 10.10.10.1:80
```

Access Logging

Implementation:

- To capture time taken to serve the **request** and **SESSION ID** in **access log**
- Add **%T & %sessionID** in **httpd.conf** under **LogFormat** directive

```
LogFormat "%h %l %u %t "%{sessionID}C" "%r" %>s %b %T" common
```

You can refer http://httpd.apache.org/docs/2.2/mod/mod_log_config.html for a complete list of parameter supported in **LogFormat** directive in Apache Web Server.

Disable Loading unwanted modules

If you have compiled and installed with all modules then there are high chances you will have many modules loaded in Apache, which may not be required. Best practice is to configure Apache with required modules in your web applications. Following modules are having security concerns and you might be interested in disabling in **httpd.conf** of Apache Web Server. WebDAV (Web-based Distributed Authoring and Versioning). This module allows remote clients to manipulate files on the server and subject to various denial-of-service attacks. To disable comment following in **httpd.conf**

#LoadModule dav_module modules/mod_dav.so

#LoadModule dav_fs_module modules/mod_dav_fs.so

#Include conf/extra/httpd-dav.conf

Info Module The mod_info module can leak sensitive information using **.htaccess** once this module is loaded. To disable comment following in **httpd.conf**

#LoadModule info_module modules/mod_info.so

Reference: This wouldn’t be possible without guidance from the following link:

- http://httpd.apache.org/docs/2.4/
- http://www.modsecurity.org/documentation/
- https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project