SedaMicro

Contents

Security settings and remarks

Drupal and Amazon S3

For Drupal, changes must be made in the IAM policies for your Amazon S3 account in order to allow access to your buckets for file uploading and to prevent unauthorized viewing or access to other buckets on your account. The reason for this has to do with Amazon's S3 implementation, and this method allows buckets to be listed, and only one bucket to be accessed with full permissions. At this time Amazon has not made a method available which would disallow listing other buckets.

To do this you will need to go into your Identity and Access Management settings in the Amazon AWS console.

From there:

1. Create a user which will have access to the buckets
2. Create a group to which this user will belong.
3. Create two policies for this group. The group should have the following first policy:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Stmt1448479011000",
         "Effect": "Allow",
         "Action": [
            "s3:*"
         ],
         "Resource": [
            "arn:aws:s3:::YOURBUCKETID"
         ]
      },
      {
         "Sid": "Stmt1448479103000",
         "Effect": "Allow",
         "Action": [
            "s3:*"
         ],
         "Resource": [
            "arn:aws:s3:::YOURBUCKETID/*"
         ]
```

```
        }
    ]
}
```

4. The second policy should be as follows:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1448479290000",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

General Security for Wordpress

File Permissions

Some neat features of WordPress come from allowing various files to be writable by the web server. However, allowing write access to your files is potentially dangerous, particularly in a shared hosting environment.

It is best to lock down your file permissions as much as possible and to loosen those restrictions on the occasions that you need to allow write access, or to create specific folders with less restrictions for the purpose of doing things like uploading files.

Here is one possible permission scheme.

All files should be owned by your user account, and should be writable by you. Any file that needs write access from WordPress should be writable by the web server, if your hosting set up requires it, that may mean those files need to be group-owned by the user account used by the web server process.

/

The root WordPress directory: all files should be writable only by your user account, except .htaccess if you want WordPress to automatically generate rewrite rules for you.

/wp-admin/

The WordPress administration area: all files should be writable only by your user account.

/wp-includes/

The bulk of WordPress application logic: all files should be writable only by your user account.

/wp-content/

User-supplied content: intended to be writable by your user account and the web server process.

Within /wp-content/ you will find:

/wp-content/themes/

Theme files. If you want to use the built-in theme editor, all files need to be writable by the web server process. If you do not want to use the built-in theme editor, all files can be writable only by your user account.

/wp-content/plugins/

Plugin files: all files should be writable only by your user account.

Other directories that may be present with /wp-content/ should be documented by whichever plugin or theme requires them. Permissions may vary.

Changing file permissions

If you have shell access to your server, you can change file permissions recursively with the following command:

For Directories:

```
find /path/to/your/wordpress/install/ -type d -exec chmod 755 {} \;
```

For Files:

find /path/to/your/wordpress/install/ -type f -exec chmod 644 {} \;

Regarding Automatic Updates

When you tell WordPress to perform an automatic update, all file operations are performed as the user that owns the files, not as the web server's user. All files are set to 0644 and all directories are set to 0755, and writable by only the user and readable by everyone else, including the web server.

Database Security

If you run multiple blogs on the same server, it is wise to consider keeping them in separate databases each managed by a different user. This is best accomplished when performing the initial WordPress installation. This is a containment strategy: if an intruder successfully cracks one WordPress installation, this makes it that much harder to alter your other blogs.

If you administer MySQL yourself, ensure that you understand your MySQL configuration and that unneeded features (such as accepting remote TCP connections) are disabled. See Secure MySQL Database Design for a nice introduction.

Restricting Database User Privileges

For normal WordPress operations, such as posting blog posts, uploading media files, posting comments, creating new WordPress users and installing WordPress plugins, the MySQL database user only needs data read and data write privileges to the MySQL database; SELECT, INSERT, UPDATE and DELETE.

Therefore any other database structure and administration privileges, such as DROP, ALTER and GRANT can be revoked. By revoking such privileges you are also improving the containment policies.

Note: Some plugins, themes and major WordPress updates might require to make database structural changes, such as add new tables or change the schema. In such case, before installing the plugin or updating a software, you will need to temporarily allow the database user the required privileges.

WARNING: Attempting updates without having these privileges can cause problems when database schema changes occur. Thus, it is **NOT** recommended to revoke these privileges. If you do feel the need to do this for security reasons, then please make sure that you have a solid backup plan in place first, with regular whole database backups which you have tested are valid and that can be easily restored. A failed database upgrade can usually be solved by restoring the database back to an old version, granting the proper permissions, and then letting WordPress try the database update again. Restoring the database will return it back to that old version and the WordPress administration screens will then detect the old version and allow you to run the necessary SQL commands on it. Most WordPress upgrades do not change the schema, but some do. Only major point upgrades (3.7 to 3.8, for example) will alter the schema. Minor upgrades (3.8 to 3.8.1) will generally not. Nevertheless, **keep a regular backup**.

Securing wp-admin

Adding server-side password protection (such as BasicAuth) to /wp-admin/ adds a second layer of protection around your blog's admin area, the login screen, and your files. This forces an attacker or bot to attack this second layer of protection instead of your actual admin files. Many WordPress attacks are carried out autonomously by malicious software bots.

Simply securing the wp-admin/ directory might also break some WordPress functionality, such as the AJAX handler at wp-admin/admin-ajax.php. See the Resources section for more documentation on how to password protect your wp-admin/ directory properly.

The most common attacks against a WordPress blog usually fall into two categories.

1. Sending specially-crafted HTTP requests to your server with specific exploit payloads for specific vulnerabilities. These include old/outdated plugins and software.
2. Attempting to gain access to your blog by using "brute-force" password guessing.

The ultimate implementation of this "second layer" password protection is to require an HTTPS SSL encrypted connection for administration, so that all communication and sensitive data is encrypted. *See Administration Over SSL.*

Securing wp-includes

A second layer of protection can be added where scripts are generally not intended to be accessed by any user. One way to do that is to block those scripts using mod_rewrite in the .htaccess file. **Note:** to ensure the code below is not overwritten by WordPress, place it outside the # BEGIN WordPress and # END WordPress tags in the .htaccess file. WordPress can overwrite anything between these tags.

```
# Block the include-only files.

<IfModule mod_rewrite.c>

RewriteEngine On

RewriteBase /

RewriteRule ^wp-admin/includes/ - [F,L]

RewriteRule !^wp-includes/ - [S=3]

RewriteRule ^wp-includes/[^/]+\.php$ - [F,L]

RewriteRule ^wp-includes/js/tinymce/langs/.+\.php - [F,L]

RewriteRule ^wp-includes/theme-compat/ - [F,L]

</IfModule>
```

# BEGIN WordPress

Note that this won't work well on Multisite, as RewriteRule ^wp-includes/[^/]+\.php$ - [F,L] would prevent the ms-files.php file from generating images. Omitting that line will allow the code to work, but offers less security.

Securing wp-config.php

You can move the wp-config.php file to the directory above your WordPress install. This means for a site installed in the root of your webspace, you can store wp-config.php outside the web-root folder.

Note: Some people assert that moving wp-config.php has minimal security benefits and, if not done carefully, may actually introduce serious vulnerabilities. Others disagree.

Note that wp-config.php can be stored ONE directory level above the WordPress (where wp-includes resides) installation. Also, make sure that only you (and the web server) can read this file (it generally means a 400 or 440 permission).

If you use a server with .htaccess, you can put this in that file (at the very top) to deny access to anyone surfing for it:

```
<files wp-config.php>

order allow,deny

deny from all

</files>
```

Disable File Editing

The WordPress Dashboard by default allows administrators to edit PHP files, such as plugin and theme files. This is often the first tool an attacker will use if able to login, since it allows code execution. WordPress has a constant to disable editing from Dashboard. Placing this line in wp-config.php is equivalent to removing the 'edit_themes', 'edit_plugins' and 'edit_files' capabilities of all users:

```
define('DISALLOW_FILE_EDIT', true);
```

This will not prevent an attacker from uploading malicious files to your site, but might stop some attacks.

Plugins

First of all, make sure your plugins are always updated. Also, if you are not using a specific plugin, delete it from the system.

Firewall

There are many plugins and services that can act as a firewall for your website. Some of them work by modifying your .htaccess file and restricting some access at the Apache level, before it is processed by WordPress. A good example is iThemes Security or All in One WP Security. Some firewall plugins act at the WordPress level, like WordFence and try to filter attacks as WordPress is loading, but before it is fully processed.

Besides plugins, you can also install a WAF (web firewall) at your web server to filter content before it is processed by WordPress. The most popular open source WAF is ModSecurity.

A website firewall can also be added as intermediary between the traffic from the internet and your hosting server. These services all function as reverse proxies, in which they accept the initial requests and reroute them to your server, stripping it of all malicious requests. They accomplish this by modifying your DNS records, via an A record or full DNS swap, allowing all traffic to pass through the new network first. This causes all traffic to be filtered by the firewall before reaching your site. A few companies offer such service, like CloudFlare, Sucuri and Incapsula.

Additionally, these third parties service providers function as Content Distribution Network (CDNs) by default, introducing performance optimization and global reach.

Plugins that need write access

If a plugin wants write access to your WordPress files and directories, please read the code to make sure it is legit or check with someone you trust. Possible places to check are the Support Forums and IRC Channel.

Code execution plugins

As we said, part of the goal of hardening WordPress is containing the damage done if there is a successful attack. Plugins which allow arbitrary PHP or other code to execute from entries in a database effectively magnify the possibility of damage in the event of a successful attack.

A way to avoid using such a plugin is to use custom page templates that call the function. Part of the security this affords is active only when you disallow file editing within WordPress.

Security through obscurity

Security through obscurity is generally an unsound primary strategy. However, there are areas in WordPress where obscuring information *might* help with security:

1. **Rename the administrative account:** When creating an administrative account, avoid easily guessed terms such as admin or webmaster as usernames because they are typically subject to attacks first. On an existing WordPress install you may rename the existing account in the MySQL command-line client with a command like UPDATE wp_users SET user_login = 'newuser' WHERE user_login = 'admin';, or by using a MySQL frontend like phpMyAdmin.

2. **Change the table_prefix:** Many published WordPress-specific SQL-injection attacks make the assumption that the table_prefix is wp_, the default. Changing this can block at least some SQL injection attacks.

Data Backups

Back up your data regularly, including your MySQL databases. See the main article: Backing Up Your Database.

Data integrity is critical for trusted backups. Encrypting the backup, keeping an independent record of MD5 hashes for each backup file, and/or placing backups on read-only media increases your confidence that your data has not been tampered with.

A sound backup strategy could include keeping a set of regularly-timed snapshots of your entire WordPress installation (including WordPress core files and your database) in a trusted location. Imagine a site that makes weekly snapshots. Such a strategy means that if a site is compromised on May 1st but the compromise is not detected until May 12th, the site owner will have pre-compromise backups that can help in rebuilding the site and possibly even post-compromise backups which will aid in determining how the site was compromised.

Logging

Logs are your best friend when it comes to understanding what is happening with your website, especially if you're trying to perform forensics. Contrary to popular beliefs, logs allow you to see what was done and by who and when. Unfortunately the logs will not tell you who, username, logged in, but it will allow you to identify the IP and time and more importantly, the actions the attacker might have taken. You will be able to see any of these attacks via the logs - Cross Site Scripting (XSS), Remote File Inclusion (RFI), Local File Inclusion (LFI) and Directory Traversal attempts. You will also be able to see brute force attempts. There are various examples and tutorials available to help guide you through the process of parsing and analyzing your raw logs.

If you get more comfortable with your logs you'll be able to see things like, when the theme and plugin editors are being used, when someone updates your widgets and when posts and pages are added. All key elements when doing forensic work on your web server. The are a few WordPress Security plugins that assist you with this as well, like the Sucuri Auditing tool or the Audit Trail plugin.

There are two key open-source solutions you'll want on your web server from a security perspective, this is a layered approach to security.

OSSEC can run on any NIX distribution and will also run on Windows. When configured correctly its very powerful. The idea is correlate and aggregate all the logs. You have to be sure to configure it to capture all access_logs and error_logs and if you have multiple websites on the server account for that. You'll also want to be sure to filter out the noise. By default you'll see a lot of noise and you'll want to configure it to be really effective.

Monitoring

Sometimes prevention is not enough and you may still be hacked. That's why intrusion detection/monitoring is very important. It will allow you to react faster, find out what happened and recover your site.

Monitoring your logs

If you are on a dedicated or virtual private server, in which you have the luxury of root access, you have the ability easily configure things so that you can see what's going on. OSSEC easily facilitates this and here is a little write up that might help you out OSSEC for Website Security - Part I.

Monitoring your files for changes

When an attack happens, it always leave traces. Either on the logs or on the file system (new files, modified files, etc). If you are using OSSEC for example, it will monitor your files and alert you when they change.

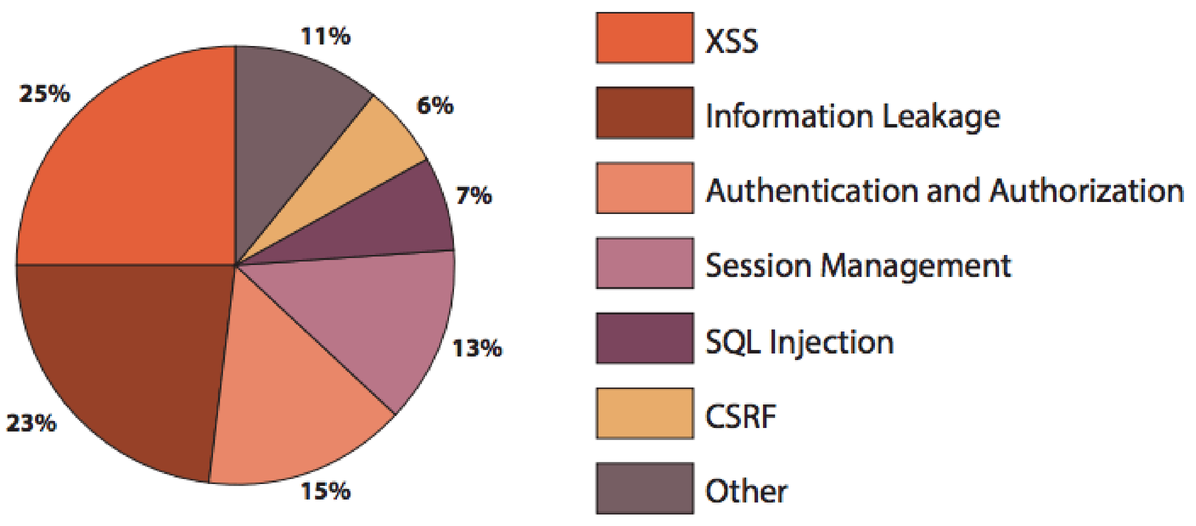
General Apache Security

Secure Apache Web Server – Practical Guide

Introduction

The Web Server is a crucial part of web-based applications. Apache Web Server is often placed at the edge of the network hence it becomes one of the most vulnerable services to attack. Having default configuration supply many sensitive information which may help hacker to prepare for an attack the web server.

The majority of web application attacks are through XSS, Info Leakage, Session Management and PHP Injection attacks which is due to weak programming code and failure to sanitize web application infrastructure. According to the security vendor Cenzic, 96% of tested applications have vulnerabilities. Below chart from Cenzic shows the vulnerability trend report of 2013.



This practical guide provides you the necessary skill set to secure Apache Web Server. In this course, we will talk about how to Harden & Secure Apache Web Server on Unix platform. Following are tested on Apache 2.4.x and I don't see any reason it won't work with Apache 2.2.x.

1. This assumes you have installed Apache on UNIX platform. If not, you can go through Installation guide. You can also refer very free video about how to Install Apache, MySQL & PHP.
2. We will call Apache installation directory /opt/apache as $Web_Server throughout this course.
3. You are advised to take a backup of existing configuration file before any modification.

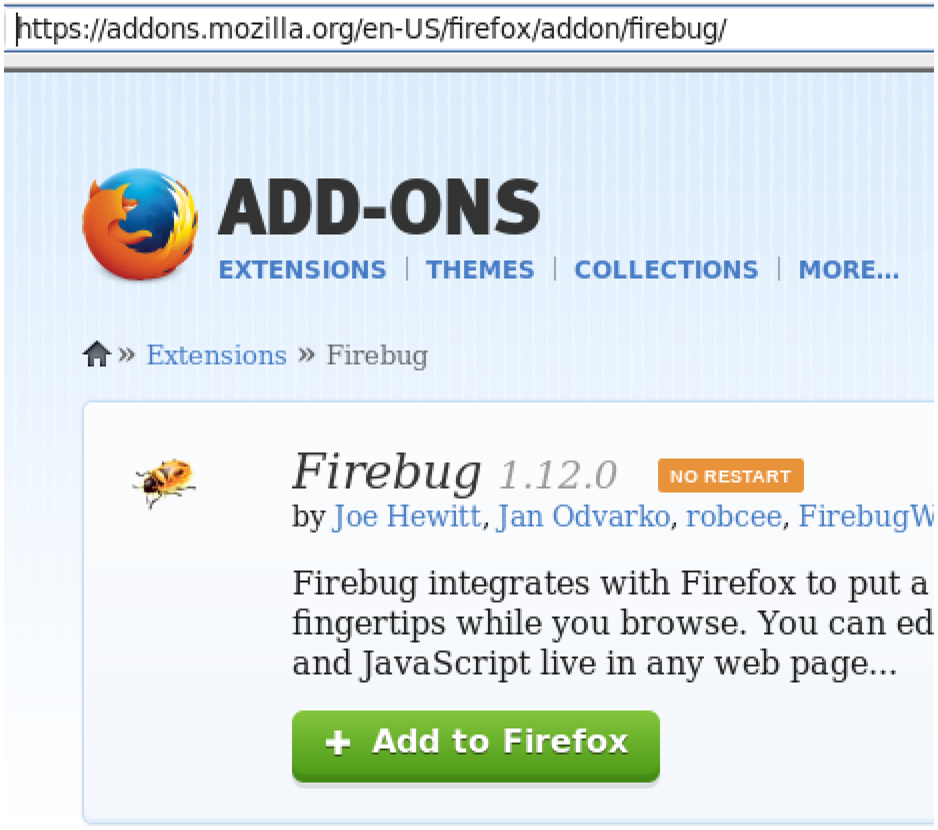
Audience

This is designed for Middleware Administrator, Application Support, System Analyst, or anyone working or eager to learn Hardening & Security guidelines. Fair knowledge of Apache Web Server & UNIX command is mandatory. This is seven page guide, click on Next to proceed. You may navigate through table of contents at right hand side.

Information Leakage

In default Apache configuration you would have many sensitive information disclosure, which can be used to prepare for an attack. It's one of the most critical tasks for administrator to understand and secure them. As per report by Cenzic, 16% of vulnerability is found in Info leakage. We require some tool to examine HTTP Headers for verification. Let's do this by install firebug add-on in Firefox.

- Open Firefox
- Access https://addons.mozilla.org/en-US/firefox/addon/firebug/
- Click on Add to Firefox


https://addons.mozilla.org/en-US/firefox/addon/firebug/

ADD-ONS

EXTENSIONS | THEMES | COLLECTIONS | MORE...

» Extensions » Firebug

Firebug 1.12.0 NO RESTART

by Joe Hewitt, Jan Odvarko, robcee, FirebugW

Firebug integrates with Firefox to put a
fingertips while you browse. You can ed
and JavaScript live in any web page...

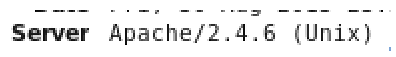
+ Add to Firefox

- Click on Install Now
- Restart Firefox
- You can see firebug icon at right top bar

We will use this icon to open firebug console to view HTTP Headers information. There are many online tools also available which helps to check in HTTP header information. Below are some of them you can try out. http://tools.geekflare.com/seo/tool.php?id=check-headers

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



```
Server  Apache/2.4.6 (Unix)
```

Implementation:

- Go to $Web_Server/conf folder
- Modify httpd.conf by using vi editor

- Add following directive and save the httpd.conf

ServerTokens Prod

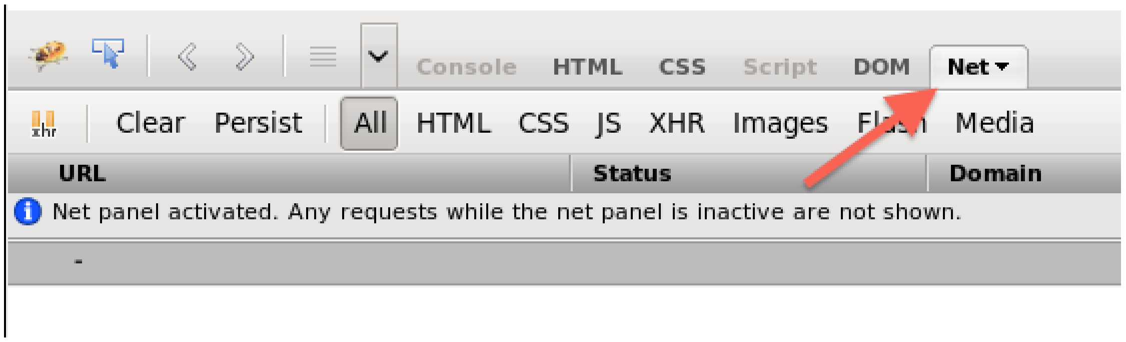
ServerSignature Off

- Restart apache

ServerSignature will remove the version information from the page generated like 403, 404, 502, etc. by apache web server. ServerTokens will change Header to production only, i.e. Apache

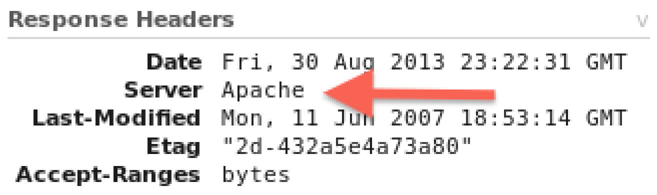
Verification:

- Open Firefox
- Activate firebug by clicking firebug icon at top right side
- Click on Net tab



Net panel activated. Any requests while the net panel is inactive are not shown.

- Hit the URL in address bar
- Expand the GET request and you could see Server directive is just showing Apache, which is much better than exposing version and OS type.



```
Response Headers
         Date  Fri, 30 Aug 2013 23:22:31 GMT
       Server  Apache
Last-Modified  Mon, 11 Jun 2007 18:53:14 GMT
         Etag  "2d-432a5e4a73a80"
Accept-Ranges  bytes
```

Disable directory browser listing

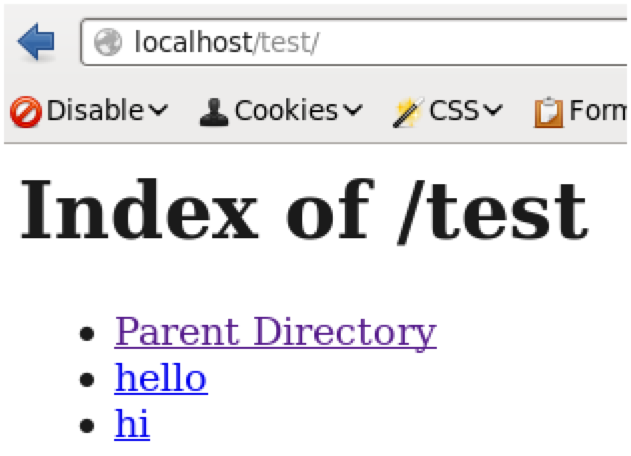
Disable directory listing in browser so visitor doesn't see what all file and folders you have under root or sub-directory. Let's test how does it look like in default settings.

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

```
# mkdir test

# touch hi
```

# touch hello

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



Index of /test

- Parent Directory
- hello
- hi

As you could see it reveals what all file/folders you have which is certainly you don’t want to expose.

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

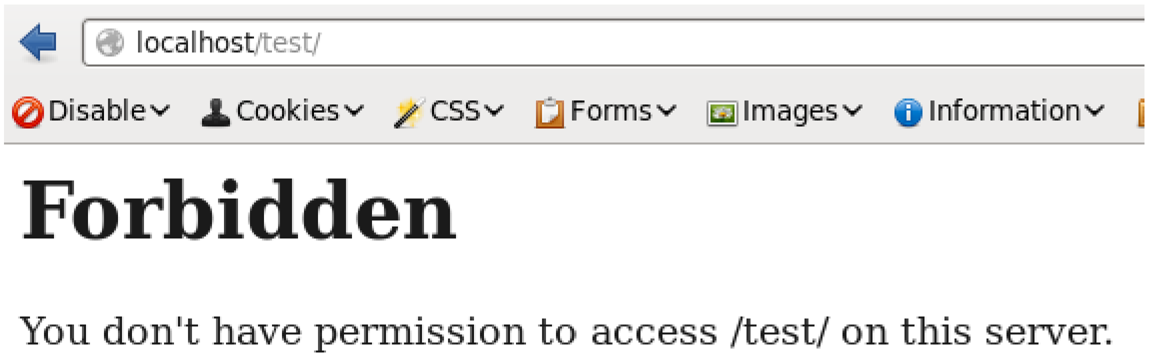
</Directory>

- Restart Apache

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

Verification:

Now, let's try to access Apache by http://localhost/test



Forbidden

You don't have permission to access /test/ on this server.

As you could see, it displays forbidden error instead 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.

Implementation:

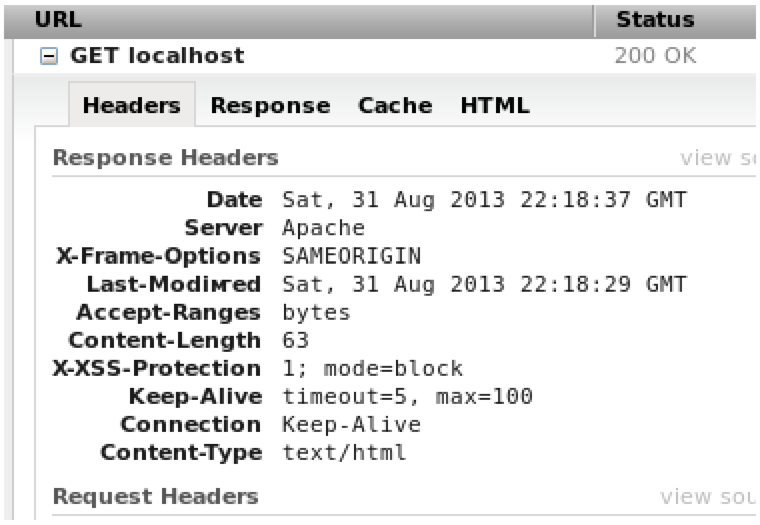
- Go to $Web_Server/conf directory
- Add following directive and save the httpd.conf

```
FileETag None
```

- Restart apache

Verification:

- Open Firefox and access your application
- Check HTTP response headers in firebug, you should not see Etag at all.



```
                 Date  Sat, 31 Aug 2013 22:18:37 GMT
               Server  Apache
      X-Frame-Options  SAMEORIGIN
        Last-Modified  Sat, 31 Aug 2013 22:18:29 GMT
        Accept-Ranges  bytes
       Content-Length  63
     X-XSS-Protection  1; mode=block
           Keep-Alive  timeout=5, max=100
           Connection  Keep-Alive
         Content-Type  text/html
```

Authorization

Run Apache from non-privileged account

Default apache configuration is to run as nobody or daemon. It’s good to use a separate non-privileged user for Apache. The idea here is to protect other services running in case of any security hole.

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

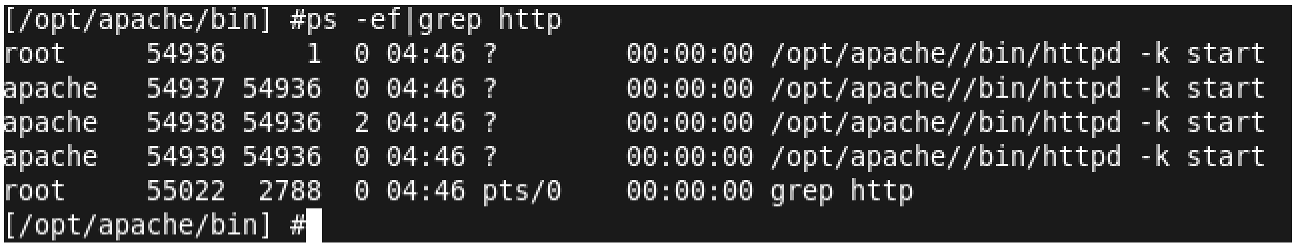
Group apache

- Save the httpd.conf
- Restart Apache

Verification:

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

# **ps –ef |grep http**



```
[/opt/apache/bin] #ps -ef|grep http
root      54936      1  0 04:46 ?        00:00:00 /opt/apache//bin/httpd -k start
apache    54937 54936  0 04:46 ?        00:00:00 /opt/apache//bin/httpd -k start
apache    54938 54936  2 04:46 ?        00:00:00 /opt/apache//bin/httpd -k start
apache    54939 54936  0 04:46 ?        00:00:00 /opt/apache//bin/httpd -k start
root      55022  2788  0 04:46 pts/0    00:00:00 grep http
[/opt/apache/bin] #
```

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.

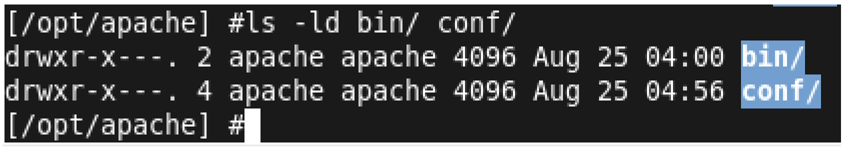
Protect binary and configuration directory permission

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

Implementation:

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

# chown –R 750 bin conf

Verification: 

```
[/opt/apache] #ls -ld bin/ conf/
drwxr-x---. 2 apache apache 4096 Aug 25 04:00 bin/
drwxr-x---. 4 apache apache 4096 Aug 25 04:56 conf/
[/opt/apache] #
```

System Settings Protection

In default installation, users can override apache configuration using .htaccess. if you want to stop users changing your apache server settings, you can add AllowOverride to None as shown below. This must be done at root level.

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

HTTP 1.1 protocol support many request methods which may not be required and some of them are having potential risk. Typically you may just need GET, HEAD, POST request methods in web application, which can be configured in respective Directory directive. Default apache configuration support OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT method in HTTP 1.1 protocol.

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>
```

Web Application Security

Apache web server misconfiguration or not hardened properly can exploit web application. It's critical to harden your web server configuration.

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 hacker to steal cookie information. Let's see how it looks like in default configuration.

- Do a telnet web server IP with listen 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.

#
```

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

Implementation:

- Go to $Web_Server/conf directory
- Add 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.

#
```

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.

Set cookie with HttpOnly and Secure flag

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

Implementation:

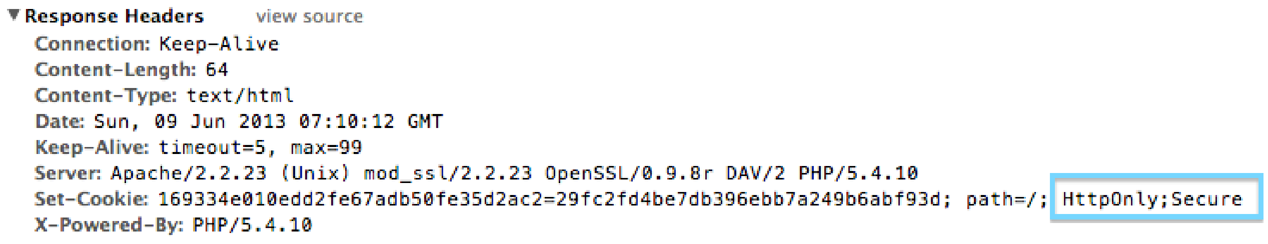
- Ensure mod_headers.so is enabled in your httpd.conf
- Go to $Web_Server/conf directory
- Add following directive and save the httpd.conf

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

- Restart apache

Verification:

- Open Firefox and access your application
- Check HTTP response headers in firebug, you should see Set-Cookie is flagged with HttpOnly and Secure as shown below.



```
▼Response Headers        view source
   Connection: Keep-Alive
   Content-Length: 64
   Content-Type: text/html
   Date: Sun, 09 Jun 2013 07:10:12 GMT
   Keep-Alive: timeout=5, max=99
   Server: Apache/2.2.23 (Unix) mod_ssl/2.2.23 OpenSSL/0.9.8r DAV/2 PHP/5.4.10
   Set-Cookie: 169334e010edd2fe67adb50fe35d2ac2=29fc2fd4be7db396ebb7a249b6abf93d; path=/; HttpOnly;Secure
   X-Powered-By: PHP/5.4.10
```

Clickjacking Attack

Clickjacking is well known web application vulnerabilities. You can refer my previous post Secure Your Web Site from Clickjacking Attack.

Implementation:

- Ensure mod_headers.so is enabled in your httpd.conf
- Go to $Web_Server/conf directory
- Add following directive and save the httpd.conf

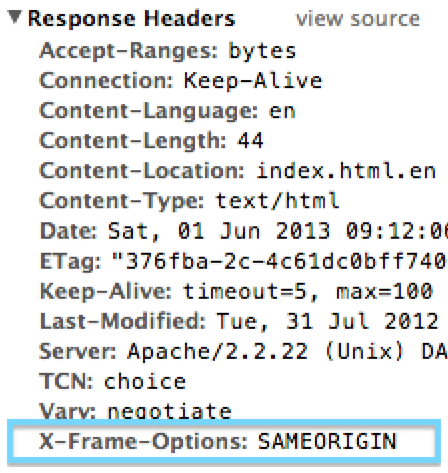
Header always append X-Frame-Options SAMEORIGIN

- Restart apache

Verification:

- Open Firefox and access your application

- Check HTTP response headers in firebug, you should see X-Frame-Options as shown below.



```
▼Response Headers    view source
  Accept-Ranges: bytes
  Connection: Keep-Alive
  Content-Language: en
  Content-Length: 44
  Content-Location: index.html.en
  Content-Type: text/html
  Date: Sat, 01 Jun 2013 09:12:06 GMT
  ETag: "376fba-2c-4c61dc0bff740"
  Keep-Alive: timeout=5, max=100
  Last-Modified: Tue, 31 Jul 2012 10:36:37 GMT
  Server: Apache/2.2.22 (Unix) DAV/2 mod_ssl/2.2.22 OpenSSL/0.9.8r
  TCN: choice
  Vary: negotiate
  X-Frame-Options: SAMEORIGIN
```

Server Side Include

Server Side Include (SSI) has a risk in increasing load on the server. If you have shared environment and heavy traffic web applications you should consider disable 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.

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

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

X-XSS Protection

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

Implementation:

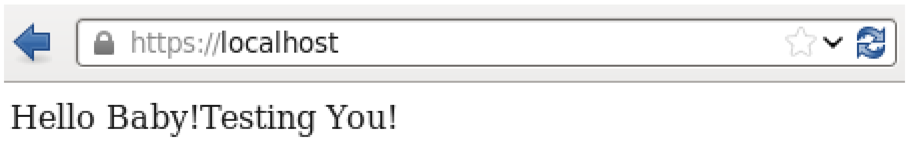
- Go to $Web_Server/conf directory
- Open httpd.conf using vi and add following Header directive

Header set X-XSS-Protection “1; mode=block”

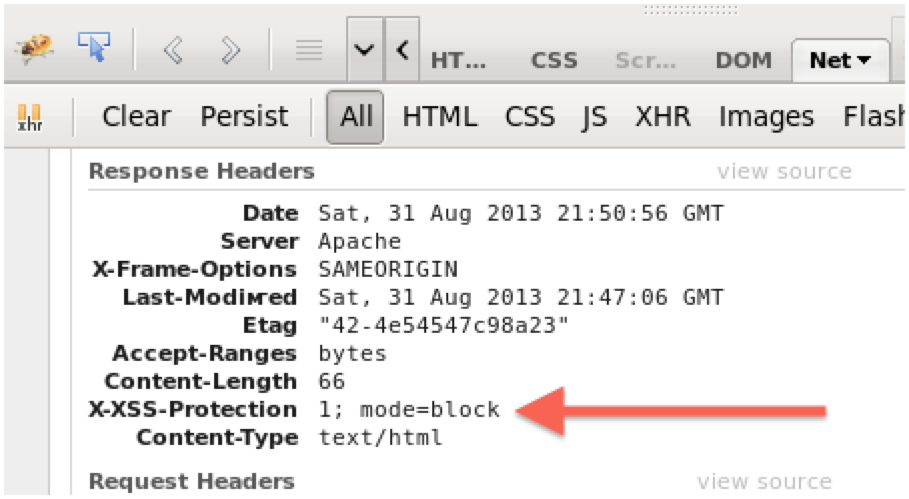
- Restart Apache

Verification:

- Open Firefox and access your application
- Check HTTP response headers in firebug, you should see XSS Protection is enabled and mode is blocked.



Hello Baby!Testing You!



```
          Date  Sat, 31 Aug 2013 21:50:56 GMT
        Server  Apache
X-Frame-Options  SAMEORIGIN
  Last-Modified  Sat, 31 Aug 2013 21:47:06 GMT
           Etag  "42-4e54547c98a23"
  Accept-Ranges  bytes
 Content-Length  66
X-XSS-Protection  1; mode=block
   Content-Type  text/html
```

Disable HTTP 1.0 Protocol

When we talk about security, we should protect as much we can. So why do we use older HTTP version of protocol, let’s disable them as well. HTTP 1.0 has security weakness related to session hijacking. We can disable this by using mod_rewrite module.

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

By default Apache timed-out value is 300 seconds, which can be victim of Slow Loris attack and DoS. To mitigate this you can lower the timeout value to maybe 60 seconds.

Implementation:

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

Timeout 60

SSL

Having SSL is additional layer of security you are adding into Web Application. However, default SSL configuration leads to certain vulnerabilities and you should consider tweaking those configurations. We require some tool to verify SSL settings. There are many available however, I would use SSL-Scan free tool. You can download from http://sourceforge.net/projects/sslscan/

SSL Key

Breaching SSL key is hard, but not impossible. Its just matter of computational power and time. As you might know using a 2009-era PC cracking away for around 73 days you can reverse engineer a 512-bit key. So the higher key length you have, the more complex it becomes to break SSL key. Majority of giant Web Companies use 2048 bit key, as below so why don't we?

- Outlook.com
- Microsoft.com
- Live.com
- Skype.com
- Apple.com
- Yahoo.com
- Bing.com
- Hotmail.com
- Twitter.com

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

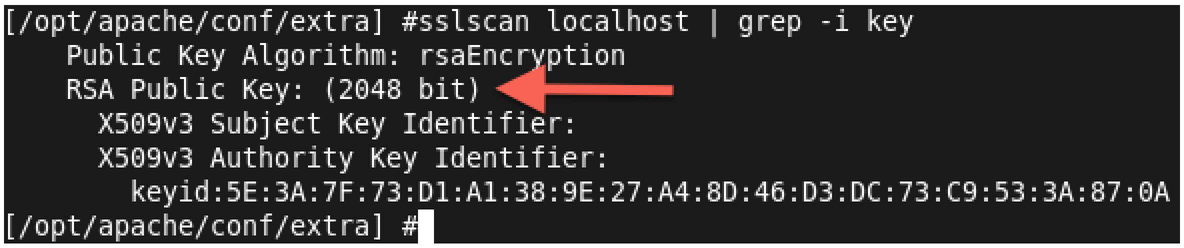
SSLCertificateKeyFile # Key File

SSLCACertificateFile # Signer Cert file

Verification:

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

sslscan localhost | grep –i key



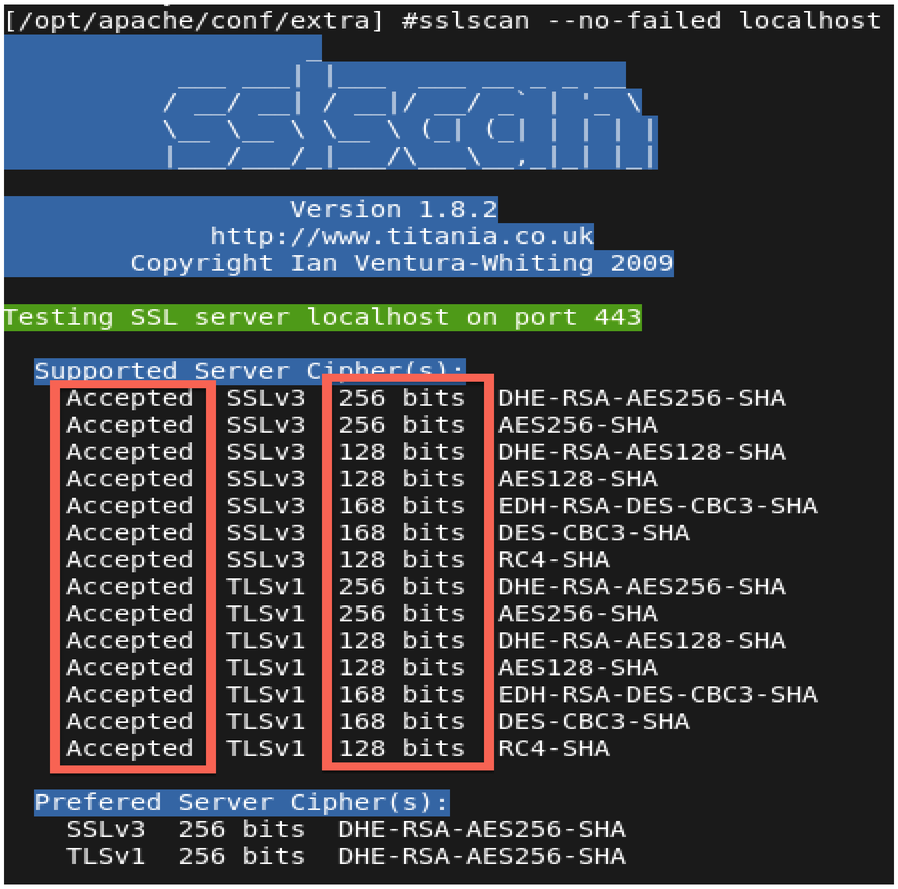
```
[/opt/apache/conf/extra] #sslscan localhost | grep -i key
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (2048 bit)
      X509v3 Subject Key Identifier:
      X509v3 Authority Key Identifier:
        keyid:5E:3A:7F:73:D1:A1:38:9E:27:A4:8D:46:D3:DC:73:C9:53:3A:87:0A
[/opt/apache/conf/extra] #
```

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

SSL Cipher

SSL Cipher is an encryption algorithm, which is used as a key between two computers over the Internet. Data encryption is the process of converting plain text into secret ciphered codes. It’s based on your web server SSL Cipher configuration the data encryption will take place. So it’s important to configure SSL Cipher, which is stronger and not vulnerable. Let’s validate the Cipher accepted in current SSL configuration. We will use sslscan utility to validate as below command. Change localhost to your actual domain name.

sslscan –no-failed localhost



```
[/opt/apache/conf/extra] #sslscan --no-failed localhost
                   _
           ___ ___| |___  ___ __ _ _ __
          / __/ __| / __|/ __/ _` | '_ \
          \__ \__ \ \__ \ (_| (_| | | | |
          |___/___/_|___/\___\__,_|_| |_|

                  Version 1.8.2
             http://www.titania.co.uk
        Copyright Ian Ventura-Whiting 2009

Testing SSL server localhost on port 443

  Supported Server Cipher(s):
    Accepted  SSLv3  256 bits  DHE-RSA-AES256-SHA
    Accepted  SSLv3  256 bits  AES256-SHA
    Accepted  SSLv3  128 bits  DHE-RSA-AES128-SHA
    Accepted  SSLv3  128 bits  AES128-SHA
    Accepted  SSLv3  168 bits  EDH-RSA-DES-CBC3-SHA
    Accepted  SSLv3  168 bits  DES-CBC3-SHA
    Accepted  SSLv3  128 bits  RC4-SHA
    Accepted  TLSv1  256 bits  DHE-RSA-AES256-SHA
    Accepted  TLSv1  256 bits  AES256-SHA
    Accepted  TLSv1  128 bits  DHE-RSA-AES128-SHA
    Accepted  TLSv1  128 bits  AES128-SHA
    Accepted  TLSv1  168 bits  EDH-RSA-DES-CBC3-SHA
    Accepted  TLSv1  168 bits  DES-CBC3-SHA
    Accepted  TLSv1  128 bits  RC4-SHA

  Prefered Server Cipher(s):
    SSLv3  256 bits  DHE-RSA-AES256-SHA
    TLSv1  256 bits  DHE-RSA-AES256-SHA
```

As you could see above, in current configuration DHE, AES, EDH, RC4 cipher is accepted. Now if you are performing penetration test or PCI compliance test, your report will say RC4 Cipher detected. Lately, it was found that RC4 is weak cipher and to pass certain security test, you must not accept RC4 or any weak cipher. You should also ensure not to accept any cipher, which is less than 128 bits.

Implementation:

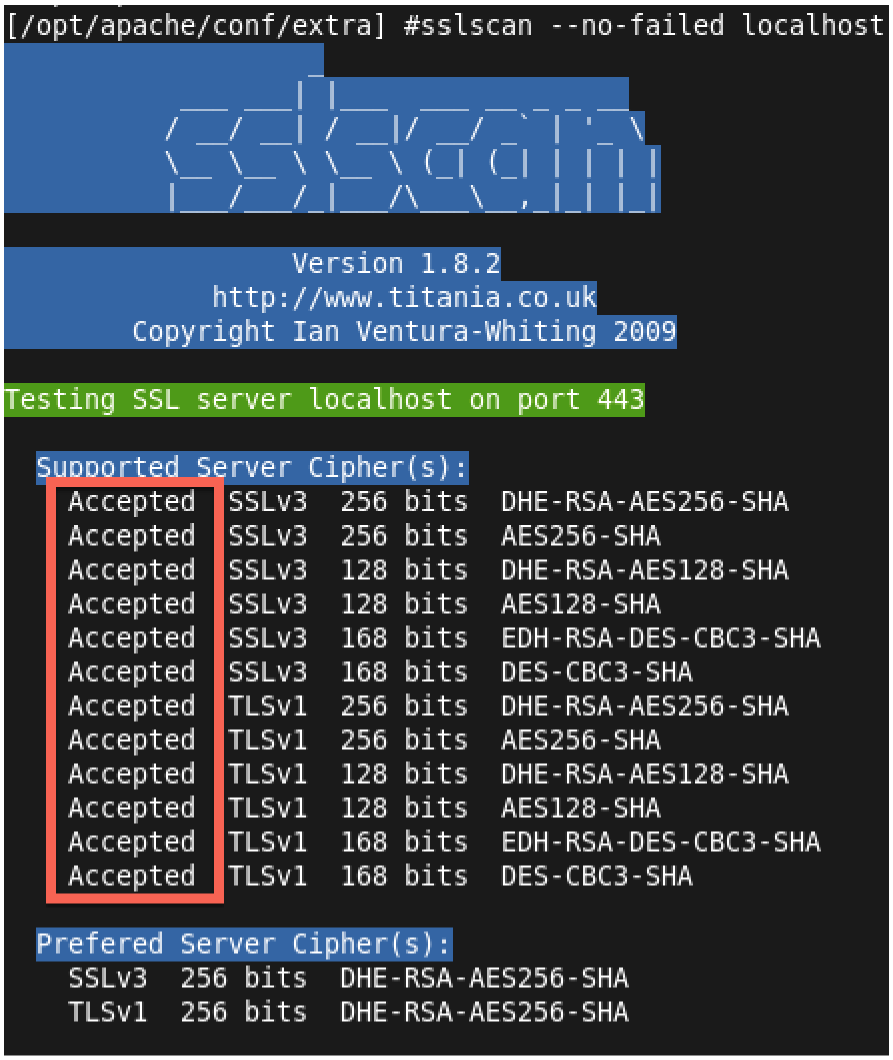
- Go to $Web_Server/conf/extra folder
- Modify SSLCipherSuite directive in httpd-ssl.conf as below to reject RC4

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

- Save the configuration file and restart apache server

Note: if you have many weak cipher 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



```
[/opt/apache/conf/extra] #sslscan --no-failed localhost
                   _
           ___ ___| |___  ___ __ _ _ __
          / __/ __| / __|/ __/ _` | '_ \
          \__ \__ \ \__ \ (_| (_| | | | |
          |___/___/_|___/\___\__,_|_| |_|

                  Version 1.8.2
             http://www.titania.co.uk
        Copyright Ian Ventura-Whiting 2009

Testing SSL server localhost on port 443

  Supported Server Cipher(s):
    Accepted  SSLv3  256 bits  DHE-RSA-AES256-SHA
    Accepted  SSLv3  256 bits  AES256-SHA
    Accepted  SSLv3  128 bits  DHE-RSA-AES128-SHA
    Accepted  SSLv3  128 bits  AES128-SHA
    Accepted  SSLv3  168 bits  EDH-RSA-DES-CBC3-SHA
    Accepted  SSLv3  168 bits  DES-CBC3-SHA
    Accepted  TLSv1  256 bits  DHE-RSA-AES256-SHA
    Accepted  TLSv1  256 bits  AES256-SHA
    Accepted  TLSv1  128 bits  DHE-RSA-AES128-SHA
    Accepted  TLSv1  128 bits  AES128-SHA
    Accepted  TLSv1  168 bits  EDH-RSA-DES-CBC3-SHA
    Accepted  TLSv1  168 bits  DES-CBC3-SHA

  Prefered Server Cipher(s):
    SSLv3  256 bits  DHE-RSA-AES256-SHA
    TLSv1  256 bits  DHE-RSA-AES256-SHA
```

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

SSL v2 has many security flaws and if you are working towards penetration test or PCI compliance then you are expected to close security finding to disable SSL v2. Any SSL v2 communication may be vulnerable to Man-in-The-Middle attack that could allow data tempering or disclosure. Let's implement apache web server to accept only latest SSL v3 and reject SSL v2 connection request.

Implementation:

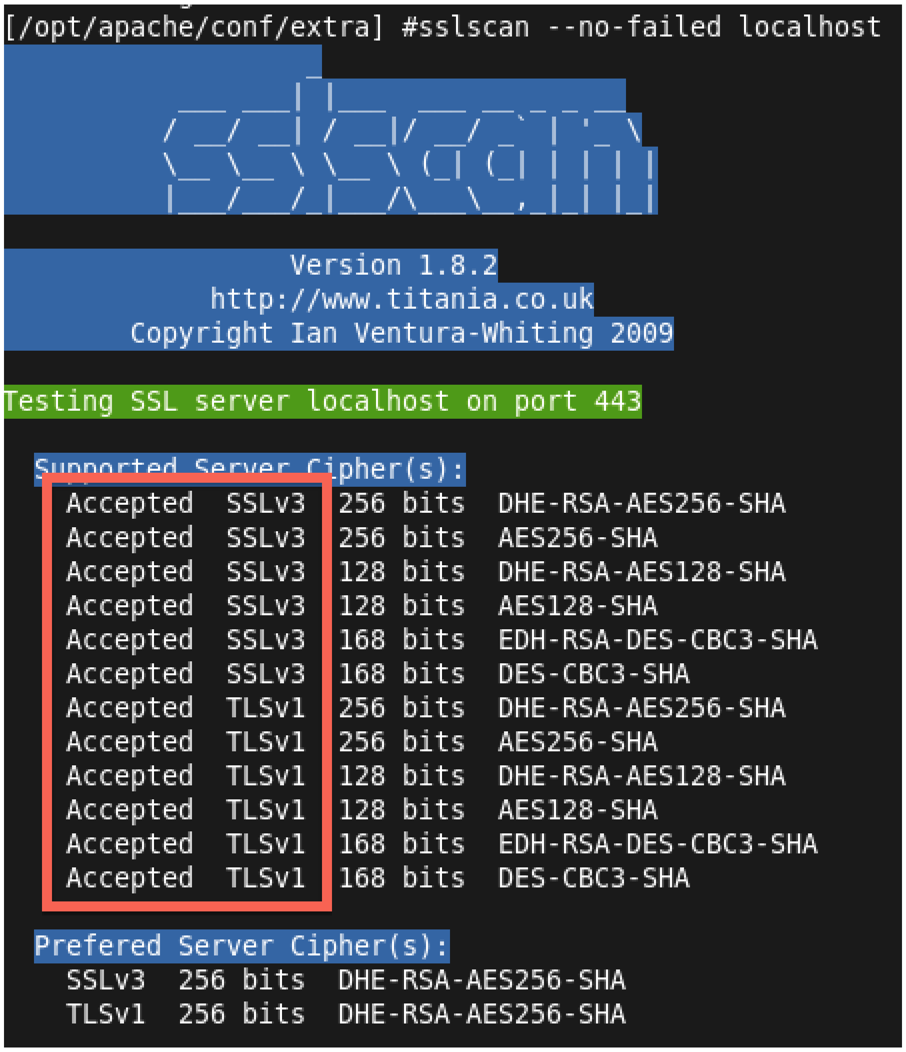
- Go to $Web_Server/conf/extra folder
- Modify SSLProtocol directive in httpd-ssl.conf as below to accept only SSL v3 and TLS v1

SSLProtocol –ALL +SSLv3 +TLSv1

Verification:

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

sslscan –no-failed localhost



```
[/opt/apache/conf/extra] #sslscan --no-failed localhost
                   _
           ___ ___| |___  ___ __ _ _ __
          / __/ __| / __|/ __/ _` | '_ \
          \__ \__ \ \__ \ (_| (_| | | | |
          |___/___/_|___/\___\__,_|_| |_|

                  Version 1.8.2
             http://www.titania.co.uk
        Copyright Ian Ventura-Whiting 2009

Testing SSL server localhost on port 443

  Supported Server Cipher(s):
    Accepted  SSLv3  256 bits  DHE-RSA-AES256-SHA
    Accepted  SSLv3  256 bits  AES256-SHA
    Accepted  SSLv3  128 bits  DHE-RSA-AES128-SHA
    Accepted  SSLv3  128 bits  AES128-SHA
    Accepted  SSLv3  168 bits  EDH-RSA-DES-CBC3-SHA
    Accepted  SSLv3  168 bits  DES-CBC3-SHA
    Accepted  TLSv1  256 bits  DHE-RSA-AES256-SHA
    Accepted  TLSv1  256 bits  AES256-SHA
    Accepted  TLSv1  128 bits  DHE-RSA-AES128-SHA
    Accepted  TLSv1  128 bits  AES128-SHA
    Accepted  TLSv1  168 bits  EDH-RSA-DES-CBC3-SHA
    Accepted  TLSv1  168 bits  DES-CBC3-SHA

  Prefered Server Cipher(s):
    SSLv3  256 bits  DHE-RSA-AES256-SHA
    TLSv1  256 bits  DHE-RSA-AES256-SHA
```

As you could see above, accepted is only SSLv3 and TLSv1, which is safe from SSLv2 vulnerabilities.

Mod Security

Mod Security is an open-source Web Application Firewall, which you can use with Apache. It comes as a module which you have to compile and install. If you can’t afford commercial web application firewall, this would be good choice to go for it. Mod Security says: In order to provide generic web applications protection, the Core Rules use the following techniques:

- **HTTP Protection** – detecting violations of the HTTP protocol and a locally defined usage policy
- **Real-time Blacklist Lookups** – utilizes 3rd Party IP Reputation
- **Web-based Malware Detection** – identifies malicious web content by check against the Google Safe Browsing API.

- **HTTP Denial of Service Protections** – defense against HTTP Flooding and Slow HTTP DoS Attacks.
- **Common Web Attacks Protection** – detecting common web application security attack
- **Automation Detection** – Detecting bots, crawlers, scanners and other surface malicious activity
- **Integration with AV Scanning for File Uploads** – detects malicious files uploaded through the web application.
- **Tracking Sensitive Data** – Tracks Credit Card usage and blocks leakages.
- **Trojan Protection** – Detecting access to Trojans horses.
- **Identification of Application Defects** – alerts on application misconfigurations.
- **Error Detection and Hiding** – Disguising error messages sent by the server.

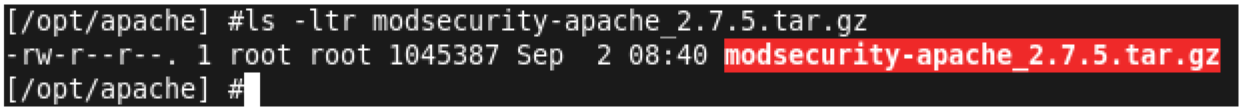
Download & Installation

Following prerequisites must be installed on server where you wish to use Mod Security with Apache. If any one of these doesn't exist then Mod Security compilation will fail. You may use yum install on Linux or Centos to install these packages.

- apache 2.x or higher
- libpcre package
- libxml2 package
- liblua package
- libcurl package
- libapr and libapr-util package
- mod_unique_id module bundled with Apache web server

Now, let's download the latest stable version of Mod Security 2.7.5 from http://www.modsecurity.org/download/

- Transfer downloaded file to /opt/apache



```
[/opt/apache] #ls -ltr modsecurity-apache_2.7.5.tar.gz
-rw-r--r--. 1 root root 1045387 Sep  2 08:40 modsecurity-apache_2.7.5.tar.gz
[/opt/apache] #
```

- 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

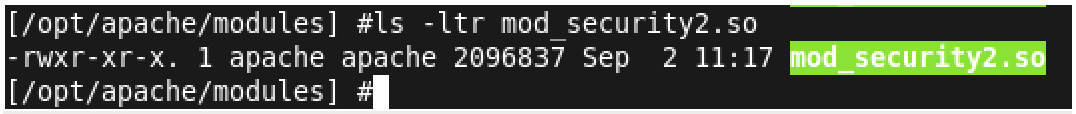
# **./configure –with-apxs=/opt/apache/bin/apxs**

- Compile & install with make script

# **make**

#make install

- Once installation is done, you would see **mod_security2.so** in modules folder under /opt/apache as shown below



```
[/opt/apache/modules] #ls -ltr mod_security2.so
-rwxr-xr-x. 1 apache apache 2096837 Sep  2 11:17 mod_security2.so
[/opt/apache/modules] #
```

Now this concludes, you have installed Mod Security module in existing Apache web server.

Configuration

In order to use Mod security feature with Apache, we have to load mod security module in httpd.conf. **mod_unique_id** module is pre-requisite for Mod Security. This module provides an environment variable with a unique identifier for each request, which is tracked and used by Mod Security.

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

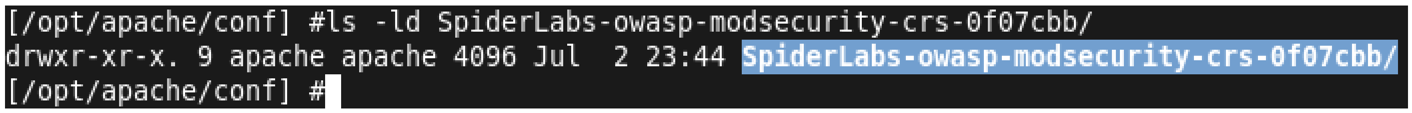
LoadModule unique_id_module modules/mod_unique_id.so

LoadModule security2_module modules/mod_security2.so

- Restart apache web server

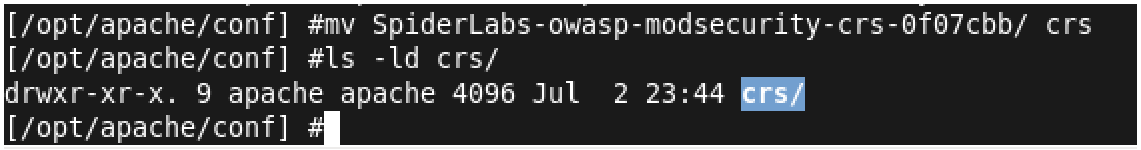
Mod Security is now installed! Next thing you have to do is to install Mod Security core rule to take a full advantage of its feature. Latest Core Rule can be downloaded from following link, which is free. https://github.com/SpiderLabs/owasp-modsecurity-crs/zipball/master

- Copy downloaded core rule zip to /opt/apache/conf folder
- Unzip core rule file, you should see the extracted folder as shown below



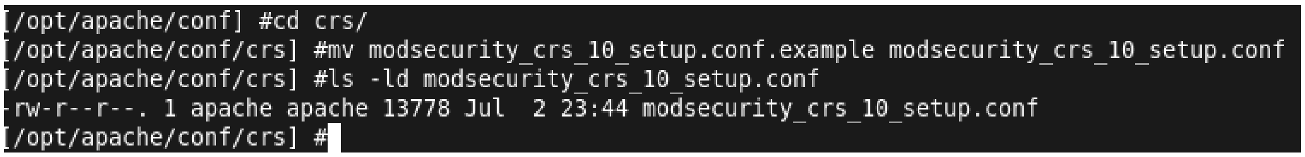
```
[/opt/apache/conf] #ls -ld SpiderLabs-owasp-modsecurity-crs-0f07cbb/
drwxr-xr-x. 9 apache apache 4096 Jul  2 23:44 SpiderLabs-owasp-modsecurity-crs-0f07cbb/
[/opt/apache/conf] #
```

- You may wish to rename the folder to something short and easy to remember. In this example, I will rename to crs.



```
[/opt/apache/conf] #mv SpiderLabs-owasp-modsecurity-crs-0f07cbb/ crs
[/opt/apache/conf] #ls -ld crs/
drwxr-xr-x. 9 apache apache 4096 Jul  2 23:44 crs/
[/opt/apache/conf] #
```

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



```
[/opt/apache/conf] #cd crs/
[/opt/apache/conf/crs] #mv modsecurity_crs_10_setup.conf.example modsecurity_crs_10_setup.conf
[/opt/apache/conf/crs] #ls -ld modsecurity_crs_10_setup.conf
-rw-r--r--. 1 apache apache 13778 Jul  2 23:44 modsecurity_crs_10_setup.conf
[/opt/apache/conf/crs] #
```

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

You have successfully configured Mod Security with Apache! **Well done**. Now, Apache Web server is protected with Mod Security web application firewall.

Getting Started

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 are the OWASP rules are provided in free. 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 experimental purpose and you may have high false alarm. It’s important to configure, test and implement in UAT before using these in 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 base, optional and experimental rules after extracting 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 request, which is missing with Host, Accept, User-Agent in header.
- **modsecurity_crs_23_request_limits.conf:**This rule has dependency on application specific like request size, upload size, length of 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 flexibility to configure Audit, Debug or both logging. By default configuration will write both logs. However, you can change based on your requirement. Log is controlled in SecDefaultAction directive. Let’s look at default logging configuration in setup.conf

SecDefaultAction “phase:1,deny,log”

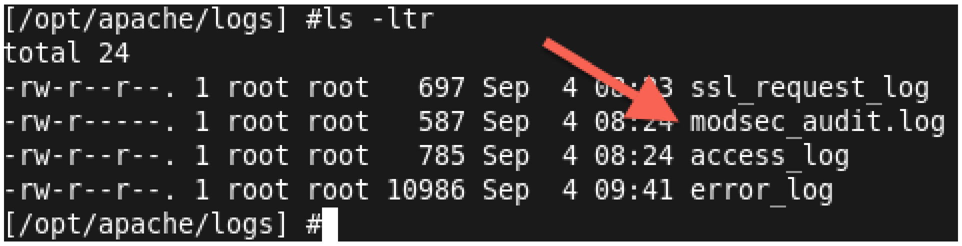
To log Debug, Audit log – use “log” To log only audit log – use “nolog,auditlog” To log only debug log – use “log,noauditlog” You can specify the Audit Log location to be stored which is controlled by SecAuditLog directive. Let’s write audit log into /opt/apache/logs/modsec_audit.log by adding as shown below.

Implementation:

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

SecAuditLog /opt/apache/logs/modsec_audit.log

- After restart, you should see modsec_audit.log getting generated as shown below.



```
[/opt/apache/logs] #ls -ltr
total 24
-rw-r--r--. 1 root root   697 Sep  4 08:23 ssl_request_log
-rw-r-----. 1 root root   587 Sep  4 08:24 modsec_audit.log
-rw-r--r--. 1 root root   785 Sep  4 08:24 access_log
-rw-r--r--. 1 root root 10986 Sep  4 09:41 error_log
[/opt/apache/logs] #
```

Enable Rule Engine

By default Engine Rule is Off that means if you don't enable Rule Engine you are not utilizing all the advantages of Mod Security. Rule Engine enabling or disabling is controlled by SecRuleEngine directive.

Implementation:

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

SecRuleEngine On

There are three values for SecRuleEngine:

- **On** – to enable Rule Engine
- **Off** – to disable Rule Engine
- **DetectionOnly** – enable Rule Engine but never executes any actions like block, deny, drop, allow, proxy or redirect

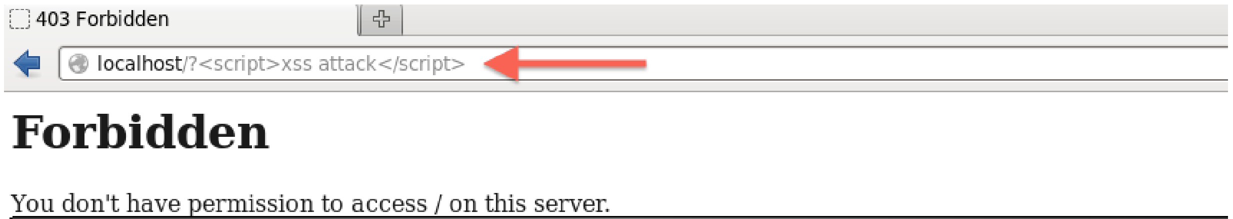
Once Rule Engine is on – Mod Security is ready to protect with some of the common attack types.

Common Attack Type Protection

Now web server is ready to protect with common attack types like XSS, SQL Injection, Protocol Violation, etc. as we have installed Core Rule and turned on Rule Engine. Let's test few of them.

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

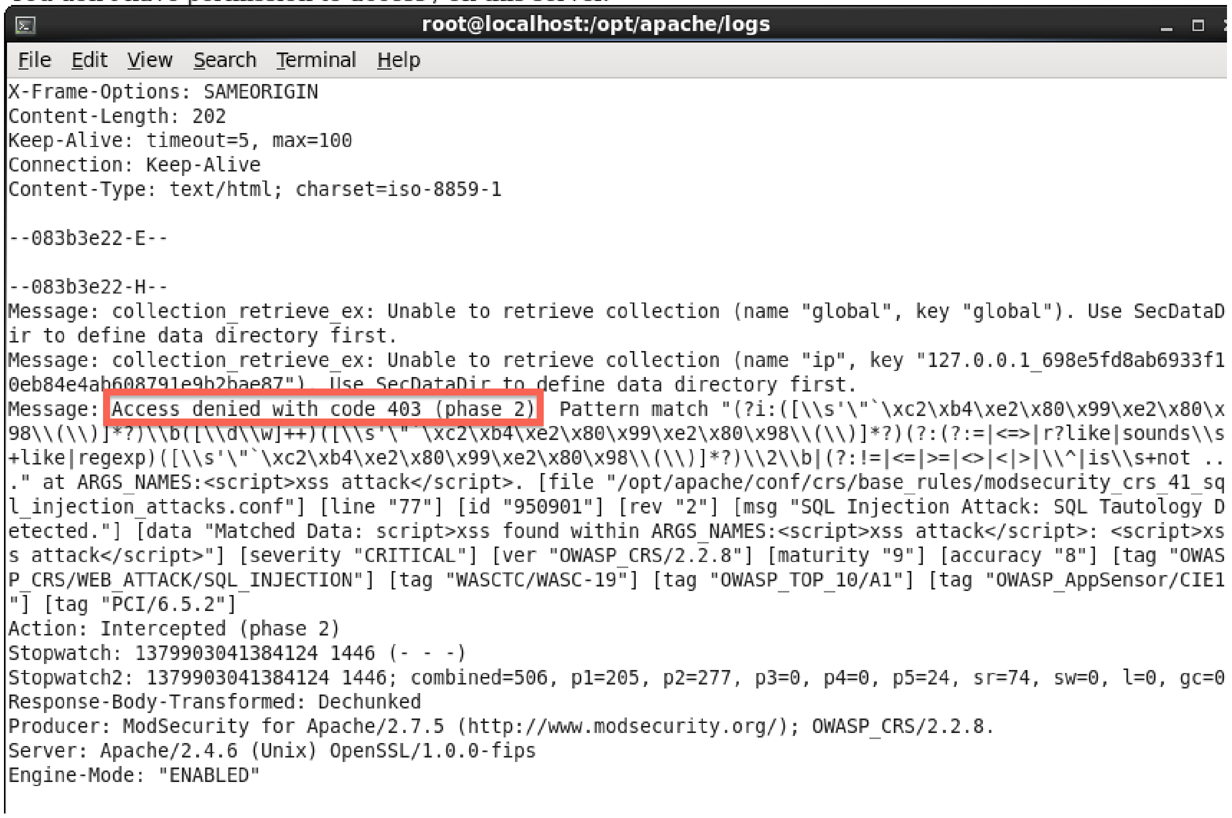


403 Forbidden

localhost/?<script>xss attack</script>

Forbidden

You don't have permission to access / on this server.



root@localhost:/opt/apache/logs
File Edit View Search Terminal Help

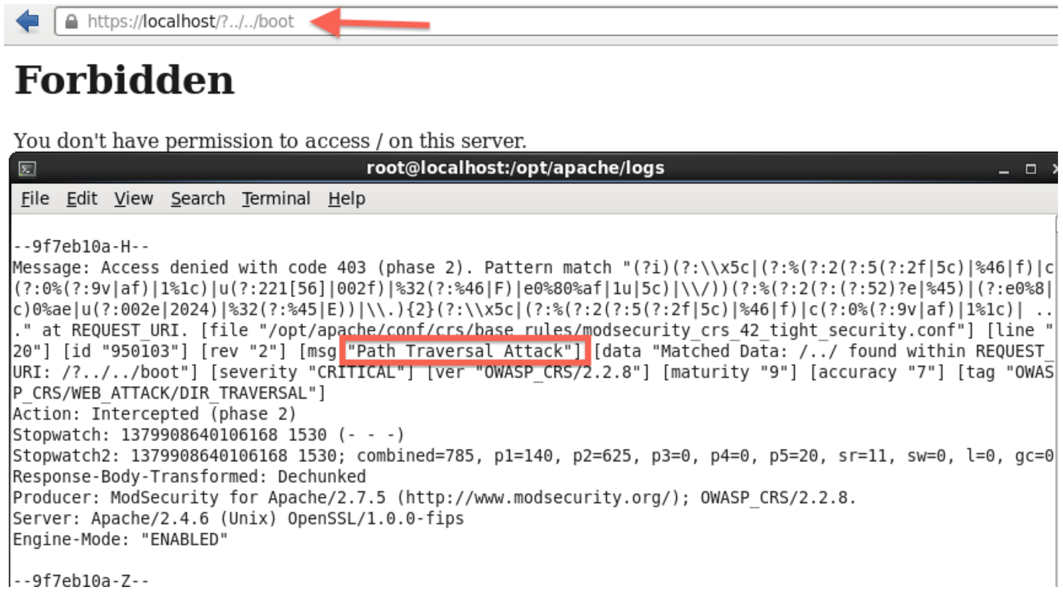
```
X-Frame-Options: SAMEORIGIN
Content-Length: 202
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

--083b3e22-E--

--083b3e22-H--
Message: collection_retrieve_ex: Unable to retrieve collection (name "global", key "global"). Use SecDataD
ir to define data directory first.
Message: collection_retrieve_ex: Unable to retrieve collection (name "ip", key "127.0.0.1_698e5fd8ab6933f1
0eb84e4ab608791e9b2bae87"). Use SecDataDir to define data directory first.
Message: Access denied with code 403 (phase 2). Pattern match "(?i:([\\s'\"`\xc2\xb4\xe2\x80\x99\xe2\x80\x
98\\(\\)]*?)\\b([\\d\\w]++)([\\s'\"`\xc2\xb4\xe2\x80\x99\xe2\x80\x98\\(\\)]*?)(?:(?:=|<=>|r?like|sounds\\s
+like|regexp)([\\s'\"`\xc2\xb4\xe2\x80\x99\xe2\x80\x98\\(\\)]*?)\\2\\b|(?:!=|<=|>=|<>|<|>|\\^|is\\s+not ..
." at ARGS_NAMES:<script>xss attack</script>. [file "/opt/apache/conf/crs/base_rules/modsecurity_crs_41_sq
l_injection_attacks.conf"] [line "77"] [id "950901"] [rev "2"] [msg "SQL Injection Attack: SQL Tautology D
etected."] [data "Matched Data: script>xss found within ARGS_NAMES:<script>xss attack</script>: <script>xs
s attack</script>"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.8"] [maturity "9"] [accuracy "8"] [tag "OWAS
P_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1
"] [tag "PCI/6.5.2"]
Action: Intercepted (phase 2)
Stopwatch: 1379903041384124 1446 (- - -)
Stopwatch2: 1379903041384124 1446; combined=506, p1=205, p2=277, p3=0, p4=0, p5=24, sr=74, sw=0, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.7.5 (http://www.modsecurity.org/); OWASP_CRS/2.2.8.
Server: Apache/2.4.6 (Unix) OpenSSL/1.0.0-fips
Engine-Mode: "ENABLED"
```

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

Directory Traversal Attack:- Directory traversal attacks can create lot of damage by taking advantage of this vulnerabilities and access system related file. Ex – /etc/passwd, .htaccess, etc.

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

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


https://localhost/?../../boot

Forbidden

You don't have permission to access / on this server.

root@localhost:/opt/apache/logs

File Edit View Search Terminal Help

```
--9f7eb10a-H--
Message: Access denied with code 403 (phase 2). Pattern match "(?i)(?:\\x5c|(?:%(?:2(?:5(?:2f|5c)|%46|f)|c
(?:0%(?:9v|af)|1%1c)|u(?:221[56]|002f)|%32(?:%46|F)|e0%80%af|1u|5c)|\\/))(?:%(?:2(?:(?:52)?e|%45)|(?:e0%8|
c)0%ae|u(?:002e|2024)|%32(?:%45|E))|\\.){2}(?:\\x5c|(?:%(?:2(?:5(?:2f|5c)|%46|f)|c(?:0%(?:9v|af)|1%1c)| ..
." at REQUEST_URI. [file "/opt/apache/conf/crs/base_rules/modsecurity_crs_42_tight_security.conf"] [line "
20"] [id "950103"] [rev "2"] [msg "Path Traversal Attack"] [data "Matched Data: /../ found within REQUEST_
URI: /?../../boot"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.8"] [maturity "9"] [accuracy "7"] [tag "OWAS
P_CRS/WEB_ATTACK/DIR_TRAVERSAL"]
Action: Intercepted (phase 2)
Stopwatch: 1379908640106168 1530 (- - -)
Stopwatch2: 1379908640106168 1530; combined=785, p1=140, p2=625, p3=0, p4=0, p5=20, sr=11, sw=0, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.7.5 (http://www.modsecurity.org/); OWASP_CRS/2.2.8.
Server: Apache/2.4.6 (Unix) OpenSSL/1.0.0-fips
Engine-Mode: "ENABLED"

--9f7eb10a-Z--
```

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

Change Server Banner

Earlier in this guide, you learnt how to remove Apache and OS type, version help of ServerTokens directive. Let's go one step ahead, how about keeping server name whatever you wish? It's possible with SecServerSignature directive in Mod Security. You see it's an interesting.

Note: in order to use Mod Security to manipulate Server Banner from header, you must set ServerTokesn to Full in httpd.conf of Apache web server.

Implementation:

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

SecServerSignature YourServerName

Ex:

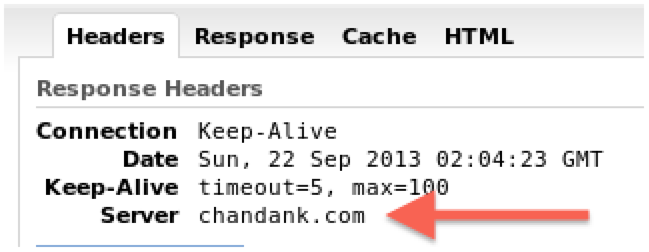
[/opt/apache/conf/crs] #grep SecServer modsecurity_crs_10_setup.conf

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.



```
Response Headers

Connection  Keep-Alive
      Date  Sun, 22 Sep 2013 02:04:23 GMT
Keep-Alive  timeout=5, max=100
    Server  chandank.com
```

General Configuration

We will talk about some of the general configuration as best practice.

Configure Listen

When you have multiple interface and IP's on 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 problem in forwarding HTTP request to some other web server. This is quite common in 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

It's essential to configure access log properly in your web server. Some of the important parameter to capture in log would be the time taken to serve the request, SESSION ID. By default apache is not configured to capture these data. You got to configure them manually as following.

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 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 to disable 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
```