ATTACK DEFENSE

by PentesterAcademy

Name	Fix Misconfigured Apache Webserver
URL	https://www.attackdefense.com/challengedetails?cid=2149
Type	Defender Labs: Basic

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

Infrastructure security is as important as application security if not more. A misconfigured or vulnerable service can allow the attacker to gain access to the system even when the application/website is secure. Hence, it is important to make service hardening configuration changes.

In this lab, an attacker simulator (with testing/verification system) is provided to the user. The CLI access to a machine running a misconfigured Apache webserver is also provided. The user has to make webserver hardening related configuration changes to the apache machine and make sure that all tests provided in the attack simulator are passed.

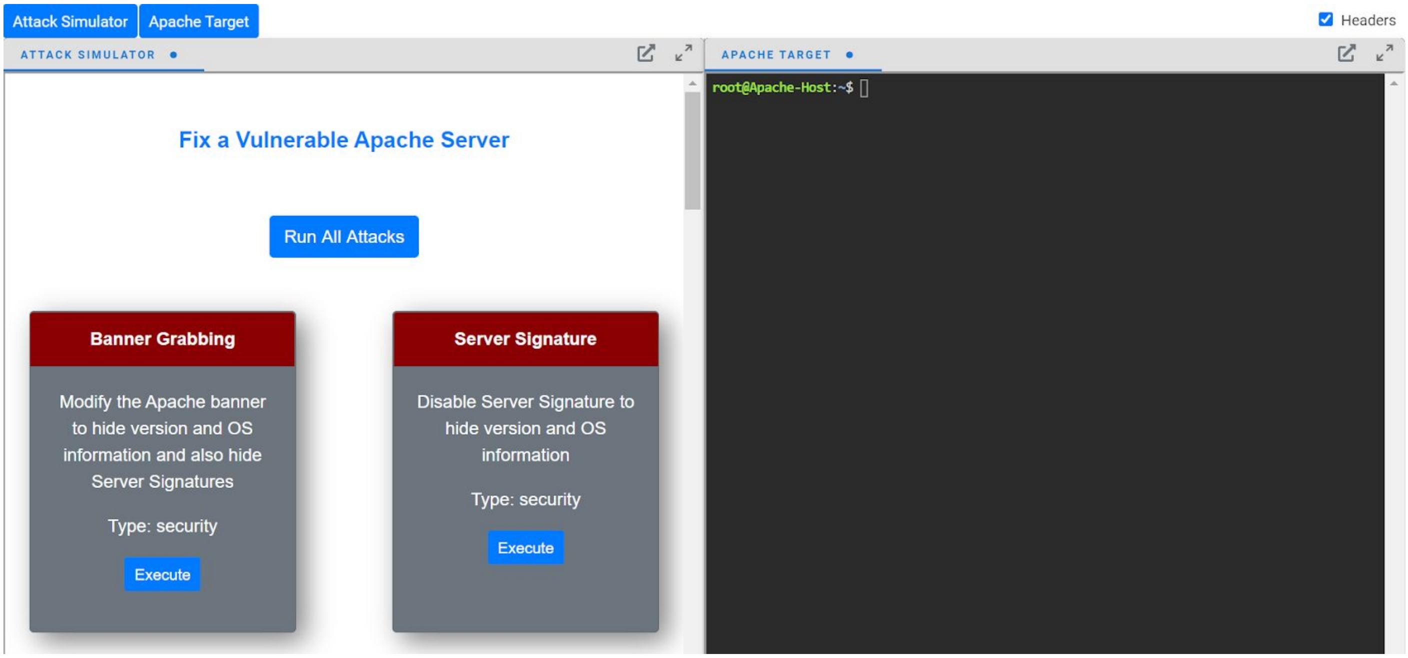
Objective: Fix the misconfiguration and pass all tests!

Guidelines

- The fixes for the tests present in the Attack Simulator are not mutually exclusive. So, changing the configuration to clear some tests might fail the previously passed tests. So it is recommended to perform these fixes in a sequential manner.
- On some browsers, the last line of the console window is not visible. To fix that, uncheck the Headers checkbox on the top right corner of the window.

Lab Setup

On starting the lab, the following interface will be accessible to the user.



Modify the Apache banner to hide version and OS information and also hide Server Signatures

Type: security

Disable Server Signature to hide version and OS information

Type: security

```
root@Apache-Host:~$
```

The **Attack Simulator** is available on the left window and CLI access to the **Apache Web Server** machine on the right window.

Solution:

The tests and fixes for each step are discussed below one by one.

Test 1: Banner Grabbing

Banner grabbing is the process of connecting to the services running on a remote machine using the appropriate tools to gain information about the machine and service.

Commonly used tools for banner grabbing:

- Netcat
- Socat
- Curl
- Nmap

Why modify the banner?

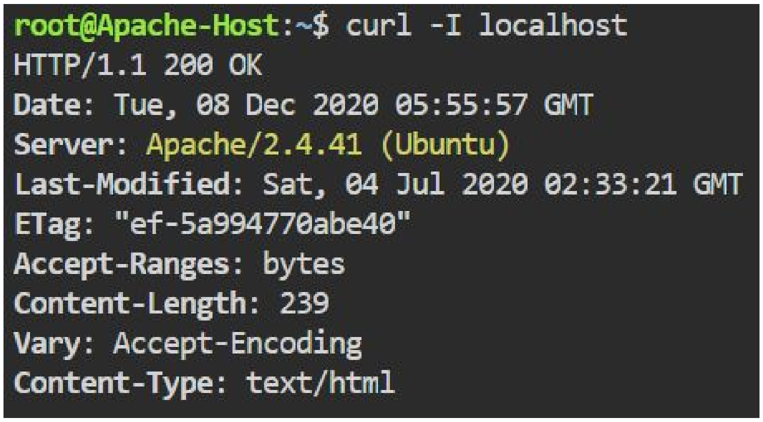
The banners commonly contain the name of the service/software, release version, architecture, etc. This information can help an attacker plan better e.g. attacker can look if this specific version has known vulnerabilities/backdoors etc. Modifying the banner to hide this information will make the attacker's life difficult.

Objective: Modify the Apache banner to hide version and OS information

Step 1: To check the banner information, make a curl request on the localhost.

Command: curl -I localhost

- -I argument instructs curl to show the response headers.



```
root@Apache-Host:~$ curl -I localhost
HTTP/1.1 200 OK
Date: Tue, 08 Dec 2020 05:55:57 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Sat, 04 Jul 2020 02:33:21 GMT
ETag: "ef-5a994770abe40"
Accept-Ranges: bytes
Content-Length: 239
Vary: Accept-Encoding
Content-Type: text/html
```

We can observe the following information:

- OS: Ubuntu Linux
- Software: Apache
- Version: 2.4.41

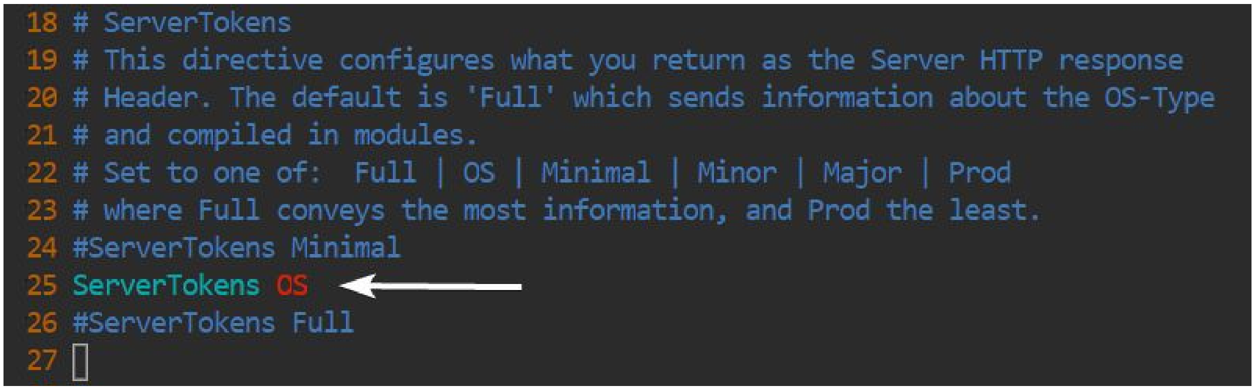
Step 2: To hide this information, open /etc/apache2/conf-available/security.conf and change

ServerTokens OS to ServerTokens Prod

ServerTokens value dictates the Server name, version, OS information to be sent back in response headers.

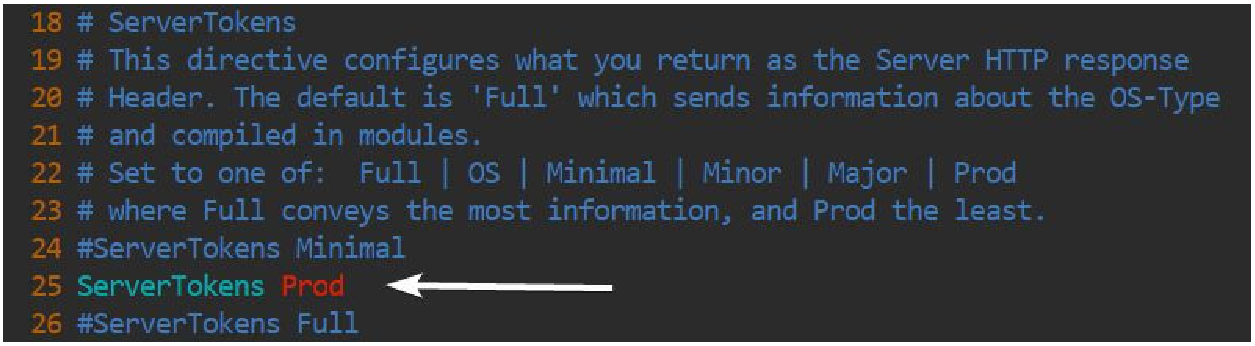
Modify it as shown below

Before modification



```
# ServerTokens
# This directive configures what you return as the Server HTTP response
# Header. The default is 'Full' which sends information about the OS-Type
# and compiled in modules.
# Set to one of:  Full | OS | Minimal | Minor | Major | Prod
# where Full conveys the most information, and Prod the least.
#ServerTokens Minimal
ServerTokens OS
#ServerTokens Full

```

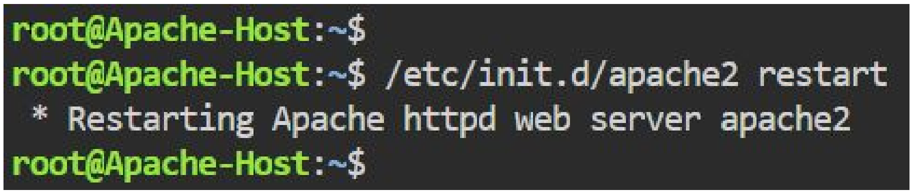
After modification



```
# ServerTokens
# This directive configures what you return as the Server HTTP response
# Header. The default is 'Full' which sends information about the OS-Type
# and compiled in modules.
# Set to one of:  Full | OS | Minimal | Minor | Major | Prod
# where Full conveys the most information, and Prod the least.
#ServerTokens Minimal
ServerTokens Prod
#ServerTokens Full
```

Step 3: Restart Apache

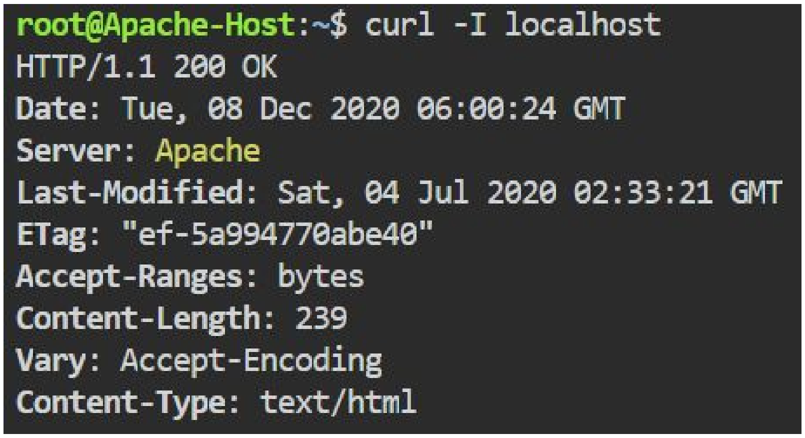
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: Verify if the modification worked

Command: curl -I localhost



```
root@Apache-Host:~$ curl -I localhost
HTTP/1.1 200 OK
Date: Tue, 08 Dec 2020 06:00:24 GMT
Server: Apache
Last-Modified: Sat, 04 Jul 2020 02:33:21 GMT
ETag: "ef-5a994770abe40"
Accept-Ranges: bytes
Content-Length: 239
Vary: Accept-Encoding
Content-Type: text/html
```

Now, only the following information is there:

- Software: Apache

Objective: Modify the Server field in the response header to Microsoft IIS.

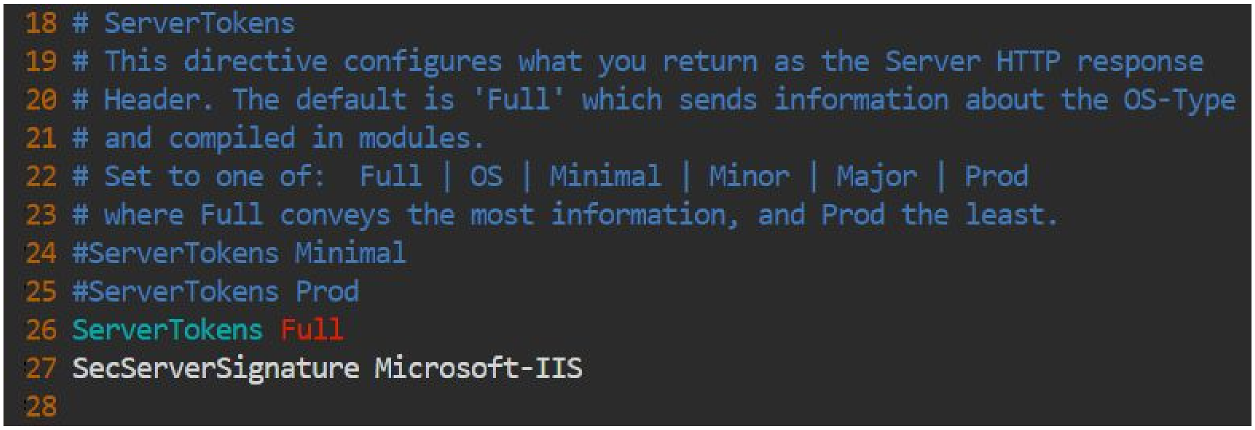
Step 5: To change this, open /etc/apache2/conf-available/security.conf and make the following change

ServerTokens Prod

To

ServerTokens Full
SecServerSignature Microsoft-IIS

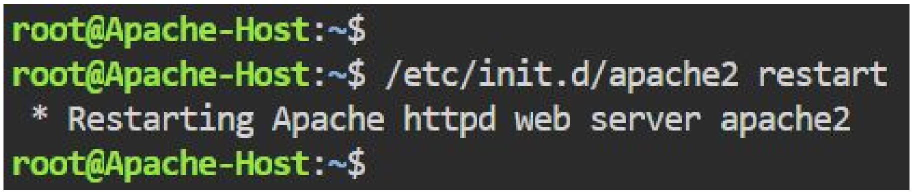
As shown below



```
# ServerTokens
# This directive configures what you return as the Server HTTP response
# Header. The default is 'Full' which sends information about the OS-Type
# and compiled in modules.
# Set to one of:  Full | OS | Minimal | Minor | Major | Prod
# where Full conveys the most information, and Prod the least.
#ServerTokens Minimal
#ServerTokens Prod
ServerTokens Full
SecServerSignature Microsoft-IIS

```

Step 6: Restart Apache

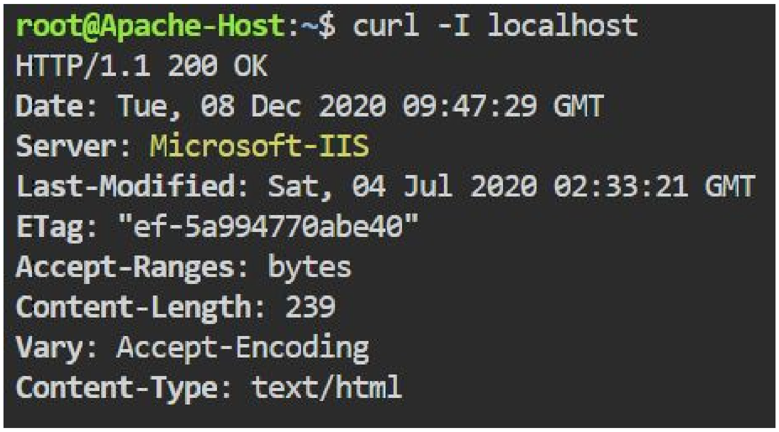
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 7: Verify if the modification worked

Command: curl -I localhost



```
root@Apache-Host:~$ curl -I localhost
HTTP/1.1 200 OK
Date: Tue, 08 Dec 2020 09:47:29 GMT
Server: Microsoft-IIS
Last-Modified: Sat, 04 Jul 2020 02:33:21 GMT
ETag: "ef-5a994770abe40"
Accept-Ranges: bytes
Content-Length: 239
Vary: Accept-Encoding
Content-Type: text/html
```

The server field is now reflecting Microsoft IIS.

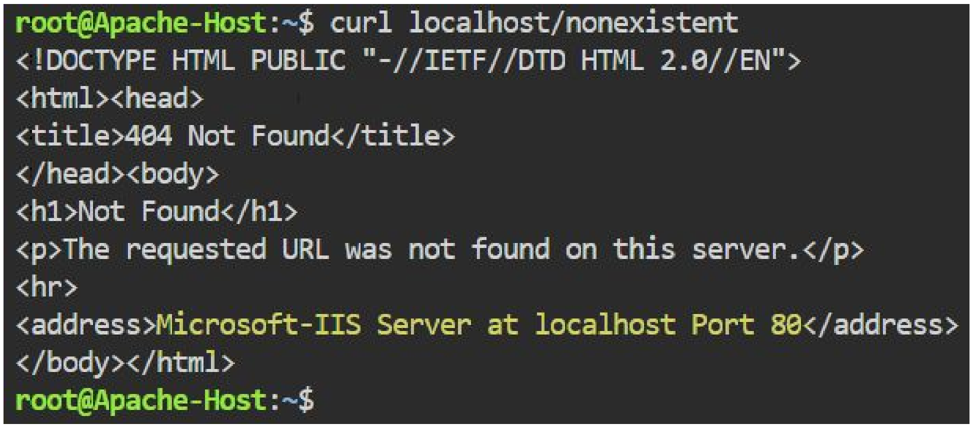
<u>Test 2: Server Signature</u>

Objective: Disable Server Signature

ServerSignature directive is to set the server or contact email information on the bottom of server-generated pages such as error pages, directory listings, etc.

Step 1: The server signature can be observed in error messages.

Command: curl localhost/nonexistent



```
root@Apache-Host:~$ curl localhost/nonexistent
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Microsoft-IIS Server at localhost Port 80</address>
</body></html>
root@Apache-Host:~$
```

Step 2: Open /etc/apache2/conf-available/security.conf and change the following line

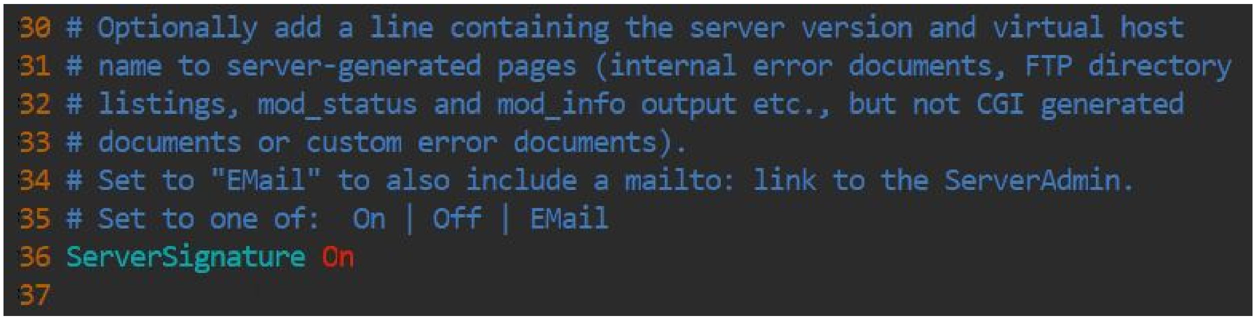
ServerSignature On

to

ServerSignature Off

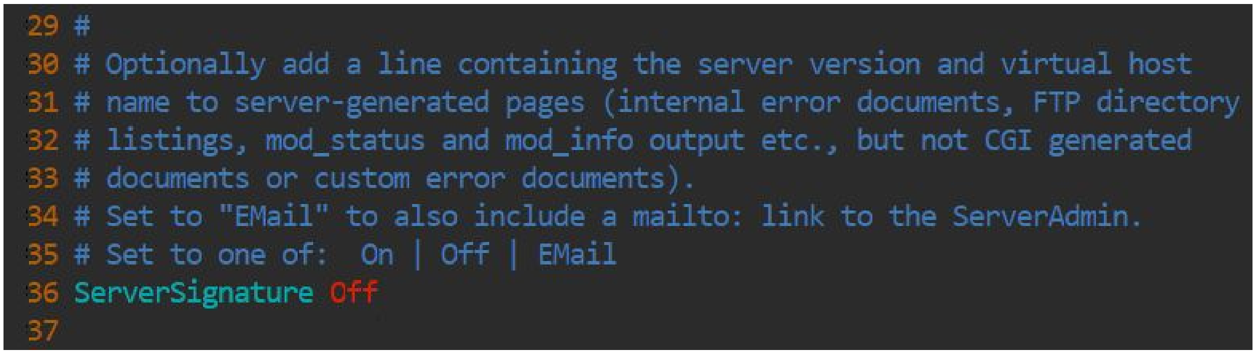
As shown below

Before modification



```
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (internal error documents, FTP directory
# listings, mod_status and mod_info output etc., but not CGI generated
# documents or custom error documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of:  On | Off | EMail
ServerSignature On

```

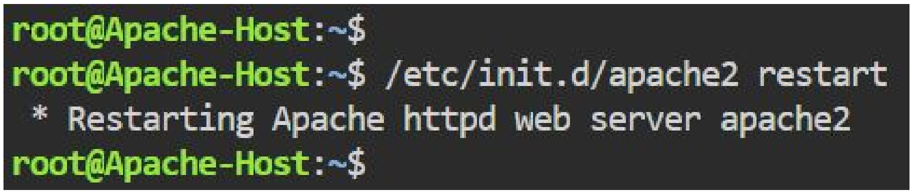
After modification



```
#
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (internal error documents, FTP directory
# listings, mod_status and mod_info output etc., but not CGI generated
# documents or custom error documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of:  On | Off | EMail
ServerSignature Off

```

Step 3: Restart Apache

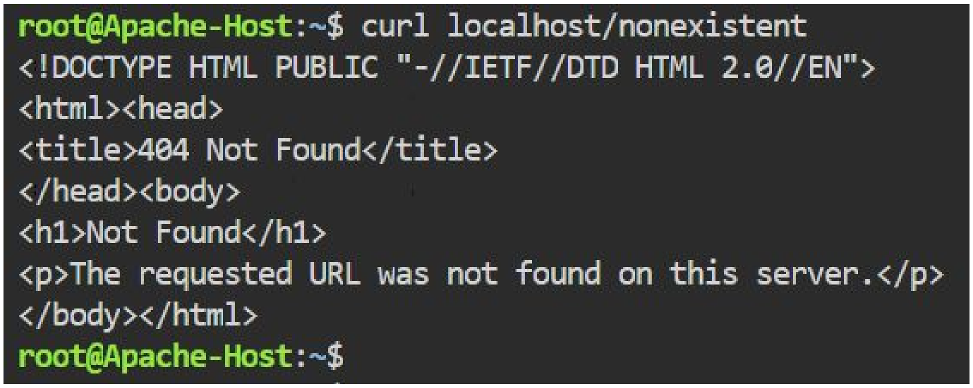
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: Verify if the modification worked

Command: curl localhost/nonexistent



```
root@Apache-Host:~$ curl localhost/nonexistent
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
</body></html>
root@Apache-Host:~$
```

The Server Signature is not present in the error message.

Read more:

- Banner Grabbing (https://en.wikipedia.org/wiki/Banner_grabbing)
- Difference between ServerToken and ServerSignature (https://ubiq.co/tech-blog/remove-server-name-apache-response-header/)
- Changing Server Signature (https://stackoverflow.com/questions/26296886/completely-hide-server-name-apache)

Test 3: Cache Etag

Etag is used to perform efficient caching. It is a unique ID generated for the resource and doesn't change till the time the corresponding resource is changed. This helps the caches to know if the resource on a particular URL is changed or not by just keeping track of the Etags.

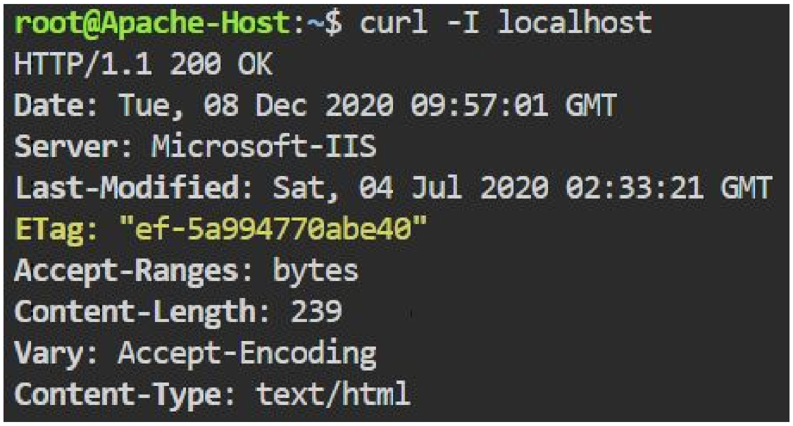
Why disable Etags?

The Apache web server has an information disclosure vulnerability related to Etags (in the default configuration). Etag for a specific file can contain an “i-node” value. This information alone is harmless but it can lead to certain attacks while combined with NFS.

Objective: Disable Etags

Step 1: To check the banner information, make a curl request on the localhost.

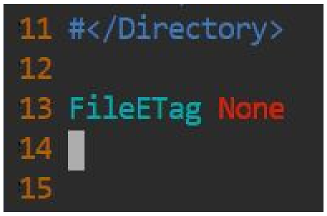
Command: curl -I localhost



```
root@Apache-Host:~$ curl -I localhost
HTTP/1.1 200 OK
Date: Tue, 08 Dec 2020 09:57:01 GMT
Server: Microsoft-IIS
Last-Modified: Sat, 04 Jul 2020 02:33:21 GMT
ETag: "ef-5a994770abe40"
Accept-Ranges: bytes
Content-Length: 239
Vary: Accept-Encoding
Content-Type: text/html
```

Step 2: Open /etc/apache2/conf-enabled/security.conf and add the following lines to it

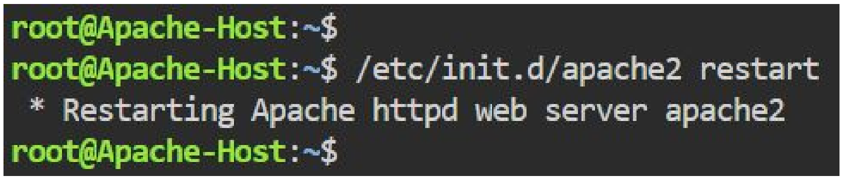
FileETag None



```
#</Directory>

FileETag None


```

Step 3: Restart Apache

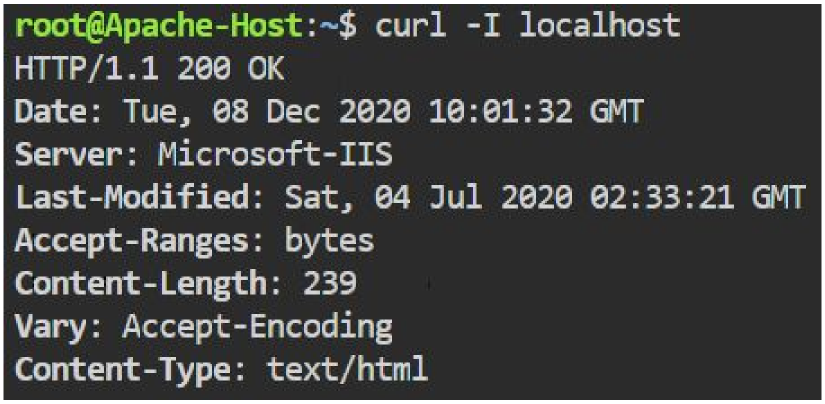
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: To check the banner information, make a curl request on the localhost.

Command: curl -I localhost



```
root@Apache-Host:~$ curl -I localhost
HTTP/1.1 200 OK
Date: Tue, 08 Dec 2020 10:01:32 GMT
Server: Microsoft-IIS
Last-Modified: Sat, 04 Jul 2020 02:33:21 GMT
Accept-Ranges: bytes
Content-Length: 239
Vary: Accept-Encoding
Content-Type: text/html
```

The Etag field is removed from the response header.

Read more:

- What is Etag header (https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/ETag)
- Why disable Etags (https://securityblog.port.ac.uk/?p=1180)
- What can happen with an exposed Etag (https://security.stackexchange.com/questions/186482/damage-of-a-leaked-etag)

Test 4: Icons Alias

Icons Alias is used in default Apache configuration to include the /icons/ alias for FancyIndexed directory listings.

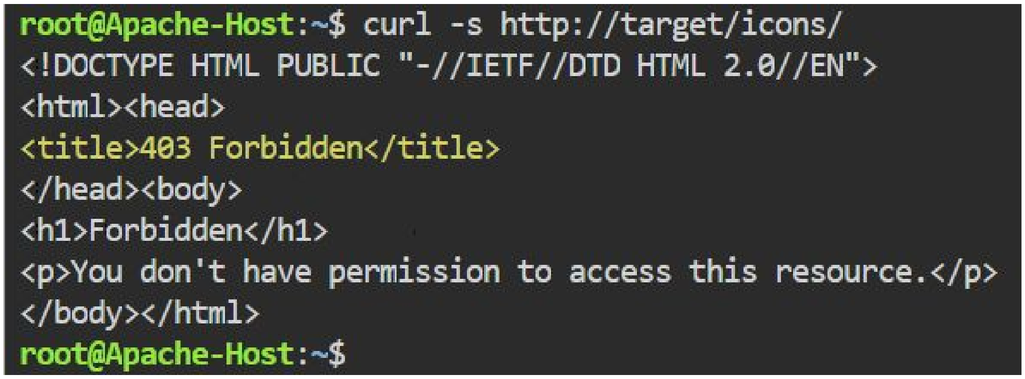
Why disable Icons Alias?

It is not a critical issue. However, it can lead to information disclosure in some cases. And, this will also be flagged by security scanners. So, it is generally suggested to disable it.

Objective: Disable the built-in icons alias

Step 1: Check if the listing on the /icons directory is enabled.

Command: curl -s http://target/icons/



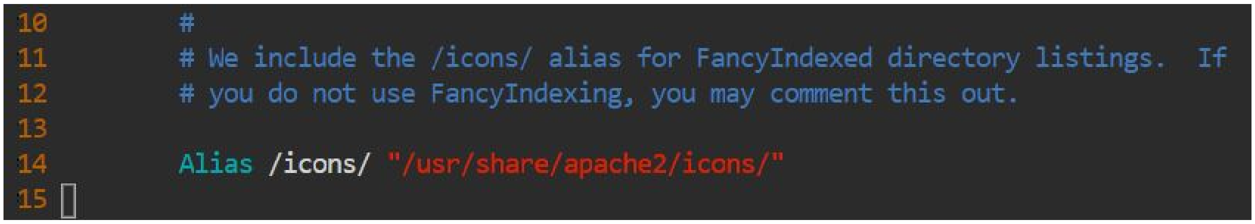
```
root@Apache-Host:~$ curl -s http://target/icons/
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
root@Apache-Host:~$
```

Step 2: Open /etc/apache2/mods-available/alias.conf and comment out the following

Alias /icons/ "/usr/share/apache2/icons/"

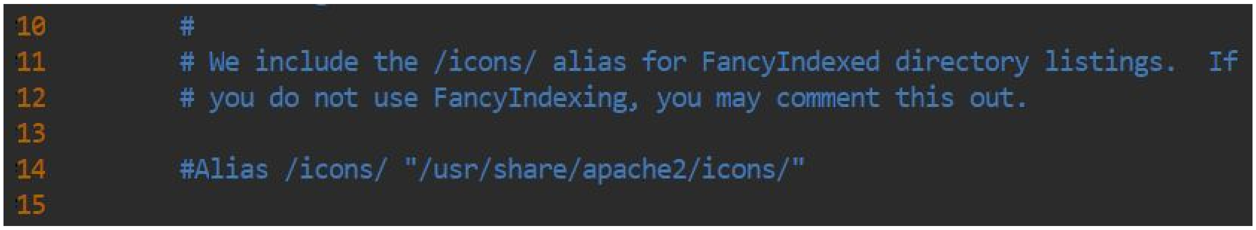
As shown below

Before modification:



```
        #
        # We include the /icons/ alias for FancyIndexed directory listings.  If
        # you do not use FancyIndexing, you may comment this out.

        Alias /icons/ "/usr/share/apache2/icons/"

```

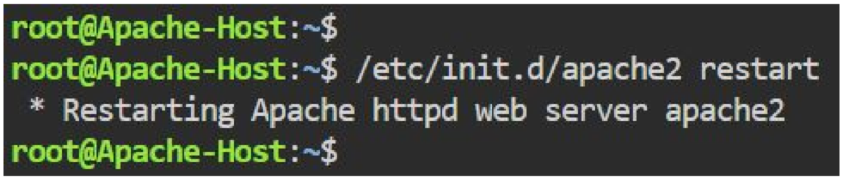
After modification:



```
        #
        # We include the /icons/ alias for FancyIndexed directory listings.  If
        # you do not use FancyIndexing, you may comment this out.

        #Alias /icons/ "/usr/share/apache2/icons/"

```

Step 3: Restart Apache

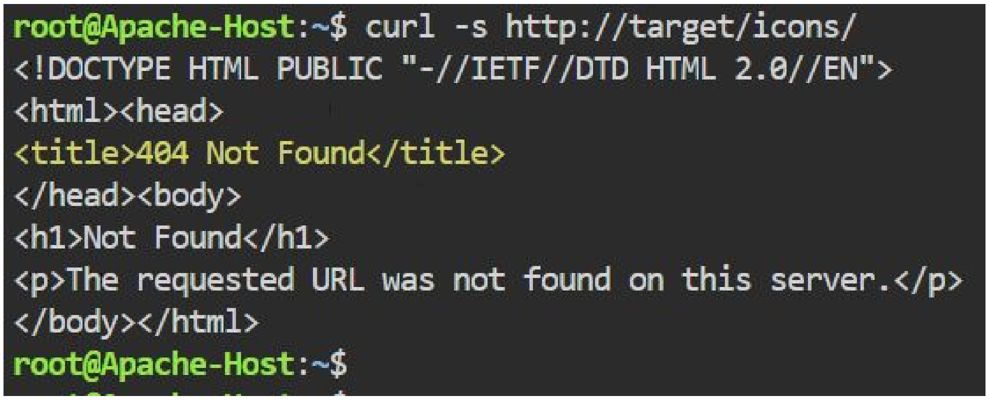
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: Verify if the icons alias is disabled and /icons directory is not accessible anymore

Command: curl -s http://target/icons/



```
root@Apache-Host:~$ curl -s http://target/icons/
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
</body></html>
root@Apache-Host:~$
```

The /icons directory is not accessible.

Read more:

- Disabling icons alias (https://ilateralweb.co.uk/web-design-news/disabling-the-icons-folder-on-an-ubuntu-web-server/)
- Example security bug report (https://hackerone.com/reports/7923)

Test 5: Directory Listing

Directory listing refers to the webserver returning the list of contents for the directories for which the index page doesn't exist.

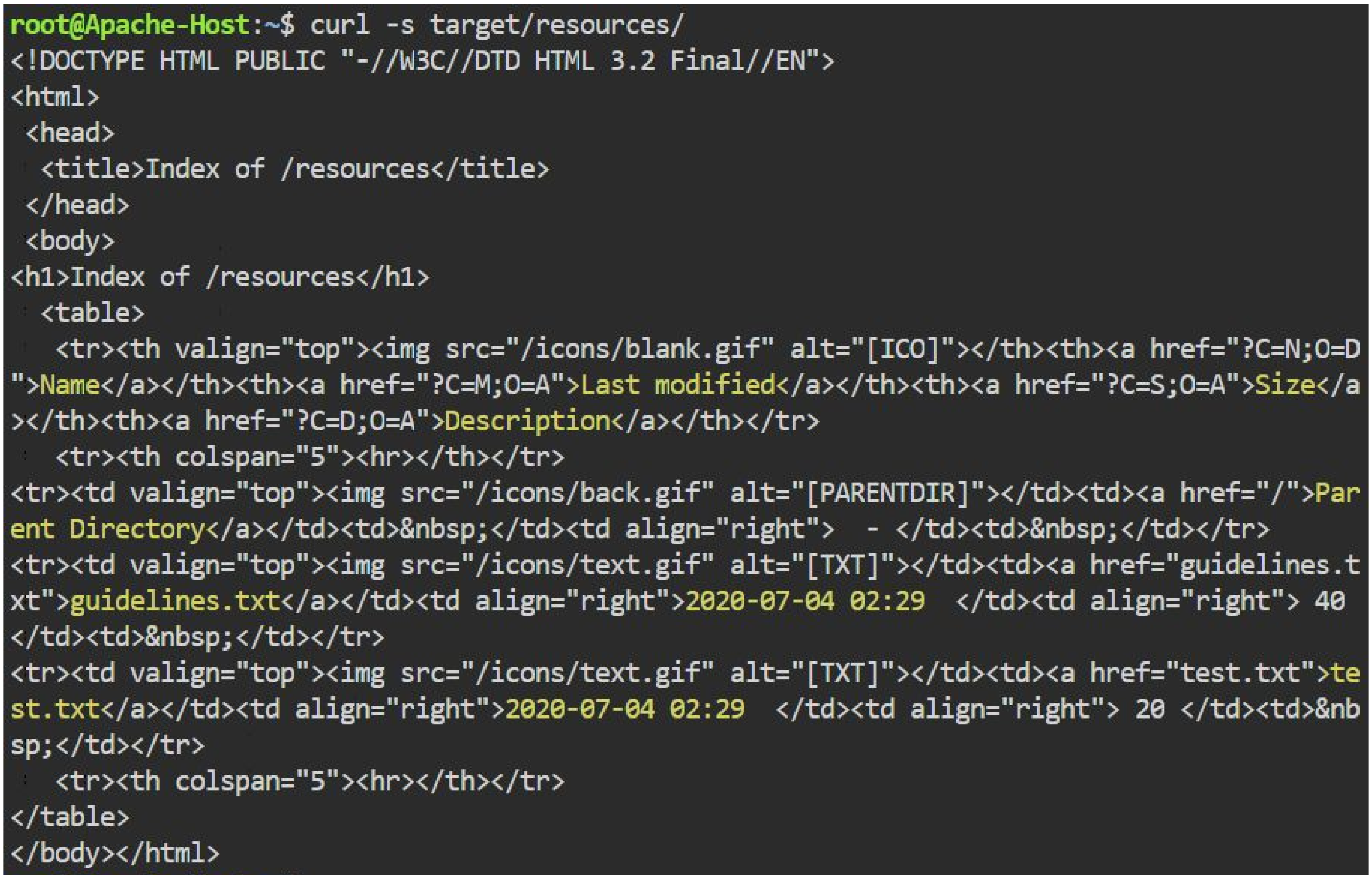
Why disable directory listing?

The directory listing can lead to information disclosure by exposing the name, type, size and modification dates of the files present in the directory. This information can help the attacker.

Objective: Disable the directory listing for /resources directory

Step 1: Check if directory listing is enabled.

Command: curl -s target/resources/



```
root@Apache-Host:~$ curl -s target/resources/
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
 <head>
  <title>Index of /resources</title>
 </head>
 <body>
<h1>Index of /resources</h1>
  <table>
   <tr><th valign="top"><img src="/icons/blank.gif" alt="[ICO]"></th><th><a href="?C=N;O=D
">Name</a></th><th><a href="?C=M;O=A">Last modified</a></th><th><a href="?C=S;O=A">Size</a
></th><th><a href="?C=D;O=A">Description</a></th></tr>
   <tr><th colspan="5"><hr></th></tr>
<tr><td valign="top"><img src="/icons/back.gif" alt="[PARENTDIR]"></td><td><a href="/">Par
ent Directory</a></td><td>&nbsp;</td><td align="right">  - </td><td>&nbsp;</td></tr>
<tr><td valign="top"><img src="/icons/text.gif" alt="[TXT]"></td><td><a href="guidelines.t
xt">guidelines.txt</a></td><td align="right">2020-07-04 02:29  </td><td align="right"> 40
</td><td>&nbsp;</td></tr>
<tr><td valign="top"><img src="/icons/text.gif" alt="[TXT]"></td><td><a href="test.txt">te
st.txt</a></td><td align="right">2020-07-04 02:29  </td><td align="right"> 20 </td><td>&nb
sp;</td></tr>
   <tr><th colspan="5"><hr></th></tr>
</table>
</body></html>
```

The directory listing is enabled.

Step 2: Open /etc/apache2/apache2.conf and replace the following line

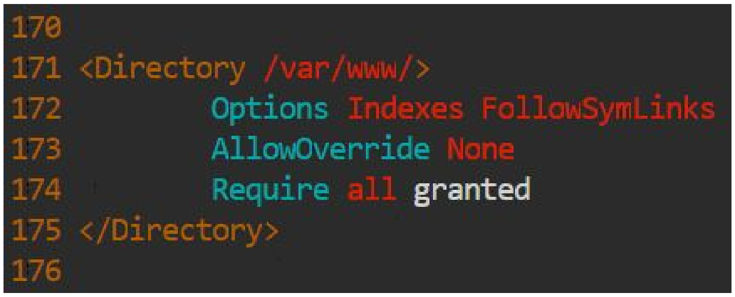
    Options Indexes FollowSymLinks

with

    Options FollowSymLinks

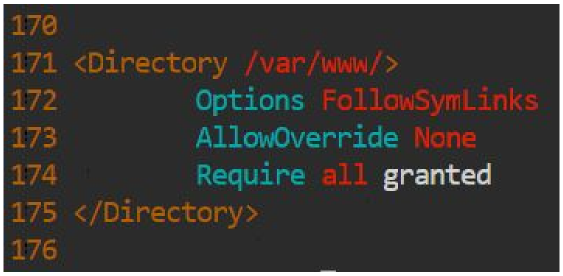
As shown below

Before Modification:



```

<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

```

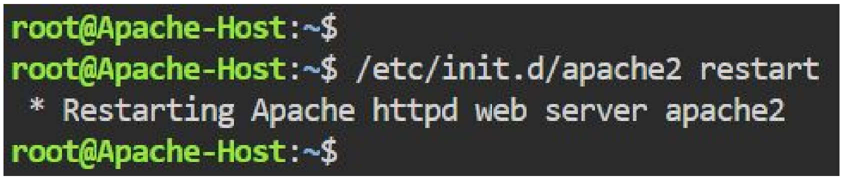
After Modification



```

<Directory /var/www/>
        Options FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

```

Step 3: Restart Apache

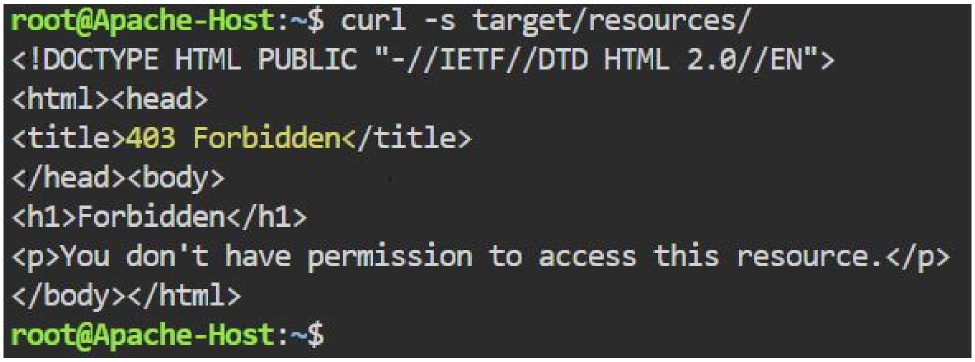
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: Verify if the directory listing is disabled.

Command: curl -s target/resources/



```
root@Apache-Host:~$ curl -s target/resources/
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
root@Apache-Host:~$
```

Directory listing is now disabled.

Read More:

- Why is directory listing dangerous (https://www.acunetix.com/blog/articles/directory-listing-information-disclosure)

Test 6: .git or .svn directries

Git and SVN version control systems are used by developers for maintaining and sharing the code. As the metadata directories used by these systems (e.g. .git and .svn) are hidden in the Linux systems, it is possible for the deployment team to copy these to the server along with the application code.

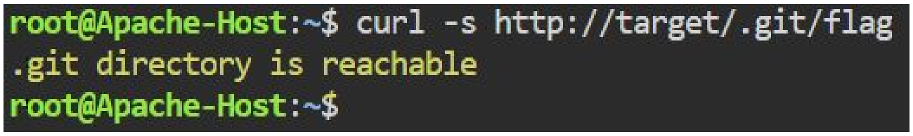
Why disable access to .git or .svn directories?

These metadata directories contain a lot of information that can be leveraged by the attacker to further their cause. For example, .git directory of a project can

Objective: Disable access to .git or .svn directories

Step 1: Check if the files present in .git directories are accessible.

Command: curl -s http://target/.git/flag



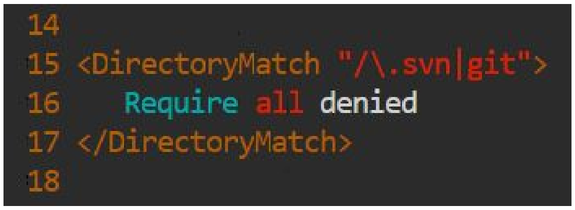
```
root@Apache-Host:~$ curl -s http://target/.git/flag
.git directory is reachable
root@Apache-Host:~$
```

Step 2: Open /etc/apache2/conf-enabled/security.conf

Add the following lines to

```
<DirectoryMatch "/\.svn|git">
   Require all denied
</DirectoryMatch>
```

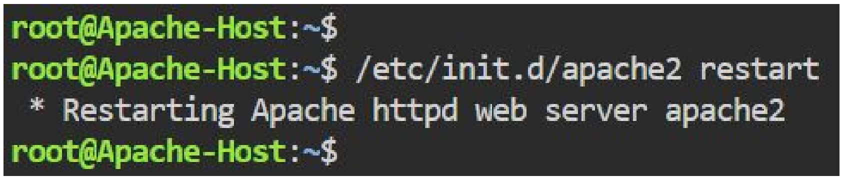
As shown below



```
<DirectoryMatch "/\.svn|git">
    Require all denied
</DirectoryMatch>
```

Step 3: Restart Apache

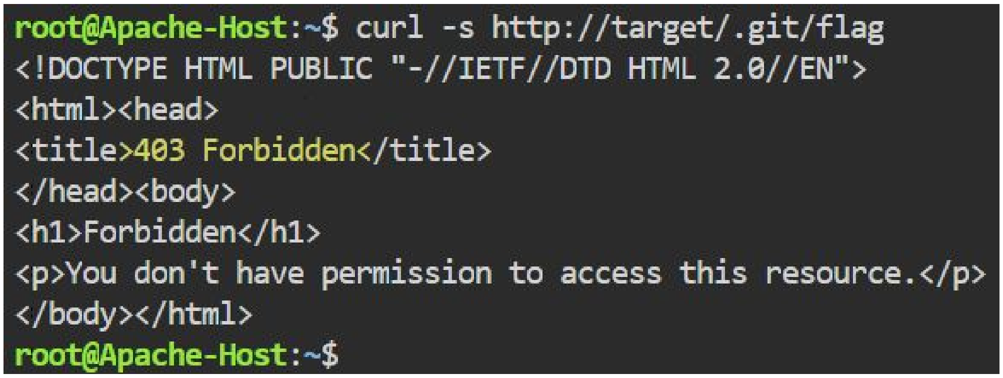
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: Verify if the directory listing is disabled.

Command: curl -s http://target/.git/flag



```
root@Apache-Host:~$ curl -s http://target/.git/flag
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
root@Apache-Host:~$
```

The access to the .git directory is blocked now.

Read more:

- Risk of exposing .git directory (https://en.internetwache.org/dont-publicly-expose-git-or-how-we-downloaded-your-websites-sourcecode-an-analysis-of-alexas-1m-28-07-2015/)
- GitTools (https://github.com/internetwache/GitTools)
- Practice lab on AttackDefense (https://attackdefense.com/challengedetails?cid=1038)

Test 7: HTTP Methods

Webservers support multiple request methods like GET, POST, DELETE, PUT etc. However, most of the time, only a few of these methods are used.

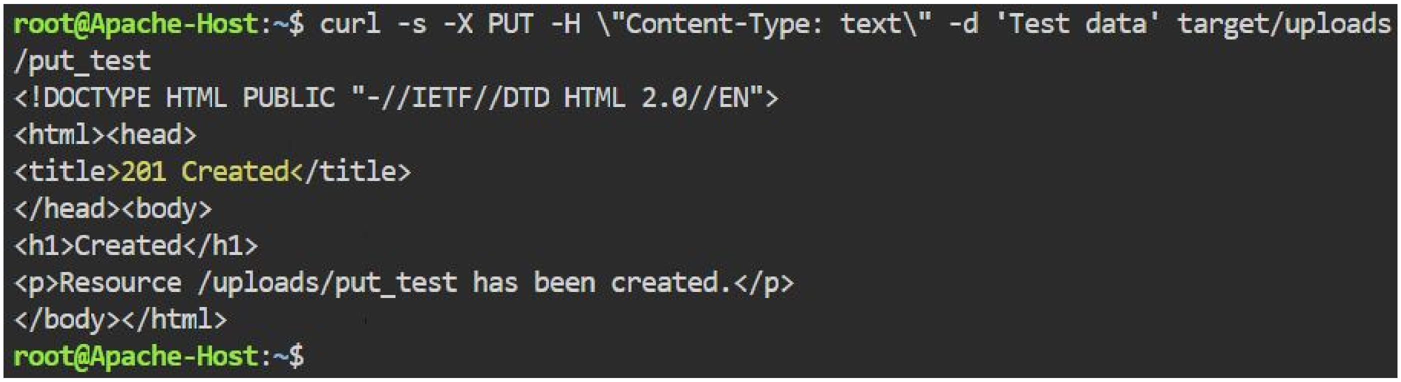
Why disable non-essential/required methods?

Some of these methods come with potential risks and can increase the attack surface. Hence, it is recommended to disable the methods, not in use.

Objective: Disable non-essential HTTP methods and requests

Step 1: Check if the PUT method is allowed on the server. Try to create a new file in /uploads directory.

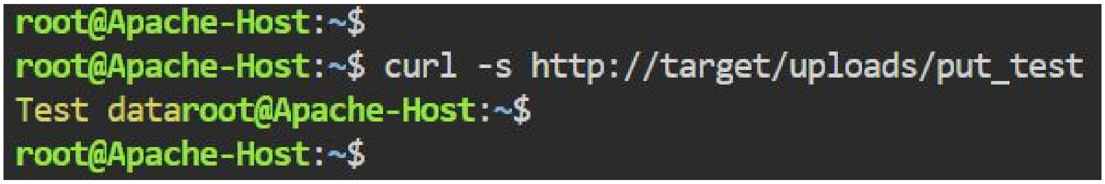
Command: curl -s -X PUT -H \"Content-Type: text\" -d 'Test data' target/uploads/put_test



```
root@Apache-Host:~$ curl -s -X PUT -H \"Content-Type: text\" -d 'Test data' target/uploads
/put_test
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>201 Created</title>
</head><body>
<h1>Created</h1>
<p>Resource /uploads/put_test has been created.</p>
</body></html>
root@Apache-Host:~$
```

Step 2: Check if the file exists

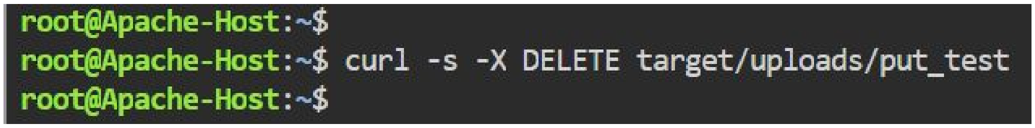
Command: curl -s http://target/uploads/put_test



```
root@Apache-Host:~$
root@Apache-Host:~$ curl -s http://target/uploads/put_test
Test dataroot@Apache-Host:~$
root@Apache-Host:~$
```

Step 3: Check if the DELETE method is allowed on the server. Try to delete the newly created file in /uploads directory.

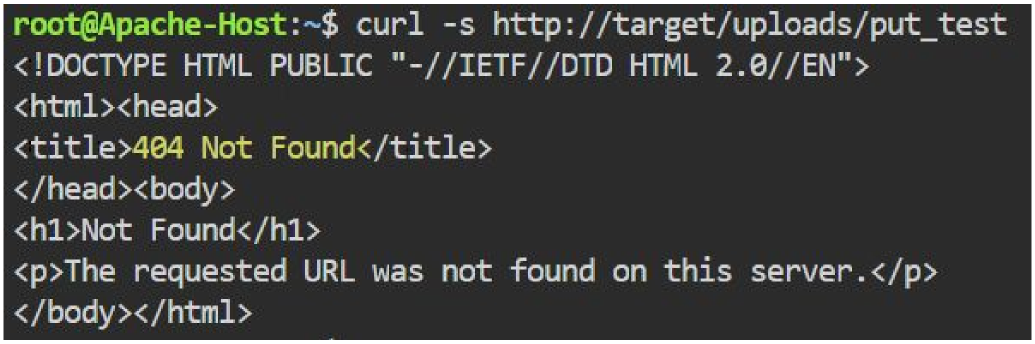
Command: curl -s -X DELETE target/uploads/put_test



```
root@Apache-Host:~$
root@Apache-Host:~$ curl -s -X DELETE target/uploads/put_test
root@Apache-Host:~$
```

Step 4: Check if the file still exists

Command: curl -s http://target/uploads/put_test



```
root@Apache-Host:~$ curl -s http://target/uploads/put_test
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
</body></html>
```

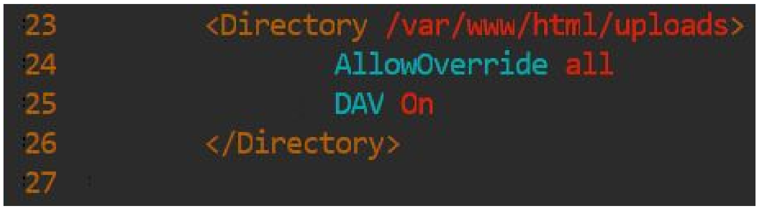
Step 5: Open /etc/apache2/sites-available/000-default.conf

Comment the following line

DAV On

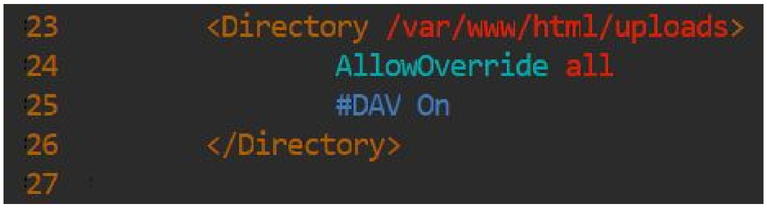
As shown below

Before Modification



```
        <Directory /var/www/html/uploads>
                AllowOverride all
                DAV On
        </Directory>

```

After Modification

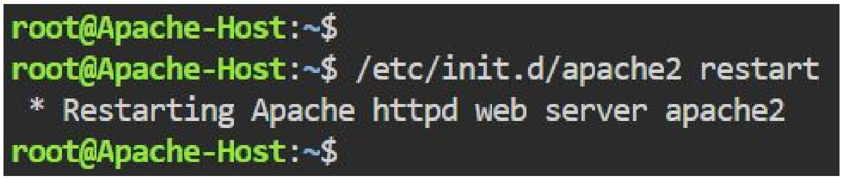


```
        <Directory /var/www/html/uploads>
                AllowOverride all
                #DAV On
        </Directory>

```

This change is to disable WebDAV (Web-based Distributed Authoring and Versioning) provided by the mod_dav module. This extension allows creating, moving, copying, and deleting resources.

Step 6: Restart Apache

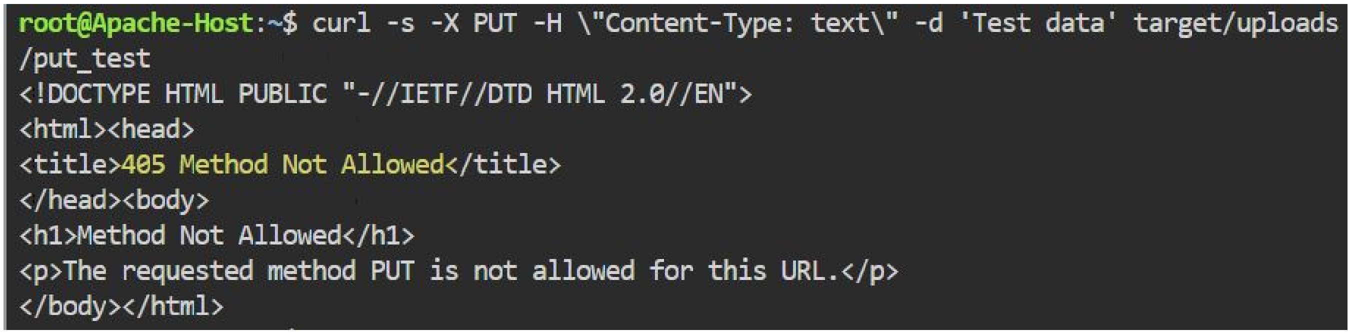
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 7: To verify the change, try to create a new file in /uploads directory.

Command: curl -s -X PUT -H \"Content-Type: text\" -d 'Test data' target/uploads/put_test



```
root@Apache-Host:~$ curl -s -X PUT -H \"Content-Type: text\" -d 'Test data' target/uploads
/put_test
<!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 PUT is not allowed for this URL.</p>
</body></html>
```

PUT and DELETE request method and not allowed.

Read more:

- Hardening Apache server (https://geekflare.com/apache-web-server-hardening-security/)
- mod_dav (http://publib.boulder.ibm.com/httpserv/manual70/mod/mod_dav.html)

The TRACE method is when enabled, makes the webserver send the will exact response it got in the response. It is used for diagnostics.

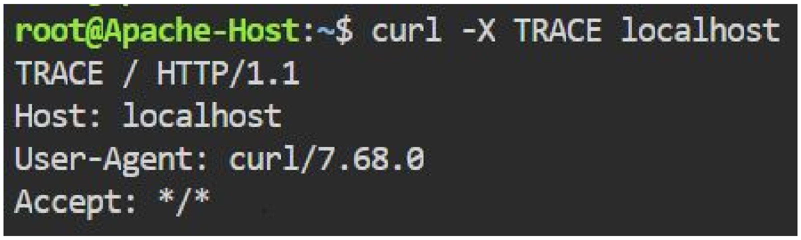
Why disable TRACE method?

Mostly this method is harmless but it can lead to information disclosure by showing information about the proxies like proxy authentication headers.

Objective: Disable the TRACE request support on the webserver

Step 1: Check if the TRACE request method is allowed on the webserver

Command: curl -X TRACE localhost



```
root@Apache-Host:~$ curl -X TRACE localhost
TRACE / HTTP/1.1
Host: localhost
User-Agent: curl/7.68.0
Accept: */*
```

Step 2: Open /etc/apache2/conf-enabled/security.conf and replace the following

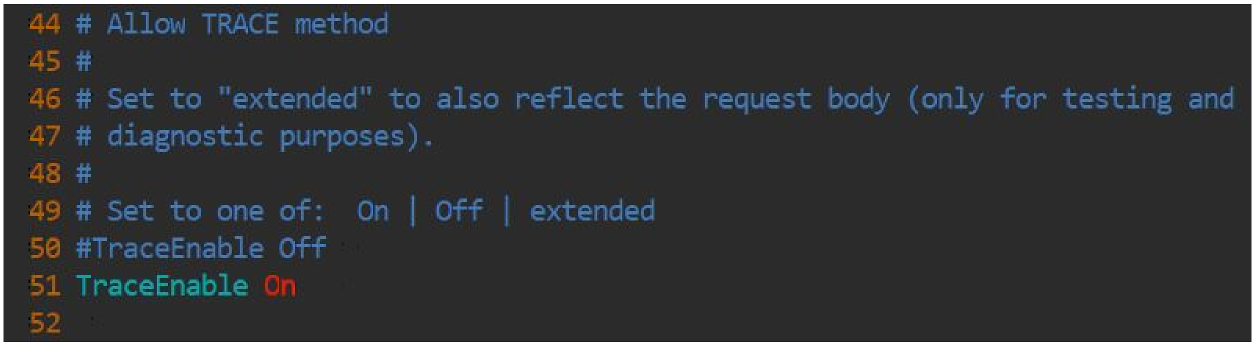
```
TraceEnable On
```

with

```
TraceEnable Off
```

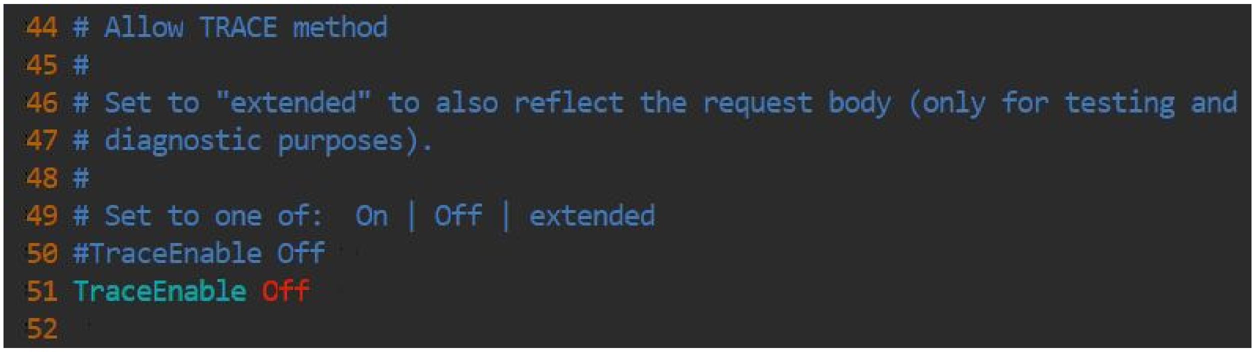
As shown below

Before Modification



```
# Allow TRACE method
#
# Set to "extended" to also reflect the request body (only for testing and
# diagnostic purposes).
#
# Set to one of:  On | Off | extended
#TraceEnable Off
TraceEnable On

```

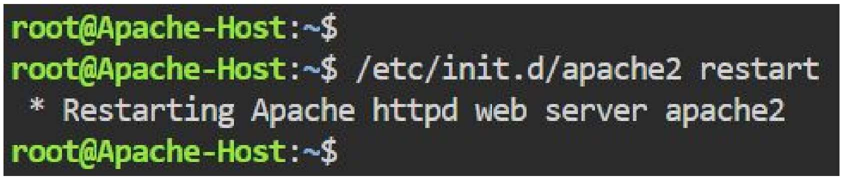
After Modification



```
# Allow TRACE method
#
# Set to "extended" to also reflect the request body (only for testing and
# diagnostic purposes).
#
# Set to one of:  On | Off | extended
#TraceEnable Off
TraceEnable Off

```

Step 3: Restart Apache

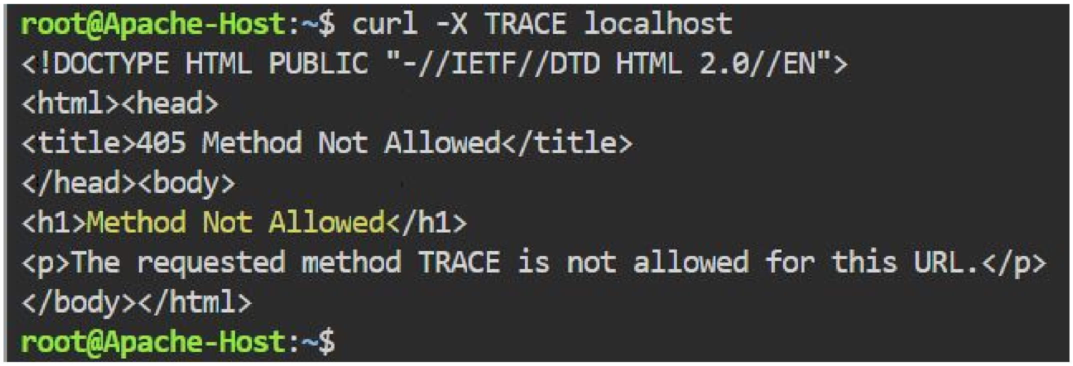
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: Verify if the TRACE request method is disabled

Command: curl -X TRACE localhost



```
root@Apache-Host:~$ curl -X TRACE localhost
<!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 this URL.</p>
</body></html>
root@Apache-Host:~$
```

The TRACE request method is now disabled.

Read more:

- TRACE method enabled (https://portswigger.net/kb/issues/00500a00_http-trace-method-is-enabled)
- TRACE method enabled vulnerability (https://www.acunetix.com/vulnerabilities/web/trace-method-is-enabled/)

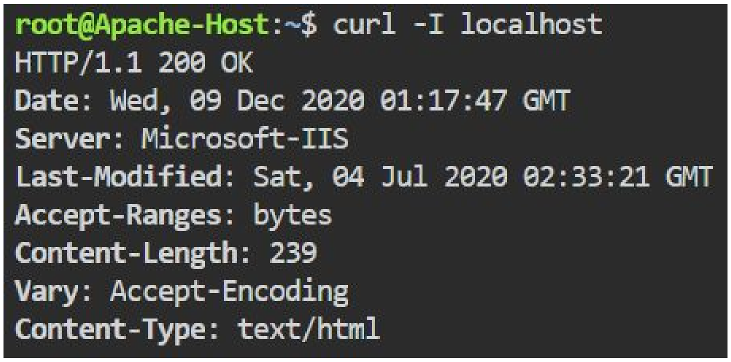
Test 8: XSS and Clickjacking

XSS (Cross-Site Scripting) and Clickjacking vulnerabilities are well known. These vulnerabilities can lead to attacks on clients to steal their information, session etc. Hence, it is important to enable built-in protection against these.

Objective: Enable the built-in protections against XSS and Clickjacking

Step 1: Check if XSS protection and clickjacking protection headers are present in the response.

Command: curl -I localhost

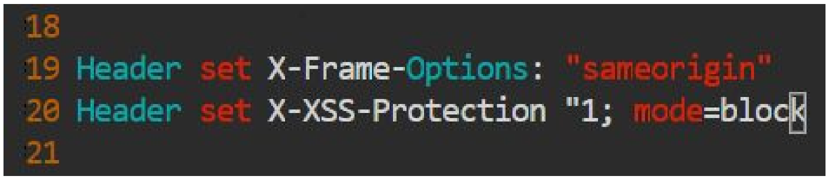


```
root@Apache-Host:~$ curl -I localhost
HTTP/1.1 200 OK
Date: Wed, 09 Dec 2020 01:17:47 GMT
Server: Microsoft-IIS
Last-Modified: Sat, 04 Jul 2020 02:33:21 GMT
Accept-Ranges: bytes
Content-Length: 239
Vary: Accept-Encoding
Content-Type: text/html
```

Step 2: Open /etc/apache2/conf-enabled/security.conf and add the following lines

Header set X-Frame-Options: "sameorigin"
Header set X-XSS-Protection "1; mode=block

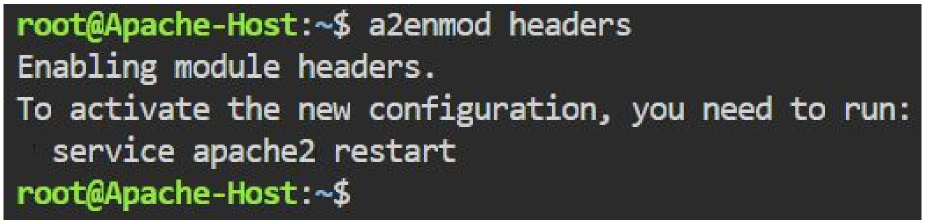
As shown below



```

Header set X-Frame-Options: "sameorigin"
Header set X-XSS-Protection "1; mode=block

```

Step 3: Enable the module

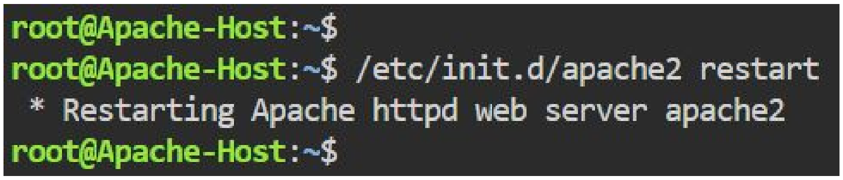
Command: a2enmod headers



```
root@Apache-Host:~$ a2enmod headers
Enabling module headers.
To activate the new configuration, you need to run:
  service apache2 restart
root@Apache-Host:~$
```

Step 4: Restart Apache

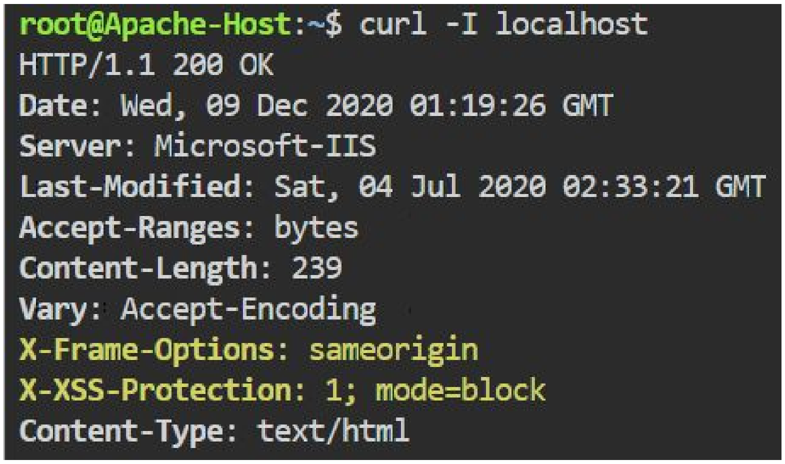
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 5: Verify if the if XSS protection and clickjacking protection headers are enabled.

Command: curl -I localhost



```
root@Apache-Host:~$ curl -I localhost
HTTP/1.1 200 OK
Date: Wed, 09 Dec 2020 01:19:26 GMT
Server: Microsoft-IIS
Last-Modified: Sat, 04 Jul 2020 02:33:21 GMT
Accept-Ranges: bytes
Content-Length: 239
Vary: Accept-Encoding
X-Frame-Options: sameorigin
X-XSS-Protection: 1; mode=block
Content-Type: text/html
```

XSS and clickjacking protection headers are now enabled.

Test 9: HTTP 1.0 Protocol

HTTP 1.0 is the older version of the HTTP protocol.

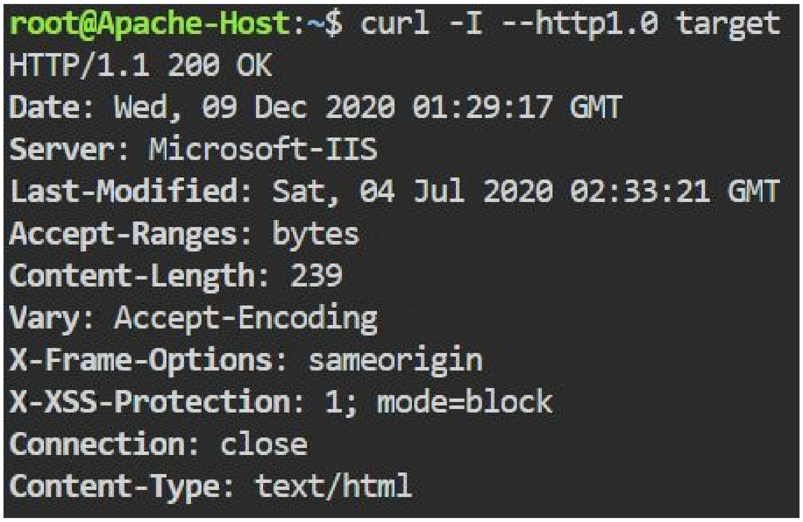
Why disable HTTP 1.0?

HTTP 1.0 is known to have a security weakness related to session hijacking. Hence, it should be disabled to protect users.

Objective: Disable the support for HTTP 1.0 protocol

Step 1: Check if HTTP 1.0 is supported on the webserver

Command: curl -I --http1.0 target

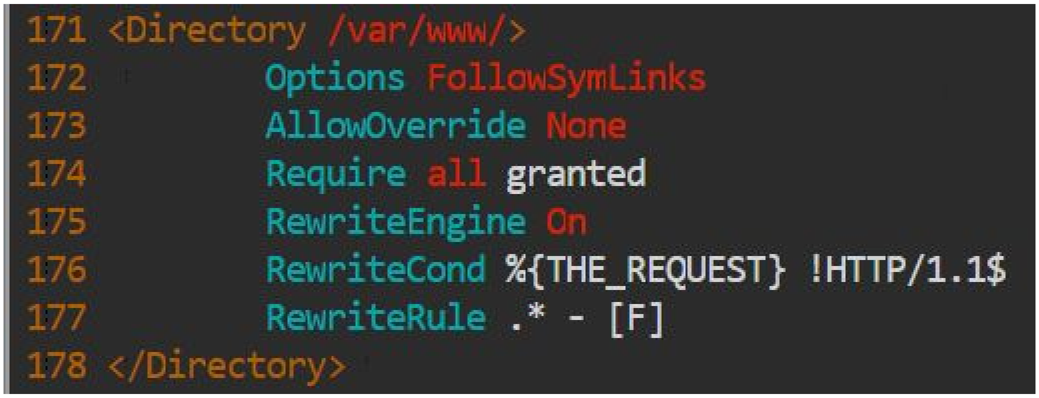


```
root@Apache-Host:~$ curl -I --http1.0 target
HTTP/1.1 200 OK
Date: Wed, 09 Dec 2020 01:29:17 GMT
Server: Microsoft-IIS
Last-Modified: Sat, 04 Jul 2020 02:33:21 GMT
Accept-Ranges: bytes
Content-Length: 239
Vary: Accept-Encoding
X-Frame-Options: sameorigin
X-XSS-Protection: 1; mode=block
Connection: close
Content-Type: text/html
```

Step 2: Open /etc/apache2/apache2.conf and add the following lines in the <Directory /var/www/>   block

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

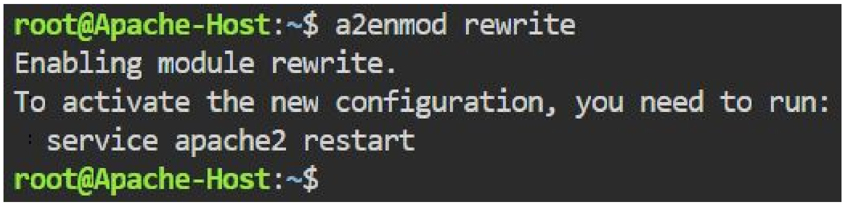
As shown below



```
<Directory /var/www/>
        Options FollowSymLinks
        AllowOverride None
        Require all granted
        RewriteEngine On
        RewriteCond %{THE_REQUEST} !HTTP/1.1$
        RewriteRule .* - [F]
</Directory>
```

Step 3: Enable the module

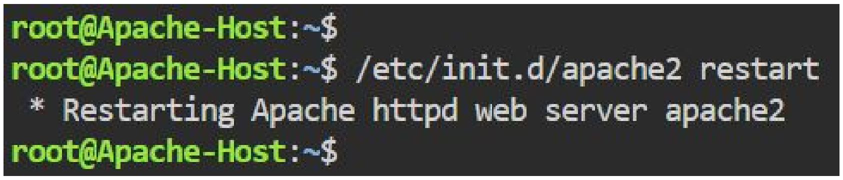
Command: a2enmod rewrite



```
root@Apache-Host:~$ a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  service apache2 restart
root@Apache-Host:~$
```

Step 4: Restart Apache

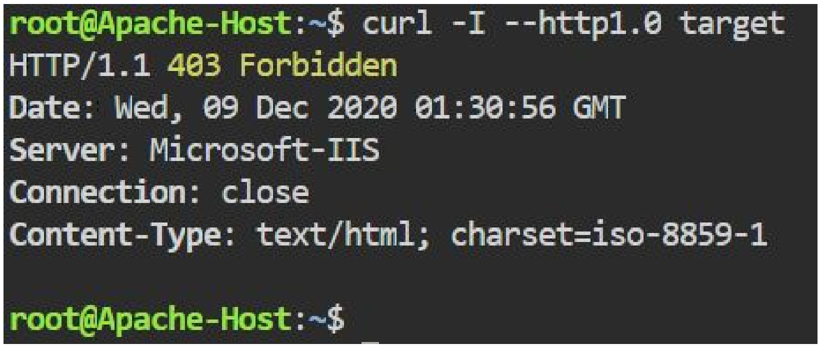
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: Verify if HTTP 1.0 support is disabled.

Command: curl -I --http1.0 target



```
root@Apache-Host:~$ curl -I --http1.0 target
HTTP/1.1 403 Forbidden
Date: Wed, 09 Dec 2020 01:30:56 GMT
Server: Microsoft-IIS
Connection: close
Content-Type: text/html; charset=iso-8859-1

root@Apache-Host:~$
```

Read more:

- Session Hijacking (https://www.imperva.com/learn/application-security/session-hijacking/)
- Why disable HTTP 1.0 (https://geekflare.com/apache-web-server-hardening-security/)

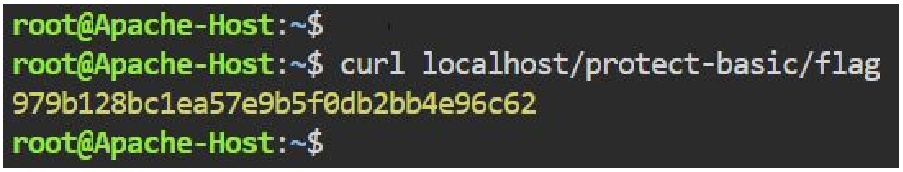
Test 10: Protecting Directories

Basic Authentication and Digest Authentication are mechanisms to protect non-public information on a public webserver. It saves the effort from the user side to install a CMS or write an application to just share some information selectively.

Objective A: Enable Basic authentication on /protect-basic directory. Use credentials admin:welcome

Step 1: Check if the protect-basic directory is protected.

Command: curl localhost/protect-basic/flag

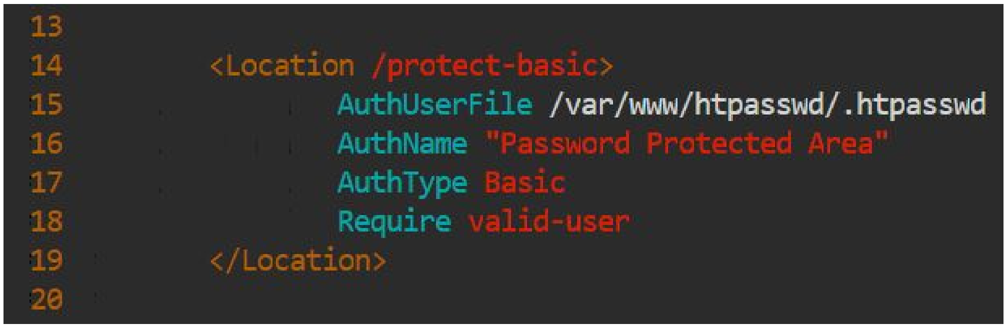


```
root@Apache-Host:~$
root@Apache-Host:~$ curl localhost/protect-basic/flag
979b128bc1ea57e9b5f0db2bb4e96c62
root@Apache-Host:~$
```

Step 2: Open /etc/apache2/sites-available/000-default.conf and add the following lines into <VirtualHost *:80> section

```
<Location /protect-basic>
        AuthUserFile /var/www/htpasswd/.htpasswd
        AuthName "Password Protected Area"
        AuthType Basic
        Require valid-user
</Location>
```

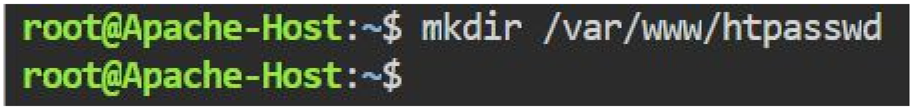
As shown below



```

        <Location /protect-basic>
                AuthUserFile /var/www/htpasswd/.htpasswd
                AuthName "Password Protected Area"
                AuthType Basic
                Require valid-user
        </Location>

```

Step 3: Create the directory

Command: mkdir /var/www/htpasswd

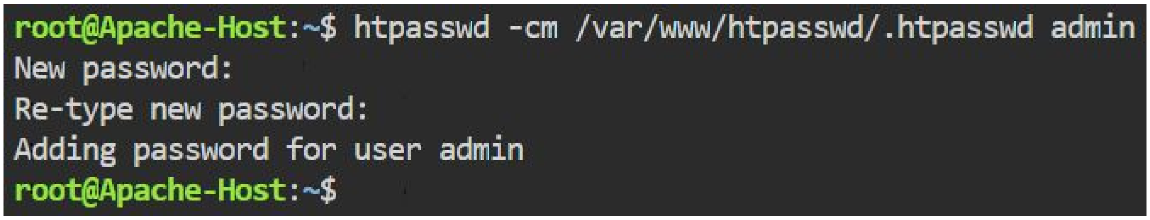


```
root@Apache-Host:~$ mkdir /var/www/htpasswd
root@Apache-Host:~$
```

Step 4: Generate the credential file

Command: htpasswd -cm /var/www/htpasswd/.htpasswd admin

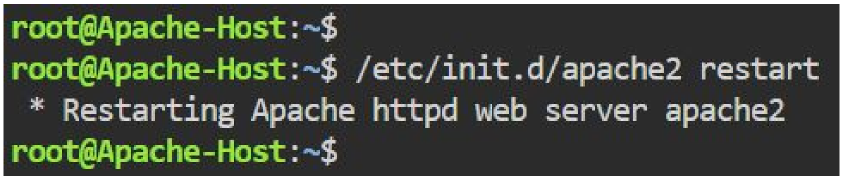
Enter password: welcome



```
root@Apache-Host:~$ htpasswd -cm /var/www/htpasswd/.htpasswd admin
New password:
Re-type new password:
Adding password for user admin
root@Apache-Host:~$
```

Step 5: Restart Apache

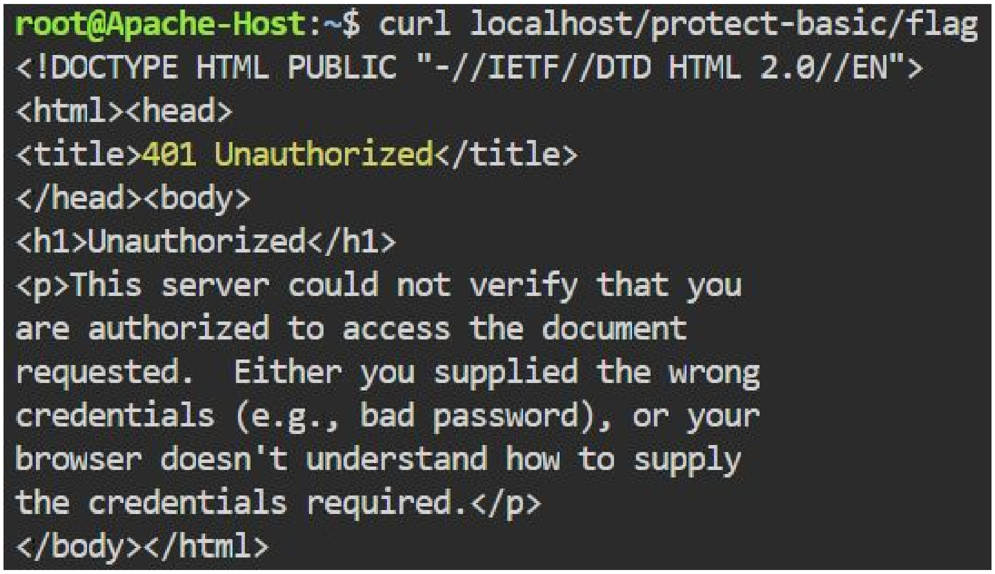
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 6: Check if the content is still accessible without authentication.

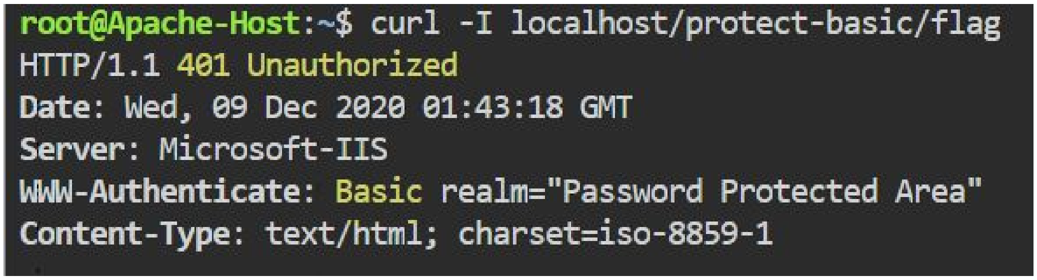
Command: curl localhost/protect-basic/flag



```
root@Apache-Host:~$ curl localhost/protect-basic/flag
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Unauthorized</title>
</head><body>
<h1>Unauthorized</h1>
<p>This server could not verify that you
are authorized to access the document
requested.  Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
</body></html>
```

Check the header to see information about the authentication

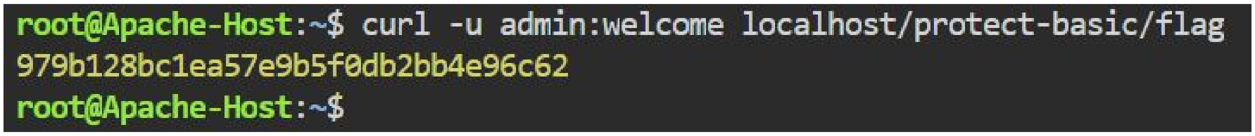
Command: curl -I localhost/protect-basic/flag



```
root@Apache-Host:~$ curl -I localhost/protect-basic/flag
HTTP/1.1 401 Unauthorized
Date: Wed, 09 Dec 2020 01:43:18 GMT
Server: Microsoft-IIS
WWW-Authenticate: Basic realm="Password Protected Area"
Content-Type: text/html; charset=iso-8859-1
```

Try to access the information using configured credentials.

Command: curl -u admin:welcome localhost/protect-basic/flag

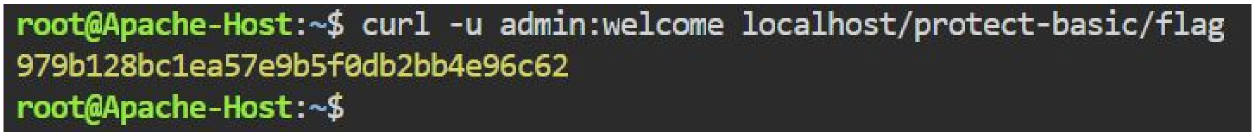


```
root@Apache-Host:~$ curl -u admin:welcome localhost/protect-basic/flag
979b128bc1ea57e9b5f0db2bb4e96c62
root@Apache-Host:~$
```

Objective B: Enable Digest authentication on protect-digest directory. Use credentials admin:thanks

Step 1: Check if the protect-basic directory is protected.

Command: curl localhost/protect-digest

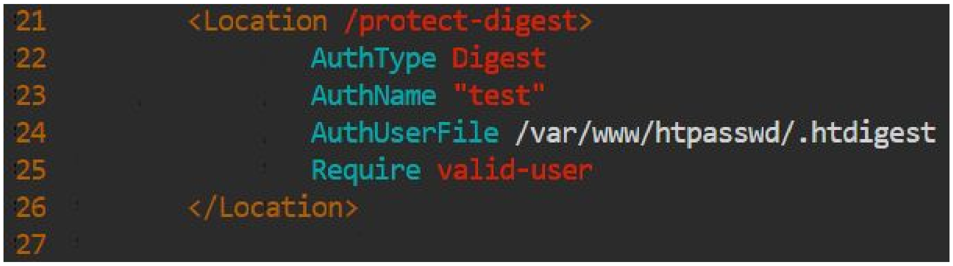


```
root@Apache-Host:~$ curl -u admin:welcome localhost/protect-basic/flag
979b128bc1ea57e9b5f0db2bb4e96c62
root@Apache-Host:~$
```

Step 2: Open /etc/apache2/sites-available/000-default.conf and add the following lines into <VirtualHost *:80> section

```
<Location /protect-digest>
        AuthType Digest
        AuthName "test"
        AuthUserFile /var/www/htpasswd/.htdigest
        Require valid-user
</Location>
```

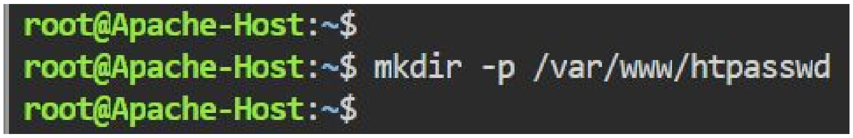
As shown below



```
        <Location /protect-digest>
                AuthType Digest
                AuthName "test"
                AuthUserFile /var/www/htpasswd/.htdigest
                Require valid-user
        </Location>

```

Step 3: Create the directory

Command: mkdir -p /var/www/htpasswd

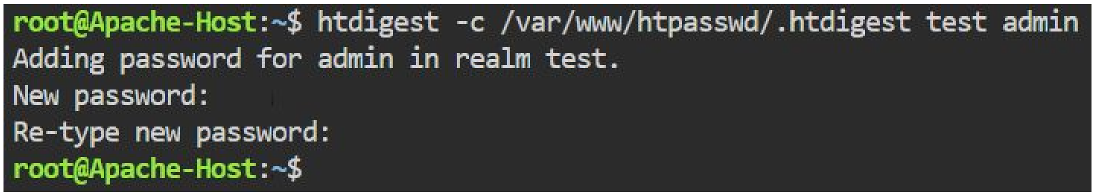


```
root@Apache-Host:~$
root@Apache-Host:~$ mkdir -p /var/www/htpasswd
root@Apache-Host:~$
```

Step 4: Generate the credential file

Command: htdigest -c /var/www/htpasswd/.htdigest test admin

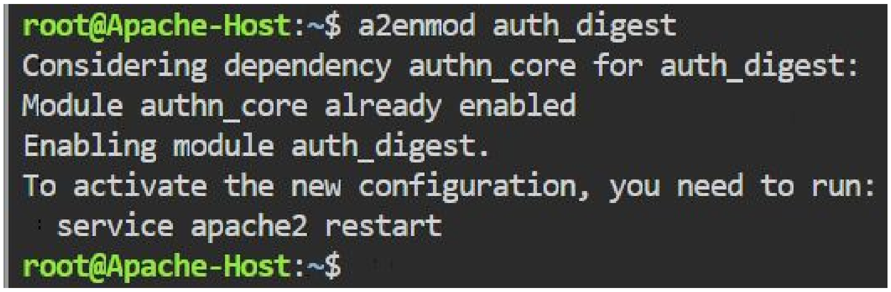
Enter password: thanks



```
root@Apache-Host:~$ htdigest -c /var/www/htpasswd/.htdigest test admin
Adding password for admin in realm test.
New password:
Re-type new password:
root@Apache-Host:~$
```

Step 5: Enable the module

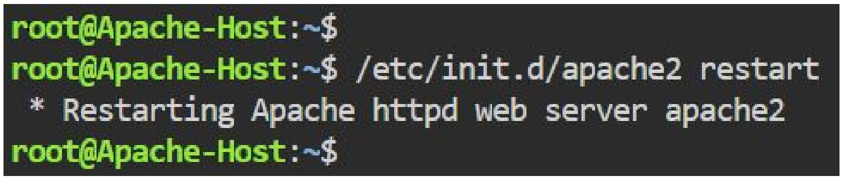
Command: a2enmod auth_digest



```
root@Apache-Host:~$ a2enmod auth_digest
Considering dependency authn_core for auth_digest:
Module authn_core already enabled
Enabling module auth_digest.
To activate the new configuration, you need to run:
  service apache2 restart
root@Apache-Host:~$
```

Step 6: Restart Apache

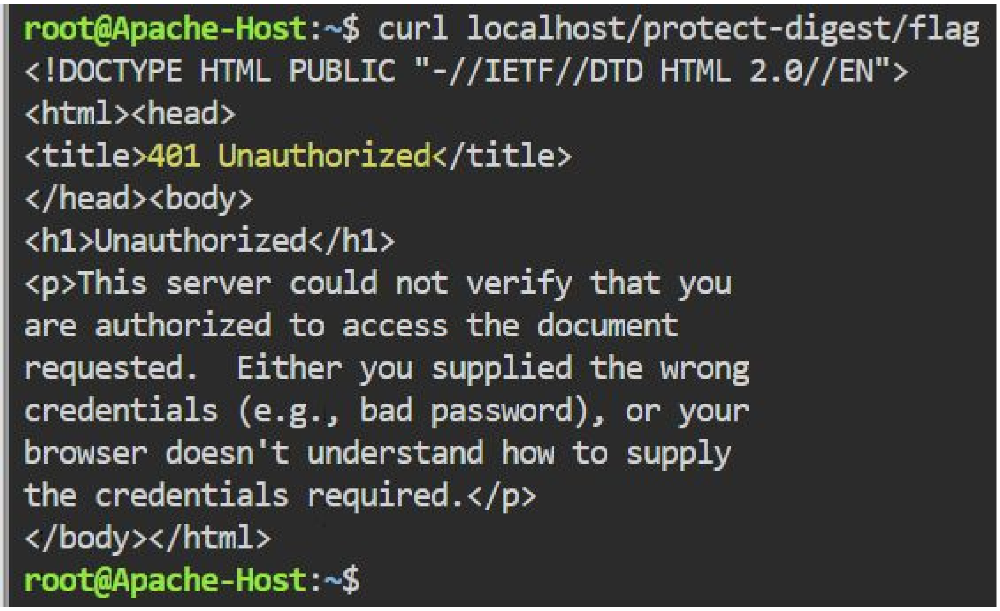
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 7: Check if the content is still accessible without authentication.

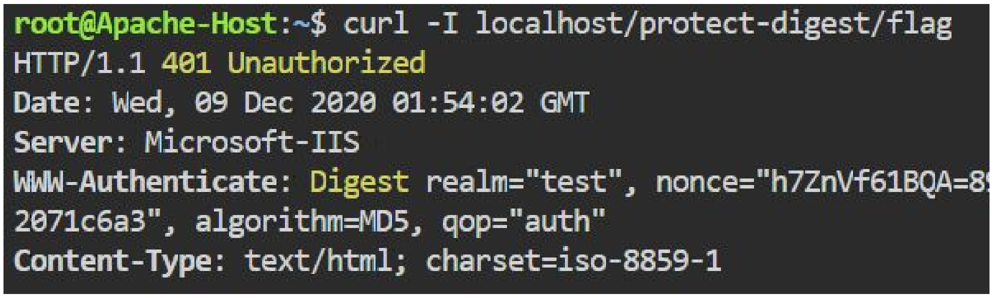
Command: curl localhost/protect-digest/flag



```
root@Apache-Host:~$ curl localhost/protect-digest/flag
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Unauthorized</title>
</head><body>
<h1>Unauthorized</h1>
<p>This server could not verify that you
are authorized to access the document
requested.  Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
</body></html>
root@Apache-Host:~$
```

Check the header to see information about the authentication

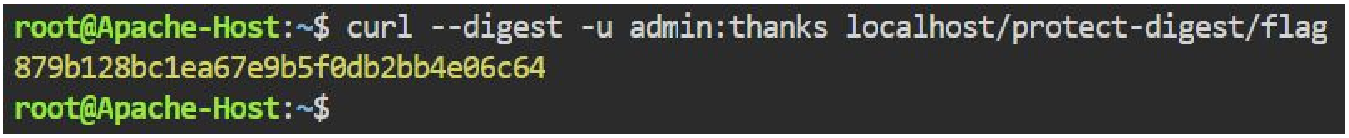
Command: curl -I localhost/protect-digest/flag



```
root@Apache-Host:~$ curl -I localhost/protect-digest/flag
HTTP/1.1 401 Unauthorized
Date: Wed, 09 Dec 2020 01:54:02 GMT
Server: Microsoft-IIS
WWW-Authenticate: Digest realm="test", nonce="h7ZnVf61BQA=8
2071c6a3", algorithm=MD5, qop="auth"
Content-Type: text/html; charset=iso-8859-1
```

Try to access the information using configured credentials.

Command: curl --digest -u admin:thanks localhost/protect-digest/flag



```
root@Apache-Host:~$ curl --digest -u admin:thanks localhost/protect-digest/flag
879b128bc1ea67e9b5f0db2bb4e06c64
root@Apache-Host:~$
```

Test 11: IP Whitelisting

IP whitelisting is used to only allow a set of IP addresses (or IP ranges/subnets) while blocking the remaining. This can be done to decrease the attack surface.

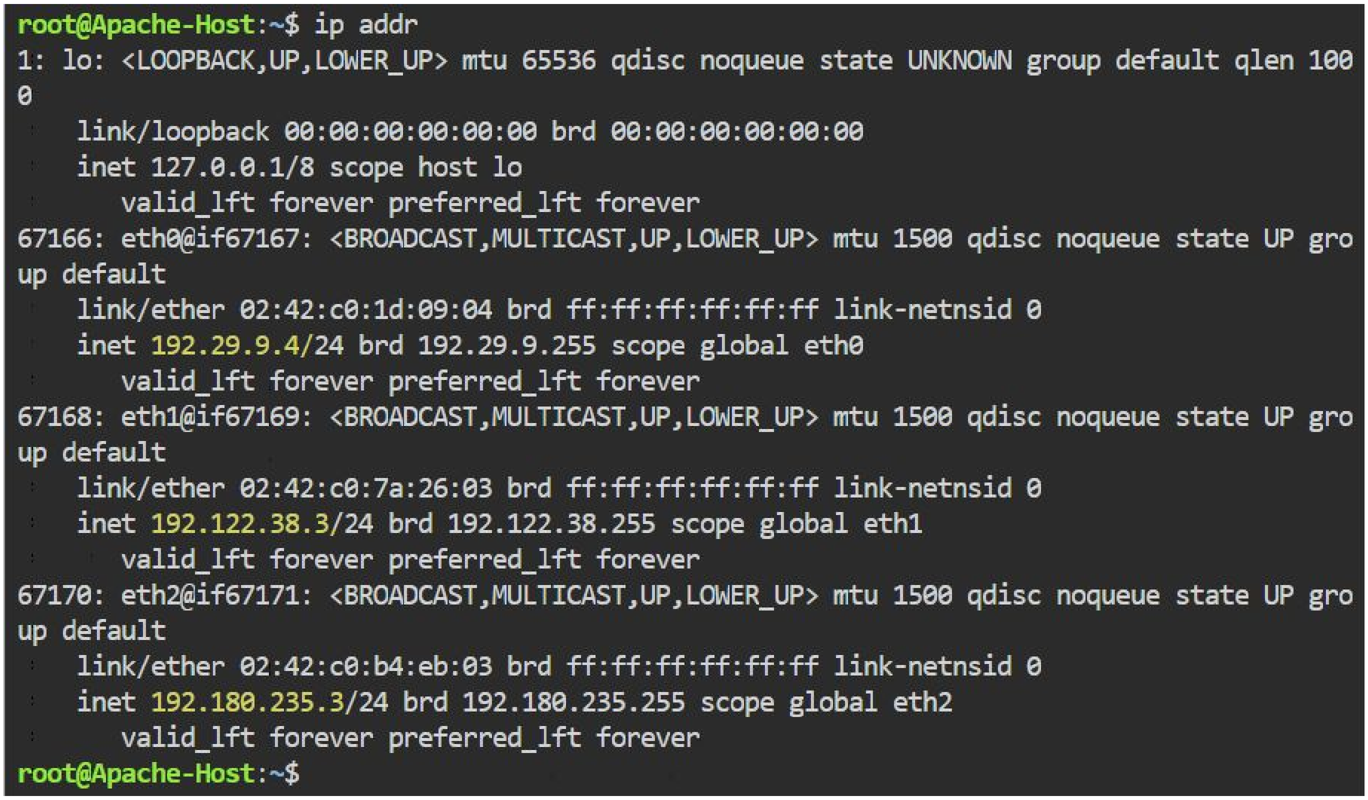
Objective: Disable access from IP range <range3 provided_in_lab> but allow access from <range2 provided_in_lab> and <range1 provided_in_lab>

The <range1>, <range2> and <range3> are dynamically generated on lab creation, so it will be different in your case.

In our case, we have to disable access from IP range 192.180.235.0 but allow access from 192.122.38.0 and 192.29.9.0

Step 1: Check the IP addresses of the Apache server machine.

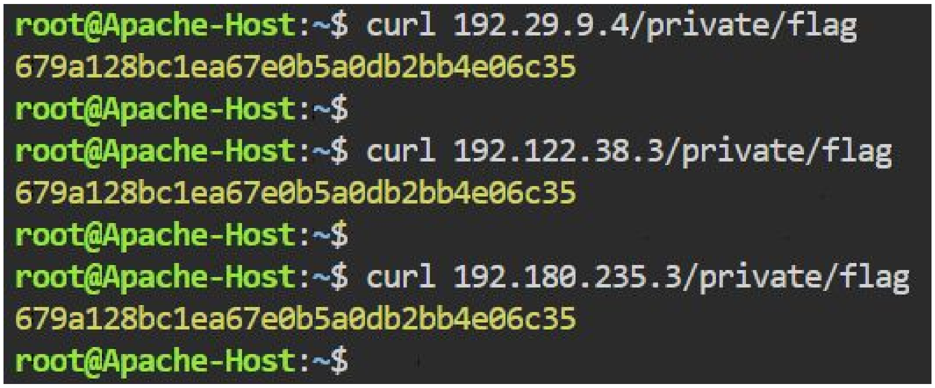
Command: ip addr



```
root@Apache-Host:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 100
0
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
67166: eth0@if67167: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP gro
up default
    link/ether 02:42:c0:1d:09:04 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.29.9.4/24 brd 192.29.9.255 scope global eth0
       valid_lft forever preferred_lft forever
67168: eth1@if67169: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP gro
up default
    link/ether 02:42:c0:7a:26:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.122.38.3/24 brd 192.122.38.255 scope global eth1
       valid_lft forever preferred_lft forever
67170: eth2@if67171: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP gro
up default
    link/ether 02:42:c0:b4:eb:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.180.235.3/24 brd 192.180.235.255 scope global eth2
       valid_lft forever preferred_lft forever
root@Apache-Host:~$
```

Step 2: Check if /private directory is accessible from all three ranges

Commands:
curl 192.29.9.4/private/flag
curl 192.122.38.3/private/flag
curl 192.180.235.3/private/flag

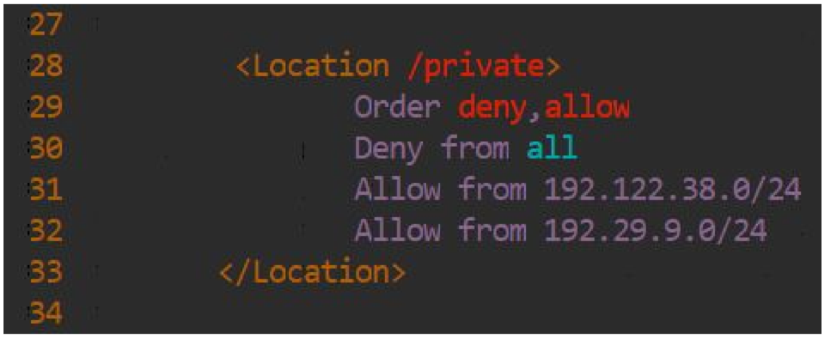


```
root@Apache-Host:~$ curl 192.29.9.4/private/flag
679a128bc1ea67e0b5a0db2bb4e06c35
root@Apache-Host:~$
root@Apache-Host:~$ curl 192.122.38.3/private/flag
679a128bc1ea67e0b5a0db2bb4e06c35
root@Apache-Host:~$
root@Apache-Host:~$ curl 192.180.235.3/private/flag
679a128bc1ea67e0b5a0db2bb4e06c35
root@Apache-Host:~$
```

Step 3: Open /etc/apache2/sites-available/000-default.conf and add the following into <VirtualHost *:80> section

```
 <Location /private>
		Order deny,allow
		Deny from all
		Allow from 192.122.38.0/24
		Allow from 192.29.9.0/24
</Location>
```

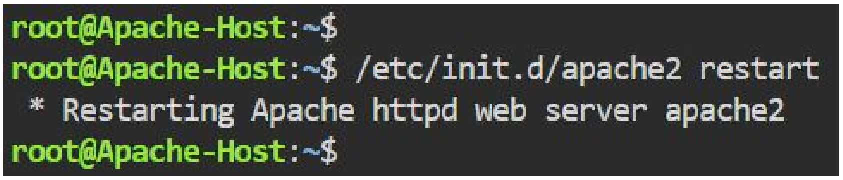
As shown below



```

        <Location /private>
                Order deny,allow
                Deny from all
                Allow from 192.122.38.0/24
                Allow from 192.29.9.0/24
        </Location>

```

Step 4: Restart Apache

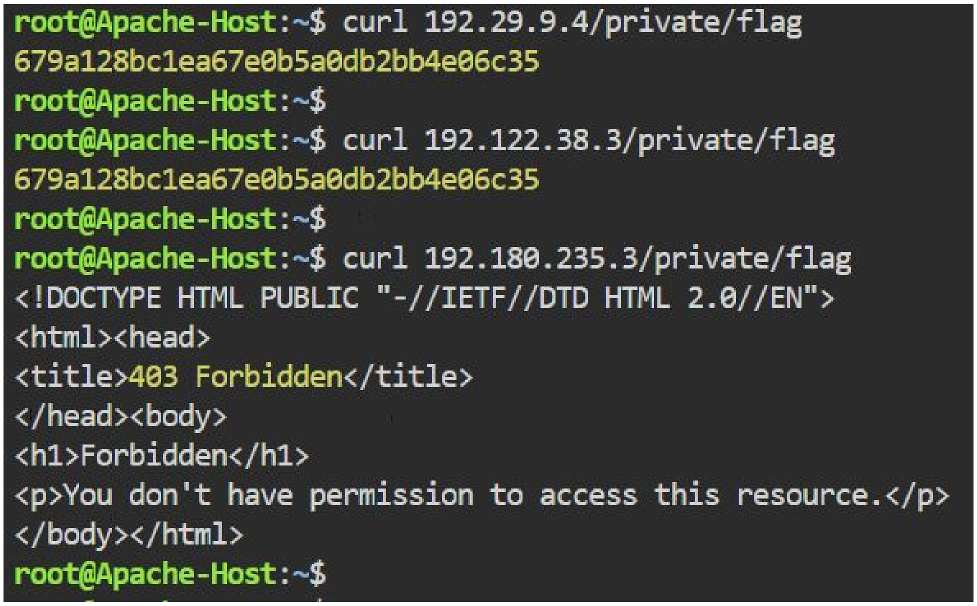
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 5: Verify if the IP whitelisting is in effect

Commands:
curl 192.29.9.4/private/flag
curl 192.122.38.3/private/flag
curl 192.180.235.3/private/flag



```
root@Apache-Host:~$ curl 192.29.9.4/private/flag
679a128bc1ea67e0b5a0db2bb4e06c35
root@Apache-Host:~$
root@Apache-Host:~$ curl 192.122.38.3/private/flag
679a128bc1ea67e0b5a0db2bb4e06c35
root@Apache-Host:~$
root@Apache-Host:~$ curl 192.180.235.3/private/flag
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
root@Apache-Host:~$
```

Test 12: Transport Layer Security

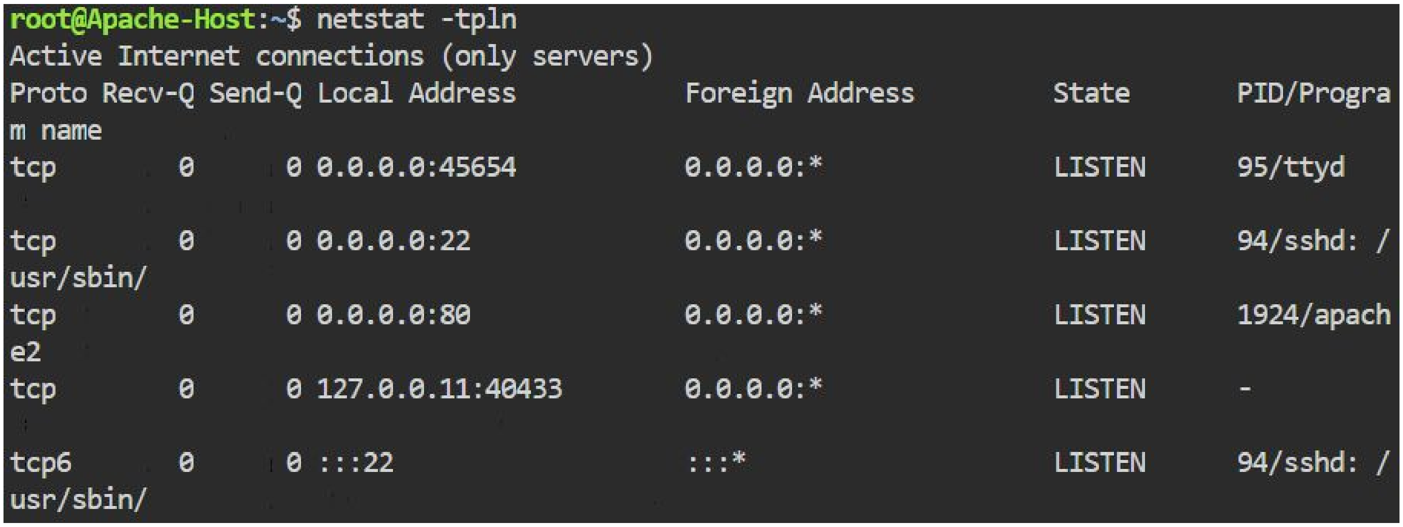
Transport Layer Security (TLS) is used to protect the data against eavesdropping and tempering during transit. Hence, it is important to enabling HTTPS on the webserver.

Objective: Enable TLS for all interactions with the portal

Step 1: Check if TLS is enabled on the server already. This can be done in two different ways

Check if port 443 is open on the localhost

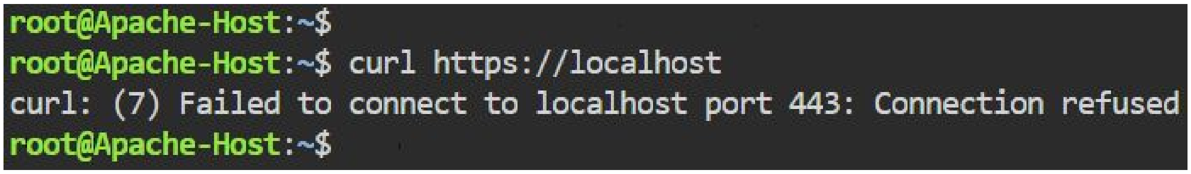
Command: netstat -tpln



```
root@Apache-Host:~$ netstat -tpln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Progra
m name
tcp        0      0 0.0.0.0:45654           0.0.0.0:*               LISTEN      95/ttyd

tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      94/sshd: /
usr/sbin/
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1924/apach
e2
tcp        0      0 127.0.0.11:40433        0.0.0.0:*               LISTEN      -

tcp6       0      0 :::22                   :::*                    LISTEN      94/sshd: /
usr/sbin/
```

Check if the server responds to HTTPS

Command: curl https://localhost



```
root@Apache-Host:~$
root@Apache-Host:~$ curl https://localhost
curl: (7) Failed to connect to localhost port 443: Connection refused
root@Apache-Host:~$
```

HTTPS is not enabled on this webserver.

Step 2: Create a configuration file (if not present already) /etc/apache2/sites-available/default-ssl.conf and add the following content.

```
<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
```

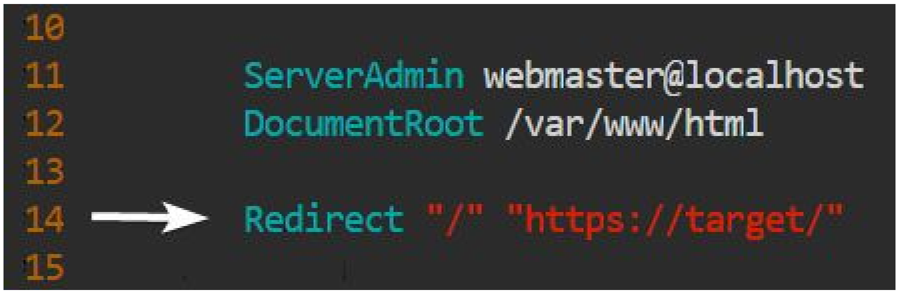
```
                ServerAdmin webmaster@localhost

                DocumentRoot /var/www/html

                # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
                # error, crit, alert, emerg.
                # It is also possible to configure the loglevel for particular
                # modules, e.g.
                #LogLevel info ssl:warn

                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined

                #   SSL Engine Switch:
                #   Enable/Disable SSL for this virtual host.
                SSLEngine on

                #   A self-signed (snakeoil) certificate can be created by installing
                #   the ssl-cert package. See
                #   /usr/share/doc/apache2/README.Debian.gz for more info.
                #   If both key and certificate are stored in the same file, only the
                #   SSLCertificateFile directive is needed.
                SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem
                SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

                #       This enables optimized SSL connection renegotiation handling when SSL
                #       directives are used in per-directory context.
                <FilesMatch "\.(cgi|shtml|phtml|php)$">
                                SSLOptions +StdEnvVars
                </FilesMatch>
        </VirtualHost>
</IfModule>
```

The file is already present in our case.

Step 3: Open the config file /etc/apache2/sites-available/000-default.conf and add the following to

```
Redirect "/" "https://<hostname of server>/"
```

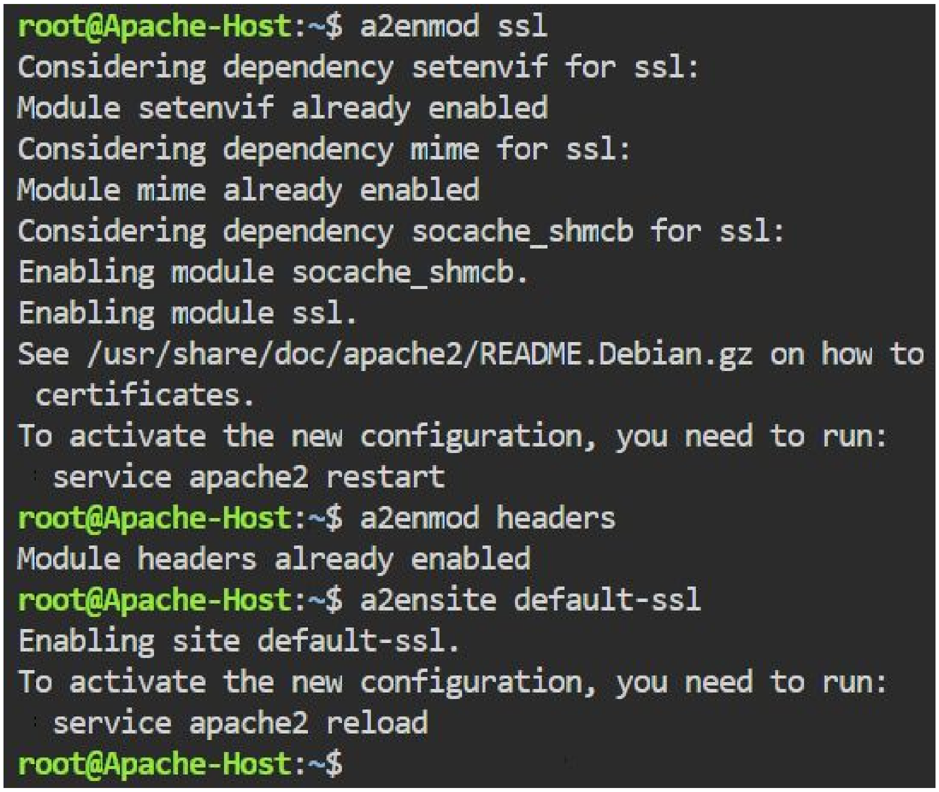
As shown below



```
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        Redirect "/" "https://target/"
```

Step 4: Enable modules to support SSL

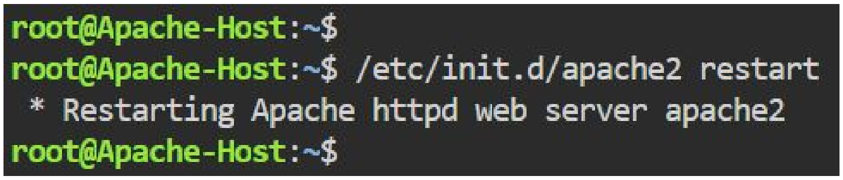
a2enmod ssl
a2enmod headers
a2ensite default-ssl



```
root@Apache-Host:~$ a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to
 certificates.
To activate the new configuration, you need to run:
  service apache2 restart
root@Apache-Host:~$ a2enmod headers
Module headers already enabled
root@Apache-Host:~$ a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  service apache2 reload
root@Apache-Host:~$
```

Step 5: Restart Apache

Command: /etc/init.d/apache2 restart

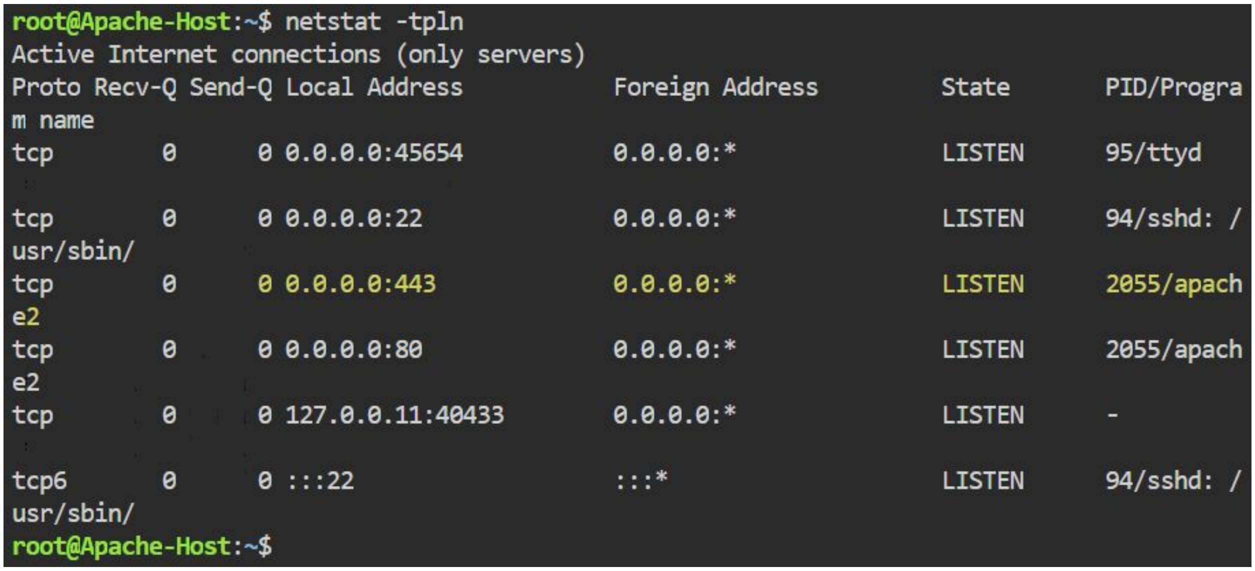


```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 6: Verify if TLS is enabled on the server now.

Check if port 443 is open on the localhost

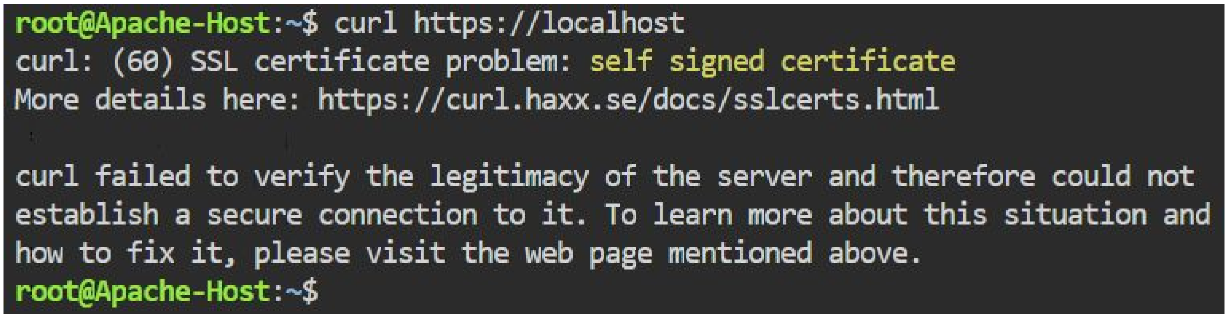
Command: netstat -tpln



```
root@Apache-Host:~$ netstat -tpln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Progra
m name
tcp        0      0 0.0.0.0:45654           0.0.0.0:*               LISTEN      95/ttyd

tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      94/sshd: /
usr/sbin/
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      2055/apach
e2
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      2055/apach
e2
tcp        0      0 127.0.0.11:40433        0.0.0.0:*               LISTEN      -

tcp6       0      0 :::22                   :::*                    LISTEN      94/sshd: /
usr/sbin/
root@Apache-Host:~$
```

Check if the server responds to HTTPS

Command: curl https://localhost



```
root@Apache-Host:~$ curl https://localhost
curl: (60) SSL certificate problem: self signed certificate
More details here: https://curl.haxx.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
root@Apache-Host:~$
```

HTTPS is now enabled on this webserver.

Test 13: Lockdown TLS

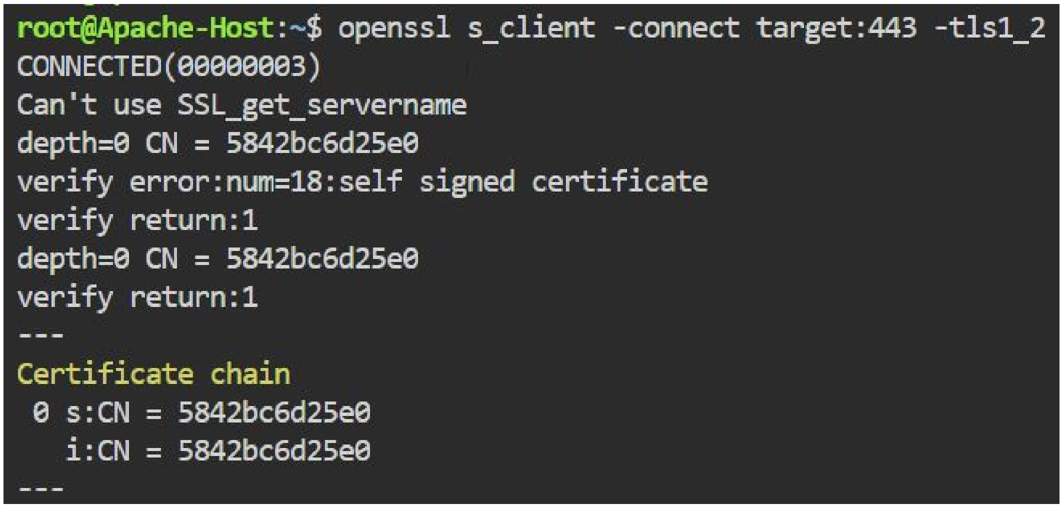
Older versions of SSL/TLS are known to be vulnerable to MiTM and other attacks. Hence it is recommended to disable the older protocols SSLv2, SSLv3, TLSv1.2.

TLSv1.2 is used by most systems and is secure but in this lab, just for the sake of practice, we have asked to disable TLS1.2.

Objective: Disable TLSv1.2 and enable TLSv1.3 on the portal

Step 1: Check if support for TLSv1.2 is enabled.

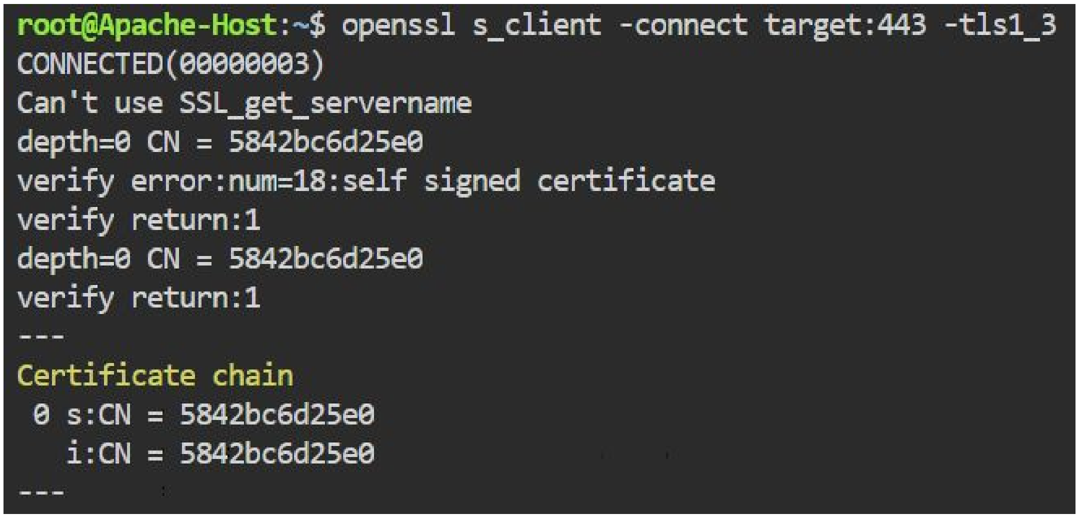
Command: openssl s_client -connect target:443 -tls1_2



```
root@Apache-Host:~$ openssl s_client -connect target:443 -tls1_2
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = 5842bc6d25e0
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = 5842bc6d25e0
verify return:1
---
Certificate chain
 0 s:CN = 5842bc6d25e0
   i:CN = 5842bc6d25e0
---
```

Check if TLSv1.3 is enabled

Command: openssl s_client -connect target:443 -tls1_3



```
root@Apache-Host:~$ openssl s_client -connect target:443 -tls1_3
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = 5842bc6d25e0
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = 5842bc6d25e0
verify return:1
---
Certificate chain
 0 s:CN = 5842bc6d25e0
   i:CN = 5842bc6d25e0
---
```

Step 2: Open /etc/apache2/mods-available/ssl.conf and add following line (if not present only)

Replace

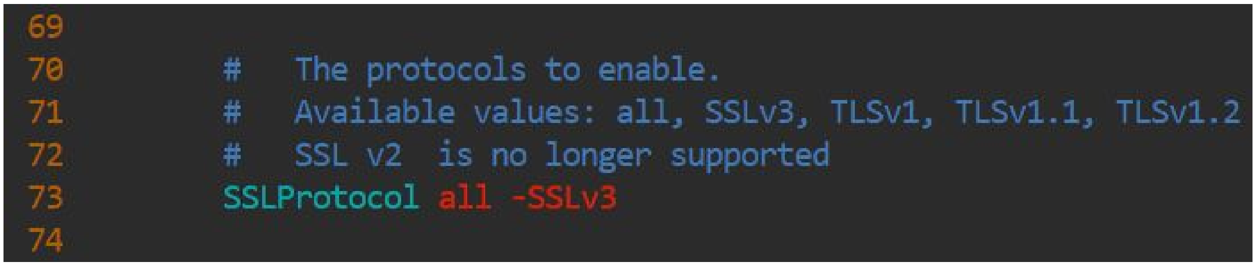
SSLProtocol all -SSLv3

With

SSLProtocol TLSv1.3

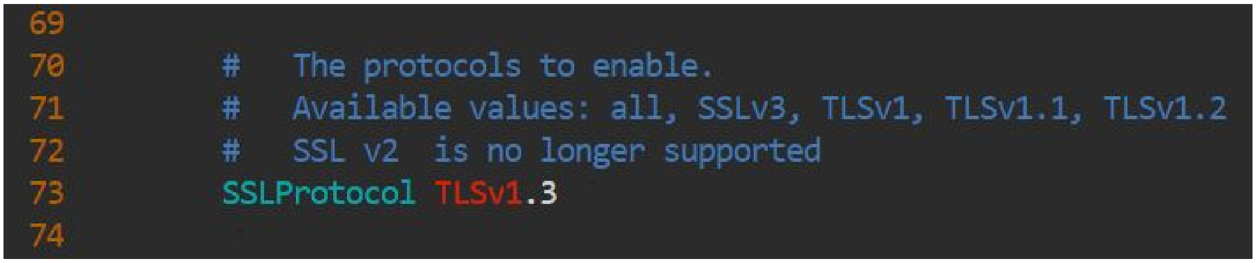
As shown below

Before modification



```

        #   The protocols to enable.
        #   Available values: all, SSLv3, TLSv1, TLSv1.1, TLSv1.2
        #   SSL v2  is no longer supported
        SSLProtocol all -SSLv3

```

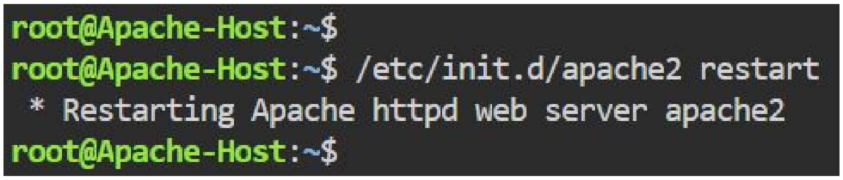
After modification



```

        #   The protocols to enable.
        #   Available values: all, SSLv3, TLSv1, TLSv1.1, TLSv1.2
        #   SSL v2  is no longer supported
        SSLProtocol TLSv1.3

```

Step 3: Restart Apache

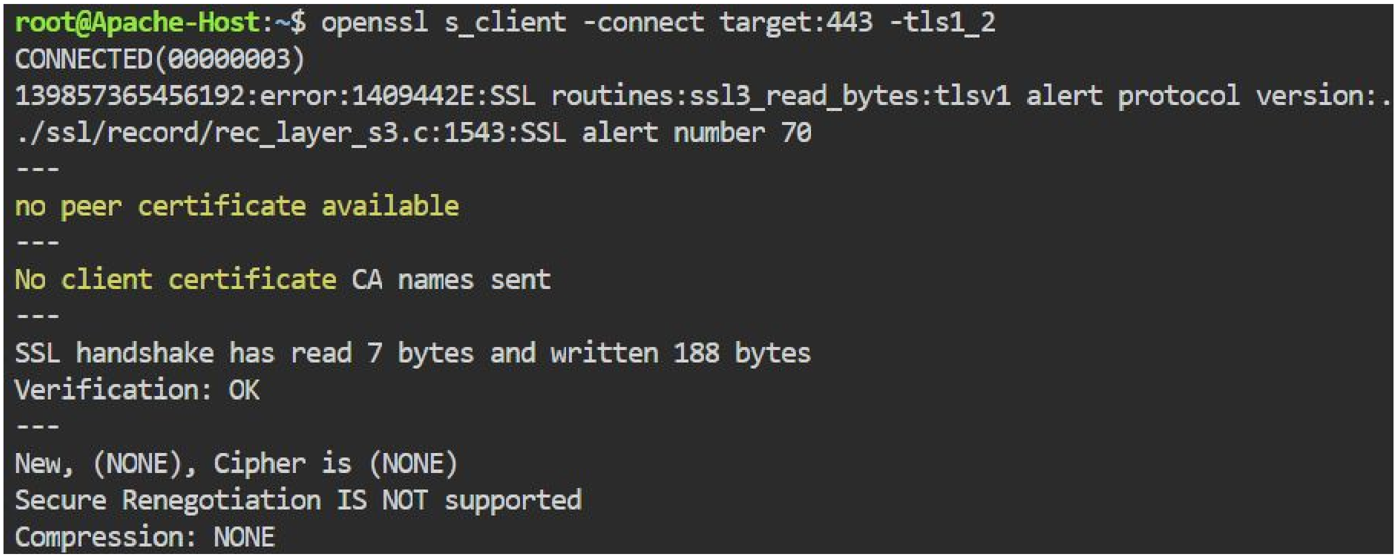
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: Check if support for TLSv1.2 is enabled.

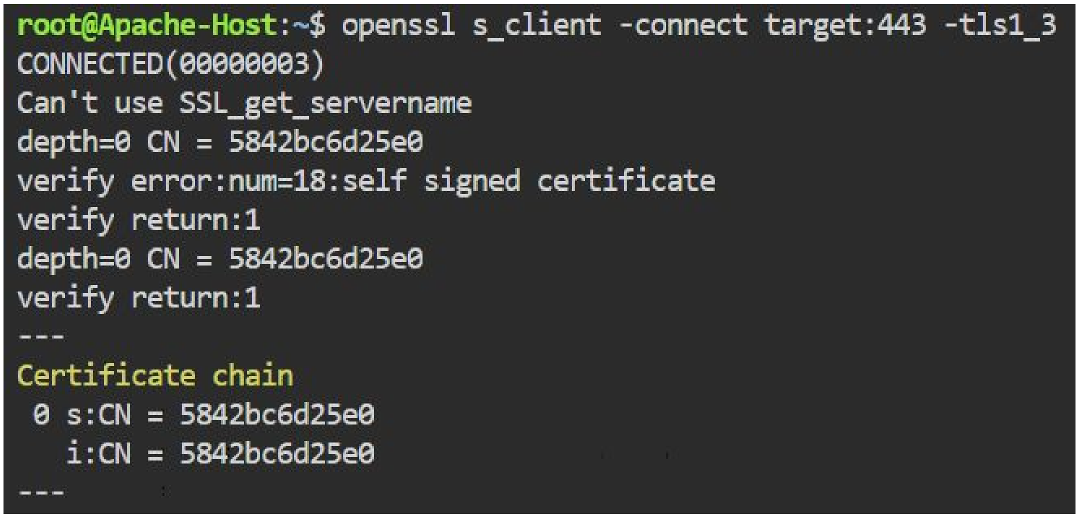
Command: openssl s_client -connect target:443 -tls1_2



```
root@Apache-Host:~$ openssl s_client -connect target:443 -tls1_2
CONNECTED(00000003)
139857365456192:error:1409442E:SSL routines:ssl3_read_bytes:tlsv1 alert protocol version:.
./ssl/record/rec_layer_s3.c:1543:SSL alert number 70
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 7 bytes and written 188 bytes
Verification: OK
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
```

Check if TLSv1.3 is enabled

Command: openssl s_client -connect target:443 -tls1_3



```
root@Apache-Host:~$ openssl s_client -connect target:443 -tls1_3
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = 5842bc6d25e0
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = 5842bc6d25e0
verify return:1
---
Certificate chain
 0 s:CN = 5842bc6d25e0
   i:CN = 5842bc6d25e0
---
```

TLS 1.2 is disabled along with all older versions and TLS 1.3 is enabled.

Test 14: Enhanced Logging

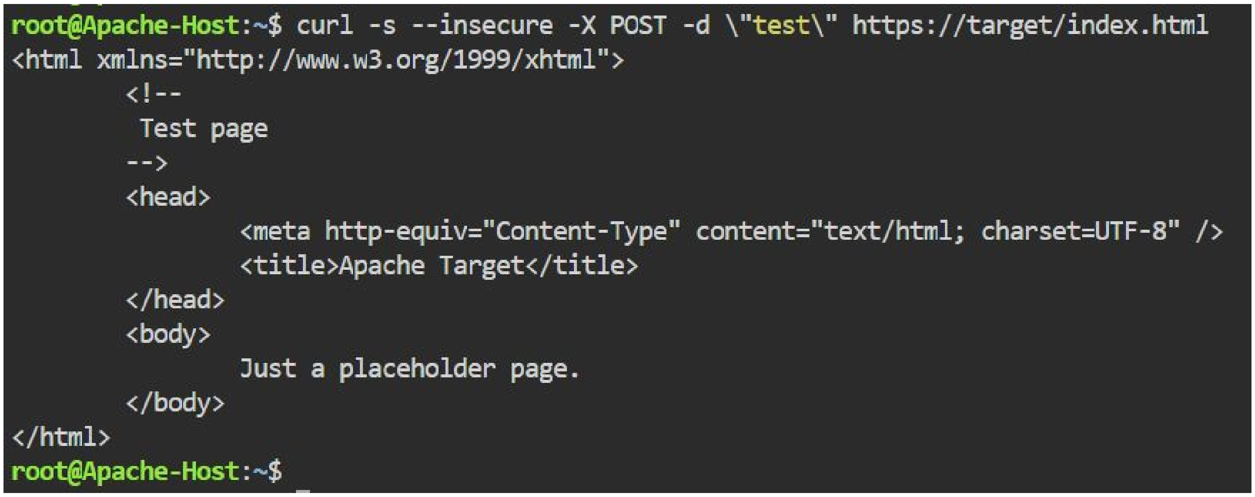
Logs help in detecting malicious activity, incident response and planning to prevent the same incident in the future. Hence, logging more information about the requests can be helpful. However, it can impact server performance.

Objective: Enable POST parameter logging and log those in /var/log/apache2/modsec_audit.log

Step 1: Check if the POST parameters are being logged on the webserver.

Make a dummy POST request on the index page with arguments

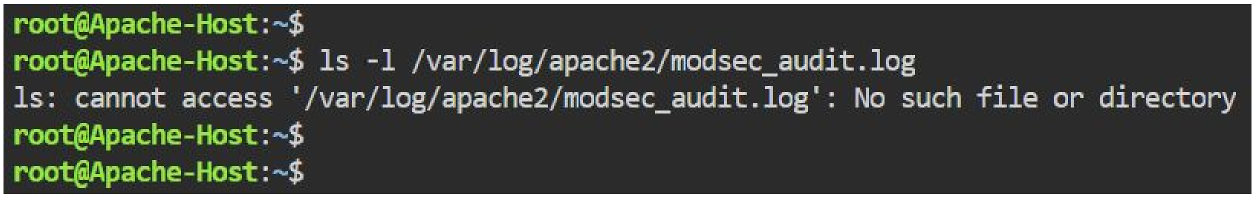
Command: curl -s --insecure -X POST -d \"test\" https://target/index.html



```
root@Apache-Host:~$ curl -s --insecure -X POST -d \"test\" https://target/index.html
<html xmlns="http://www.w3.org/1999/xhtml">
        <!--
         Test page
        -->
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
                <title>Apache Target</title>
        </head>
        <body>
                Just a placeholder page.
        </body>
</html>
root@Apache-Host:~$ _
```

Check of the POST parameters was logged

Command: ls -l /var/log/apache2/modsec_audit.log



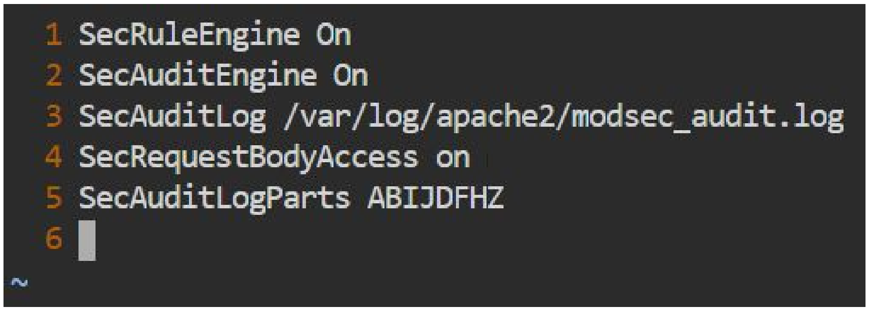
```
root@Apache-Host:~$
root@Apache-Host:~$ ls -l /var/log/apache2/modsec_audit.log
ls: cannot access '/var/log/apache2/modsec_audit.log': No such file or directory
root@Apache-Host:~$
root@Apache-Host:~$
```

This file doesn’t exist.

Step 2: Create /etc/modsecurity/modsecurity.conf file with the following content

SecRuleEngine On
SecAuditEngine On
SecAuditLog /var/log/apache2/modsec_audit.log
SecRequestBodyAccess on
SecAuditLogParts ABIJDFHZ

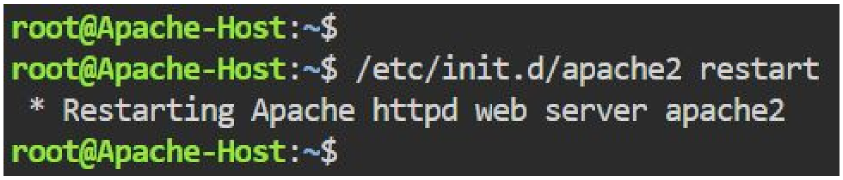
As shown below



```
SecRuleEngine On
SecAuditEngine On
SecAuditLog /var/log/apache2/modsec_audit.log
SecRequestBodyAccess on
SecAuditLogParts ABIJDFHZ

~
```

Step 3: Restart Apache

Command: /etc/init.d/apache2 restart

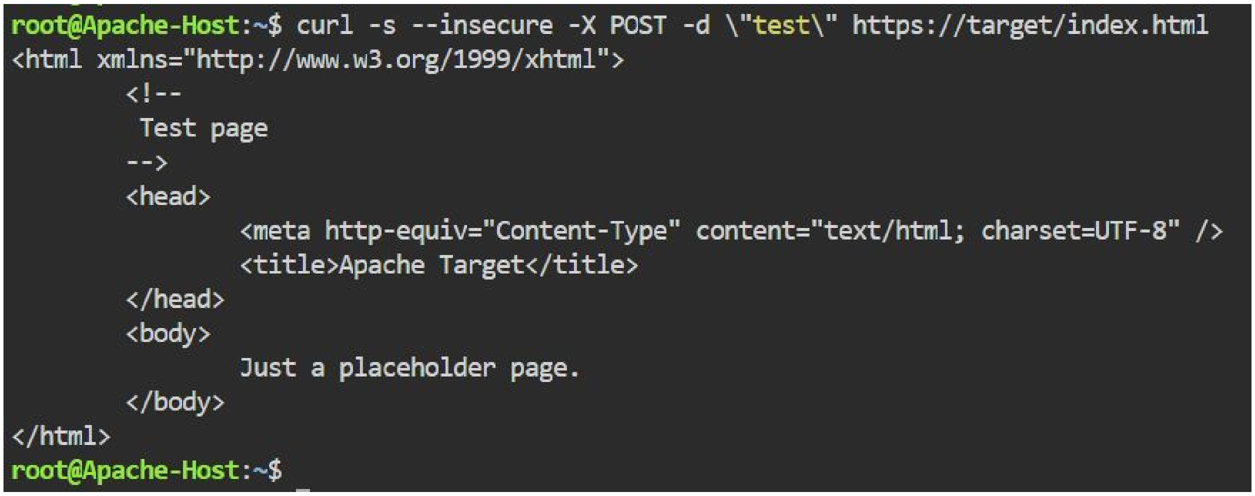


```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: Verify if the argument logging is enabled now

Make a dummy POST request on the index page with arguments

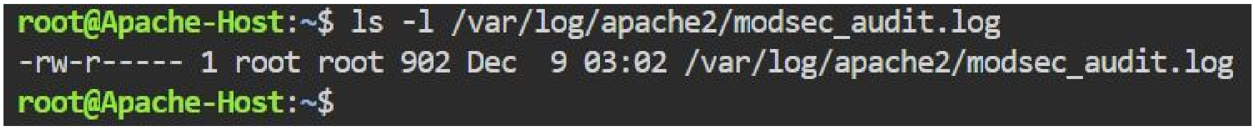
Command: curl -s --insecure -X POST -d \"test\" https://target/index.html



```
root@Apache-Host:~$ curl -s --insecure -X POST -d \"test\" https://target/index.html
<html xmlns="http://www.w3.org/1999/xhtml">
        <!--
         Test page
        -->
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
                <title>Apache Target</title>
        </head>
        <body>
                Just a placeholder page.
        </body>
</html>
root@Apache-Host:~$
```

Check of the POST parameters was logged

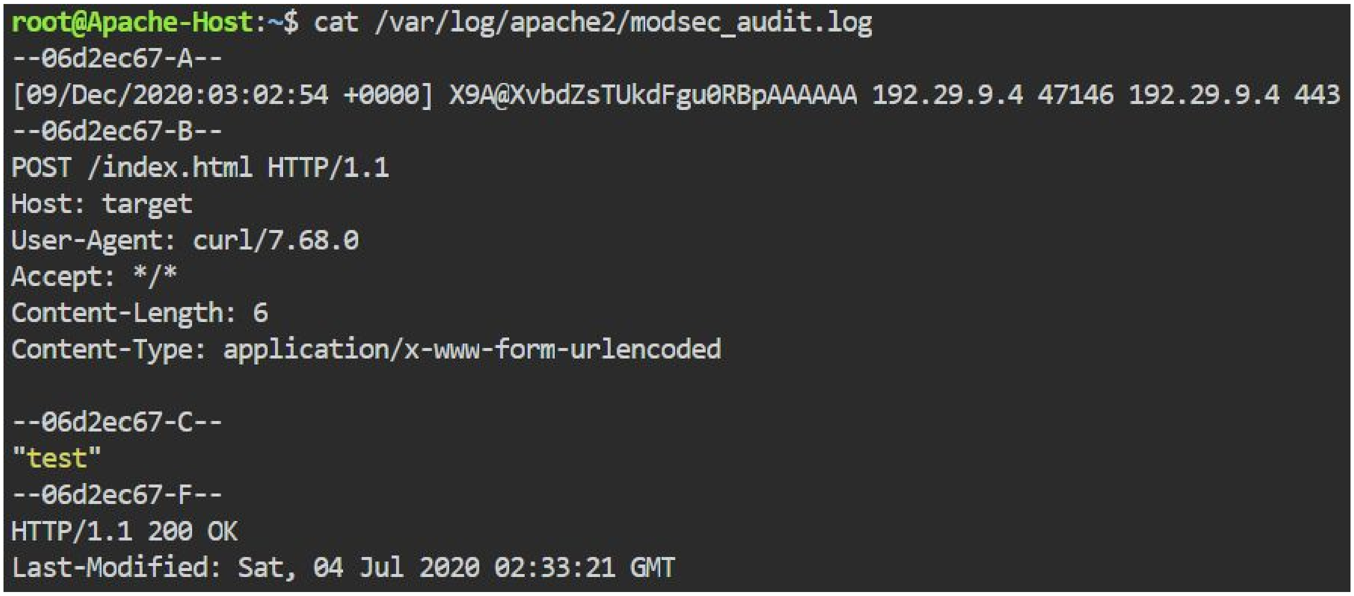
Command: ls -l /var/log/apache2/modsec_audit.log



```
root@Apache-Host:~$ ls -l /var/log/apache2/modsec_audit.log
-rw-r----- 1 root root 902 Dec  9 03:02 /var/log/apache2/modsec_audit.log
root@Apache-Host:~$
```

This file is created. Check the content of the file.

Command: cat /var/log/apache2/modsec_audit.log



```
root@Apache-Host:~$ cat /var/log/apache2/modsec_audit.log
--06d2ec67-A--
[09/Dec/2020:03:02:54 +0000] X9A@XvbdZsTUkdFgu0RBpAAAAAA 192.29.9.4 47146 192.29.9.4 443
--06d2ec67-B--
POST /index.html HTTP/1.1
Host: target
User-Agent: curl/7.68.0
Accept: */*
Content-Length: 6
Content-Type: application/x-www-form-urlencoded

--06d2ec67-C--
"test"
--06d2ec67-F--
HTTP/1.1 200 OK
Last-Modified: Sat, 04 Jul 2020 02:33:21 GMT
```

The dummy payload “test” is logged by the webserver.

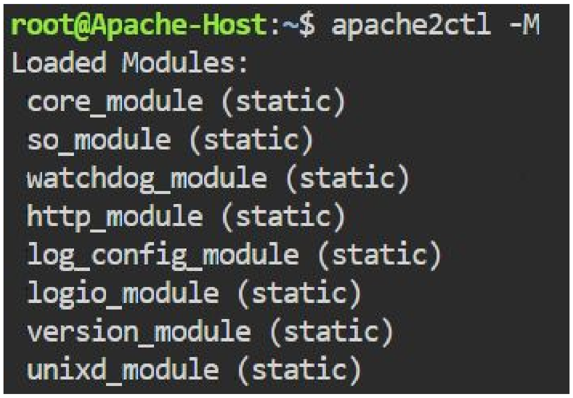
<u>Test 15: Unnecessary Modules</u>

Unnecessary apache modules increase the attack surface area and can also hamper system performance. Hence, it is recommended to remove or disable the non-needed modules.

Objective: Disable the cache module

Step 1: Check the currently loaded modules

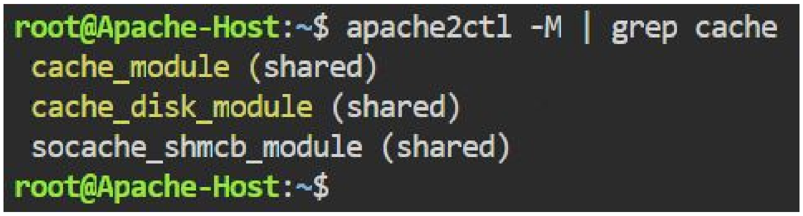
Commands: apache2ctl -M



```
root@Apache-Host:~$ apache2ctl -M
Loaded Modules:
 core_module (static)
 so_module (static)
 watchdog_module (static)
 http_module (static)
 log_config_module (static)
 logio_module (static)
 version_module (static)
 unixd_module (static)
```

Filter for cache-related modules

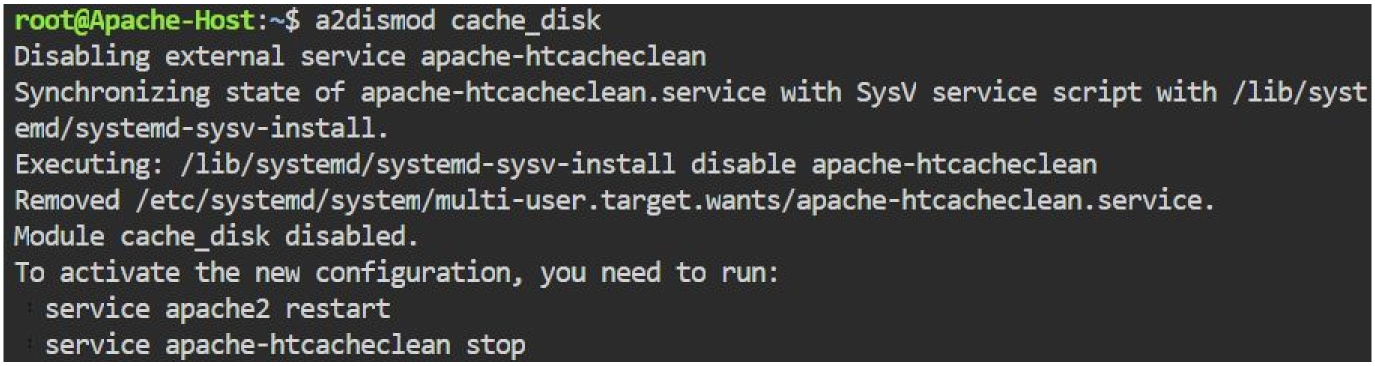
Commands: apache2ctl -M | grep cache



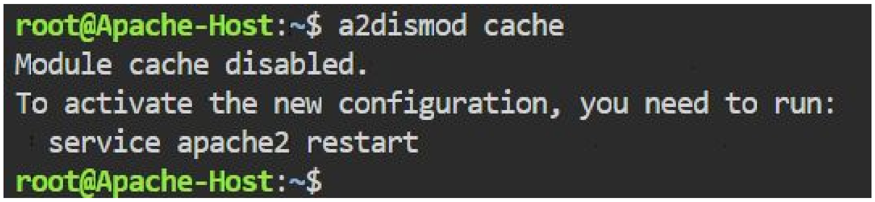
```
root@Apache-Host:~$ apache2ctl -M | grep cache
 cache_module (shared)
 cache_disk_module (shared)
 socache_shmcb_module (shared)
root@Apache-Host:~$
```

Step 2: Disable cache modules

Commands:
a2dismod cache_disk
a2dismod cache



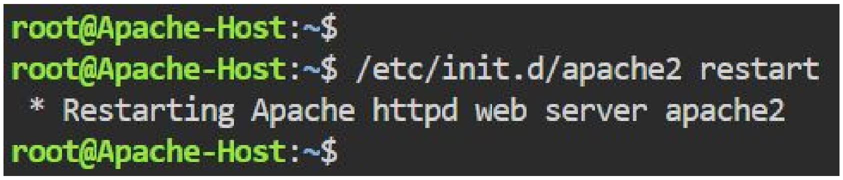
```
root@Apache-Host:~$ a2dismod cache_disk
Disabling external service apache-htcacheclean
Synchronizing state of apache-htcacheclean.service with SysV service script with /lib/syst
emd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable apache-htcacheclean
Removed /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.service.
Module cache_disk disabled.
To activate the new configuration, you need to run:
  service apache2 restart
  service apache-htcacheclean stop
```



```
root@Apache-Host:~$ a2dismod cache
Module cache disabled.
To activate the new configuration, you need to run:
  service apache2 restart
root@Apache-Host:~$
```

Step 3: Restart Apache

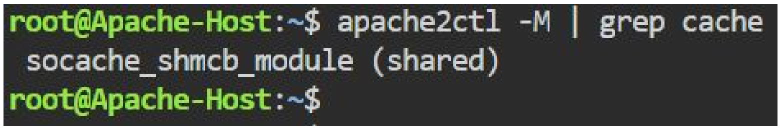
Command: /etc/init.d/apache2 restart



```
root@Apache-Host:~$
root@Apache-Host:~$ /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
root@Apache-Host:~$
```

Step 4: Verify if the modules are disabled

Commands: apache2ctl -M | grep cache



```
root@Apache-Host:~$ apache2ctl -M | grep cache
 socache_shmcb_module (shared)
root@Apache-Host:~$
```

The other two cache-related modules are disabled. This module is required by TLS, so can’t be disabled.

Learning

Fixing different misconfigurations and enabling security settings on Apache webserver.