**.htaccess files.**

We need the **htpasswd** command to somewhat hash our password. Install it by running

$ sudo apt-get update

$ sudo apt-get install **apache2-utils**

We want to set out .htpasswd file inside the apache folder. Our .htaccess file will look at the contents of .htpasswd for authentication.

$sudo htpasswd -c /etc/apache2/.htpasswd **ADMIN\_USER**

* The ADMIN\_USER string can be replaced with whatever user you want. This does not have to be a user already on the system. It can be a random user name.

By default debian/ubuntu has a virtualhost setup already, this is what we will want to edit if we don't have anything setup. Make sure you edit the **actual virtual host file** when you are in the actual competition.

$sudo nano /etc/apache2/sites-enabled/000-default.conf

Find the virtual host block and add the following text that is in red. Also make sure that the **Directory/.htaccess file String** is correctly set to whatever it is you are trying to authenticate with .htaccess.

<VirtualHost \*:80>

ServerAdmin webmaster@localhost

DocumentRoot /var/www/html

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

<Directory "/var/www/html">

AuthType Basic

**AuthName "Restricted Content"**

**AuthUserFile /etc/apache2/.htpasswd**

Require valid-user

</Directory>

</VirtualHost>

Make sure the **Directory /var/www/** block inside **/etc/apache2/apache2.conf** is set to AllowOverride All

<Directory /var/www/>

Options Indexes FollowSymLinks

**AllowOverride All**

Require all granted

</Directory>

Now create the actual .htaccess file and add the following lines in red. Also make sure the path to the .htpasswd is correct.

$ sudo nano /var/www/html/.htaccess

AuthType Basic

AuthName "Restricted Content"

AuthUserFile /etc/apache2/.htpasswd

Require valid-user

Done! Restart the server.

$ service apache2 restart

# 

**SSL Certificates.**

Let’s first get openssl to self sign our certificates

$ apt-get install openssl

Enable the open\_ssl module

$ a2enmod ssl

We need to create a directory to hold our key and cert. Or we can use the default /etc/pki/ssl. folder Whatever you choose to do. Make sure you back it up.

$ mkdir /etc/apache2/ssl

We can create the entire key and cert with one command. It's better to use RSA over des3

$ openssl req -x509 -nodes -days 365 -newkey rsa:4096 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt

Now it will be better to edit our virtual host openssl file then to just use the virtualhost file itself. Let's keep it simple and change everything that is in this file that is in red.

$ sudo nano /etc/apache2/sites-available/default-ssl.conf

<IfModule mod\_ssl.c>

<VirtualHost \_default\_:443>

ServerAdmin admin@example.com

ServerName your\_domain.com

ServerAlias www.your\_domain.com

DocumentRoot /var/www/html

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

SSLEngine on

SSLCertificateFile /etc/apache2/ssl/apache.crt

SSLCertificateKeyFile /etc/apache2/ssl/apache.key

<FilesMatch "\.(cgi|shtml|phtml|php)$">

SSLOptions +StdEnvVars

</FilesMatch>

<Directory /usr/lib/cgi-bin>

SSLOptions +StdEnvVars

</Directory>

BrowserMatch "MSIE [2-6]" \

nokeepalive ssl-unclean-shutdown \

downgrade-1.0 force-response-1.0

BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

</VirtualHost>

</IfModule>

Now we need to activate the “site” and restart! Everything is configured. Make sure to add the .htaccess file to this virtualhost, if you decide to do this. The .htaccess file is only made for the 000-default.conf virtualhost.

$ sudo a2ensite default-ssl.conf

$ sudo service apache2 restart

## 

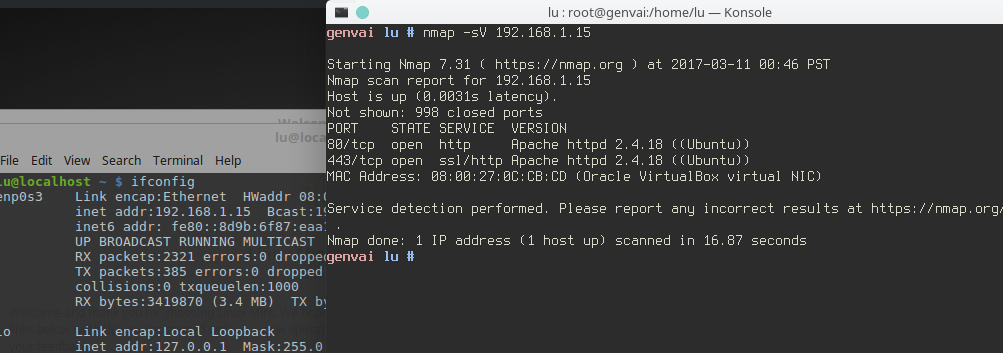
ModSecurity

OWASP ModSecurity Rules

<https://www.netnea.com/cms/apache-tutorial-7_including-modsecurity-core-rules/>

Let’s start by nmaping our target that is hosting multiple services. We are going to try and disguise these services to make out server look like windows. In my case i'm scanning my virtualbox on my local machine.

$ sudo nmap -sV 192.x.x.x



As you can see the nmap scan returns the proper services with the proper machine name.

Let’s install mod\_security

$ sudo apt-get install libapache-mod-security Ubuntu=libapache-mod-security2

$ sudo a2enmod mod-security

$ sudo a2enmod security2

Now lets edit the default mod security file and change some lines.

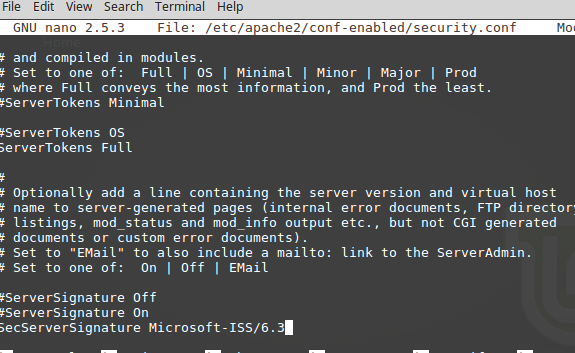
**sudo nano /etc/apache2/conf-enabled/security.conf**

#ServerTokens OS

ServerTokens Full

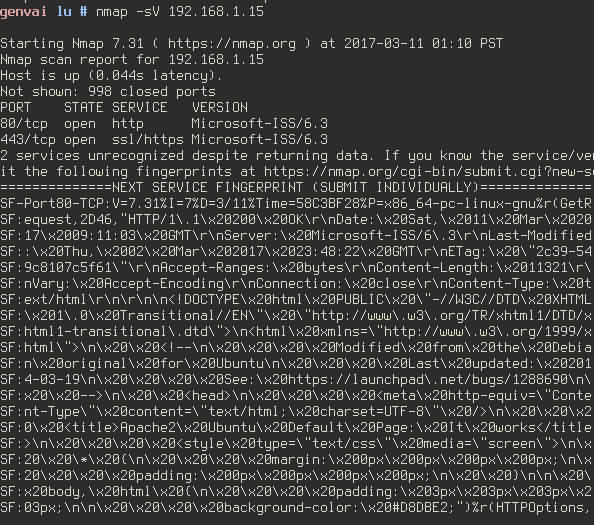
#ServerSigniture On

SecServerSignature Microsoft-IIS/6.3



Let's restart and run another nmap scan

$ sudo service apache2 restart



At this point Nmap does not know what the heck its doing and our ubuntu box is coming up as a windows server. Mission complete.

Disable SSHD BAnner on debian. This will make an nmap scan say nothing.

sudo nano /etc/ssh/sshd\_config

DebianBanner no

**1. Install ModSecurity for Apache**

**yum install mod\_security**

2. Ensure that ModSecurity is loading correctly by checking **/var/log/mod\_security/error.log**

at startup for lines indicating ModSecurity is installed. An example

might appear as follows:

**```ModSecurity for Apache/2.9.1 (http://www.modsecurity.org/) configured.```**

3. The most common method of deploying ModSecurity we have seen is

to create a new folder underneath the Apache directory (typically

**/usr/local/apache/, /etc/httpd/, or /etc/apache2**). Often this folder

is called '**modsecurity.d**'. Create this folder and cd into it.

4. Clone the repository into the modsecurity.d folder using:

```**git clone https://github.com/SpiderLabs/owasp-modsecurity-crs** .```

This will create a new owasp-modsecurity-crs folder.

5. Move the **crs-setup.conf.example file to crs-setup.conf.**

Please take the time to go through this file and customize the settings

for your local environment. Failure to do so may result in false

negatives and false positives. See the section entitled OWASP CRS

Configuration for more detail.

6. **Rename rules/REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf.example and**

**rules/RESPONSE-999-EXCLUSION-RULES-AFTER-CRS.conf.example to remove the**

**'.example' extension**. This will allow you to add exclusions without updates

overwriting them in the future.

7. Add the following line to your httpd.conf/apache2.conf (the following

assumes you've cloned CRS into modsecurity.d/owasp-modsecurity-crs). You

can alternatively place these in any config file included by Apache:

```

**<IfModule security2\_module>**

**Include modsecurity.d/owasp-modsecurity-crs/crs-setup.conf**

**Include modsecurity.d/owasp-modsecurity-crs/rules/\*.conf**

**</IfModule>**

```

8. Restart web server and ensure it starts without errors

9. Make sure your web sites are still running fine.

10. Proceed to the section "Testing the Installation" below.

## Securing Apache Indexes.