
Certificado profesional en seguridad informática
Módulo 2: Auditoría de seguridad informática

Examen práctico

**PROTECCIÓN DE PÁGINAS WEB CON
MODSECURITY**

Miquel Rodríguez González

10 de mayo de 2024



Qué es Modsecurity

ModSecurity es un firewall de aplicaciones web (WAF) de código abierto. Fue diseñado para monitorear, registrar y filtrar tráfico HTTP en tiempo real con el objetivo de proteger aplicaciones web de ataques y amenazas. En este caso será implementado a través del servidor web Apache y utilizaremos las normas de OWASP

Índice

1- Preparación MariaDB y Apache2.....	2
1.1- Comprobación de Apache2 y MariaDB.....	2
1.2- Configuración de MariaDB.....	4
2- Documento vuln.conf e inyección de código.....	7
3- Instalación y configuración de modsecurity.....	10
4- Implementación de OWAS en modsecurity.....	13
5- Configuraciones de seguridad en Apache2.....	18
5.1- Ocultar el banner address.....	18
5.2- Creación de reglas de bloqueo específicas.....	20
5.3- Búsqueda de logs.....	22
6- Anexo.....	23
6.1- Contenido vuln.php:.....	23

1- Preparación MariaDB y Apache2

1.1- Comprobación de Apache2 y MariaDB

Lo primero de todo es comprobar si tenemos MariaDB y apache instalado en nuestro equipo. Intentamos iniciar apache 2 con **systemctl start apache2** y comprobamos que esté activo con **systemctl status apache2** (Si no está instalado hacemos un **apt get install apache2**)

```
(root@kali)-[/home/kali]
# systemctl start apache2

(root@kali)-[/home/kali]
# systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; vendor preset: enabled)
   Active: active (running) since Tue 2024-05-07 09:28:03 UTC; 1min 1s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 15091 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 15108 (apache2)
    Tasks: 6 (limit: 4610)
   Memory: 19.8M (peak: 20.3M)
      CPU: 99ms
   CGroup: /system.slice/apache2.service
           └─15108 /usr/sbin/apache2 -k start
           └─15113 /usr/sbin/apache2 -k start
           └─15114 /usr/sbin/apache2 -k start
           └─15115 /usr/sbin/apache2 -k start
           └─15116 /usr/sbin/apache2 -k start
           └─15117 /usr/sbin/apache2 -k start

May 07 09:28:03 kali systemd[1]: Starting apache2.service: The Apache HTTP Server.
May 07 09:28:03 kali apachectl[15107]: AH00558: httpd: could not open error log /usr/local/apache2/logs/error.log: Permission denied
May 07 09:28:03 kali systemd[1]: Started apache2.service: The Apache HTTP Server.
```

Hacemos el mismo proceso con MariaDB con los comandos de **systemctl start mariadb** para iniciar y **systemctl status mariadb** para comprobar que está activo (Si no está instalado hacemos un **apt get install mariadb**)

```
(root@kali)-[/home/kali]
# systemctl start mariadb

(root@kali)-[/home/kali]
# systemctl status mariadb
● mariadb.service - MariaDB 10.11.6 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; vendor preset: enabled)
   Active: active (running) since Tue 2024-05-07 09:33:31 UTC; 1min 1s ago
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 17969 ExecStartPre=/usr/bin/install -m 755 /etc/mysql/debian-start (code=0)
   Process: 17971 ExecStartPre=/bin/sh -c systemctl --no-pager status mariadb.service (code=0)
   Process: 17973 ExecStartPre=/bin/sh -c [ ! -e /usr/lib/mysql ] && /usr/bin/mysqld_safe --mysql-libdir=/usr/lib/mysql --mysql-data-dir=/usr/lib/mysql --mysql-log-error=/usr/lib/mysql/mariadb.log --mysql-error-log=/usr/lib/mysql/mariadb.log --mysql-user=mysql --mysql-test-dir=/usr/lib/mysql/test --mysql-test-libdir=/usr/lib/mysql/test/lib (code=0)
   Process: 18054 ExecStartPost=/bin/sh -c systemctl --no-pager status mariadb.service (code=0)
   Process: 18056 ExecStartPost=/etc/mysql/debian-start (code=0)
  Main PID: 18034 (mariabdd)
    Status: "Taking your SQL requests now..."
     Tasks: 12 (limit: 4610)
  Memory: 206.4M (peak: 209.8M)
     CPU: 678ms
    CGroup: /system.slice/mariadb.service
            └─18034 /usr/sbin/mariabdd

Almacenamiento (usado...):
May 07 09:33:31 kali mariabdd[18034]: 2024-05-07  9:33:31 [Warning] InnoDB: Buffer pool(s) memory resized from 128M to 209.8M
May 07 09:33:31 kali mariabdd[18034]: 2024-05-07  9:33:31 [Warning] InnoDB: Buffer pool(s) memory resized from 128M to 209.8M
May 07 09:33:31 kali mariabdd[18034]: 2024-05-07  9:33:31 [Warning] InnoDB: Buffer pool(s) memory resized from 128M to 209.8M
May 07 09:33:31 kali mariabdd[18034]: 2024-05-07  9:33:31 [Warning] InnoDB: Buffer pool(s) memory resized from 128M to 209.8M
May 07 09:33:31 kali mariabdd[18034]: 2024-05-07  9:33:31 [Warning] InnoDB: Buffer pool(s) memory resized from 128M to 209.8M
May 07 09:33:31 kali mariabdd[18034]: 2024-05-07  9:33:31 [Warning] InnoDB: Buffer pool(s) memory resized from 128M to 209.8M
May 07 09:33:31 kali mariabdd[18034]: Version: '10.11.6-MariaDB'  source: https://mariadb.com
May 07 09:33:31 kali systemd[1]: Started mariadb.service: MariaDB 10.11.6 database server.
May 07 09:33:31 kali /etc/mysql/debian-start[18059]: 2024-05-07  9:33:31 [Warning] InnoDB: Buffer pool(s) memory resized from 128M to 209.8M
May 07 09:33:31 kali /etc/mysql/debian-start[18071]: 2024-05-07  9:33:31 [Warning] InnoDB: Buffer pool(s) memory resized from 128M to 209.8M
lines 1-28/28 (END)
```

1.2- Configuración de MariaDB

Entraremos a MariaDB con **mariadb -u root -p**, nos pedirá crear una contraseña si es la primera vez.

```
(root@kali)-[/var/www/html]
# mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 34
Server version: 10.11.6-MariaDB-2 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> █
```

Después crearemos el usuario **root** con el comando **ALTER USER 'root'@'localhost' IDENTIFIED BY '123456';**

```
MariaDB [(none)]> ALTER USER 'root'@'localhost' IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.012 sec)

MariaDB [(none)]> █
```

Con **show databases** podemos ver las bases de datos actuales en MariaDB

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.010 sec)
```

Para crear nuestra BBDD con el nombre **dfir**, debemos hacer un **create database dfir;**

```
MariaDB [(none)]> create database dfir;  
Query OK, 1 row affected (0.000 sec)
```

Comprobamos que la tabla se ha creado

```
MariaDB [(none)]> show databases;  
+-----+  
| Database |  
+-----+  
| dfir      |  
| information_schema |  
| mysql     |  
| performance_schema |  
| sys       |  
+-----+  
5 rows in set (0.000 sec)
```

Para crear los distintos usuarios en nuestra BBDD, primero debemos acceder a ella con **connect dfir;** (o **use dfir;**)

```
MariaDB [(none)]> connect dfir;  
Connection id: 36  
Current database: dfir  
  
MariaDB [dfir]> █
```

Una vez dentro crearemos la tabla de usuarios con una columna de identificación de usuario y otra para la contraseña. Utilizaremos **CREATE TABLE usuarios (userid VARCHAR(100), password VARCHAR(100));**

```
MariaDB [dfir]> CREATE TABLE usuarios (userid VARCHAR(100), password VARCHAR(100));  
Query OK, 0 rows affected (0.019 sec)
```

Comprobamos que se ha creado la tabla con **show tables;**

```
MariaDB [dfir]> show tables;  
+-----+  
| Tables_in_dfir |  
+-----+  
| usuarios        |  
+-----+  
1 row in set (0.000 sec)
```

Con esto ya podemos empezar a crear distintos usuarios a través de **insert into usuarios values ('dfir1','123456');**

```
MariaDB [dfir]> insert into usuarios values('dfir1','123456');  
Query OK, 1 row affected (0.014 sec)
```

```
MariaDB [dfir]> insert into usuarios values ('dfir2','1234567');  
Query OK, 1 row affected (0.010 sec)
```

```
MariaDB [dfir]> insert into usuarios values ('dfir3','12345678');  
Query OK, 1 row affected (0.002 sec)
```

Comprobamos que se han creado los usuarios con **select * from usuarios**

```
MariaDB [dfir]> select * from usuarios;  
+-----+-----+  
| userid | password |  
+-----+-----+  
| dfir1  | 123456   |  
| dfir2  | 1234567  |  
| dfir3  | 12345678 |  
+-----+-----+  
3 rows in set (0.000 sec)
```

2- Documento vuln.conf e inyección de código

Lo primero de todo es entrar a la ruta `/var/www/html` en la cual crearemos el archivo `vuln.php` (`nano vuln.php`) con el código de nuestra página web. [contenido de vuln.php](#)

```
(root@kali)-[/home/kali]
# cd ..

(root@kali)-[/home]
# cd ..

(root@kali)-[/]
# cd var/www

(root@kali)-[/var/www]
# ls
html

(root@kali)-[/var/www]
# cd html

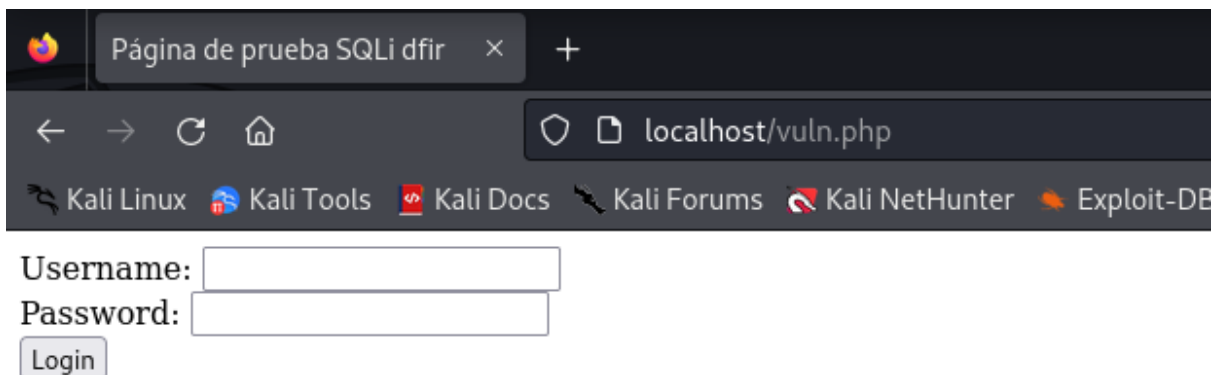
(root@kali)-[/var/www/html]
# nano vuln.php.txt

(root@kali)-[/var/www/html]
```



```
1<html>
2<head>
3<title>
4Página de prueba SQLi dfir
5</title>
6</head>
7<body>
8<?php
9    if(isset($_POST['login']))
10    {
11        $username = $_POST['username'];
12        $password = $_POST['password'];
13        $con = mysqli_connect('localhost','root','123456','dfir');
14        $result = mysqli_query($con, "SELECT * FROM `usuarios` WHERE userid='$username' AND
password='$password'");
15        if(mysqli_num_rows($result) == 0)
16            echo 'Usuario o Password Incorrecto, prueba otra vez';
17        else
18            echo '<h1>Dentro!!!</h1><p>Este texto lo ven sólo aquellos que han hecho un login correcto.</p>';
19    }
20    else
21    {
22        ?>
23        <form action="" method="post">
24            Username: <input type="text" name="username" /><br />
25            Password: <input type="password" name="password" /><br />
26            <input type="submit" name="login" value="Login" />
27        </form>
28    <?php
29    }
30    ?>
31</body>
32</html>
33
```

Una vez con esto creado podemos acceder a un navegador introduciendo la URL **localhost/vuln.php**

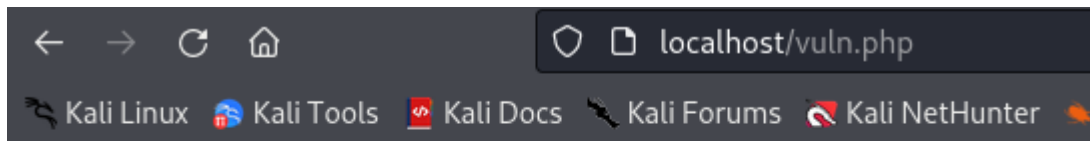


Con el usuario y contraseñas correctas podemos acceder dentro de la página de “bienvenida”

Username:

Password:

Login



Dentro!!!

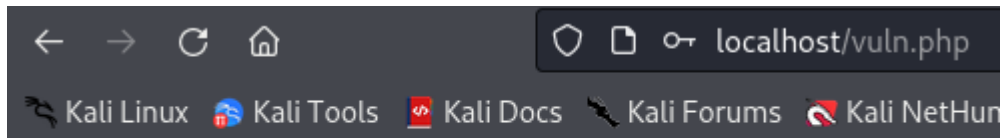
Este texto lo ven sólo aquellos que han hecho un login correcto.

Si accedemos con credenciales incorrectas encontraremos la siguiente página de error

Username:

Password:

Login



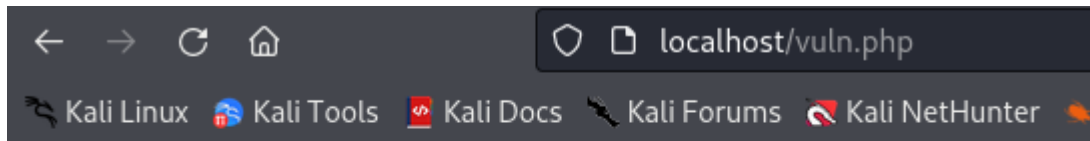
Usuario o Password Incorrecto, prueba otra vez

De todas maneras podemos acceder a la página haciendo una inyección de código con el comando **'or 1=1#'**

Username:

Password:

Login



Dentro!!!

Este texto lo ven sólo aquellos que han hecho un login correcto.

3- Instalación y configuración de modsecurity

Instalamos modsecurity con **apt install libapache2-mod-security2 -y**

```
(root@kali)-[/home/kali/Downloads]
# apt install libapache2-mod-security2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Activaremos uno de sus módulos para leer cabeceras con **a2enmod headers**

```
(root@kali)-[/home/kali/Downloads]
# a2enmod headers
Enabling module headers.
To activate the new configuration, you need to run:
systemctl restart apache2
```

Tal y como nos pide, reiniciamos apache2 haciendo un **systemctl restart apache2**

```
(root@kali)-[/home/kali/Downloads]
# systemctl restart apache2
```

Una vez hecho esto accederemos a **/etc/modsecurity** y crearemos el archivo de configuración de reglas de seguridad. Para hacerlo copiaremos el archivo **modsecurity.conf-recommended** a **modsecurity.conf** con el comando **cp modsecurity.conf-recommended modsecurity.conf**

```
(root@kali)-[/etc/modsecurity]
# cp modsecurity.conf-recommended modsecurity.conf
```

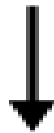
Entraremos dentro del archivo con **nano modsecurity.conf**

```
(root@kali)-[/etc/modsecurity]
# nano modsecurity.conf
```

El único cambio ha hacer en el archivo es activar el **SecRuleEngine**

```
GNU nano 7.2
# -- Rule engine initialization
# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine DetectionOnly

# -- Request body handling
```



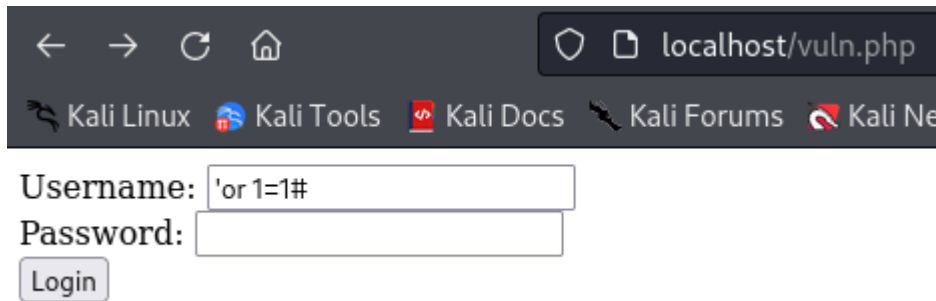
```
GNU nano 7.2
# -- Rule engine initialization
# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine On

# -- Request body handling
```

Reseteamos apache2 para aplicar los cambios

```
(root@kali)-[/etc/modsecurity]
# systemctl restart apache2
```

Comprobamos si ahora podemos acceder con la inyección de código, si se ha hecho correctamente no podremos acceder



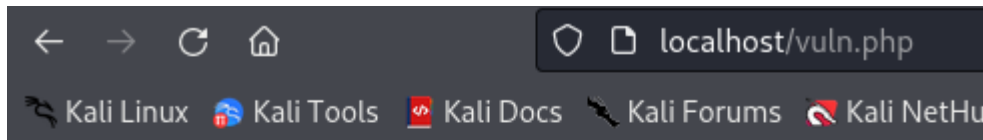
← → ↻ 🏠 localhost/vuln.php

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHu

Username:

Password:

Login



← → ↻ 🏠 localhost/vuln.php

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHu

Forbidden

You don't have permission to access this resource.

Apache/2.4.58 (Debian) Server at localhost Port 80

como podemos observar no nos deja entrar pero nos da información sobre qué servidor utilizamos, dando capacidad a un atacante de encontrar vulnerabilidades (ir a: [Quitar información de "addres"](#))

4- Implementación de OWAS en modsecurity

Podemos instalar las configuraciones de seguridad OWAS para protegernos mejor (como por ejemplo de que nos abran un terminal en un lugar en concreto)

Para ello lo primero que tenemos que hacer es acceder a `cd /usr/share/modsecurity-crs` y eliminar la carpeta **modsecurity-crs** con `rm -rf modsecurity-crs/`

```
(root@kali)-[/etc/modsecurity]
# cd /usr/share

(root@kali)-[/usr/share]
# cd modsecurity-crs

(root@kali)-[/usr/share/modsecurity-crs]
# ls
owasp-crs.load  rules  util

(root@kali)-[/usr/share/modsecurity-crs]

(root@kali)-[/usr/share]
# rm -rf modsecurity-crs/
```

Una vez hecho esto nos descargamos un conjunto de normas de un repositorio git. Lo primero de todo será instalar git con `apt install git`

```
(root@kali)-[/usr/share]
# apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1).
git set to manually installed.
The following packages were automatically installed and are no longer
needed:
  libadwaita-1-0 libappstream5 libatk-adaptor libstemmer0d libxmlb2
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 16 not upgraded.

(root@kali)-[/usr/share]
```

Descargamos desde un repositorio git las distintas normas de seguridad a través de **git clone** <https://github.com/coreruleset/coreruleset> /usr/share/modsecurity-crs

```
(root@kali)-[/usr/share]
# git clone https://github.com/coreruleset/coreruleset /usr/share/modsecurity-crs
Cloning into '/usr/share/modsecurity-crs' ...
remote: Enumerating objects: 30393, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 30393 (delta 0), reused 1 (delta 0), pack-reused 30392
Receiving objects: 100% (30393/30393), 8.02 MiB | 4.51 MiB/s, done.
Resolving deltas: 100% (23754/23754), done.

(root@kali)-[/usr/share]
```

Entramos en el nuevo **modsecurity-crs** y copiamos el archivo de setup para ponerlo como default con **cp crs-setup.conf.example crs-setup.conf**

```
(root@kali)-[/usr/share]
# cd modsecurity-crs

(root@kali)-[/usr/share/modsecurity-crs]
# cp crs-setup.conf.example crs-setup.conf
```

Dentro de **modsecurity-crs** hay una carpeta de **rules** con un archivo que también tendremos que clonar para ponerlo como default. Entramos con **cd rules** y clonamos el archivo con **cp REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf.example REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf**

```
(root@kali)-[/usr/share/modsecurity-crs]
# cd rules
```

```
(root@kali)-[/usr/share/modsecurity-crs/rules]
# cp REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf.example REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf
```

Lo siguiente ha hacer es modificar el **security2.conf** de apache2 para coger así empezar a utilizar las nuevas normas.

primero vamos a **cd/etc/apache2/mods-available**

```
(root@kali)-[/usr/share/modsecurity-crs/rules]
# cd /etc/apache2

(root@kali)-[/etc/apache2]
```

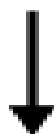
```
(root@kali)-[/etc/apache2]
# ls
apache2.conf  conf-available  conf-enabled  envvars  magi

(root@kali)-[/etc/apache2]
# cd mods-available
```

Editamos el fichero con **nano security2.conf**. Comentamos la última línea y añadimos **Include /etc/modsecurity/rules/*.conf**

```
(root@kali)-[/etc/apache2/mods-available]
# nano security2.conf
```

```
GNU nano 7.2 packages have been kept back:
<IfModule security2_module>
# Default Debian dir for modsecurity's persistent data
SecDataDir /var/cache/modsecurity
# Include all the *.conf files in /etc/modsecurity.
# Keeping your local configuration in that directory
# will allow for an easy upgrade of THIS file and
# make your life easier
IncludeOptional /etc/modsecurity/*.conf
# Include OWASP ModSecurity CRS rules if installed
IncludeOptional /usr/share/modsecurity-crs/*.load
</IfModule>
```



```
GNU nano 7.2 packages have been kept back:
<IfModule security2_module>
# Default Debian dir for modsecurity's persistent data
SecDataDir /var/cache/modsecurity
# Include all the *.conf files in /etc/modsecurity.
# Keeping your local configuration in that directory
# will allow for an easy upgrade of THIS file and
# make your life easier
IncludeOptional /etc/modsecurity/*.conf
Include /etc/modsecurity/rules/*.conf
# Include OWASP ModSecurity CRS rules if installed
#IncludeOptional /usr/share/modsecurity-crs/*.load
</IfModule>
```


Por último podemos implementar complementos de las OWAS para mayor seguridad. Descargamos las normas de gir con **git clone https://github.com/coreruleset/coreruleset.git**

```
(root@kali)-[/etc/apache2/mods-available]
# cd /home/kali/Downloads

(root@kali)-[/home/kali/Downloads]
# git clone https://github.com/coreruleset/coreruleset.git
Cloning into 'coreruleset' ...
remote: Enumerating objects: 30393, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 30393 (delta 0), reused 1 (delta 0), pack-reused 30392
Receiving objects: 100% (30393/30393), 8.03 MiB | 5.26 MiB/s, done.
Resolving deltas: 100% (23755/23755), done.

(root@kali)-[/home/kali/Downloads]
#
```

Entramos dentro de **coreruleset** y pasamos el archivo **crs-setup-conf.example** a **modsecurity** con

cp crs-setup.conf.example /etc/modsecurity/crs-setup.conf

```
(root@kali)-[/home/kali/Downloads]
# cd coreruleset

(root@kali)-[/home/kali/Downloads/coreruleset]
# ls
CHANGES.md  CONTRIBUTING.md  CONTRIBUTORS.md  INSTALL.md  KNOWN_BUGS.md

(root@kali)-[/home/kali/Downloads/coreruleset]
# cp crs-setup.conf.example /etc/modsecurity/crs-setup.conf

(root@kali)-[/home/kali/Downloads/coreruleset]
#
```

Dentro del mismo **coreruleset** encontramos la carpeta **rules**, la cual también pondremos dentro de **modsecurity** con **cp -av rules /etc/modsecurity/**

```
(root@kali)-[/home/kali/Downloads/coreruleset]
# cp -av rules /etc/modsecurity/
```

Para verificar que todas estas normas de seguridad están activadas podemos intentar abrir un terminal dentro de index.html con el siguiente comando:

curl http://localhost/index.html?exec=/bin/bash

```
(root@kali)-[/etc/modsecurity]
# curl http://localhost/index.html?exec=/bin/bash
<!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>
<hr>
<address>Apache/2.4.58 (Debian) Server at localhost Port 80</address>
</body></html>
```

Como podemos ver no nos lanza un mensaje de que no tenemos permisos para acceder y nos saca otro mensaje de error 403

podemos ver el log lanzado por este intento de acceso dentro de **/var/log/apache2**

```
(root@kali)-[/etc/apache2/sites-available]
# cd /var/log

(root@kali)-[/var/log]
# ls
README      alternatives.log  btmp          inetsim       macchang
Xorg.0.log  apache2         dpkg.log      installer     mosquitt
Xorg.0.log.old apt             faillog       journal       nginx
Xorg.1.log  boot.log        fontconfig.log lastlog       notus-sc
Xorg.1.log.old boot.log.1.html  gvm           lightdm       openvpn

(root@kali)-[/var/log]
# cd apache2

(root@kali)-[/var/log/apache2]
# ls
access.log  error.log  modsec_audit.log  other_vhosts_access.log
```

leemos el fichero **error.log** con un **cat error.log | grep "/bin/bash"**

```
(root@kali)-[/var/log/apache2]
# cat error.log | grep "/bin/bash"
[Wed May 08 10:07:26.668547 2024] [security2:error] [pid 1681] [client ::1:42096] [client ::1] ModSecurity: Warning
. Matched phrase "bin/bash" at ARGS:exec. [file "/etc/modsecurity/rules/REQUEST-932-APPLICATION-ATTACK-RCE.conf"] [
line "575"] [id "932160"] [msg "Remote Command Execution: Unix Shell Code Found"] [data "Matched Data: bin/bash fou
nd within ARGS:exec: /bin/bash"] [severity "CRITICAL"] [ver "OWASP CRS/4.3.0-dev"] [tag "application-multi"] [tag "
language-shell"] [tag "platform-unix"] [tag "attack-rce"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/10
00/152/248/88"] [tag "PCI/6.5.2"] [hostname "localhost"] [uri "/index.html"] [unique_id "Zjsyvh14Mut6eo5_PLYpyQAAAA
A"]

(root@kali)-[/var/log/apache2]
```

5- Configuraciones de seguridad en Apache2

5.1- Ocultar el banner address

Cuando hacemos el curl, aunque no nos deje entrar ni poner iniciar la shell, nos da la información sobre el servidor que estamos utilizando. Información que pueden utilizar contra nosotros. Para evitar esto vamos a entrar a **/etc/apache2/conf-available**

```
(root@kali)-[/etc/modsecurity]
# cd /etc/apache2

(root@kali)-[/etc/apache2]
# cd conf-available

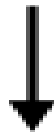
(root@kali)-[/etc/apache2/conf-available]
#
```

Hacemos **nano security.conf** para editar el archivo

```
(root@kali)-[/etc/apache2/conf-available]
# nano security.conf
```

Añadimos **SecServerSignature** “mensaje” para que se muestre lo que nosotros deseemos, o simplemente ponemos el **ServerSignature** en **Off** para que no se vea nada.

```
Preparing to unpack .../python3-pyqtgraph_0.13.7-1_all.deb ...  
# packing python3-pyqtgraph (0.13.7-1) over (0.13.6-2) ...  
# 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  
ServerSignature On amd64 (1:3.12.0-1) ...  
Setting up python3-pyasnl (0.5.1-1) ...  
Setting up nsis (3.10-2) ...  
# Setting up pciutils (1:3.12.0-1) ...  
# Allow TRACE method for libc-bin (2.37-15) ...  
# processing triggers for man-db (2.12.0-3) ...  
# Set to "extended" to also reflect the request body (only for testing and  
# diagnostic purposes).
```



```
Preparing to unpack .../python3-pyqtgraph_0.13.7-1_all.deb ...  
# packing python3-pyqtgraph (0.13.7-1) over (0.13.6-2) ...  
# 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  
ServerSignature On amd64 (1:3.12.0-1) ...  
SecServerSignature Windows .5.1-1) ...  
Setting up nsis (3.10-2) ...  
# Setting up pciutils (1:3.12.0-1) ...  
# Allow TRACE method for libc-bin (2.37-15) ...  
# processing triggers for man-db (2.12.0-3) ...  
# Set to "extended" to also reflect the request body (only for testing and  
# diagnostic purposes).
```

Reseteamos apache

```
(root@kali)-[/etc/apache2/sites-available]  
# systemctl restart apache2
```

Volvemos a hacer el curl para ver cómo se han aplicado los cambios. En este caso podemos observar que en vez de apache, ahora nos dice que es un windows server, mareando así a nuestros atacantes y dificultando la intrusión a la web.

```
(root@kali)-[/etc/apache2/conf-available]
# curl http://localhost/index.html?exec=/bin/bash
<!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>
<hr>
<address>Windows Server at localhost Port 80</address>
</body></html>

(root@kali)-[/etc/apache2/conf-available]
```

5.2- Creación de reglas de bloqueo específicas

Para poder crear reglas específicas tendremos que dirigirnos a `/etc/apache2/sites-available`

```
(root@kali)-[/etc/apache2/conf-available]
# cd /etc/apache2

(root@kali)-[/etc/apache2]
# ls
apache2.conf  conf-enabled  magic          mods-enabled  sites-available
conf-available  envvars      mods-available  ports.conf    sites-enabled

(root@kali)-[/etc/apache2]
# cd sites-available

(root@kali)-[/etc/apache2/sites-available]
#
```

Des de allí editamos uno de los ficheros con **nano 000-default.conf**

```
(root@kali)-[/etc/apache2/sites-available]
# nano 000-default.conf
```

Aquí dentro podemos crear nuestras propias reglas, vamos a crear una de ejemplo editando el fichero de la siguiente manera:

SecRuleEngine On

SecRule ARGS:testparam "@contains test" "id:254,deny,status:403,msg:'Test con exito'"

ARGS: decimos que la regla se aplicará si la solicitud tiene un parámetro llamado "testparam"

@contains: indicamos de que la regla se activa si el parámetro "testparam" contiene la cadena test

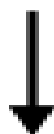
id: se le da un id único a la regla

deny: indica que queremos hacer cuando se cumplan las condiciones, en este caso denegamos

status:403: código de error que se enviará como respuesta y que podremos ver dentro de los logs

msg: texto que se enviará como respuesta y que podremos ver dentro de los logs

```
Setting  ErrorLog ${APACHE_LOG_DIR}/error.log
Setting  CustomLog ${APACHE_LOG_DIR}/access.log combined
locate-updatedb.service is a disabled or a static unit not running, not starting it.
Setting  # For most configuration files from conf-available/, which are
Setting  # enabled or disabled at a global level, it is possible to
Setting  # include a line for only one particular virtual host. For example the
Setting  # following line enables the CGI configuration for this host only
Process  # after it has been globally disabled with "a2disconf".
Process  #Include conf-available/serve-cgi-bin.conf
Processing triggers for kali-menu (2023.4.7) ...
</VirtualHost>
/var/www/html
```

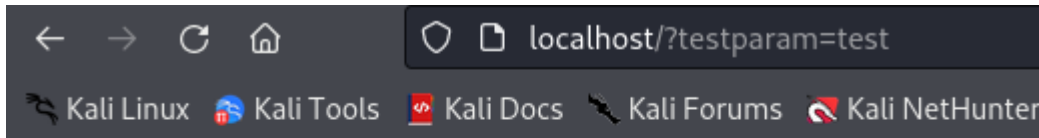


```
Setting  ErrorLog ${APACHE_LOG_DIR}/error.log
Setting  CustomLog ${APACHE_LOG_DIR}/access.log combined
locate-updatedb.service is a disabled or a static unit not running, not starting it.
Setting  # For most configuration files from conf-available/, which are
Setting  # enabled or disabled at a global level, it is possible to
Setting  # include a line for only one particular virtual host. For example the
Setting  # following line enables the CGI configuration for this host only
Process  # after it has been globally disabled with "a2disconf".
Process  #Include conf-available/serve-cgi-bin.conf
Processing triggers for kali-menu (2023.4.7) ...
    SecRuleEngine On
    SecRule ARGS:testparam "@contains test" "id:254,deny,status:403,msg:'Test con exito'"
</VirtualHost>
completing local directory
```

Resetearmos apache 2

```
(root@kali)-[/etc/apache2/sites-available]
# systemctl restart apache2
```

Si vamos al navegador y ponemos la siguiente dirección, **localhost/?testparam=test**, nos bloqueará



Forbidden

You don't have permission to access this resource.

5.3- Búsqueda de logs

Para encontrar los logs que nos hace apache2, y así encontrar los mensajes de error del mensaje anterior, debemos acceder a **/var/log/apache2**

```
(root@kali)-[/home/kali]
# cd /var/log/apache2
```

Los logs los encontramos en el archivo error.log. Accedemos al archivo y a la vez aplicamos un filtro para encontrar los logs de la norma creada con anterioridad. Hacemos **cat error.log | grep "test con éxito"**

```
(root@kali)-[/var/log/apache2]
# cat error.log | grep "Test con éxito"
[Tue May 07 16:26:09.601396 2024] [security2:error] [pid 220638] [client 127.0.0.1:41656] [client 127.0.0.1] ModSecurity: Access denied with code 403 (phase 2). String match "test" at ARGS:testparam. [file "/etc/apache2/sites-enabled/000-default.conf"] [line "31"] [id "254"] [msg "Test con éxito"] [hostname "localhost"] [uri "/"] [unique_id "Zjo6AT7tm4p-H0TorT_XyAAAAAE"]

(root@kali)-[/var/log/apache2]
#
```

Podemos observar que el log contiene el código "403" y el mensaje "Test con éxito" puestos en la rule, además de información extra.

6- Anexo

6.1- Contenido vuln.php:

[\(regresar\)](#)

```
<html>
<head>
<title>
Página de prueba SQLi dfir
</title>
</head>
<body>
<?php
    if(isset($_POST['login']))
    {
        $username = $_POST['username'];
        $password = $_POST['password'];
        $con = mysqli_connect('localhost','root','123456','dfir');
        $result = mysqli_query($con, "SELECT * FROM `usuarios` WHERE
userid='$username' AND password='$password'");
        if(mysqli_num_rows($result) == 0)
            echo 'Usuario o Password Incorrecto, prueba otra vez';
        else
            echo '<h1>Dentro!!!</h1><p>Este texto lo ven sólo aquellos que
han hecho un login correcto.</p>';
    }
    else
    {
        ?>
        <form action="" method="post">
            Username: <input type="text" name="username"/><br />
            Password: <input type="password" name="password"/><br />
            <input type="submit" name="login" value="Login"/>
        </form>
    }
    ?>
</body>
</html>
```