Alejandro Hernandez

Trabajo 3: Instalar y configurar un firewall de aplicaciones web Open Source

El siguiente trabajo consiste en la instalación y configuración del Firewall de aplicaciones web que permita proteger a un sitio web de ataques conocidos, para ello se hará uso del conjunto de reglas de OWASP ModSecurity Core Rule Set (CRS).

Antes de empezar es preciso tener en la misma red el servidor y el cliente para que se puedan conectar:

Red: 192.168.1.0/24

Pasos

1. Una vez instalado el Ubuntu Server y el cliente Linux Mint en VirtualBox procedemos a instalar la aplicación web en nuestro Ubuntu Server.

   En este caso usaremos una aplicación web vulnerable previamente creada que funciona para simular estos ataques de SQL injection. Esta aplicación fue desarrollada en PHP/MYSQL.

   Para que la dicha aplicación funcione instalamos en nuestro servidor lo siguiente:

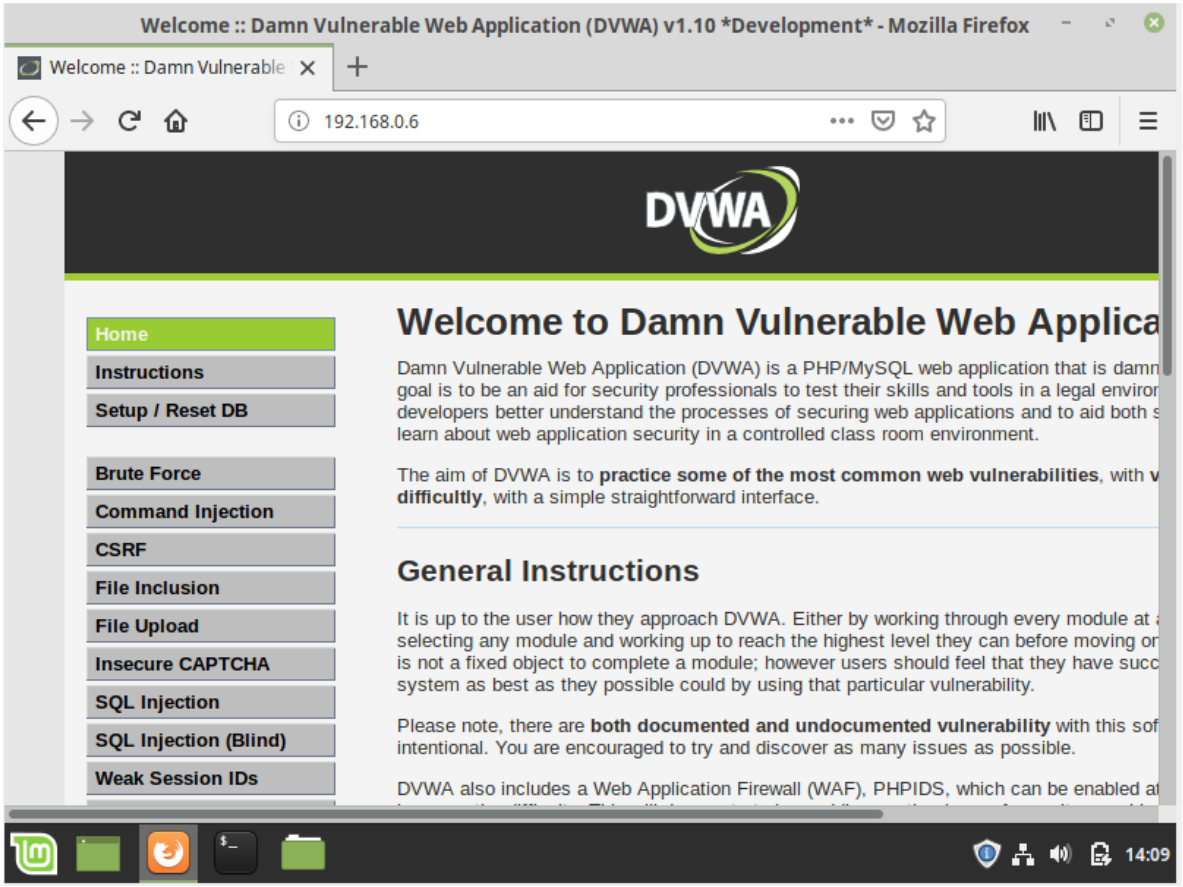
   ```
   sudo apt install apache2 mysql-server php php-mysqli php-gd
   libapache2-mod-php git
   ```

   Repositorio de la aplicación:

   ```
   git clone --recursive https://github.com/ethicalhack3r/DVWA.git
   ```

   ```
   alejo@alejo:~$ git clone --recursive https://github.com/ethicalhack3r/DVWA.git
   Cloning into 'DVWA'...
   remote: Enumerating objects: 2995, done.
   remote: Total 2995 (delta 0), reused 0 (delta 0), pack-reused 2995
   Receiving objects: 100% (2995/2995), 1.52 MiB | 663.00 KiB/s, done.
   Resolving deltas: 100% (1318/1318), done.
   ```

2. Una vez instalada la aplicación, ingresamos desde nuestro navegador en una maquina a parte que este en la misma red, con la siguiente IP: 192.168.100.23.



3. Ahora configuramos las reglas de OWASP ModSecurity Core Rule Set(CRS) instalando primero Apache y luego las reglas de ModSecurity usando estos comandos.

```
sudo apt-get install Apache2
```

```
sudo apt-get install libapache2-mod-security2
```

Cuando ya este instalado reiniciamos el Apache

```
sudo service apache2 restart
```

Y verificamos que este funcionando

```
sudo apachectl -M | grep security
```

```
alejo@alejo:~$ sudo apachectl -M | grep security
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0
.1.1. Set the 'ServerName' directive globally to suppress this message
 security2_module (shared)
```

4. Luego, configuramos el ModSecurity usando los siguientes comandos en el servidor.

```
Sudo cp /etc/modsecurity/modsecurity.conf-recommended
/etc/modsecurity/modsecurity.conf
```

Y abrimos el archive modsecurity.conf y lo modifico el atributo.

```
sudo nano /etc/modsecurity/modsecurity.conf
```

SecRuleEngine On

Reiniciamos el servidor Apache.

```
sudo systemctl restart apache2
```

Luego bajamos las reglas del git.

```
sudo mv /usr/share/modsecurity-crs /usr/share/modsecurity-
crs.bk
```

```
sudo git clone https://github.com/SpiderLabs/owasp-
modsecurity-crs.git /usr/share/modsecurity-crs
```

```
sudo cp /usr/share/modsecurity-crs/crs-setup.conf.example
/usr/share/modsecurity-crs/crs-setup.conf
```

Y se edita el archive security2 de la siguiente manera:

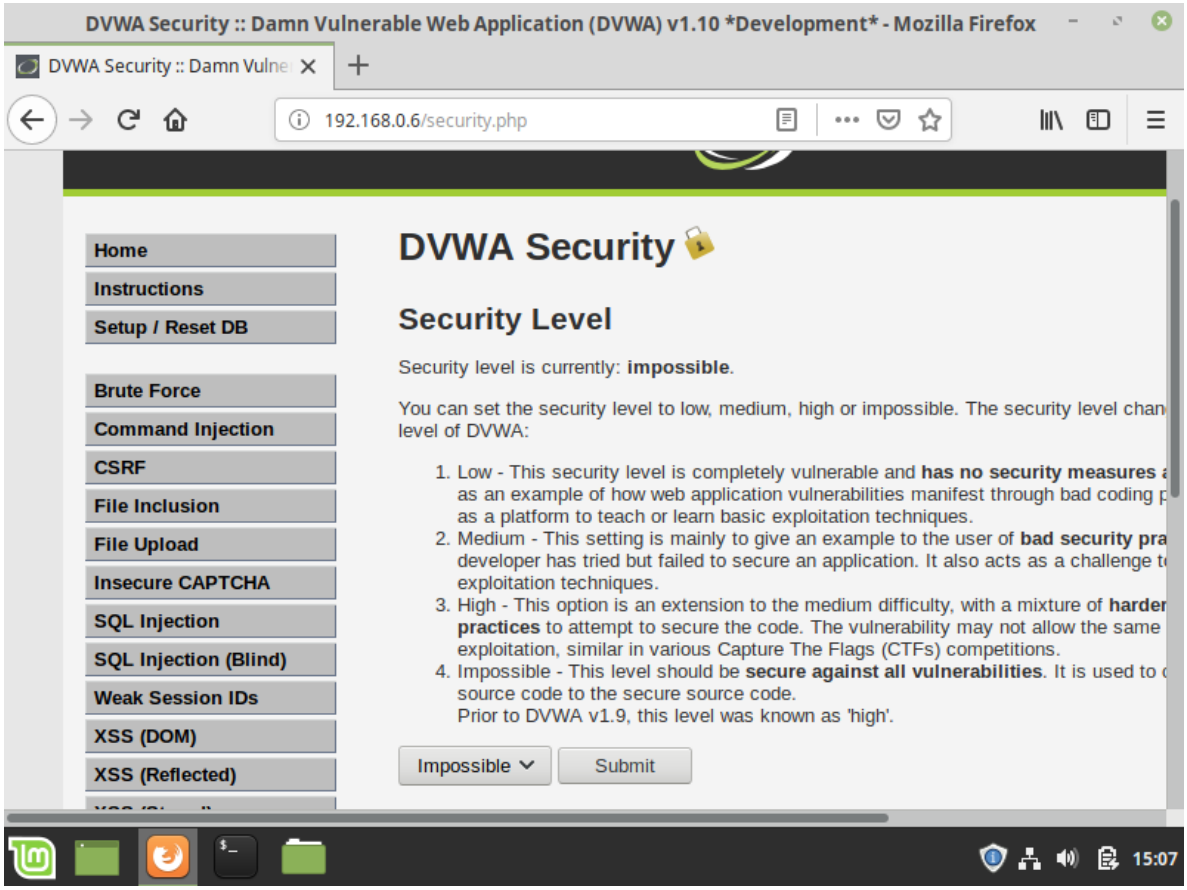
```
sudo nano /etc/apache2/mods-enabled/security2.conf
```

```
IncludeOptional /usr/share/modsecurity-crs/*.conf
```

```
IncludeOptional "/usr/share/modsecurity-crs/rules/*.conf
```

Y finalmente volvemos a reiniciar el servidor.

5. Una vez configuradas las normas en OWASP en el servidor, la aplicación web nos permite el nivel de seguridad de nuestra aplicación.

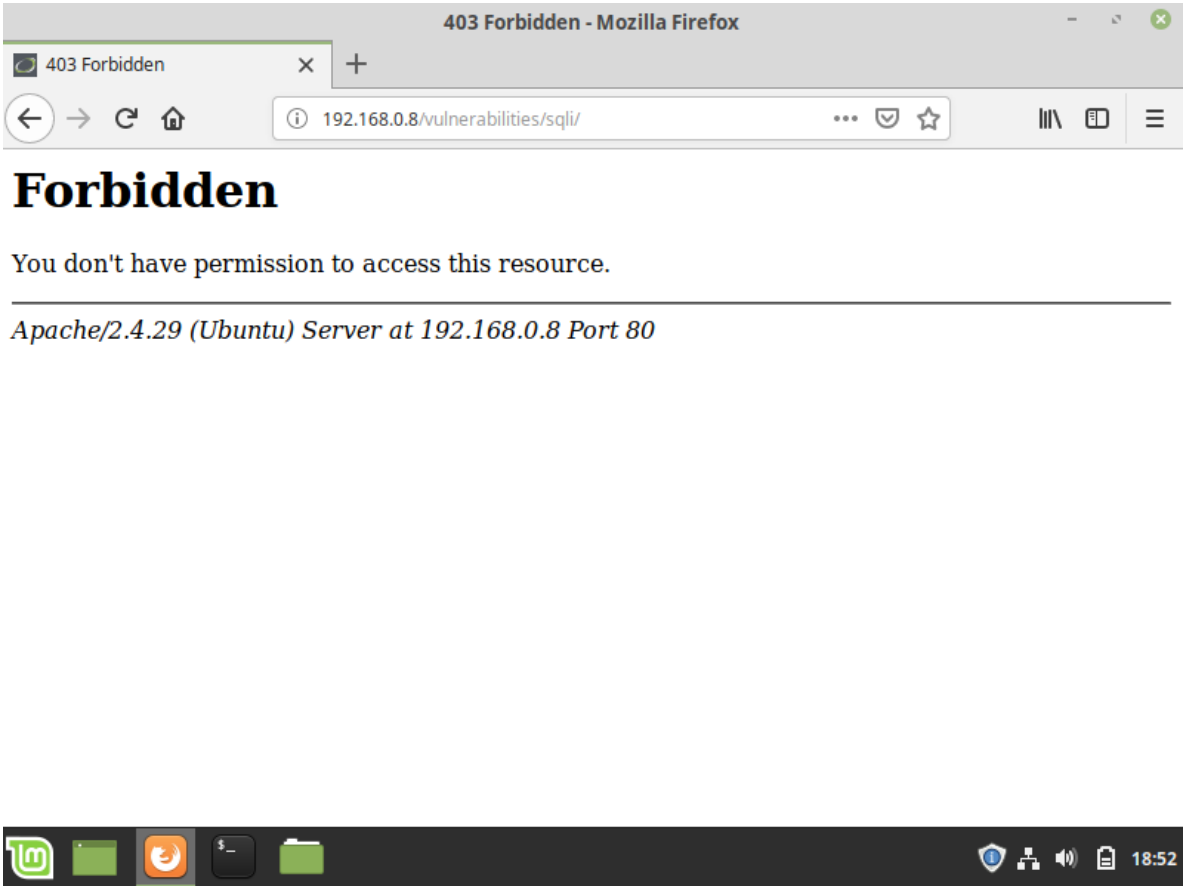


Este nivel de seguridad debemos cambiarlo a low para poder aplicar el SQL injection.

Para simular la inyección a la base de datos de nuestra aplicación, debemos introducir la siguiente consulta.

' or 1=1 union select user, password from dvwa.users#

Haciendo eso, nos dará el siguiente error



Si apagamos el WAF dará lo siguiente:


Vulnerability: SQL Injection :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Mozill...
Vulnerability: SQL Injection ::
192.168.0.8/vulnerabilities/sqli/?id='+or+1%3D1
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
Vulnerability: SQL Injection
User ID:
Submit
ID: ' or 1=1 union select user, password from dvwa.users#
First name: admin
Surname: admin
ID: ' or 1=1 union select user, password from dvwa.users#
First name: Gordon
Surname: Brown
ID: ' or 1=1 union select user, password from dvwa.users#
First name: Hack
Surname: Me
ID: ' or 1=1 union select user, password from dvwa.users#
First name: Pablo
Surname: Picasso
ID: ' or 1=1 union select user, password from dvwa.users#
First name: Bob
Surname: Smith
18:55

Bibliografía

https://hostadvice.com/how-to/how-to-setup-modsecurity-for-apache-on-ubuntu-18-04/

https://www.thomaslaurenson.com/blog/2018/07/12/installing-and-configuring-damn-vulnerable-web-application/#initial-steps

http://director-it.com/index.php/es/ssoluciones/seguridad/firewall-y-dmz/118-que-es-un-waf-web-application-firewall.html

https://www.w3schools.com/sql/sql_injection.asp