3. Active Directory PowerView

AD Enumeration Toolkit

As discussed in the Active Directory LDAP module, in-depth enumeration is arguably the most important phase of any security assessment. Attackers are continuing to find new (and old) techniques and methodologies for abusing and attacking AD. In AD, this phase helps us to get a "lay of the land" and understand the design of the internal network, including the number of OUs, users, groups, computers, ACLs, and other AD objects and the hundreds and thousands of relationships that make up an AD environment. Our job is to untangle these often very complex relationships by gathering relevant data in various formats and organizing in a way that helps us uncover the flaws and misconfigurations hiding inside the network.

The Active Directory LDAP module provided an overview of Active Directory, introduced a variety of built-in tools that can be extremely useful when performing AD enumeration, and perhaps the most important, covered LDAP and AD search filters which, when combined with these built-in tools, provide us with a powerful arsenal to drill down into the intricacies of AD and discover nuanced, but serious, misconfigurations before the attackers do. While it is important for us to be able to "live off the land" when performing assessments, it is equally important to understand the wide variety of third-party open-source tools available to us for enumerating and attacking AD. Each of the tools that we will cover in this module performs AD enumeration in slightly different ways. We often need to gather, analyze, and interpret data from many of them iteratively throughout and assessment. The knowledge of and ability to use built-in tools and third-party tools effectively is what can set us apart from other assessors.

Tools of the Trade

Depending on the type of engagement we are on, there are various tools available to us to perform AD enumeration. Some of the most important ones for us to be able to use effectively are:

Tool	Description
BloodHound	Used to visually map out AD relationships and help plan attack paths that may otherwise go unnoticed. Uses the SharpHound PowerShell or C# ingestor to gather data to later be imported into the BloodHound JavaScript (Electron) application with a Neo4j database for graphical analysis of the AD environment.
BloodHound.py	A Python-based BloodHound ingestor based on the Impacket toolkit. It supports most BloodHound collection methods and can be run from a non-domain joined attack box. The output can be ingested into BloodHound 3.0 for analysis.
PowerView/ SharpView	A PowerShell tool and a .NET port of the same used to gain situational awareness in AD. These tools can be used as replacements for various Windows `net*` commands and more. PowerView and SharpView can help us gather much of the data that BloodHound does, but it requires more work to make meaningful relationships among all of the data points. These tools are great for checking what additional access we may have with a new set of credentials, targeting specific users or computers, or finding some "quick wins" such as users that can be attacked via Kerberoasting or ASREPRoasting
CrackMapExec (CME)	CME is an enumeration, attack, and post-exploitation toolkit which can help us greatly in enumeration and performing attacks with the data we gather. CME attempts to "live off the land" and abuse built-in AD features and protocols such as SMB, WMI, WinRM, and more.
PingCastle	Used for auditing the security level of an AD environment based on a risk assessment and maturity framework (based on CMMI adapted to AD security).
PowerUpSQL	This tool is used for SQL Server discovery, configuration auditing, privilege escalation, and post-exploitation.
Snaffler	Useful for finding information (such as credentials) in Active Directory on computers with accessible file shares.
Group3r	Group3r is useful for auditing and finding security misconfigurations in AD Group Policy Objects (GPO)
MailSniper	A tool for searching through email inboxes in a Microsoft Exchange environment for specific keywords/terms that may be used to enumerate sensitive data (such as credentials) which could be used for lateral movement and privilege escalation. It can search a user's individual mailbox or by a user with Exchange Administrator privileges to enumerate all mailboxes in a domain. It can also be used for password spraying, enumerating domain users/domains, checking mailbox permissions, and gathering the Global Address List (GAL) from Outlook Web Access (OWA) and Exchange Web Services (EWS).
windapsearch	A Python script used to enumerate AD users, groups, and computers using LDAP queries. Useful for automating custom LDAP queries.

Tool	Description
ADRecon	A tool used to extract various data from a target AD environment. The data can be output in Microsoft Excel format with summary views and analysis to assist with analysis and paint a picture of the environment's overall security state.
Active Directory Explorer	Active Directory Explorer (AD Explorer) is an AD viewer and editor. It can be used to navigate an AD database and view object properties and attributes. It can also be used to save a snapshot of an AD database for off-line analysis. When an AD snapshot is loaded, it can be explored as a live version of the database. It can also be used to compare two AD database snapshots to see changes in objects, attributes, and security permissions.

This module will focus on the `PowerView` and `SharpView` tools to cover various AD enumeration techniques. As penetration testers, it is important to have a wide range of tools available to us and understand how they work to troubleshoot if we are not getting expected results. While we may not use every one of these tools on an engagement, it is important to understand how they work, complement each other, and can be combined to provide the deepest possible coverage of the target AD environment, based on the goals of the assessment. The tools listed above will be covered in other modules.

Next Steps

During this module, we will target a fictional company called `INLANEFREIGHT` with the internal domain `INLANEFREIGHT.LOCAL`. The module sections will build on each other, culminating in a mock penetration testing skills assessment to showcase our skills before moving on to the next module in this series. For all exercises, we will assume that the target company Inlanefreight has hired us to perform an in-depth penetration test with a heavy focus on AD security, where stealth and bypassing stringent security controls are not a requirement.

Module Exercises

Throughout this module, you will connect to various target hosts via the Remote Desktop Protocol (RDP) to complete the exercises. Any necessary credentials will be provided with each exercise, and the RDP connection can be made via `xfreerdp` from the Pwnbox as follows:

```
xfreerdp /v:<target IP address> /u:htb-student /p:<password>
```

Any necessary tools can be found in the `c:\tools` directory after logging in to the target host.

PowerView/SharpView Overview & Usage

SharpView is a .NET port of PowerView, one of many tools contained within the now deprecated PowerSploit offensive PowerShell toolkit. This Read the Docs page explains the function naming schema and provides information about the various parameters that can be passed to each function.

Note: Since writing this module, we noticed that BC-Security has started pushing updates to PowerView as part of their Empire project. This course still uses the Development PowerView module out of PowerSploit's GitHub, but by the end of the year, we plan to migrate this to the version that Empire uses.

In the past, PowerShell was the scripting language of choice for offensive tools, but it has become more security transparent, with better detection optics available for both consumer and enterprise-level endpoint protection products. For this reason, offensive security practitioners have evolved their tradecraft to mitigate improved security monitoring capabilities and have ported their tooling to C# inline, which is less security transparent. While `PowerView` is no longer officially maintained, it is still an extremely powerful AD enumeration tool and can be useful when performing engagements where stealth is not a requirement. It also remains useful for defenders who are looking to gain a better understanding of their AD environment. We will cover the history and general usage of `PowerView`, but this module (and related modules) will focus on `SharpView` in line with current .NET tradecraft to be more applicable to real-life, modern engagements. We will cover general `PowerView` and `SharpView` usage in this module because both still have their uses, depending on the situation.

Both tools can perform enumeration, gain situational awareness, and perform attacks within a Windows domain. `PowerView` utilizes PowerShell AD hooks and Win32 API functions, and, among other functions, replaces a variety of net commands called by the built-in Windows tools. `SharpView` is a .NET port that provides all of the `PowerView` functions and arguments in a .NET assembly. One major difference between `PowerView` and `SharpView` is the ability to pipe commands. `SharpView` uses strings instead of PowerShell objects. Therefore we cannot specify properties using `Select` or `Select-Object`, to parse the output or select specific AD objects as easily.

```
C:\htb> net accounts

Force user logoff how long after time expires?:       Never
Minimum password age (days):                          1
Maximum password age (days):                          Unlimited
```

```
Minimum password length:                              7
Length of password history maintained:                24
Lockout threshold:                                    Never
Lockout duration (minutes):                           30
Lockout observation window (minutes):                 30
Computer role:                                        PRIMARY
The command completed successfully.
```

Here we can see that a command similar to `net accounts` can be performed with the `PowerView` or `SharpView` command `Get-DomainPolicy`.

```
PS C:\htb> Get-DomainPolicy

Unicode        : @{Unicode=yes}
SystemAccess   : @{MinimumPasswordAge=1; MaximumPasswordAge=-1;
MinimumPasswordLength=7; PasswordComplexity=0;
                 PasswordHistorySize=24; LockoutBadCount=0;
RequireLogonToChangePassword=0;
                 ForceLogoffWhenHourExpire=0; ClearTextPassword=0;
LSAAnonymousNameLookup=0}
KerberosPolicy : @{MaxTicketAge=10; MaxRenewAge=7; MaxServiceAge=600;
MaxClockSkew=5; TicketValidateClient=1}
Version        : @{signature="$CHICAGO$"; Revision=1}
RegistryValues :
@{MACHINE\System\CurrentControlSet\Control\Lsa\NoLMHash=System.Object[]}
Path           :
\\INLANEFREIGHT.LOCAL\sysvol\INLANEFREIGHT.LOCAL\Policies\{31B2F340-016D-
11D2-945F-00C04FB984F9}\MACHI
                 NE\Microsoft\Windows NT\SecEdit\GptTmpl.inf
GPOName        : {31B2F340-016D-11D2-945F-00C04FB984F9}
GPODisplayName : Default Domain Policy
```

The functionality of both tools can be grouped into different "buckets". While we will not cover every single function in this section, we will cover some of the most important ones under each. Both tools use the same functions and arguments, but the output can differ. This Read the Docs documentation provides an in-depth description of each function and command syntax and various examples for how the functions can be used.

Misc Functions

The misc functions offer various useful tools such as converting UAC values, SID conversion, user impersonation, Kerberoasting, and more. The entire list of functions with explanations from the tool documentation is as follows:

```
Export-PowerViewCSV             -   thread-safe CSV append
Resolve-IPAddress               -   resolves a hostname to an IP
ConvertTo-SID                   -   converts a given user/group name to a
security identifier (SID)
Convert-ADName                  -   converts object names between a
variety of formats
ConvertFrom-UACValue            -   converts a UAC int value to human
readable form
Add-RemoteConnection            -   pseudo "mounts" a connection to a
remote path using the specified credential object
Remove-RemoteConnection         -   destroys a connection created by New-
RemoteConnection
Invoke-UserImpersonation        -   creates a new "runas /netonly" type
logon and impersonates the token
Invoke-RevertToSelf             -   reverts any token impersonation
Get-DomainSPNTicket             -   request the kerberos ticket for a
specified service principal name (SPN)
Invoke-Kerberoast               -   requests service tickets for
kerberoast-able accounts and returns extracted ticket hashes
Get-PathAcl                     -   get the ACLs for a local/remote file
path with optional group recursion
```

We can use `SharpView` or `PowerView` to convert a username to the corresponding SID.

```
PS C:\htb> .\SharpView.exe ConvertTo-SID -Name sally.jones

S-1-5-21-2974783224-3764228556-2640795941-1724
```

And vice-versa:

```
PS C:\htb> .\SharpView.exe Convert-ADName -ObjectName S-1-5-21-2974783224-
3764228556-2640795941-1724

INLANEFREIGHT\sally.jones
```

When we enumerate UAC values using the `useraccountcontrol` value, the values are displayed back to us as numerical values, not in a human-readable format. We can use the `ConvertFrom-UACValue` function. If we add the `-showall` property, all common UAC values are shown, and the ones that are set for the user are marked with a `+`. This can be saved as a reference on a cheat sheet for future engagements.

```
PS C:\htb> Get-DomainUser harry.jones  | ConvertFrom-UACValue -showall
```

```
Name                            Value
----                            -----
SCRIPT                          1
ACCOUNTDISABLE                  2
HOMEDIR_REQUIRED                8
LOCKOUT                         16
PASSWD_NOTREQD                  32+
PASSWD_CANT_CHANGE              64
ENCRYPTED_TEXT_PWD_ALLOWED      128
TEMP_DUPLICATE_ACCOUNT          256
NORMAL_ACCOUNT                  512+
INTERDOMAIN_TRUST_ACCOUNT       2048
WORKSTATION_TRUST_ACCOUNT       4096
SERVER_TRUST_ACCOUNT            8192
DONT_EXPIRE_PASSWORD            65536+
MNS_LOGON_ACCOUNT               131072
SMARTCARD_REQUIRED              262144
TRUSTED_FOR_DELEGATION          524288
NOT_DELEGATED                   1048576
USE_DES_KEY_ONLY                2097152
DONT_REQ_PREAUTH                4194304
PASSWORD_EXPIRED                8388608
TRUSTED_TO_AUTH_FOR_DELEGATION  16777216
PARTIAL_SECRETS_ACCOUNT         67108864
```


Domain/LDAP Functions

```
Get-DomainDNSZone               -   enumerates the Active Directory DNS
zones for a given domain
Get-DomainDNSRecord             -   enumerates the Active Directory DNS
records for a given zone
Get-Domain                      -   returns the domain object for the
current (or specified) domain
Get-DomainController            -   return the domain controllers for the
current (or specified) domain
Get-Forest                      -   returns the forest object for the
current (or specified) forest
Get-ForestDomain                -   return all domains for the current (or
specified) forest
Get-ForestGlobalCatalog         -   return all global catalogs for the
current (or specified) forest
Find-DomainObjectPropertyOutlier-   inds user/group/computer objects in AD
that have 'outlier' properties set
Get-DomainUser                  -   return all users or specific user
objects in AD
```

```
New-DomainUser                  -   creates a new domain user (assuming
appropriate permissions) and returns the user object
Set-DomainUserPassword          -   sets the password for a given user
identity and returns the user object
Get-DomainUserEvent             -   enumerates account logon events (ID
4624) and Logon with explicit credential events
Get-DomainComputer              -   returns all computers or specific
computer objects in AD
Get-DomainObject                -   returns all (or specified) domain
objects in AD
Set-DomainObject                -   modifies a given property for a
specified active directory object
Get-DomainObjectAcl             -   returns the ACLs associated with a
specific active directory object
Add-DomainObjectAcl             -   adds an ACL for a specific active
directory object
Find-InterestingDomainAcl       -   finds object ACLs in the current (or
specified) domain with modification rights set to non-built in objects
Get-DomainOU                    -   search for all organization units
(OUs) or specific OU objects in AD
Get-DomainSite                  -   search for all sites or specific site
objects in AD
Get-DomainSubnet                -   search for all subnets or specific
subnets objects in AD
Get-DomainSID                   -   returns the SID for the current domain
or the specified domain
Get-DomainGroup                 -   return all groups or specific group
objects in AD
New-DomainGroup                 -   creates a new domain group (assuming
appropriate permissions) and returns the group object
Get-DomainManagedSecurityGroup  -   returns all security groups in the
current (or target) domain that have a manager set
Get-DomainGroupMember           -   return the members of a specific
domain group
Add-DomainGroupMember           -   adds a domain user (or group) to an
existing domain group, assuming appropriate permissions to do so
Get-DomainFileServer            -   returns a list of servers likely
functioning as file servers
Get-DomainDFSShare              -   returns a list of all fault-tolerant
distributed file systems for the current (or specified) domain
```

The LDAP functions provide us with a wealth of useful commands. The `Get-Domain` function will provide us with information about the domain, such as the name, any child domains, a list of domain controllers, domain controller roles, and more.

```
PS C:\htb> .\SharpView.exe Get-Domain
```

```
Forest                         : INLANEFREIGHT.LOCAL
DomainControllers              : {DC01.INLANEFREIGHT.LOCAL}
Children                       : {LOGISTICS.INLANEFREIGHT.LOCAL}
DomainMode                     : Unknown
DomainModeLevel                : 7
PdcRoleOwner                   : DC01.INLANEFREIGHT.LOCAL
RidRoleOwner                   : DC01.INLANEFREIGHT.LOCAL
InfrastructureRoleOwner        : DC01.INLANEFREIGHT.LOCAL
Name                           : INLANEFREIGHT.LOCAL
```

We can begin to get the lay of the land with the `Get-DomainOU` function and return the names of all Organizational Units (OUs), which can help us map out the domain structure. We can enumerate these names with `SharpView`.

```
PS C:\htb> .\SharpView.exe Get-DomainOU | findstr /b "name"

name                           : Domain Controllers
name                           : Admin
name                           : Employees
name                           : Servers
name                           : Workstations
name                           : Quarantine
name                           : Disabled Accounts
name                           : Help Desk
name                           : Executives
name                           : Interns
name                           : IT
name                           : Temp
name                           : Operations
name                           : Sales
name                           : Marketing
name                           : Warehouse
name                           : Legal
name                           : HR
name                           : Web Servers
name                           : SQL Servers
name                           : File Servers
name                           : Contractor Laptops
name                           : Staff Workstations
name                           : Executive Workstations
name                           : Security
name                           : Server Team
name                           : Network Ops
name                           : Service Accounts
name                           : Developers
name                           : Mail Servers
name                           : Accounting
```

```
name                          : Privileged Access
name                          : Mail Room
name                          : Freight
name                          : Finance
name                          : Contractors
name                          : Vendors
name                          : Microsoft Exchange Security Groups
```

We can gather information about domain users with the `Get-DomainUser` function and specify properties such as `PreauthNotRequired` to try planning out attacks.

```
PS C:\htb> .\SharpView.exe Get-DomainUser -KerberosPreauthNotRequired

[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainUser] Searching for user accounts that do not require kerberos
preauthentication
[Get-DomainUser] filter string: (&(samAccountType=805306368)
(userAccountControl:1.2.840.113556.1.4.803:=4194304))
objectsid                     : {S-1-5-21-2974783224-3764228556-
2640795941-1859}
samaccounttype                : USER_OBJECT
objectguid                    : f4493b78-55f0-488f-b21b-1dfd9069407d
useraccountcontrol            : PASSWD_NOTREQD, NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD, DONT_REQ_PREAUTH
accountexpires                : NEVER
lastlogon                     : 8/13/2020 4:59:09 AM
lastlogontimestamp            : 8/12/2020 10:22:30 AM
pwdlastset                    : 7/27/2020 3:35:52 PM
lastlogoff                    : 12/31/1600 7:00:00 PM
badPasswordTime               : 12/31/1600 7:00:00 PM
name                          : Amber Smith
distinguishedname             : CN=Amber
Smith,OU=Contractors,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
whencreated                   : 7/27/2020 7:35:52 PM
whenchanged                   : 8/12/2020 2:22:30 PM
samaccountname                : amber.smith
cn                            : {Amber Smith}
objectclass                   : {top, person, organizationalPerson, user}
displayname                   : Amber Smith
givenname                     : amber
codepage                      : 0
objectcategory                :
CN=Person,CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
dscorepropagationdata         : {7/30/2020 3:09:16 AM, 7/30/2020 3:09:16
AM, 7/28/2020 1:45:00 AM, 7/28/2020 1:34:13 AM
, 7/14/1601 10:36:49 PM}
usnchanged                    : 107351
```

```
instancetype                    : 4
logoncount                      : 4
msds-supportedencryptiontypes   : 0
badpwdcount                     : 0
usncreated                      : 18877
sn                              : smith
countrycode                     : 0
primarygroupid                  : 513
userprincipalname               : amber.smith@inlanefreight

<SNIP>
```

We can also begin gathering information about individual hosts using the `Get-DomainComputer` function.

```
PS C:\htb> Get-DomainComputer | select dnshostname, useraccountcontrol

dnshostname
useraccountcontrol
-----------                                                    -------
-----------
DC01.INLANEFREIGHT.LOCAL                 SERVER_TRUST_ACCOUNT,
TRUSTED_FOR_DELEGATION
EXCHG01.INLANEFREIGHT.LOCAL
WORKSTATION_TRUST_ACCOUNT
SQL01.INLANEFREIGHT.LOCAL   WORKSTATION_TRUST_ACCOUNT,
TRUSTED_TO_AUTH_FOR_DELEGATION
WS01.INLANEFREIGHT.LOCAL
WORKSTATION_TRUST_ACCOUNT
DC02.INLANEFREIGHT.LOCAL                       ACCOUNTDISABLE,
WORKSTATION_TRUST_ACCOUNT
```

GPO functions

```
Get-DomainGPO                           -   returns all GPOs or specific
GPO objects in AD
Get-DomainGPOLocalGroup                 -   returns all GPOs in a domain
that modify local group memberships through 'Restricted Groups' or Group
Policy preferences
Get-DomainGPOUserLocalGroupMapping      -   enumerates the machines where
a specific domain user/group is a member of a specific local group, all
through GPO correlation
Get-DomainGPOComputerLocalGroupMapping  -   takes a computer (or GPO)
```

```
object and determines what users/groups are in the specified local group
for the machine through GPO correlation
Get-DomainPolicy                        -   returns the default domain
policy or the domain controller policy for the current domain or a
specified domain/domain controller
```

Moving on to GPO functions, we can use `Get-DomainGPO` to return all Group Policy Objects (GPOs) names.

```
PS C:\htb> .\SharpView.exe Get-DomainGPO | findstr displayname

displayname                     : Default Domain Policy
displayname                     : Default Domain Controllers Policy
displayname                     : LAPS Install
displayname                     : LAPS
displayname                     : Disable LM Hash
displayname                     : Disable CMD.exe
displayname                     : Disallow removable media
displayname                     : Prevent software installs
displayname                     : Disable guest account
displayname                     : Disable SMBv1
displayname                     : Map home drive
displayname                     : Disable Forced Restarts
displayname                     : Screensaver
displayname                     : Applocker
displayname                     : Fine-grained password policy
displayname                     : Restrict Control Panel
displayname                     : User - MS Office
displayname                     : User - Browser Settings
displayname                     : Audit Policy
displayname                     : PowerShell logging
displayname                     : Disable Defender
```

We can also determine which GPOs map back to which hosts.

```
PS C:\htb> Get-DomainGPO -ComputerIdentity WS01 | select displayname

displayname
-----------
LAPS
Default Domain Policy
```

Computer Enumeration Functions

```
Get-NetLocalGroup                   -   enumerates the local groups on the
local (or remote) machine
Get-NetLocalGroupMember             -   enumerates members of a specific
local group on the local (or remote) machine
Get-NetShare                        -   returns open shares on the local
(or a remote) machine
Get-NetLoggedon                     -   returns users logged on the local
(or a remote) machine
Get-NetSession                      -   returns session information for
the local (or a remote) machine
Get-RegLoggedOn                     -   returns who is logged onto the
local (or a remote) machine through enumeration of remote registry keys
Get-NetRDPSession                   -   returns remote desktop/session
information for the local (or a remote) machine
Test-AdminAccess                    -   rests if the current user has
administrative access to the local (or a remote) machine
Get-NetComputerSiteName             -   returns the AD site where the
local (or a remote) machine resides
Get-WMIRegProxy                     -   enumerates the proxy server and
WPAD contents for the current user
Get-WMIRegLastLoggedOn              -   returns the last user who logged
onto the local (or a remote) machine
Get-WMIRegCachedRDPConnection       -   returns information about RDP
connections outgoing from the local (or remote) machine
Get-WMIRegMountedDrive              -   returns information about saved
network mounted drives for the local (or remote) machine
Get-WMIProcess                      -   returns a list of processes and
their owners on the local or remote machine
Find-InterestingFile                -   searches for files on the given
path that match a series of specified criteria
```


The computer enumeration functions can gather information about user sessions, test for local admin access, search for file shares and interesting files, and more. The `Test-AdminAccess` function can check if our current user has local admin rights on any remote hosts.

```
PS C:\htb> Test-AdminAccess -ComputerName SQL01

ComputerName IsAdmin
------------ -------
SQL01           True
```

We can use the `Net-Share` function to enumerate open shares on a remote computer. Shares can hold a wealth of information, and the importance of enumerating file shares should not be overlooked.

```
PS C:\htb> .\SharpView.exe Get-NetShare -ComputerName DC01

Name                           : ADMIN$
Type                           : 2147483648
Remark                         : Remote Admin
ComputerName                   : DC01

Name                           : C$
Type                           : 2147483648
Remark                         : Default share
ComputerName                   : DC01

Name                           : Department Shares
Type                           : 0
Remark                         :
ComputerName                   : DC01
```

Threaded 'Meta'-Functions

```
Find-DomainUserLocation        -   finds domain machines where
specific users are logged into
Find-DomainProcess             -   finds domain machines where
specific processes are currently running
Find-DomainUserEvent           -   finds logon events on the current
(or remote domain) for the specified users
Find-DomainShare               -   finds reachable shares on domain
machines
Find-InterestingDomainShareFile    -   searches for files matching
specific criteria on readable shares in the domain
Find-LocalAdminAccess          -   finds machines on the local domain
where the current user has local administrator access
Find-DomainLocalGroupMember    -   enumerates the members of
specified local group on machines in the domain
```

The 'meta' functions can be used to find where domain users are logged in, look for specific processes on remote hosts, find domain shares, find files on domain shares, and test where our current user has local admin rights. We can use the `Find-DomainUserLocation` function to find domain machines that users are logged into.

```
PS C:\htb> Find-DomainUserLocation

UserDomain      : INLANEFREIGHT
UserName        : Administrator
ComputerName    : DC01.INLANEFREIGHT.LOCAL
IPAddress       : 172.16.1.3
SessionFrom     :
SessionFromName :
LocalAdmin      :

UserDomain      : INLANEFREIGHT
UserName        : harry.jones
ComputerName    : SQL01.INLANEFREIGHT.LOCAL
IPAddress       : 172.16.1.30
SessionFrom     :
SessionFromName :
LocalAdmin      :

UserDomain      : INLANEFREIGHT
UserName        : cliff.moore
ComputerName    : WS01.INLANEFREIGHT.LOCAL
IPAddress       : 172.16.1.40
SessionFrom     :
SessionFromName :
LocalAdmin      :
```


Domain Trust Functions

```
Get-DomainTrust                     -   returns all domain trusts for the
current domain or a specified domain
Get-ForestTrust                     -   returns all forest trusts for the
current forest or a specified forest
Get-DomainForeignUser               -   enumerates users who are in groups
outside of the user's domain
Get-DomainForeignGroupMember        -   enumerates groups with users
outside of the group's domain and returns each foreign member
Get-DomainTrustMapping              -   this function enumerates all
trusts for the current domain and then enumerates all trusts for each
domain it finds
```

The domain trust functions provide us with the tools we need to enumerate information that can be used to mount cross-trust attacks. The most basic of these commands, `Get-DomainTrust` will return all domain trusts for our current domain.

```
PS C:\htb> Get-DomainTrust

SourceName      : INLANEFREIGHT.LOCAL
TargetName      : LOGISTICS.INLANEFREIGHT.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : WITHIN_FOREST
TrustDirection  : Bidirectional
WhenCreated     : 7/27/2020 2:06:07 AM
WhenChanged     : 7/27/2020 2:06:07 AM

SourceName      : INLANEFREIGHT.LOCAL
TargetName      : freightlogistics.local
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FOREST_TRANSITIVE
TrustDirection  : Bidirectional
WhenCreated     : 7/28/2020 4:46:40 PM
WhenChanged     : 7/28/2020 4:46:40 PM
```

Closing Thoughts

`PowerView` / `SharpView` can also be used to perform Kerberoasting and ASREPRoasting attacks and abuse Kerberos delegation, which will be covered in later modules.

`PowerView` can leverage token impersonation. Instead of spawning a new process, it enables running commands as another user by temporarily impersonating the user and then reverting to the current user. The credentials can be specified using the `-Credential` flag.

Note: If trying to remain stealthy, invoking the user impersonation does generate a logon event which could generate an alert if using a sensitive account with administrative level or equivalent privileges.

Module Lab Usage

To connect to the lab targets the best option is to use `xfreerdp`. You can connect with the following command:

```
xfreerdp /v:<TARGET IP> /u:htb-student /p:Academy_student_AD!
```

Note: When spawning your target, we ask you to wait for 3-5 minutes until the whole Active Directory lab spawns and all services start before attempting to connect via RDP.

Enumerating AD Users

When starting enumeration in an AD environment, arguably, the most important objects are domain users. Users have access to computers and are assigned permissions to perform a variety of functions throughout the domain. We need to control user accounts to move laterally and vertically within a network to reach the assessment goal.

Key AD User Data Points

We can use `PowerView` and `SharpView` to enumerate a wealth of information about AD users. We can start by getting a count of how many users are in the target domain. We switch between the two tools frequently in this course because it is important to know them both. Sometimes you won't be able to run powershell commands and at other times you may not be able to run executable's. Any time you see SharpView usage, it should be possible to do it in PowerView by just removing SharpView.exe. SharpView does not have the latest PowerSploit features, so it may not be possible to run PowerView commands within SharpView.

```
PS C:\htb> (Get-DomainUser).count

1038
```

Next, let's explore the `Get-DomainUser` function. If we provide the `-Help` flag to any `SharpView function, we can see all of the parameters that the function accepts.

```
PS C:\htb> .\SharpView.exe Get-DomainUser -Help

Get_DomainUser -Identity <String[]> -DistinguishedName <String[]> -
SamAccountName <String[]> -Name <String[]> -MemberDistinguishedName
<String[]> -MemberName <String[]> -SPN <Boolean> -AdminCount <Boolean> -
AllowDelegation <Boolean> -DisalowDelegation <Boolean> -TrustedToAuth
<Boolean> -PreauthNotRequired <Boolean> -KerberosPreauthNotRequired
<Boolean> -Noreauth <Boolean> -Domain <String> -LDAPFilter <String> -
Filter <String> -Properties <String[]> -SearchBase <String> -ADPath
<String> -Server <String> -DomainController <String> -SearchScope
<SearchScope> -ResultPageSize <Int32> -ServerTimLimit <Nullable`1> -
SecurityMasks <Nullable`1> -Tombstone <Boolean> -FindOne <Boolean> -
```

```
ReturnOne <Boolean> -Credential <NetworkCredential> -Raw <Boolean> -
UACFilter <UACEnum>
```

Below are some of the most important properties to gather about domain users. Let's take a look at the `harry.jones` user.

```
PS C:\htb> Get-DomainUser -Identity harry.jones -Domain
inlanefreight.local | Select-Object -Property
name,samaccountname,description,memberof,whencreated,pwdlastset,lastlogont
imestamp,accountexpires,admincount,userprincipalname,serviceprincipalname,
mail,useraccountcontrol

name                 : Harry Jones
samaccountname       : harry.jones
description          :
memberof             : {CN=Network
Team,CN=Users,DC=INLANEFREIGHT,DC=LOCAL, CN=Help Desk,OU=Microsoft
Exchange
                       Security Groups,DC=INLANEFREIGHT,DC=LOCAL,
CN=Security
                       Operations,CN=Users,DC=INLANEFREIGHT,DC=LOCAL,
CN=LAPS
                       Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL...}
whencreated          : 7/27/2020 7:35:59 PM
pwdlastset           : 7/30/2020 2:33:04 PM
lastlogontimestamp   : 8/9/2020 10:52:42 PM
accountexpires       : 12/31/1600 7:00:00 PM
admincount           : 1
userprincipalname    : harry.jones@inlanefreight
serviceprincipalname :
mail                 :
useraccountcontrol   : PASSWD_NOTREQD, NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD
```

It is useful to enumerate these properties for ALL domain users and export them to a CSV file for offline processing.

```
PS C:\htb> Get-DomainUser * -Domain inlanefreight.local | Select-Object -
Property
name,samaccountname,description,memberof,whencreated,pwdlastset,lastlogont
imestamp,accountexpires,admincount,userprincipalname,serviceprincipalname,
mail,useraccountcontrol | Export-Csv .\inlanefreight_users.csv -
NoTypeInformation
```

Once we have gathered information on all users, we can begin to perform more specific user enumeration by obtaining a list of users that do not require Kerberos pre-authentication and can be subjected to an ASREPRoast attack.

```
PS C:\htb> .\SharpView.exe Get-DomainUser -KerberosPreauthNotRequired -
Properties samaccountname,useraccountcontrol,memberof

[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainUser] Searching for user accounts that do not require kerberos
preauthenticate
[Get-DomainUser] filter string: (&(samAccountType=805306368)
(userAccountControl:1.2.840.113556.1.4.803:=4194304))
useraccountcontrol             : PASSWD_NOTREQD, NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD, DONT_REQ_PREAUTH
samaccountname                 : amber.smith
memberof                       : {CN=Help Desk,OU=Microsoft Exchange
Security Groups,DC=INLANEFREIGHT,DC=LOCAL}

useraccountcontrol             : PASSWD_NOTREQD, NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD, DONT_REQ_PREAUTH
samaccountname                 : jenna.smith
memberof                       : {CN=Schema
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
```

Let's also gather information about users with Kerberos constrained delegation.

```
PS C:\htb> .\SharpView.exe Get-DomainUser -TrustedToAuth -Properties
samaccountname,useraccountcontrol,memberof

[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainUser] Searching for users that are trusted to authenticate for
other principals
[Get-DomainUser] filter string: (&(samAccountType=805306368)(msds-
allowedtodelegateto=*))
useraccountcontrol             : NORMAL_ACCOUNT
samaccountname                 : sqlprod
memberof                       : {CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}

useraccountcontrol             : PASSWD_NOTREQD, NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD
samaccountname                 : adam.jones
```

While we're at it, we can look for users that allow unconstrained delegation.

```
PS C:\htb> .\SharpView.exe Get-DomainUser -LDAPFilter "
(userAccountControl:1.2.840.113556.1.4.803:=524288)"

[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainUser] Using additional LDAP filter:
(userAccountControl:1.2.840.113556.1.4.803:=524288)
[Get-DomainUser] filter string: (&(samAccountType=805306368)
(userAccountControl:1.2.840.113556.1.4.803:=524288))
objectsid                      : {S-1-5-21-2974783224-3764228556-
2640795941-1110}
samaccounttype                 : USER_OBJECT
objectguid                     : f71224a5-baa7-4aec-bfe9-56778184dc63
useraccountcontrol             : NORMAL_ACCOUNT, TRUSTED_FOR_DELEGATION
accountexpires                 : 12/31/1600 7:00:00 PM
lastlogon                      : 12/31/1600 7:00:00 PM
pwdlastset                     : 7/27/2020 2:46:20 PM
lastlogoff                     : 12/31/1600 7:00:00 PM
badPasswordTime                : 12/31/1600 7:00:00 PM
name                           : sqldev
distinguishedname              : CN=sqldev,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
whencreated                    : 7/27/2020 6:46:20 PM
whenchanged                    : 8/14/2020 1:30:37 PM
samaccountname                 : sqldev
memberof                       : {CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
cn                             : {sqldev}
objectclass                    : {top, person, organizationalPerson, user}
ServicePrincipalName           : CIFS/roguecomputer.inlanefreight.local
logoncount                     : 0
codepage                       : 0
objectcategory                 :
CN=Person,CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
dscorepropagationdata          : {7/30/2020 3:09:16 AM, 7/30/2020 3:09:16
AM, 7/28/2020 1:45:00 AM, 7/28/2020 1:34:13 AM
, 7/14/1601 10:36:49 PM}
usnchanged                     : 110783
instancetype                   : 4
badpwdcount                    : 0
usncreated                     : 14648
countrycode                    : 0
primarygroupid                 : 513
```

We can also check for any domain users with sensitive data such as a password stored in the description field.

```
PS C:\htb> Get-DomainUser -Properties samaccountname,description | Where
{$_.description -ne $null}

samaccountname description
-------------- -----------
Administrator  Built-in account for administering the computer/domain
Guest          Built-in account for guest access to the computer/domain
DefaultAccount A user account managed by the system.
krbtgt         Key Distribution Center Service Account
svc-sccm       **Do not change password** 03/04/2015 N3ssu$_svc2014!
```

Next, let's enumerate any users with Service Principal Names (SPNs) that could be subjected to a Kerberoasting attack.

```
PS C:\htb> .\SharpView.exe Get-DomainUser -SPN -Properties
samaccountname,memberof,serviceprincipalname

[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainUser] Searching for non-null service principal names
[Get-DomainUser] filter string: (&(samAccountType=805306368)
(servicePrincipalName=*))
samaccountname                : sqldev
memberof                      : {CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
ServicePrincipalName          : CIFS/roguecomputer.inlanefreight.local

samaccountname                : adam.jones
ServicePrincipalName          : IIS_dev/inlanefreight.local:80

samaccountname                : krbtgt
memberof                      : {CN=Denied RODC Password Replication
Group,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
ServicePrincipalName          : kadmin/changepw

samaccountname                : sqlqa
memberof                      : {CN=Domain
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
ServicePrincipalName          : MSSQL_svc_qa/inlanefreight.local:1443

samaccountname                : sql-test
ServicePrincipalName          : MSSQL_svc_test/inlanefreight.local:1443

samaccountname                : sqlprod
```

```
memberof                       : {CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
ServicePrincipalName           : MSSQLSvc/sql01:1433
```

Finally, we can enumerate any users from other (foreign) domains with group membership within any groups in our current domain. We can see that the user `harry.jones` from the `FREIGHTLOGISTICS.LOCAL` domain is in our current domain's `administrators` group. If we compromise the current domain, we may obtain credentials for this user from the NTDS database and authenticate into the `FREIGHTLOGISTICS.LOCAL` domain.

```
PS C:\htb> Find-ForeignGroup

GroupDomain             : INLANEFREIGHT.LOCAL
GroupName               : Administrators
GroupDistinguishedName  :
CN=Administrators,CN=Builtin,DC=INLANEFREIGHT,DC=LOCAL
MemberDomain            : INLANEFREIGHT.LOCAL
MemberName              : S-1-5-21-888139820-103978830-333442103-1602
MemberDistinguishedName : CN=S-1-5-21-888139820-103978830-333442103-
1602,CN=ForeignSecurityPrincipals,DC=INLANEFREIGHT,
                          DC=LOCAL
```

```
PS C:\htb> Convert-SidToName S-1-5-21-888139820-103978830-333442103-1602

FREIGHTLOGISTIC\harry.jones
```


Another useful command is checking for users with Service Principal Names (SPNs) set in other domains that we can authenticate into via inbound or bi-directional trust relationships with forest-wide authentication allowing all users to authenticate across a trust or selective-authentication set up which allows specific users to authenticate. Here we can see one account in the `FREIGHTLOGISTICS.LOCAL` domain, which could be leveraged to Kerberoast across the forest trust.

```
PS C:\htb> Get-DomainUser -SPN -Domain freightlogistics.local | select
samaccountname,memberof,serviceprincipalname | fl

samaccountname       : krbtgt
memberof             : CN=Denied RODC Password Replication
Group,CN=Users,DC=freightlogistics,DC=local
serviceprincipalname : kadmin/changepw

samaccountname       : svc_azure
memberof             : CN=Account
```

```
Operators,CN=Builtin,DC=freightlogistics,DC=local
serviceprincipalname : freightlogistics/azureconnect:443
```

Password Set Times

Analyzing the Password Set times is incredibly important when performing password sprays. Organizations are much more likely to find an automated password spray across all accounts than at a few guesses towards a small group of accounts.

- If you see a `several passwords set at the same time`, this indicates they were set by the Help Desk and may be the same. Because of Password Lockout Policies, you may not be able to exceed four failed passwords in fifteen minutes. However, if you think the password is the same across 20 accounts, for one user, you can guess passwords along the line of "Password2020" for a different use, you can use the company name like "Freight2020!".
- Additionally, if you see the password was set in July of 2019; then you can normally exclude "2020" from your password guessing and probably shouldn't guess variations that wouldn't make sense, such as "Winter2019."
- If you see an `old password that was set 2 years ago`, chances are this password is weak and also one of the first accounts I would recommend guessing the password to before launching a large Password Spray.
- In most organizations, administrators have multiple accounts. If you see the administrator changing his "user account" around the same time as his "Administrator Account", they are highly likely to use the same password for both accounts.

The following command will display all password set times.

```
PS C:\htb> Get-DomainUser -Properties samaccountname,pwdlastset,lastlogon
-Domain InlaneFreight.local | select samaccountname, pwdlastset, lastlogon
| Sort-Object -Property pwdlastset
```

If you want only to show passwords set before a certain date:

```
PS C:\htb> Get-DomainUser -Properties samaccountname,pwdlastset,lastlogon
-Domain InlaneFreight.local | select samaccountname, pwdlastset, lastlogon
| where { $_.pwdlastset -lt (Get-Date).addDays(-90) }
```

Blue team tip: Whenever you deal with a compromise or complete a Penetration Test. It is always a good idea to use the above command to verify all passwords have been rotated!

You should never have passwords older than a year in your Active Directory.

Next Steps

Now that we have gathered a wealth of information about domain users let's look at group memberships to map out the domain further.

Enumerating AD Groups

Armed with the domain user information, it is next important to gather AD group information to see what privileges members of a group may have and even find nested groups or issues with group membership that could lead to unintended rights.

Domain Groups

A quick check shows that our target domain, `INLANEFREIGHT.LOCAL` has 72 groups.

```
PS C:\htb> Get-DomainGroup -Properties Name

name
----
Administrators
Users
Guests
Print Operators
Backup Operators
Replicator
Remote Desktop Users
Network Configuration Operators
Performance Monitor Users
Performance Log Users
Distributed COM Users
IIS_IUSRS
Cryptographic Operators
Event Log Readers
Certificate Service DCOM Access
RDS Remote Access Servers
RDS Endpoint Servers
RDS Management Servers
Hyper-V Administrators
Access Control Assistance Operators
```

```
Remote Management Users
System Managed Accounts Group
Storage Replica Administrators
Domain Computers
Domain Controllers
Schema Admins
Enterprise Admins
Cert Publishers
Domain Admins
Domain Users
Domain Guests
Group Policy Creator Owners
RAS and IAS Servers
Server Operators
Account Operators
Pre-Windows 2000 Compatible Access
Incoming Forest Trust Builders
Windows Authorization Access Group
Terminal Server License Servers
Allowed RODC Password Replication Group
Denied RODC Password Replication Group
Read-only Domain Controllers
Enterprise Read-only Domain Controllers
Cloneable Domain Controllers
Protected Users
Key Admins
Enterprise Key Admins
DnsAdmins
DnsUpdateProxy
LAPS Admins
Security Operations
Organization Management
Recipient Management
View-Only Organization Management
Public Folder Management
UM Management
Help Desk
Records Management
Discovery Management
Server Management
Delegated Setup
Hygiene Management
Compliance Management
Security Reader
Security Administrator
Exchange Servers
Exchange Trusted Subsystem
Managed Availability Servers
Exchange Windows Permissions
ExchangeLegacyInterop
```

```
Exchange Install Domain Servers
Network Team
```

Let's grab a full listing of the group names. Many of these are built-in, standard AD groups. The presence of some group shows us that Microsoft Exchange is present in the environment. An Exchange installation adds several groups to AD, some of which such as `Exchange Trusted Subsystem` and `Exchange Windows Permissions` are considered high-value targets due to the permissions that membership in these groups grants a user or computer. Other groups such as `Protected Users`, `LAPS Admins`, `Help Desk`, and `Security Operations` should be noted down for review.

We can use `Get-DomainGroupMember` to examine group membership in any given group. Again, when using the `SharpView` function for this, we can pass the `-Help` flag to see all of the parameters that this function accepts.

```
PS C:\htb> .\SharpView.exe Get-DomainGroupMember -Help

Get_DomainGroupMember -Identity <String[]> -DistinguishedName <String[]> -
SamAccountName <String[]> -Name <String[]> -MemberDistinguishedName
<String[]> -MemberName <String[]> -Domain <String> -Recurse <Boolean> -
RecurseUsingMatchingRule <Boolean> -LDAPFilter <String> -Filter <String> -
SearchBase <String> -ADSPath <String> -Server <String> -DomainController
<String> -SearchScope <SearchScope> -ResultPageSize <Int32> -
ServerTimeLimit <Nullable`1> -SecurityMasks <Nullable`1> -Tombstone
<Boolean> -Credential <NetworkCredential>
```


A quick examination of the `Help Desk` group shows us that there are two members.

```
PS C:\htb> .\SharpView.exe Get-DomainGroupMember -Identity 'Help Desk'

[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainGroupMember] Get-DomainGroupMember filter string: (&
(objectCategory=group)(|(samAccountName=Help Desk)))
[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainObject] Extracted domain 'INLANEFREIGHT.LOCAL' from 'CN=Harry
Jones,OU=Network Ops,OU=IT,OU=Employees,DC=INLA
NEFREIGHT,DC=LOCAL'
[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainObject] Get-DomainComputer filter string: (&(|
(distinguishedname=CN=Harry Jones,OU=Network Ops,OU=IT,OU=Emplo
yees,DC=INLANEFREIGHT,DC=LOCAL)))
[Get-DomainSearcher] search base:
```

```
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainObject] Extracted domain 'INLANEFREIGHT.LOCAL' from 'CN=Amber
Smith,OU=Contractors,OU=Employees,DC=INLANEFREI
GHT,DC=LOCAL'
[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainObject] Get-DomainComputer filter string: (&(|
(distinguishedname=CN=Amber Smith,OU=Contractors,OU=Employees,D
C=INLANEFREIGHT,DC=LOCAL)))
GroupDomain                   : INLANEFREIGHT,LOCAL
GroupName                     : Help Desk
GroupDistinguishedName        : CN=Help Desk,OU=Microsoft Exchange
Security Groups,DC=INLANEFREIGHT,DC=LOCAL
MemberDomain                  : INLANEFREIGHT.LOCAL
MemberName                    : harry.jones
MemberDistinguishedName       : CN=Harry Jones,OU=Network
Ops,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
MemberObjectClass             : user
MemberSID                     : S-1-5-21-2974783224-3764228556-
2640795941-2040

GroupDomain                   : INLANEFREIGHT,LOCAL
GroupName                     : Help Desk
GroupDistinguishedName        : CN=Help Desk,OU=Microsoft Exchange
Security Groups,DC=INLANEFREIGHT,DC=LOCAL
MemberDomain                  : INLANEFREIGHT.LOCAL
MemberName                    : amber.smith
MemberDistinguishedName       : CN=Amber
Smith,OU=Contractors,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
MemberObjectClass             : user
MemberSID                     : S-1-5-21-2974783224-3764228556-
2640795941-1859
```


Protected Groups

Next, we can look for all AD groups with the `AdminCount` attribute set to 1, signifying that this is a protected group.

```
PS C:\htb> .\SharpView.exe Get-DomainGroup -AdminCount

[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainGroup] Searching for adminCount=1
[Get-DomainGroup] filter string: (&(objectCategory=group)(admincount=1))
objectsid                     : {S-1-5-32-544}
```

```
grouptype                       : CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE,
SECURITY
samaccounttype                  : ALIAS_OBJECT
objectguid                      : 4f86f787-7173-4a34-a317-3f69e2263f0d
name                            : Administrators
distinguishedname               :
CN=Administrators,CN=Builtin,DC=INLANEFREIGHT,DC=LOCAL
whencreated                     : 7/26/2020 8:13:52 PM
whenchanged                     : 8/23/2020 4:28:44 AM
samaccountname                  : Administrators
member                          : {CN=S-1-5-21-888139820-103978830-
333442103-1602,CN=ForeignSecurityPrincipals,D
REIGHT,DC=LOCAL, CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL,
CN=Enterprise Admins,CN=Users,DC=INLANEFR
LOCAL, CN=Administrator,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
cn                              : {Administrators}
objectclass                     : {top, group}
objectcategory                  :
CN=Group,CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
usnchanged                      : 124889
description                     : Administrators have complete and
unrestricted access to the computer/domain
instancetype                    : 4
usncreated                      : 8200
admincount                      : 1
iscriticalsystemobject          : True
systemflags                     : -1946157056
dscorepropagationdata           : {7/30/2020 3:52:30 AM, 7/30/2020 3:09:16
AM, 7/30/2020 3:09:16 AM, 7/28/2020 1
, 1/1/1601 12:00:00 AM}

objectsid                       : {S-1-5-32-550}
grouptype                       : CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE,
SECURITY
samaccounttype                  : ALIAS_OBJECT
objectguid                      : ae974502-7850-44ab-9518-f909f9526daa
name                            : Print Operators
distinguishedname               : CN=Print
Operators,CN=Builtin,DC=INLANEFREIGHT,DC=LOCAL
whencreated                     : 7/26/2020 8:13:52 PM
whenchanged                     : 7/30/2020 3:52:30 AM
samaccountname                  : Print Operators
cn                              : {Print Operators}
objectclass                     : {top, group}
iscriticalsystemobject          : True
usnchanged                      : 61476
instancetype                    : 4
usncreated                      : 8212
objectcategory                  :
CN=Group,CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
```

```
admincount                      : 1
description                     : Members can administer printers installed
on domain controllers
systemflags                     : -1946157056
dscorepropagationdata           : {7/30/2020 3:52:30 AM, 7/30/2020 3:09:16
AM, 7/30/2020 3:09:16 AM, 7/28/2020 1
, 1/1/1601 12:00:00 AM}

<...SNIP...>
```

Another important check is to look for any managed security groups. These groups have delegated non-administrators the right to add members to AD security groups and distribution groups and is set by modifying the `managedBy` attribute. This check looks to see if a group has a manager set and if the user can add users to the group. This could be useful for lateral movement by gaining us access to additional resources. First, let's take a look at the list of managed security groups.

```
PS C:\htb> Find-ManagedSecurityGroups | select GroupName

GroupName
---------
Security Operations
Organization Management
Recipient Management
View-Only Organization Management
Public Folder Management
UM Management
Help Desk
Records Management
Discovery Management
Server Management
Delegated Setup
Hygiene Management
Compliance Management
Security Reader
Security Administrator
```

Next, let's look at the `Security Operations` group and see if the group has a manager set. We can see that the user `joe.evans` is set as the group manager.

```
PS C:\htb> Get-DomainManagedSecurityGroup

GroupName                : Security Operations
GroupDistinguishedName   : CN=Security
Operations,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
```

```
ManagerName               : joe.evans
ManagerDistinguishedName : CN=Joe
Evans,OU=Security,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ManagerType               : User
ManagerCanWrite           : UNKNOWN

GroupName                 : Organization Management
GroupDistinguishedName    : CN=Organization Management,OU=Microsoft
Exchange Security Groups,DC=INLANEFREIGHT,DC=LOCAL
ManagerName               : Organization Management
ManagerDistinguishedName : CN=Organization Management,OU=Microsoft
Exchange Security Groups,DC=INLANEFREIGHT,DC=LOCAL
ManagerType               : Group
ManagerCanWrite           : UNKNOWN

GroupName                 : Recipient Management
GroupDistinguishedName    : CN=Recipient Management,OU=Microsoft Exchange
Security Groups,DC=INLANEFREIGHT,DC=LOCAL
ManagerName               : Organization Management
ManagerDistinguishedName : CN=Organization Management,OU=Microsoft
Exchange Security Groups,DC=INLANEFREIGHT,DC=LOCAL
ManagerType               : Group
ManagerCanWrite           : UNKNOWN

GroupName                 : View-Only Organization Management
GroupDistinguishedName    : CN=View-Only Organization
Management,OU=Microsoft Exchange Security
                            Groups,DC=INLANEFREIGHT,DC=LOCAL
ManagerName               : Organization Management
ManagerDistinguishedName : CN=Organization Management,OU=Microsoft
Exchange Security Groups,DC=INLANEFREIGHT,DC=LOCAL
ManagerType               : Group
ManagerCanWrite           : UNKNOWN

<...SNIP...>
```

Enumerating the ACLs set on this group, we can see that this user has `GenericWrite` privileges meaning that this user can modify group membership (add or remove users). If we gain control of this user account, we can add this account or any other account that we control to the group and inherit any privileges that it has in the domain.

```
PS C:\htb> ConvertTo-SID joe.evans

S-1-5-21-2974783224-3764228556-2640795941-1238
```

```
PS C:\htb> $sid = ConvertTo-SID joe.evans
PS C:\htb> Get-DomainObjectAcl -Identity 'Security Operations' | ?{
$_.SecurityIdentifier -eq $sid}

ObjectDN              : CN=Security
Operations,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
ObjectSID             : S-1-5-21-2974783224-3764228556-2640795941-2127
ActiveDirectoryRights : ListChildren, ReadProperty, GenericWrite
BinaryLength          : 36
AceQualifier          : AccessAllowed
IsCallback            : False
OpaqueLength          : 0
AccessMask            : 131132
SecurityIdentifier    : S-1-5-21-2974783224-3764228556-2640795941-1238
AceType               : AccessAllowed
AceFlags              : ContainerInherit
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None
AuditFlags            : None
```


Local Groups

It is also important to check local group membership. Is our current user local admin or part of local groups on any hosts? We can get a list of the local groups on a host using `Get-NetLocalGroup`.

```
PS C:\htb> Get-NetLocalGroup -ComputerName WS01 | select GroupName

GroupName
---------
Access Control Assistance Operators
Administrators
Backup Operators
Certificate Service DCOM Access
Cryptographic Operators
Distributed COM Users
Event Log Readers
Guests
Hyper-V Administrators
IIS_IUSRS
Network Configuration Operators
Performance Log Users
Performance Monitor Users
```

```
Power Users
Print Operators
RDS Endpoint Servers
RDS Management Servers
RDS Remote Access Servers
Remote Desktop Users
Remote Management Users
Replicator
Storage Replica Administrators
System Managed Accounts Group
Users
```

We can also enumerate the local group members on any given host using the `Get-NetLocalGroupMember` function.

```
PS C:\htb> .\SharpView.exe Get-NetLocalGroupMember -ComputerName WS01

ComputerName                  : WS01
GroupName                     : Administrators
MemberName                    : WS01\Administrator
SID                           : S-1-5-21-3098764391-2955872655-
3533479253-500
IsGroup                       : False
IsDomain                      : false

ComputerName                  : WS01
GroupName                     : Administrators
MemberName                    : INLANEFREIGHT\
SID                           : S-1-5-21-2974783224-3764228556-
2640795941-512
IsGroup                       : False
IsDomain                      : true

ComputerName                  : WS01
GroupName                     : Administrators
MemberName                    : INLANEFREIGHT\
SID                           : S-1-5-21-2974783224-3764228556-
2640795941-2040
IsGroup                       : False
IsDomain                      : true

ComputerName                  : WS01
GroupName                     : Administrators
MemberName                    : INLANEFREIGHT\
SID                           : S-1-5-21-2974783224-3764228556-
2640795941-513
IsGroup                       : False
```

```
IsDomain                      : true
```

We see one non-RID 500 user in the local administrators group and use the `Convert-SidToName` function to convert the SID and reveal the `harry.jones` user.

```
PS C:\htb> Convert-SidToName S-1-5-21-2974783224-3764228556-2640795941-2040

INLANEFREIGHT\harry.jones
```

We use this same function to check all the hosts that a given user has local admin access, though this can be done much quicker with another `PowerView` / `SharpView` function that we will cover later in this module.

```
PS C:\htb> $sid = Convert-NameToSid harry.jones
PS C:\htb> $computers = Get-DomainComputer -Properties dnshostname | select -ExpandProperty dnshostname
PS C:\htb> foreach ($line in $computers) {Get-NetLocalGroupMember -ComputerName $line | ? {$_.SID -eq $sid}}

ComputerName : WS01.INLANEFREIGHT.LOCAL
GroupName    : Administrators
MemberName   : INLANEFREIGHT\harry.jones
SID          : S-1-5-21-2974783224-3764228556-2640795941-2040
IsGroup      : False
IsDomain     : True
```

Pulling Date User Added to Group

PowerView cannot pull the date when a user was added to a group, but since we are enumerating groups here, we wanted to include it. This information isn't too helpful for an attacker. Still, adding information that can aid in Incident Response will make your report stand out and hopefully lead to repeat business. Having this information, if you notice a strange user as part of a group, defenders can search for Event ID 4728/ 4738 on that date to find out who added the user, or Event ID 4624 since the date added to see if anyone has logged in.

The module we generally use to pull this information is called `Get-ADGroupMemberDate` and can be downloaded here. Load this module up the same way you would `PowerView`.

Then run `Get-ADGroupMemberDate -Group "Help Desk" -DomainController DC01.INLANEFREIGHT.LOCAL`, if there is a specific user you want to pull, we recommend running `Get-ADGroupMemberDate -Group "Help Desk" -DomainController DC01.INLANEFREIGHT.LOCAL | ? { ($_.Username -match 'harry.jones') -And ($_.State -NotMatch 'ABSENT') }`

Continuing on

We have now covered AD users and groups. Let's start piecing things together and take a look at some key enumeration techniques around domain computers.

Enumerating AD Computers

Now that we have gathered user and group information, we need to find out information about the various hosts our target users can log in to, and if gaining SYSTEM access on any given host will open up different attack paths.

Domain Computer Information

We can use the `Get-DomainComputer` function to enumerate many details about domain computers.

```
PS C:\htb> .\SharpView.exe Get-DomainComputer -Help

Get_DomainComputer -Identity <String[]> -SamAccountName <String[]> -
Unconstrained <Boolean> -TrustedToAuth <Boolean> -Printers <Boolean> -SPN
<String> -ServicePrincipalName <String> -OperatingSystem <String> -
ServicePack <String> -SiteName <String> -Ping <Boolean> -Domain <String> -
LDAPFilter <String> -Filter <String> -Properties <String[]> -SearchBase
<String> -ADSPath <String> -Server <String> -DomainController <String> -
SearchScope <SearchScope> -ResultPageSize <Int32> -ServerTimeLimit
<Nullable`1> -SecurityMasks <Nullable`1> -Tombstone <Boolean> -FindOne
<Boolean> -ReturnOne <Boolean> -Credential <NetworkCredential> -Raw
<Boolean> -UACFilter <UACEnum>
```

Some of the most useful information we can gather is the hostname, operating system, and User Account Control (UAC) attributes.

```
PS C:\htb>.\SharpView.exe Get-DomainComputer -Properties
dnshostname,operatingsystem,lastlogontimestamp,useraccountcontrol

[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainComputer] Get-DomainComputer filter string: (&
(samAccountType=805306369))
useraccountcontrol             : SERVER_TRUST_ACCOUNT,
TRUSTED_FOR_DELEGATION
lastlogontimestamp             : 8/17/2020 6:43:25 AM
dnshostname                    : DC01.INLANEFREIGHT.LOCAL
operatingsystem                : Windows Server 2016 Standard

useraccountcontrol             : WORKSTATION_TRUST_ACCOUNT
lastlogontimestamp             : 8/15/2020 9:49:12 PM
dnshostname                    : EXCHG01.INLANEFREIGHT.LOCAL
operatingsystem                : Windows Server 2016 Standard

useraccountcontrol             : WORKSTATION_TRUST_ACCOUNT,
TRUSTED_TO_AUTH_FOR_DELEGATION
lastlogontimestamp             : 8/15/2020 7:42:00 PM
dnshostname                    : SQL01.INLANEFREIGHT.LOCAL
operatingsystem                : Windows Server 2016 Standard

useraccountcontrol             : WORKSTATION_TRUST_ACCOUNT
lastlogontimestamp             : 8/15/2020 5:55:24 PM
dnshostname                    : WS01.INLANEFREIGHT.LOCAL
operatingsystem                : Windows Server 2016 Standard

useraccountcontrol             : ACCOUNTDISABLE, WORKSTATION_TRUST_ACCOUNT
lastlogontimestamp             : 7/26/2020 9:58:15 PM
dnshostname                    : DC02.INLANEFREIGHT.LOCAL
```


Let's save this data to a CSV for our records using `PowerView`.

```
PS C:\htb> Get-DomainComputer -Properties
dnshostname,operatingsystem,lastlogontimestamp,useraccountcontrol |
Export-Csv .\inlanefreight_computers.csv -NoTypeInformation
```

Finding Exploitable Machines

The most obvious thing in the above screenshot is within the "User Account Control" setting, and we will get into that shortly. However, tools like `Bloodhound` will quickly point this setting

out, and it may become uncommon to find in organizations that have regular penetration tests performed. The following flags can be combined to help come up with attacks:

- `LastLogonTimeStamp`: This field exists to let administrators find stale machines. If this field is 90 days old for a machine, it has not been turned on and is missing both operating system and application patches. Due to this, administrators may want to automatically disable machines upon this field hitting 90 days of age. Attackers can use this field in combination with other fields such as `Operating System` or `When Created` to identify targets.
- `OperatingSystem`: This lists the Operating System. The obvious attack path is to find a Windows 7 box that is still active (LastLogonTimeStamp) and try attacks like Eternal Blue. Even if Eternal Blue is not applicable, older versions of Windows are ideal spots to work from as there are fewer logging/antivirus capabilities on older Windows. It's also important to know the differences between flavors of Windows. For example, Windows 10 Enterprise is the only version that comes with "Credential Guard" (Prevents Mimikatz from Stealing Passwords) Enabled by default. If you see Administrators logging into Windows 10 Professional and Windows 10 Enterprise, the Professional box should be targeted.
- `WhenCreated`: This field is created when a machine joins Active Directory. The older the box is, the more likely it is to deviate from the "Standard Build." Old workstations could have weaker local administration passwords, more local admins, vulnerable software, more data, etc.

Computer Attacks

We can see if any computers in the domain are configured to allow unconstrained delegation and find one, the domain controller, which is standard.

```
PS C:\htb> .\SharpView.exe Get-DomainComputer -Unconstrained -Properties
dnshostname,useraccountcontrol

[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainComputer] Searching for computers with for unconstrained
delegation
[Get-DomainComputer] Get-DomainComputer filter string: (&
(samAccountType=805306369)(userAccountControl:1.2.840.113556.1.
4.803:=524288))
useraccountcontrol             : SERVER_TRUST_ACCOUNT,
TRUSTED_FOR_DELEGATION
dnshostname                    : DC01.INLANEFREIGHT.LOCAL
```

Finally, we can check for any hosts set up to allow for constrained delegation.

```
PS C:\htb> Get-DomainComputer -TrustedToAuth | select -Property
dnshostname,useraccountcontrol

dnshostname
useraccountcontrol
-----------                                                    -------
-----------
EXCHG01.INLANEFREIGHT.LOCAL
WORKSTATION_TRUST_ACCOUNT
SQL01.INLANEFREIGHT.LOCAL   WORKSTATION_TRUST_ACCOUNT,
TRUSTED_TO_AUTH_FOR_DELEGATION
```

Upwards and Onwards

Now let's study the `access control lists` (`ACLs`) set up in the domain to see how we can further leverage any access we have obtained.

Enumerating Domain ACLs

Access Control Lists (ACLs)

Access Control List (ACL) settings themselves are called Access Control Entries (ACEs). Each ACE refers back to a user, group, or process (security principal) and defines the principal's rights.

There are two types of ACLs.

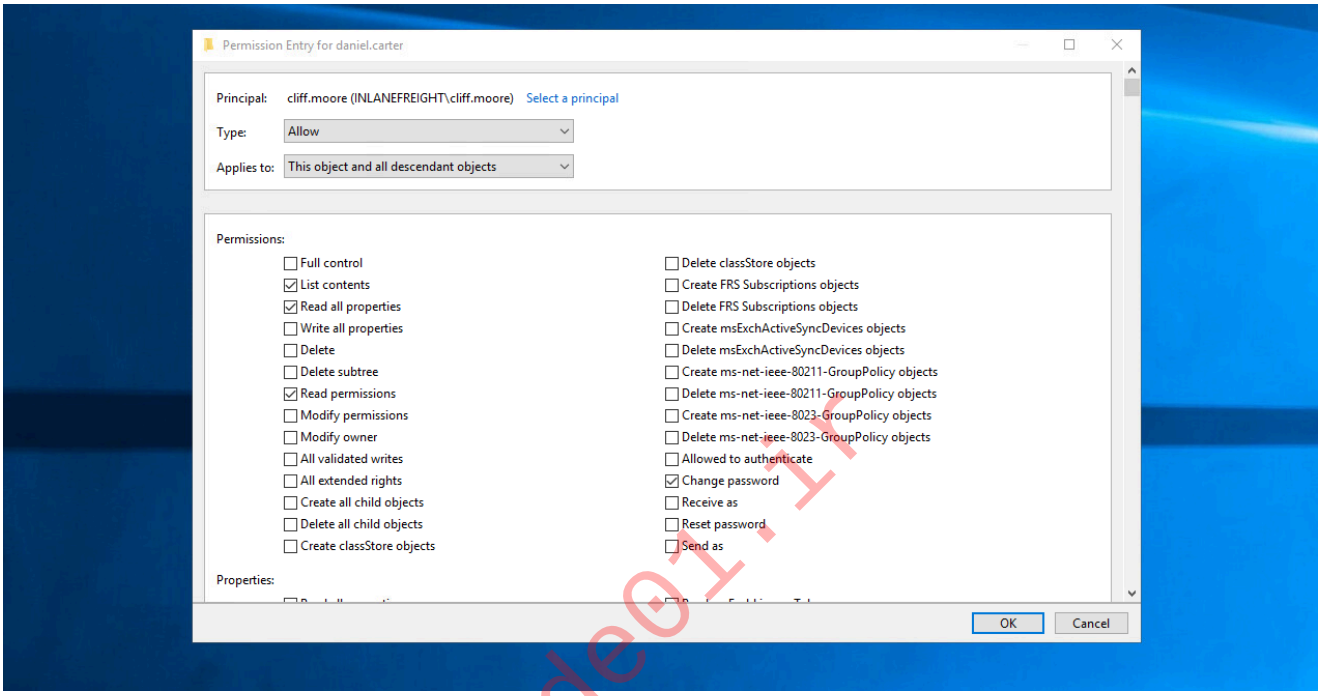
ACL	Description
`Discretionary Access Control List (DACL)`	This defines which security principals are granted or denied access to an object.
`System Access Control Lists (SACL)`	These allow administrators to log access attempts made to secured objects.

ACL (mis)-configurations may allow for chained object-to-object control. We can visualize unrolled membership of target groups, so-called `derivative admins`, who can derive admin rights from exploiting an AD attack chain.

AD Attack chains may include the following components:

- "Unprivileged" users (shadow admins) having administrative access on member servers or workstations.
- Privileged users having a logon session on these workstations and member servers.
- Other forms of object-to-object control include force password change, add group member, change owner, write ACE, and full control.

Below is an example of just some of the ACLs that can be set on a user object.



ACL Abuse

Why do we care about ACLs? ACL abuse is a powerful attack vector for us as penetration testers. These types of misconfigurations often go unnoticed in corporate environments because they can be difficult to monitor and control. An organization may be unaware of overly permissive ACL settings for years before (hopefully) we discover them. Below are some of the example Active Directory object security permissions (supported by `BloodHound` and abusable with `SharpView` / `PowerView`):

- ForceChangePassword abused with `Set-DomainUserPassword`
- Add Members abused with `Add-DomainGroupMember`
- GenericAll abused with `Set-DomainUserPassword` or `Add-DomainGroupMember`
- GenericWrite abused with `Set-DomainObject`
- WriteOwner abused with `Set-DomainObjectOwner`
- WriteDACL abused with `Add-DomainObjectACL`

- AllExtendedRights abused with `Set-DomainUserPassword` or `Add-DomainGroupMember`

Enumerating ACLs with Built-In Cmdlets

We can use the built-in `Get-ADUser` cmdlet to enumerate ACLs. For example, we can look at the ACL for a single domain user `daniel.carter` with this command.

```
PS C:\htb> (Get-ACL "AD:$((Get-ADUser
daniel.carter).distinguishedname)").access  | ? {$_.IdentityReference -eq
"INLANEFREIGHT\cliff.moore"}

ActiveDirectoryRights : ReadProperty, WriteProperty, GenericExecute
InheritanceType       : All
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : INLANEFREIGHT\cliff.moore
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None

ActiveDirectoryRights : ExtendedRight
InheritanceType       : All
ObjectType            : ab721a53-1e2f-11d0-9819-00aa0040529b
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : ObjectAceTypePresent
AccessControlType     : Allow
IdentityReference     : INLANEFREIGHT\cliff.moore
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None
```

We can drill down further on this user to find all users with `WriteProperty` or `GenericAll` rights over the target user.

```
PS C:\htb> (Get-ACL "AD:$((Get-ADUser
daniel.carter).distinguishedname)").access  | ? {$_.ActiveDirectoryRights
-match "WriteProperty" -or $_.ActiveDirectoryRights -match "GenericAll"} |
Select IdentityReference,ActiveDirectoryRights -Unique | ft -W

IdentityReference
ActiveDirectoryRights
```

```
-----------------
----------------------
BUILTIN\Administrators                          CreateChild, DeleteChild,
Self, WriteProperty, ExtendedRight, Delete, GenericRead,

WriteDacl, WriteOwner
INLANEFREIGHT\Domain Admins                  CreateChild, DeleteChild,
Self, WriteProperty, ExtendedRight, GenericRead, WriteDacl,

WriteOwner
INLANEFREIGHT\Enterprise Admins              CreateChild, DeleteChild,
Self, WriteProperty, ExtendedRight, GenericRead, WriteDacl,

WriteOwner
INLANEFREIGHT\cliff.moore
ReadProperty, WriteProperty, GenericExecute
NT AUTHORITY\SELF
ReadProperty, WriteProperty, ExtendedRight
BUILTIN\Terminal Server License Servers
ReadProperty, WriteProperty
INLANEFREIGHT\Cert Publishers
ReadProperty, WriteProperty
INLANEFREIGHT\Organization Management
WriteProperty
INLANEFREIGHT\Exchange Servers
WriteProperty
INLANEFREIGHT\Exchange Servers                          CreateChild,
DeleteChild, ListChildren, ReadProperty, WriteProperty, ListObject
INLANEFREIGHT\Exchange Servers
ReadProperty, WriteProperty, ListObject, Delete
INLANEFREIGHT\Exchange Trusted Subsystem           CreateChild,
DeleteChild, ListChildren, ReadProperty, WriteProperty, ListObject
INLANEFREIGHT\Exchange Trusted Subsystem
WriteProperty
```

Enumerating ACLs with PowerView and SharpView

We can use `PowerView` / `SharpView` to perform the previous command much quicker. For example, `Get-DomainObjectACL` can be used on a user to return similar data.

```
PS C:\htb> Get-DomainObjectAcl -Identity harry.jones -Domain
inlanefreight.local -ResolveGUIDs

AceQualifier           : AccessAllowed
ObjectDN               : CN=Harry Jones,OU=Network
```

```
Ops,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : CreateChild, DeleteChild, ListChildren
ObjectAceType          : ms-Exch-Active-Sync-Devices
ObjectSID              : S-1-5-21-2974783224-3764228556-2640795941-2040
InheritanceFlags       : ContainerInherit
BinaryLength           : 72
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent,
InheritedObjectAceTypePresent
IsCallback             : False
PropagationFlags       : InheritOnly
SecurityIdentifier     : S-1-5-21-2974783224-3764228556-2640795941-2615
AccessMask             : 7
AuditFlags             : None
IsInherited            : False
AceFlags               : ContainerInherit, InheritOnly
InheritedObjectAceType : inetOrgPerson
OpaqueLength           : 0

AceQualifier           : AccessAllowed
ObjectDN               : CN=Harry Jones,OU=Network
Ops,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : CreateChild, DeleteChild, ListChildren
ObjectAceType          : ms-Exch-Active-Sync-Devices
ObjectSID              : S-1-5-21-2974783224-3764228556-2640795941-2040
InheritanceFlags       : ContainerInherit
BinaryLength           : 72
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent,
InheritedObjectAceTypePresent
IsCallback             : False
PropagationFlags       : InheritOnly
SecurityIdentifier     : S-1-5-21-2974783224-3764228556-2640795941-2615
AccessMask             : 7
AuditFlags             : None
IsInherited            : False
AceFlags               : ContainerInherit, InheritOnly
InheritedObjectAceType : User
OpaqueLength           : 0

<...SNIP...>
```

We can seek out ACLs on specific users and filter out results using the various AD filters covered in the `Active Directory LDAP` module. We can use the `Find-InterestingDomainAcl` to search out objects in the domain with modification rights over non-built-in objects. This command, too, produces a large amount of data and can either be filtered on for information about specific objects or saved to be examined offline.

```
PS C:\htb> Find-InterestingDomainAcl -Domain inlanefreight.local -
ResolveGUIDs

ObjectDN                : DC=INLANEFREIGHT,DC=LOCAL
AceQualifier            : AccessAllowed
ActiveDirectoryRights   : ExtendedRight
ObjectAceType           : User-Change-Password
AceFlags                : ContainerInherit
AceType                 : AccessAllowedObject
InheritanceFlags        : ContainerInherit
SecurityIdentifier      : S-1-5-21-2974783224-3764228556-2640795941-2618
IdentityReferenceName   : Exchange Windows Permissions
IdentityReferenceDomain : INLANEFREIGHT.LOCAL
IdentityReferenceDN     : CN=Exchange Windows Permissions,OU=Microsoft
Exchange Security
                          Groups,DC=INLANEFREIGHT,DC=LOCAL
IdentityReferenceClass  : group

ObjectDN                : DC=INLANEFREIGHT,DC=LOCAL
AceQualifier            : AccessAllowed
ActiveDirectoryRights   : ExtendedRight
ObjectAceType           : User-Force-Change-Password
AceFlags                : ContainerInherit
AceType                 : AccessAllowedObject
InheritanceFlags        : ContainerInherit
SecurityIdentifier      : S-1-5-21-2974783224-3764228556-2640795941-2618
IdentityReferenceName   : Exchange Windows Permissions
IdentityReferenceDomain : INLANEFREIGHT.LOCAL
IdentityReferenceDN     : CN=Exchange Windows Permissions,OU=Microsoft
Exchange Security
                          Groups,DC=INLANEFREIGHT,DC=LOCAL
IdentityReferenceClass  : group

ObjectDN                : DC=INLANEFREIGHT,DC=LOCAL
AceQualifier            : AccessAllowed
ActiveDirectoryRights   : CreateChild, DeleteChild, ListChildren
ObjectAceType           : ms-Exch-Active-Sync-Devices
AceFlags                : ContainerInherit, InheritOnly
AceType                 : AccessAllowedObject
InheritanceFlags        : ContainerInherit
SecurityIdentifier      : S-1-5-21-2974783224-3764228556-2640795941-2615
IdentityReferenceName   : Exchange Servers
IdentityReferenceDomain : INLANEFREIGHT.LOCAL
IdentityReferenceDN     : CN=Exchange Servers,OU=Microsoft Exchange
Security Groups,DC=INLANEFREIGHT,DC=LOCAL
IdentityReferenceClass  : group
```

```
<...SNIP...>
```

Aside from users and computers, we should also look at the ACLs set on file shares. This could provide us with information about which users can access a specific share or permissions are set too loosely on a specific share, which could lead to sensitive data disclosure or other attacks.

```
PS C:\htb> Get-NetShare -ComputerName SQL01

Name             Type Remark        ComputerName
----             ---- ------        ------------
ADMIN$     2147483648 Remote Admin  SQL01
C$         2147483648 Default share SQL01
DB_backups          0               SQL01
IPC$       2147483651 Remote IPC    SQL01
```

```
PS C:\htb> Get-PathAcl "\\SQL01\DB_backups"

Path              : \\SQL01\DB_backups
FileSystemRights  : Read
IdentityReference : Local System
IdentitySID       : S-1-5-18
AccessControlType : Allow

Path              : \\SQL01\DB_backups
FileSystemRights  : Read
IdentityReference : BUILTIN\Administrators
IdentitySID       : S-1-5-32-544
AccessControlType : Allow

Path              : \\SQL01\DB_backups
FileSystemRights  : Read
IdentityReference : BUILTIN\Users
IdentitySID       : S-1-5-32-545
AccessControlType : Allow

Path              : \\SQL01\DB_backups
FileSystemRights  : AppendData/AddSubdirectory
IdentityReference : BUILTIN\Users
IdentitySID       : S-1-5-32-545
AccessControlType : Allow

Path              : \\SQL01\DB_backups
FileSystemRights  : WriteData/AddFile
IdentityReference : BUILTIN\Users
```

```
IdentitySID       : S-1-5-32-545
AccessControlType : Allow

Path              : \\SQL01\DB_backups
FileSystemRights  : GenericAll
IdentityReference : Creator Owner
IdentitySID       : S-1-3-0
AccessControlType : Allow
```

Aside from ACLs of specific users and computers that may allow us to fully control or grant us other permissions, we should also check the ACL of the domain object. A common attack called DCSync requires a user to be delegated a combination of the following three rights:

- Replicating Directory Changes (DS-Replication-Get-Changes)
- Replicating Directory Changes All (DS-Replication-Get-Changes-All)
- Replicating Directory Changes In Filtered Set (DS-Replication-Get-Changes-In-Filtered-Set)

We can use the `Get-ObjectACL` function to search for all users that have these rights.

```
PS C:\htb> Get-ObjectACL "DC=inlanefreight,DC=local" -ResolveGUIDs | ? {
($_.ActiveDirectoryRights -match 'GenericAll') -or ($_.ObjectAceType -
match 'Replication-Get')} | Select-Object SecurityIdentifier | Sort-Object
-Property SecurityIdentifier -Unique

SecurityIdentifier
------------------
S-1-5-18
S-1-5-21-2974783224-3764228556-2640795941-1883
S-1-5-21-2974783224-3764228556-2640795941-2601
S-1-5-21-2974783224-3764228556-2640795941-2616
S-1-5-21-2974783224-3764228556-2640795941-498
S-1-5-21-2974783224-3764228556-2640795941-516
S-1-5-21-2974783224-3764228556-2640795941-519
S-1-5-32-544
S-1-5-9
```

Once we have the SIDs we can convert the SID back to the user to see which accounts have these rights and determine whether or not this is intended and/or if we can abuse these rights.

```
PS C:\htb> convertfrom-sid S-1-5-21-2974783224-3764228556-2640795941-1883
```

```
INLANEFREIGHT\frederick.walton
```

This can be done quickly to enumerate all users with this right.

```
PS C:\htb> $dcsync = Get-ObjectACL "DC=inlanefreight,DC=local" -
ResolveGUIDs | ? { ($_.ActiveDirectoryRights -match 'GenericAll') -or
($_.ObjectAceType -match 'Replication-Get')} | Select-Object -
ExpandProperty SecurityIdentifier | Select -ExpandProperty value
PS C:\htb> Convert-SidToName $dcsync

INLANEFREIGHT\frederick.walton
INLANEFREIGHT\Enterprise Read-only Domain Controllers
INLANEFREIGHT\Domain Controllers
INLANEFREIGHT\Organization Management
INLANEFREIGHT\Exchange Trusted Subsystem
BUILTIN\Administrators
Enterprise Domain Controllers
INLANEFREIGHT\Enterprise Admins
Local System
```

Leveraging ACLs

As seen in this section, various ACE entries can be set within AD. Administrators may set some on purpose to grant fine-grained privileges over an object or set of objects. In contrast, others may result from misconfigurations or installation of a service such as Exchange, which makes many changes ACLs within the domain by default.

We may compromise a user with `GenericWrite` over a user or group and can leverage this to force change a user's password or add our account to a specific group to further our access. Any modifications such as these should be carefully noted down and mentioned in the final report so the client can make sure changes are reverted if we cannot during the assessment period. Also, a "destructive" action, such as changing a user's password, should be used sparingly and coordinated with the client to avoid disruptions.

If we find a user, group, or computer with `WriteDacl` privileges over an object, we can leverage this in several ways. For example, if we can compromise a member of an Exchange-related group such as `Exchange Trusted Subsystem` we will likely have `WriteDacl` privileges over the domain object itself and be able to grant an account we control `Replicating Directory Changes` and `Replicating Directory Change` permissions to an account that we control and perform a DCSync attack to fully compromise the domain by mimicking a Domain Controller to retrieve user NTLM password hashes for any account we choose.

If we find ourselves with `GenericAll` / `GenericWrite` privileges over a target user, a less destructive attack would be to set a fake SPN on the account and perform a targeted `Kerberoasting` attack or modify the account's `userAccountControl` not to require Kerberos pre-authentication and perform a targeted `ASREPRoasting attack`. These examples require the account to be using a weak password that can be cracked offline using a tool such as `Hashcat` with minimal effort but are much less destructive than changing a user's password and have a higher likelihood of going unnoticed.

If you perform a destructive action such as changing a user's password and can compromise the domain, you can `DCSync`, obtain the account's password history, and use `Mimikatz` to reset the account to the previous password using `LSADUMP::ChangeNTLM` or `LSADUMP::SetNTLM`.

If we find ourselves with `GenericAll` / `GenericWrite` on a computer, we can perform a Kerberos Resource-based Constrained Delegation attack.

Sometimes we will find that a user or even the entire `Domain Users` group has been granted write permissions over a specific group policy object. If we find this type of misconfiguration, and the GPO is linked to one or more users or computers, we can use a tool such as SharpGPOAbuse to modify the target GPO to perform actions such as provisioning additional privileges to a user (such as `SeDebugPrivilege` to be able to perform targeted credential theft, or `SeTakeOwnershipPrivilege` to gain control over a sensitive file or file share), add a user we control as a local admin to a target host, add a computer startup script, and more. As discussed above, these modifications should be performed carefully in coordination with the client and noted in the final report to minimize disruptions.

This is a summary of the many options we have for abusing ACLs. This topic will be covered more in-depth in later modules.

Wrap Up

ACLs are an often overlooked area of AD security, but they can provide powerful intended and unintended rights over objects in the domain environment, as we have seen here. Even a small AD network has thousands of ACLs, so we must be targeted with our searches to uncover useful data. Next, we will take a look at Group Policy Objects (GPOs).

Enumerating Group Policy Objects (GPOs)

Group Policy provides systems administrators with a centralized way to manage configuration settings and manage operating systems and user and computer settings in a

Windows environment. A Group Policy Object (GPO) is a collection of policy settings. GPOs include policies such as screen lock timeout, disabling USB ports, domain password policy, push out software, manage applications, and more. GPOs can be applied to individual users and hosts or groups by being applied directly to an Organizational Unit (OU). Gaining rights over a GPO can lead to lateral vertical movement up to full domain compromise and can also be used as a persistence mechanism. Like ACLs, GPOs are often overlooked, and one misconfigured GPO can have catastrophic results.

We can use `Powerview` / `Sharpview`, `BloodHound`, and Group3r to enumerate Group Policy security misconfigurations. This section will show some of the enumeration techniques we can perform on the command line using `PowerView` and `SharpView`.

GPO Abuse

GPOs can be abused to perform attacks such as adding additional rights to a user, adding a local admin, or creating an immediate scheduled task. There are several ways to gain persistence via GPOs:

- Configure a GPO to run any of the above attacks.
- Create a scheduled task to modify group membership, add an account, run DCSync, or send back a reverse shell connection.
- Install targeted malware across the entire Domain.

SharpGPOAbuse is an excellent tool that can be used to take advantage of GPO misconfigurations. This section will help arm us with the data that we need to use tools such as this.

Gathering GPO Data

Let's start by gathering GPO names. In our test domain `INLANEFREIGHT.LOCAL`, there are 20 GPOs applied to various OUs.

```
PS C:\htb> Get-DomainGPO | select displayname

displayname
-----------
Default Domain Policy
Default Domain Controllers Policy
LAPS Install
LAPS
Disable LM Hash
Disable CMD.exe
```

```
Disallow removable media
Prevent software installs
Disable guest account
Disable SMBv1
Map home drive
Disable Forced Restarts
Screensaver
Applocker
Fine-grained password policy
Restrict Control Panel
User - MS Office
User - Browser Settings
Audit Policy
PowerShell logging
```

We can also check which GPOs apply to a specific computer.

```
PS C:\htb> Get-DomainGPO -ComputerName WS01 | select displayname

displayname
-----------
LAPS
Restrict Control Panel
Applocker
Disable Forced Restarts
Prevent software installs
Disallow removable media
Disable CMD.exe
Disable LM Hash
Disable Defender
PowerShell logging
Audit Policy
User - Browser Settings
User - MS Office
Fine-grained password policy
Screensaver
Map home drive
Disable SMBv1
Disable guest account
Default Domain Policy
```


Analyzing the GPO names can give us an idea of some of the security configurations in the target domain, such as LAPS, AppLocker, PowerShell Logging, cmd.exe disabled for workstations, etc. We can check for hosts/users that these GPOs are not applied to and plan out our attack paths for circumventing these controls.

If we do not have tools available to us, we can use gpresult, which is a built-in tool that determines GPOs that have been applied to a given user or computer and their settings. We can use specific commands to see the GPOs applied to a specific user and computer, respectively, such as:

```
C:\> gpresult /r /user:harry.jones
```

```
C:\> gpresult /r /S WS01
```

The tool can output in HTML format with a command such as `gpresult /h gpo_report.html`.

Let's use `gpresult` to see what GPOs are applied to a workstation in the domain.

```
C:\htb> gpresult /r /S WS01

Microsoft (R) Windows (R) Operating System Group Policy Result tool v2.0
© 2016 Microsoft Corporation. All rights reserved.

Created on 8/27/2020 at 12:05:47 AM

RSOP data for INLANEFREIGHT\Administrator on WS01 : Logging Mode
-----------------------------------------------------------

OS Configuration:            Member Server
OS Version:                  10.0.14393
Site Name:                   Default-First-Site-Name
Roaming Profile:             N/A
Local Profile:               C:\Users\administrator.INLANEFREIGHT
Connected over a slow link?: No

COMPUTER SETTINGS
------------------

    Last time Group Policy was applied: 8/26/2020 at 10:57:00 PM
    Group Policy was applied from:      DC01.INLANEFREIGHT.LOCAL
    Group Policy slow link threshold:   500 kbps
    Domain Name:                        INLANEFREIGHT
    Domain Type:                        Windows 2008 or later

    Applied Group Policy Objects
    -----------------------------
        LAPS
        LAPS
        Disable LM Hash
```

```
        Prevent software installs
        Default Domain Policy
        LAPS
        Disable LM Hash
        Prevent software installs
        Disable guest account

    The following GPOs were not applied because they were filtered out
    -------------------------------------------------------------------
        Local Group Policy
            Filtering:  Not Applied (Empty)

    The computer is a part of the following security groups
    -------------------------------------------------------
        BUILTIN\Administrators
        Everyone
        BUILTIN\Users
        NT AUTHORITY\NETWORK
        NT AUTHORITY\Authenticated Users
        This Organization
        Authentication authority asserted identity
        System Mandatory Level

USER SETTINGS
--------------
    CN=Administrator,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
    Last time Group Policy was applied: 7/30/2020 at 3:10:28 PM
    Group Policy was applied from:      N/A
    Group Policy slow link threshold:   500 kbps
    Domain Name:                        INLANEFREIGHT
    Domain Type:                        Windows 2008 or later

    Applied Group Policy Objects
    ----------------------------
        N/A

    The following GPOs were not applied because they were filtered out
    ------------------------------------------------------------------
        Local Group Policy
            Filtering:  Not Applied (Empty)

    The user is a part of the following security groups
    ---------------------------------------------------
        Domain Users
        Everyone
        BUILTIN\Users
        BUILTIN\Administrators
        NT AUTHORITY\INTERACTIVE
        CONSOLE LOGON
        NT AUTHORITY\Authenticated Users
```

```
        This Organization
        LOCAL
        Domain Admins
        Authentication authority asserted identity
        High Mandatory Level
```

GPO Permissions

After reviewing all of the GPOs applied throughout the domain, it is always good to look at GPO permissions. We can use the `Get-DomainGPO` and `Get-ObjectAcl` using the SID for the `Domain Users` group to see if this group has any permissions assigned to any GPOs.

```
PS C:\htb> Get-DomainGPO | Get-ObjectAcl | ? {$_.SecurityIdentifier -eq
'S-1-5-21-2974783224-3764228556-2640795941-513'}

ObjectDN              : CN={831DE3ED-40B1-4703-ABA7-
8EA13B2EB118},CN=Policies,CN=System,DC=INLANEFREIGHT,DC=LOCAL
ObjectSID             :
ActiveDirectoryRights : CreateChild, DeleteChild, ReadProperty,
WriteProperty, GenericExecute
BinaryLength          : 36
AceQualifier          : AccessAllowed
IsCallback            : False
OpaqueLength          : 0
AccessMask            : 131127
SecurityIdentifier    : S-1-5-21-2974783224-3764228556-2640795941-513
AceType               : AccessAllowed
AceFlags              : ContainerInherit
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None
AuditFlags            : None
```

From the result, we can see that one GPO allows all Domain Users full write access. We can then confirm the name of the GPO using the built-in cmdlet `Get-GPO`.

```
PS C:\htb> Get-GPO -Guid 831DE3ED-40B1-4703-ABA7-8EA13B2EB118

DisplayName      : Screensaver
DomainName       : INLANEFREIGHT.LOCAL
Owner            : INLANEFREIGHT\Domain Admins
Id               : 831de3ed-40b1-4703-aba7-8ea13b2eb118
GpoStatus        : AllSettingsEnabled
```

```
Description      :
CreationTime     : 8/26/2020 10:46:46 PM
ModificationTime : 8/26/2020 11:11:01 PM
UserVersion      : AD Version: 0, SysVol Version: 0
ComputerVersion  : AD Version: 0, SysVol Version: 0
WmiFilter        :
```

This misconfigured GPO could be exploited using a tool such as `SharpGPOAbuse` and the `--AddUserRights` attack to give a user unintended rights or the `--AddLocalAdmin` attack to add a user as a local admin on a machine where the GPO is applied and use it to move laterally towards our target.

Hidden GPO Code Execution Paths

Group Policy is the most basic way System Administrators can command many Computers to perform a task. It is not the most common way to do things as many organizations will use commercial applications such as:

- Microsoft SCCM - System Center Configuration Manager
- PDQInventory/Deploy
- NinjaRMM (Remote Management and Monitoring)
- Ansible/Puppet/Salt

However, each one of these applications is non-default, and when an Administrator googles for a solution, their answer probably won't include the technology they use. Often, you may find one-off configurations an administrator did to accomplish a task quickly. For example, on multiple occasions, I have run across a "Machine/User Startup" script to collect inventory and write it to a domain share. I have seen this policy execute both BAT and VBScript files that were either write-able by the `machine account` or `domain users`. Whenever I dig into file shares and see files write-able by Everyone, Authenticated Users, Domain Users, Domain Computers, etc., containing what looks like log files, I dig into Group Policy, specifically looking for Startup Scripts.

That is just one way an Administrators use "Code Execution via GP" legitimately. Here is a list of the path's I know about:

- Add Registry Autoruns
- Software Installation (Install MSI Package that exists on a share)
- Scripts in the Startup/Shutdown for a Machine or User
- Create Shortcuts on Desktops that point to files
- Scheduled Tasks

If anyone of these paths points to a file on a share, enumerate the permissions to check if non-administrators can edit the file. Your tools will often miss this because it only looks at if the Group Policy itself is write-able, not if the executables/scripts the group policy references are writeable.

Next Steps

At this point, we have enumerated users, groups, computers, ACLs, and GPOs within the target domain and uncovered many misconfigurations that we could use to move through the domain towards our target. Now that we have seen a few ways to take over the `INLANEFREIGHT.LOCAL` domain, we can look at domain trusts and see what partner domains/forests exist and the relationships. This will help us plan our attacks to move from our current domain and potentially compromise any trusting domains.

Enumerating AD Trusts

A trust is used to establish forest-forest or domain-domain authentication, allowing users to access resources in (or administer) another domain outside of the domain their account resides in.

`A trust creates a link between the authentication systems of two domains.`

Trusts can be transitive or non-transitive.

- A transitive trust means that trust is extended to objects which the child domain trusts.
- In a non-transitive trust, only the child domain itself is trusted.

Trusts can be set up to be one-way or two-way (bidirectional).

- In bidirectional trusts, users from both trusting domains can access resources.
- In a one-way trust, only users in a trusted domain can access resources in a trusting domain, not vice-versa. `The direction of trust is opposite to the direction of access.`

There are several trust types.

Trust Type	Description
Parent-child	Domains within the same forest. The child domain has a two-way transitive trust with the parent domain.

Trust Type	Description
Cross-link	A trust between child domains to speed up authentication.
External	A non-transitive trust between two separate domains in separate forests that are not already joined by a forest trust. This type of trust utilizes SID filtering.
Tree-root	A two-way transitive trust between a forest root domain and a new tree root domain. They are created by design when you set up a new tree root domain within a forest.
Forest	A transitive trust between two forest root domains.

Often, domain trusts are set up improperly and provide unintended attack paths. Also, trusts set up for ease of use may not be reviewed later for potential security implications. M&A can result in bidirectional trusts with acquired companies, unknowingly introducing risk into the acquiring company's environment. It is not uncommon to perform an attack such as Kerberoasting against a domain outside the principal domain and obtain a user with administrative access within the principal domain.

Enumerating Trust Relationships

Aside from using built-in AD tools such as the Active Directory PowerShell module, `PowerView`/ `SharpView` and `BloodHound` can be utilized to enumerate trust relationships, the type of trusts established, as well as the authentication flow.

BloodHound creates a graphical view of trust relationships, which helps both attackers and defenders understand potential trust-related vectors.

`PowerView` can be used to perform a domain trust mapping and provide information such as the type of trust (parent/child, external, forest), as well as the direction of the trust (one-way or bidirectional). All of this information is extremely useful once a foothold is obtained, and you are planning to compromise the environment further.

We can use the function `Get-DomainTrust` to quickly check which trusts exist, the type, and the direction of the trusts.

```
PS C:\htb> Get-DomainTrust

SourceName      : INLANEFREIGHT.LOCAL
TargetName      : LOGISTICS.INLANEFREIGHT.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : WITHIN_FOREST
TrustDirection  : Bidirectional
```

```
WhenCreated     : 7/27/2020 2:06:07 AM
WhenChanged     : 7/27/2020 2:06:07 AM

SourceName      : INLANEFREIGHT.LOCAL
TargetName      : freightlogistics.local
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FOREST_TRANSITIVE
TrustDirection  : Bidirectional
WhenCreated     : 7/28/2020 4:46:40 PM
WhenChanged     : 7/28/2020 4:46:40 PM
```

We can use the function `Get-DomainTrustMapping` to enumerate all trusts for our current domain and other reachable domains.

```
PS C:\htb> Get-DomainTrustMapping

SourceName      : INLANEFREIGHT.LOCAL
TargetName      : LOGISTICS.INLANEFREIGHT.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : WITHIN_FOREST
TrustDirection  : Bidirectional
WhenCreated     : 7/27/2020 2:06:07 AM
WhenChanged     : 7/27/2020 2:06:07 AM

SourceName      : INLANEFREIGHT.LOCAL
TargetName      : freightlogistics.local
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FOREST_TRANSITIVE
TrustDirection  : Bidirectional
WhenCreated     : 7/28/2020 4:46:40 PM
WhenChanged     : 7/28/2020 4:46:40 PM

SourceName      : freightlogistics.local
TargetName      : INLANEFREIGHT.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FOREST_TRANSITIVE
TrustDirection  : Bidirectional
WhenCreated     : 7/28/2020 4:46:41 PM
WhenChanged     : 7/28/2020 4:46:41 PM

SourceName      : LOGISTICS.INLANEFREIGHT.LOCAL
TargetName      : INLANEFREIGHT.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : WITHIN_FOREST
TrustDirection  : Bidirectional
WhenCreated     : 7/27/2020 2:06:07 AM
```

```
WhenChanged     : 7/27/2020 2:06:07 AM
```

Depending on the trust type, there are several attacks that we may be able to perform, such as the `ExtraSids` attack to compromise a parent domain once the child domain has been compromised or cross-forest trust attacks such as `Kerberoasting` and `ASREPRoasting` and `SID History` abuse. Each of these attacks will be covered in-depth in later modules.

Attacking Trusts

Organizations set up a trust for various reasons, i.e., ease of management, quickly "plugging in" a new forest obtained through a merger & acquisition, enabling communications between multiple branches of a company, etc. Managed service providers often set up trusts between their domain and those of their clients to facilitate administration.

Some examples for why an organization may set up a trust are:

- Keeping management local to regions. You may see `FLORIDA.INLANEFREIGHT.LOCAL`. By having the `FLORIDA` Domain, it is easy for administrators to ensure those users access resources in their LAN.
- Acquisitions - When a company acquires another company and wants a quick way to manage the new equipment without rebuilding anything. They may establish a trust. This can lead to issues, especially if the acquired company has not had regular security assessments performed, has legacy hosts in its environment, has different/no security monitoring controls in place, etc.
- Keeping development, testing, etc., logically separated. If `DEVELOPMENT.INLANEFREIGHT.LOCAL` has little privileges over `INLANEFREIGHT.LOCAL`, it is unlikely for beta code to have any adverse effects on production.

In all of these cases, Domain Trusts are set up to minimize the number of accounts required. It is much easier to manage multiple domains when you can reference adjacent domains' groups/users. If configured wrong, with lax permissions, etc., a trust relationship can be attacked to further our access, compromising one or many domains in the process.

In our example environment, the domain `INLANEFREIGHT.LOCAL` has a bidirectional trust with the `LOGISTICS.INLANEFREIGHT.LOCAL` domain and is set up as a parent-child trust relationship (both domains within the same forest with `INLANEFREIGHT.LOCAL` acting as the forest root domain.). If we can gain a foothold in either domain, we will be able to perform attacks such as `Kerberoasting` or `ASREPRoasting` across the trust in either direction because our compromised user would be able to authenticate to/from the parent domain, therefore querying any AD objects in the other domain.

Furthermore, if we can compromise the child domain `LOGISTICS.INLANEFREIGHT.LOCAL` we will be able to compromise the parent domain using the `ExtraSids` attack. This is possible because the `sidHistory` property is respected due to a lack of "SID Filtering" protection. Therefore, a user in a child domain with their `sidHistory` set to the Enterprise Admins group (which only exists in the parent domain) is treated as a member of this group, which allows for administrative access to the entire forest.

Our lab environment also shows a bidirectional forest trust between the `INLANEFREIGHT.LOCAL` and `freightlogistics.local` forests, meaning that users from either forest can authenticate across the trust and query any AD object within the partner forest. Aside from attacks such as `Kerberoasting` and `ASREPRoasting`, we may also be able to abuse `SID History` to compromise the trusting forest.

The `SID history` attribute is used in migration scenarios. If a user in one domain is migrated to another domain, a new account is created in the second domain. The original user's SID will be added to the new user's SID history attribute, ensuring that they can still access resources in the original domain.

`SID history` is intended to work across domains but can actually work in the same domain. Using `Mimikatz`, it is possible to perform SID history injection and add an administrator account to the SID History attribute of an account that they control. When logging in with this account, all of the SIDs associated with the account are added to the user's token.

This token is used to determine what resources the account can access. If the SID of a Domain Admin account is added to the SID History attribute of this account, this account will be able to perform DCSync and create golden tickets for further persistence.

This can also be abused across a forest trust. If a user is migrated from one forest to another and SID Filtering is not enabled, it becomes possible to add a SID from the other forest, and this SID will be added to the user's token when authenticating across the trust. If the SID of an account having administrative privileges in Forest A is added to the SID history attribute of an account in Forest B, assuming they can authenticate across the forest, this account will have administrative privileges when accessing resources in the partner forest.

Another common way to cross trust boundaries is by leveraging password re-use. Let's say we compromise the `INLANEFREIGHT.LOCAL` forest and find a user account named `BSIMMONS_ADM` that also exists in the `freightlogistics.local` forest. There is a good chance that this administrator re-uses their password across environments. Also, it is always worth checking for foreign users/foreign group membership. We may find accounts belonging to administrative (or non-administrative) groups in Forest A that are actually part of Forest B and can be used to gain a foothold in the partner forest.

Each of these attacks will be covered in-depth in later modules.

Wrapping Up

We have now seen how `PowerView` and `SharpView` can be used to enumerate standard AD objects such as users, computers, and groups, as well as more complex relationships such as ACLS. The tools can be used to inform a variety of AD attacks, enumerate accessible file shares, find local admin access, find logged in users, and more. The skills assessment that follows will test the application of the skills taught throughout this module.

Active Directory PowerView - Skills Assessment

The `INLANEFREIGHT` organization has contracted your firm to perform an Active Directory security assessment. Use the `PowerView` and `SharpView` tools to perform targeted enumeration of the client's domain environment.

Connect to the target host and perform the enumeration tasks listed below to complete this module.