14-Credentialed Enumeration - from Windows

ActiveDirectory PowerShell Module

The ActiveDirectory PowerShell module is a group of PowerShell cmdlets for administering an Active Directory environment from the command line. It consists of 147 different cmdlets at the time of writing. We can't cover them all here, but we will look at a few that are particularly useful for enumerating AD environments. Feel free to explore other cmdlets included in the module in the lab built for this section, and see what interesting combinations and output you can create.

Before we can utilize the module, we have to make sure it is imported first. The Get-Module cmdlet, which is part of the Microsoft.PowerShell.Core module, will list all available modules, their version, and potential commands for use. This is a great way to see if anything like Git or custom administrator scripts are installed. If the module is not loaded, run `Import-Module ActiveDirectory` to load it for use.

Get-module: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/get-module?view=powershell-7.4&viewFallbackFrom=powershell-7.2

Microsoft.PowerShell.Core module : https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/?view=powershell-7.4&viewFallbackFrom=powershell-7.2

Get-Module

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

ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Manifest   3.1.0.0    Microsoft.PowerShell.Utility        {Add-Member, Add-
Type, Clear-Variable, Compare-Object...}
Script     2.0.0      PSReadline                          {Get-
PSReadLineKeyHandler, Get-PSReadLineOption, Remove-PS...
```

We'll see that the ActiveDirectory module is not yet imported. Let's go ahead and import it.

Load ActiveDirectory Module

```
PS C:\htb> Import-Module ActiveDirectory
PS C:\htb> Get-Module

ModuleType Version    Name                                ExportedCommands
```

```
---------- -------    ----                                 ---------------
Manifest   1.0.1.0    ActiveDirectory                      {Add-
ADCentralAccessPolicyMember, Add-ADComputerServiceAcc...
Manifest   3.1.0.0    Microsoft.PowerShell.Utility         {Add-Member, Add-
Type, Clear-Variable, Compare-Object...}
Script     2.0.0      PSReadline                           {Get-
PSReadLineKeyHandler, Get-PSReadLineOption, Remove-PS...
```

Now that our modules are loaded, let's begin. First up, we'll enumerate some basic information about the domain with the [Get-ADDomain https://docs.microsoft.com/en-us/powershell/module/activedirectory/get-addomain?view=windowsserver2022-ps[] (https://docs.microsoft.com/en-us/powershell/module/activedirectory/get-addomain?view=windowsserver2022-ps)](https://docs.microsoft.com/en-us/powershell/module/activedirectory/get-addomain?view=windowsserver2022-ps) cmdlet.

Get Domain Info

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

AllowedDNSSuffixes                : {}
ChildDomains                      : {LOGISTICS.INLANEFREIGHT.LOCAL}
ComputersContainer                : CN=Computers,DC=INLANEFREIGHT,DC=LOCAL
DeletedObjectsContainer           : CN=Deleted
Objects,DC=INLANEFREIGHT,DC=LOCAL
DistinguishedName                 : DC=INLANEFREIGHT,DC=LOCAL
DNSRoot                           : INLANEFREIGHT.LOCAL
DomainControllersContainer        : OU=Domain
Controllers,DC=INLANEFREIGHT,DC=LOCAL
DomainMode                        : Windows2016Domain
DomainSID                         : S-1-5-21-3842939050-3880317879-
2865463114
ForeignSecurityPrincipalsContainer :
CN=ForeignSecurityPrincipals,DC=INLANEFREIGHT,DC=LOCAL
Forest                            : INLANEFREIGHT.LOCAL
InfrastructureMaster              : ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
LastLogonReplicationInterval      :
LinkedGroupPolicyObjects          : {cn={DDBB8574-E94E-4525-8C9D-
ABABE31223D0},cn=policies,cn=system,DC=INLANEFREIGHT,
                                    DC=LOCAL, CN={31B2F340-016D-11D2-945F-
00C04FB984F9},CN=Policies,CN=System,DC=INLAN
                                    EFREIGHT,DC=LOCAL}
LostAndFoundContainer             :
CN=LostAndFound,DC=INLANEFREIGHT,DC=LOCAL
ManagedBy                         :
Name                              : INLANEFREIGHT
```

```
NetBIOSName                           : INLANEFREIGHT
ObjectClass                           : domainDNS
ObjectGUID                            : 71e4ecd1-a9f6-4f55-8a0b-e8c398fb547a
ParentDomain                          :
PDCEmulator                           : ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
PublicKeyRequiredPasswordRolling      : True
QuotasContainer                       : CN=NTDS
Quotas,DC=INLANEFREIGHT,DC=LOCAL
ReadOnlyReplicaDirectoryServers       : {}
ReplicaDirectoryServers               : {ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL}
RIDMaster                             : ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
SubordinateReferences                 :
{DC=LOGISTICS,DC=INLANEFREIGHT,DC=LOCAL,

DC=ForestDnsZones,DC=INLANEFREIGHT,DC=LOCAL,

DC=DomainDnsZones,DC=INLANEFREIGHT,DC=LOCAL,

CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL}
SystemsContainer                      : CN=System,DC=INLANEFREIGHT,DC=LOCAL
UsersContainer                        : CN=Users,DC=INLANEFREIGHT,DC=LOCAL
```

This will print out helpful information like the domain SID, domain functional level, any child domains, and more. Next, we'll use the Get-ADUser https://learn.microsoft.com/en-us/powershell/module/activedirectory/get-aduser?view=windowsserver2022-ps[] (https://docs.microsoft.com/en-us/powershell/module/activedirectory/get-aduser?view=windowsserver2022-ps) cmdlet. We will be filtering for accounts with the `ServicePrincipalName` property populated. This will get us a listing of accounts that may be susceptible to a Kerberoasting attack, which we will cover in-depth after the next section.

Get-ADUser

```
PS C:\htb> Get-ADUser -Filter {ServicePrincipalName -ne "$null"} -Properties
ServicePrincipalName

DistinguishedName    : CN=adfs,OU=Service
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
Enabled              : True
GivenName            : Sharepoint
Name                 : adfs
ObjectClass          : user
ObjectGUID           : 49b53bea-4bc4-4a68-b694-b806d9809e95
SamAccountName       : adfs
ServicePrincipalName : {adfsconnect/azure01.inlanefreight.local}
```

```
SID                   : S-1-5-21-3842939050-3880317879-2865463114-5244
Surname               : Admin
UserPrincipalName     :

DistinguishedName     : CN=BACKUPAGENT,OU=Service
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
Enabled               : True
GivenName             : Jessica
Name                  : BACKUPAGENT
ObjectClass           : user
ObjectGUID            : 2ec53e98-3a64-4706-be23-1d824ff61bed
SamAccountName        : backupagent
ServicePrincipalName : {backupjob/veam001.inlanefreight.local}
SID                   : S-1-5-21-3842939050-3880317879-2865463114-5220
Surname               : Systemmailbox 8Cc370d3-822A-4Ab8-A926-Bb94bd0641a9
UserPrincipalName     :

<SNIP>
```

Another interesting check we can run utilizing the ActiveDirectory module, would be to verify domain trust relationships using the Get-ADTrust cmdlet

Checking For Trust Relationships

```
PS C:\htb> Get-ADTrust -Filter *

Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       :
CN=LOGISTICS.INLANEFREIGHT.LOCAL,CN=System,DC=INLANEFREIGHT,DC=LOCAL
ForestTransitive        : False
IntraForest             : True
IsTreeParent            : False
IsTreeRoot              : False
Name                    : LOGISTICS.INLANEFREIGHT.LOCAL
ObjectClass             : trustedDomain
ObjectGUID              : f48a1169-2e58-42c1-ba32-a6ccb10057ec
SelectiveAuthentication : False
SIDFilteringForestAware : False
SIDFilteringQuarantined : False
Source                  : DC=INLANEFREIGHT,DC=LOCAL
Target                  : LOGISTICS.INLANEFREIGHT.LOCAL
TGTDelegation           : False
TrustAttributes         : 32
```

```
TrustedPolicy           :
TrustingPolicy          :
TrustType               : Uplevel
UplevelOnly             : False
UsesAESKeys             : False
UsesRC4Encryption       : False

Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       :
CN=FREIGHTLOGISTICS.LOCAL,CN=System,DC=INLANEFREIGHT,DC=LOCAL
ForestTransitive        : True
IntraForest             : False
IsTreeParent            : False
IsTreeRoot              : False
Name                    : FREIGHTLOGISTICS.LOCAL
ObjectClass             : trustedDomain
ObjectGUID              : 1597717f-89b7-49b8-9cd9-0801d52475ca
SelectiveAuthentication : False
SIDFilteringForestAware : False
SIDFilteringQuarantined : False
Source                  : DC=INLANEFREIGHT,DC=LOCAL
Target                  : FREIGHTLOGISTICS.LOCAL
TGTDelegation           : False
TrustAttributes         : 8
TrustedPolicy           :
TrustingPolicy          :
TrustType               : Uplevel
UplevelOnly             : False
UsesAESKeys             : False
UsesRC4Encryption       : False
```

1. العلاقة مع LOGISTICS.INLANEFREIGHT.LOCAL:
Direction:
BiDirectional: العلاقة بين الدومينات ثنائية الاتجاه (كل دومين يثق بالآخر).
ForestTransitive:

```
False: العلاقة غير قابلة للانتقال (Transitive) بين الغابات (Forests).
IntraForest:
True: العلاقة داخل نفس الغابة (Forest).
TrustType:
Uplevel: يشير إلى أن الدومينات تستخدم إصدارًا حديثًا من Active Directory (مثل Windows Server 2003 فما فوق).
UsesAESKeys / UsesRC4Encryption:
False: هذا يعني أن العلاقة لا تستخدم التشفير AES أو RC4 لتأمين الاتصالات.
2. العلاقة مع FREIGHTLOGISTICS.LOCAL:
Direction:
BiDirectional: العلاقة ثنائية الاتجاه.
ForestTransitive:
True: العلاقة قابلة للانتقال بين الغابات، مما يسمح بمشاركة الثقة عبر عدة غابات.
IntraForest:
False: العلاقة ليست داخل نفس الغابة (تتعلق بغابتين منفصلتين).
TrustType:
Uplevel: إصدارات حديثة من Active Directory.
UsesAESKeys / UsesRC4Encryption:
False: التشفير غير مستخدم هنا أيضًا.
نقاط مهمة من التحليل:
العلاقة مع LOGISTICS.INLANEFREIGHT.LOCAL:

هذه العلاقة داخل الغابة (IntraForest)، لكنها غير قابلة للانتقال (Non-Transitive).
تستخدم Active Directory نفس البنية الأساسية بين الدومينين.
العلاقة مع FREIGHTLOGISTICS.LOCAL:

هذه العلاقة بين غابتين منفصلتين (InterForest)، وهي قابلة للانتقال (Transitive).
هذا يعني أن الدومينات الأخرى داخل الغابات المرتبطة يمكن أن تشارك الثقة أيضًا.
أمان التشفير:

لم يتم تفعيل تشفير AES أو RC4. من الجيد مراجعة في أي من العلاقات المذكورة سياسات الأمان لتفعيل التشفير إذا لم تكن هناك قيود تؤدي لمنع ذلك.
```

نتائج Get-ADTrust:

- **Direction**: اتجاه الثقة (اتجاه الوصول).
  - `Inbound`: يعني أن الدومين الآخر يثق بك.

- `Outbound`: يعني أنك تثق بالدومين الآخر.
- `Bidirectional`: ثقة متبادلة.

- **TrustType**: نوع العلاقة:
  - **ParentChild**: علاقة بين دومين رئيسي وفرعي.
  - **Forest**: علاقة بين الغابات (Forests).
  - **External**: علاقة مع دومين خارجي.
- **TrustingDomainName**: اسم الدومين الذي يمنح الثقة.
- **TrustedDomainName**: اسم الدومين الذي يحصل على الثقة.

افترض أن لديك دومين باسم **company.local** وتريد التحقق من علاقات الثقة:

```
Get-ADTrust | Format-Table Name, TrustType, Direction, IsTransitive
```

- **Name**: اسم الدومين المرتبط.
- **TrustType**: يوضح نوع العلاقة (ParentChild, External, Forest).
- **Direction**: اتجاه العلاقة (Inbound, Outbound, Bidirectional).
- **IsTransitive**: هل العلاقة قابلة للتوسع عبر أكثر من دومين؟ (True/False).

---

1. **تحليل العلاقات:**
   - يمكنك استخدام هذه البيانات لتحديد الثغرات المحتملة بين الدومينات.
   - مثال: إذا كان هناك دومين ثقة غير ضروري أو علاقات مفتوحة.
2. **تقييم المخاطر:**
   - تحديد إذا كان هناك أي دومين غير آمن قد يُعرّض بقية الشبكة للخطر.
3. **التدقيق الداخلي:**
   - التحقق من أن علاقات الثقة متوافقة مع سياسة الأمان المؤسسية.

This cmdlet will print out any trust relationships the domain has. We can determine if they are trusts within our forest or with domains in other forests, the type of trust, the direction of the trust, and the name of the domain the relationship is with. This will be useful later on when looking to take advantage of child-to-parent trust relationships and attacking across forest trusts. Next, we can gather AD group information using the Get-ADGroup cmdlet.

Group Enumeration

```
PS C:\htb> Get-ADGroup -Filter * | select 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
Storage Replica Administrators
Domain Computers
Domain Controllers
Schema Admins
Enterprise Admins
Cert Publishers
Domain Admins

<SNIP>
```

We can take the results and feed interesting names back into the cmdlet to get more detailed information about a particular group like so:

Detailed Group Info

```
PS C:\htb> Get-ADGroup -Identity "Backup Operators"

DistinguishedName : CN=Backup Operators,CN=Builtin,DC=INLANEFREIGHT,DC=LOCAL
GroupCategory     : Security
GroupScope        : DomainLocal
Name              : Backup Operators
ObjectClass       : group
ObjectGUID        : 6276d85d-9c39-4b7c-8449-cad37e8abc38
```

```
SamAccountName     : Backup Operators
SID                : S-1-5-32-551
```

Now that we know more about the group, let's get a member listing using the Get-ADGroupMember cmdlet.

Group Membership

```
PS C:\htb> Get-ADGroupMember -Identity "Backup Operators"

distinguishedName : CN=BACKUPAGENT,OU=Service
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
name              : BACKUPAGENT
objectClass       : user
objectGUID        : 2ec53e98-3a64-4706-be23-1d824ff61bed
SamAccountName    : backupagent
SID               : S-1-5-21-3842939050-3880317879-2865463114-5220
```

We can see that one account, `backupagent`, belongs to this group. It is worth noting this down because if we can take over this service account through some attack, we could use its membership in the Backup Operators group to take over the domain. We can perform this process for the other groups to fully understand the domain membership setup. Try repeating the process with a few different groups. You will see that this process can be tedious, and we will be left with an enormous amount of data to sift through. We must know how to do this with built-in tools such as the ActiveDirectory PowerShell module, but we will see later in this section just how much tools like BloodHound can speed up this process and make our results far more accurate and organized.

2-PowerView

PowerView is a tool written in PowerShell to help us gain situational awareness within an AD environment. Much like BloodHound, it provides a way to identify where users are logged in on a network, enumerate domain information such as users, computers, groups, ACLS, trusts, hunt for file shares and passwords, perform Kerberoasting, and more. It is a highly versatile tool that can provide us with great insight into the security posture of our client's domain. It requires more manual work to determine misconfigurations and relationships within the domain than BloodHound but, when used right, can help us to identify subtle misconfigurations.

Let's examine some of PowerView's capabilities and see what data it returns. The table below describes some of the most useful functions PowerView offers.

PowerView : https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon

install tool:

```
1- git clone https://github.com/PowerShellMafia/PowerSploit.git

2- cd PowerSploit/Recon

3- Import-Module .\PowerView.ps1


4-Get-Command -Module PowerView
```

Command	Description
`Export-PowerViewCSV`	Append results to a CSV file
`ConvertTo-SID`	Convert a User or group name to its SID value
`Get-DomainSPNTicket`	Requests the Kerberos ticket for a specified Service Principal Name (SPN) account
Domain/LDAP Functions:	
`Get-Domain`	Will return the AD object for the current (or specified) domain
`Get-DomainController`	Return a list of the Domain Controllers for the specified domain
`Get-DomainUser`	Will return all users or specific user objects in AD
`Get-DomainComputer`	Will return all computers or specific computer objects in AD
`Get-DomainGroup`	Will return all groups or specific group objects in AD
`Get-DomainOU`	Search for all or specific OU objects in AD
`Find-InterestingDomainAcl`	Finds object ACLs in the domain with modification rights set to non-built in objects
`Get-DomainGroupMember`	Will return the members of a specific domain group
`Get-DomainFileServer`	Returns a list of servers likely functioning as file servers
`Get-DomainDFSShare`	Returns a list of all distributed file systems for the current (or specified) domain
GPO Functions:	
`Get-DomainGPO`	Will return all GPOs or specific GPO objects in AD
`Get-DomainPolicy`	Returns the default domain policy or the domain controller policy for the current domain
Computer Enumeration Functions:	
`Get-NetLocalGroup`	Enumerates local groups on the local or a remote machine

Command	Description
`Get-NetLocalGroupMember`	Enumerates members of a specific local group
`Get-NetShare`	Returns open shares on the local (or a remote) machine
`Get-NetSession`	Will return session information for the local (or a remote) machine
`Test-AdminAccess`	Tests if the current user has administrative access to the local (or a remote) machine
Threaded 'Meta'-Functions:	
`Find-DomainUserLocation`	Finds machines where specific users are logged in
`Find-DomainShare`	Finds reachable shares on domain machines
`Find-InterestingDomainShareFile`	Searches for files matching specific criteria on readable shares in the domain
`Find-LocalAdminAccess`	Find machines on the local domain where the current user has local administrator access
Domain Trust Functions:	
`Get-DomainTrust`	Returns 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`	Will enumerate all trusts for the current domain and any others seen.

This table is not all-encompassing for what PowerView offers, but it includes many of the functions we will use repeatedly. For more on PowerView, check out the Active Directory PowerView module. Below we will experiment with a few of them.

First up is the Get-DomainUser function. This will provide us with information on all users or specific users we specify. Below we will use it to grab information about a specific user, `mmorgan`.

https://academy.hackthebox.com/course/preview/active-directory-powerview

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

```
name                  : Matthew Morgan
samaccountname        : mmorgan
description           :
memberof              : {CN=VPN Users,OU=Security
Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=Shared Calendar
                        Read,OU=Security
Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=Printer Access,OU=Security
                        Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=File
Share H Drive,OU=Security
                        Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL...}
whencreated           : 10/27/2021 5:37:06 PM
pwdlastset            : 11/18/2021 10:02:57 AM
lastlogontimestamp    : 2/27/2022 6:34:25 PM
accountexpires        : NEVER
admincount            : 1
userprincipalname     : mmorgan@inlanefreight.local
serviceprincipalname  :
mail                  :
useraccountcontrol    : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD,
DONT_REQ_PREAUTH
```

We saw some basic user information with PowerView. Now let's enumerate some domain group information. We can use the [Get-DomainGroupMember](https://powersploit.readthedocs.io/en/latest/Recon/Get-DomainGroupMember/[](https://powersploit.readthedocs.io/en/latest/Recon/Get-DomainGroupMember/) function to retrieve group-specific information. Adding the `-Recurse` switch tells PowerView that if it finds any groups that are part of the target group (nested group membership) to list out the members of those groups. For example, the output below shows that the `Secadmins` group is part of the `Domain Admins` group through nested group membership. In this case, we will be able to view all of the members of that group who inherit Domain Admin rights via their group membership.

تؤدي إضافة مفتاح التبديل -Recurse إلى إخبار PowerView بأنه إذا عثر على أي مجموعات تشكل جزءًا من المجموعة المستهدفة (عضوية المجموعة المتداخلة) فسيقوم بإدراج أعضاء تلك المجموعات. على سبيل المثال، يوضح الإخراج أدناه أن مجموعة Secadmins جزء من مجموعة Domain Admins من خلال عضوية المجموعة المتداخلة. في هذه الحالة، سنكون قادرين على عرض جميع أعضاء تلك المجموعة الذين يرثون حقوق مسؤول المجال عبر عضوية المجموعة الخاصة بهم

```
PS C:\htb>  Get-DomainGroupMember -Identity "Domain Admins" -Recurse

GroupDomain             : INLANEFREIGHT.LOCAL
GroupName               : Domain Admins
GroupDistinguishedName  : CN=Domain
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
MemberDomain            : INLANEFREIGHT.LOCAL
```

```
MemberName              : svc_qualys
MemberDistinguishedName : CN=svc_qualys,OU=Service
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
MemberObjectClass       : user
MemberSID               : S-1-5-21-3842939050-3880317879-2865463114-5613

GroupDomain             : INLANEFREIGHT.LOCAL
GroupName               : Domain Admins
GroupDistinguishedName  : CN=Domain
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
MemberDomain            : INLANEFREIGHT.LOCAL
MemberName              : sp-admin
MemberDistinguishedName : CN=Sharepoint Admin,OU=Service
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
MemberObjectClass       : user
MemberSID               : S-1-5-21-3842939050-3880317879-2865463114-5228

GroupDomain             : INLANEFREIGHT.LOCAL
GroupName               : Secadmins
GroupDistinguishedName  : CN=Secadmins,OU=Security
Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
MemberDomain            : INLANEFREIGHT.LOCAL
MemberName              : spong1990
MemberDistinguishedName : CN=Maggie
                          Jablonski,OU=Operations,OU=Logistics-
HK,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
MemberObjectClass       : user
MemberSID               : S-1-5-21-3842939050-3880317879-2865463114-1965

<SNIP>
```

Above we performed a recursive look at the `Domain Admins` group to list its members. Now we know who to target for potential elevation of privileges. Like with the AD PowerShell module, we can also enumerate domain trust mappings.

Trust Enumeration

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

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

```
WhenCreated     : 11/1/2021 6:20:22 PM
WhenChanged     : 2/26/2022 11:55:55 PM

SourceName      : INLANEFREIGHT.LOCAL
TargetName      : FREIGHTLOGISTICS.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FOREST_TRANSITIVE
TrustDirection  : Bidirectional
WhenCreated     : 11/1/2021 8:07:09 PM
WhenChanged     : 2/27/2022 12:02:39 AM

SourceName      : LOGISTICS.INLANEFREIGHT.LOCAL
TargetName      : INLANEFREIGHT.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : WITHIN_FOREST
TrustDirection  : Bidirectional
WhenCreated     : 11/1/2021 6:20:22 PM
WhenChanged     : 2/26/2022 11:55:55 PM
```

We can use the Test-AdminAccess : https://powersploit.readthedocs.io/en/latest/Recon/Test-AdminAccess/[](https://powersploit.readthedocs.io/en/latest/Recon/Test-AdminAccess/) function to test for local admin access on either the current machine or a remote one.

Testing for Local Admin Access

```
PS C:\htb> Test-AdminAccess -ComputerName ACADEMY-EA-MS01

ComputerName    IsAdmin
------------    -------
ACADEMY-EA-MS01    True


2-Get-DomainComputer | Test-AdminAccess
```

Above, we determined that the user we are currently using is an administrator on the host ACADEMY-EA-MS01. We can perform the same function for each host to see where we have administrative access. We will see later how well BloodHound performs this type of check. Now we can check for users with the SPN attribute set, which indicates that the account may be subjected to a Kerberoasting attack.

Finding Users With SPN Set

```
PS C:\htb> Get-DomainUser -SPN -Properties
samaccountname,ServicePrincipalName
```

```
serviceprincipalname                              samaccountname
--------------------                              --------------
adfsconnect/azure01.inlanefreight.local           adfs
backupjob/veam001.inlanefreight.local             backupagent
d0wngrade/kerberoast.inlanefreight.local          d0wngrade
kadmin/changepw                                   krbtgt
MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433     sqldev
MSSQLSvc/SPSJDB.inlanefreight.local:1433          sqlprod
MSSQLSvc/SQL-CL01-01inlanefreight.local:49351     sqlqa
sts/inlanefreight.local                           solarwindsmonitor
testspn/kerberoast.inlanefreight.local            testspn
testspn2/kerberoast.inlanefreight.local           testspn2
```

3-SharpView

PowerView is part of the now deprecated PowerSploit offensive PowerShell toolkit. The tool has been receiving updates by BC-Security as part of their Empire 4 framework. Empire 4 is BC-Security's fork of the original Empire project and is actively maintained as of April 2022. We show examples throughout this module using the development version of PowerView because it is an excellent tool for recon in an Active Directory environment, and is still extremely powerful and helpful in modern AD networks even though the original version is not maintained. The BC-SECURITY version of PowerView has some new functions such as `Get-NetGmsa`, used to hunt for Group Managed Service Accounts, which is out of scope for this module. It is worth playing around with both versions to see the subtle differences between the old and currently maintained versions.

Another tool worth experimenting with is SharpView, a .NET port of PowerView. Many of the same functions supported by PowerView can be used with SharpView. We can type a method name with `-Help` to get an argument list.

SharpView : https://github.com/tevora-threat/SharpView

```
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> -DisallowDelegation <Boolean> -TrustedToAuth
<Boolean> -PreauthNotRequired <Boolean> -KerberosPreauthNotRequired
<Boolean> -NoPreauth <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>
```

Here we can use SharpView to enumerate information about a specific user, such as the user `forend`, which we control.

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

[Get-DomainSearcher] search base: LDAP://ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainUser] filter string: (&(samAccountType=805306368)(|
(samAccountName=forend)))
objectsid                     : {S-1-5-21-3842939050-3880317879-2865463114-
5614}
samaccounttype                : USER_OBJECT
objectguid                    : 53264142-082a-4cb8-8714-8158b4974f3b
useraccountcontrol            : NORMAL_ACCOUNT
accountexpires                : 12/31/1600 4:00:00 PM
lastlogon                     : 4/18/2022 1:01:21 PM
lastlogontimestamp            : 4/9/2022 1:33:21 PM
pwdlastset                    : 2/28/2022 12:03:45 PM
lastlogoff                    : 12/31/1600 4:00:00 PM
badPasswordTime               : 4/5/2022 7:09:07 AM
name                          : forend
distinguishedname             : CN=forend,OU=IT Admins,OU=IT,OU=HQ-
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
whencreated                   : 2/28/2022 8:03:45 PM
whenchanged                   : 4/9/2022 8:33:21 PM
samaccountname                : forend
memberof                      : {CN=VPN Users,OU=Security
Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=Shared Calendar
Read,OU=Security Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=Printer
Access,OU=Security Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=File Share H
Drive,OU=Security Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=File Share G
Drive,OU=Security Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL}
cn                            : {forend}
objectclass                   : {top, person, organizationalPerson, user}
badpwdcount                   : 0
countrycode                   : 0
usnchanged                    : 3259288
logoncount                    : 26618
primarygroupid                : 513
objectcategory                :
CN=Person,CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
```

```
dscorepropagationdata           : {3/24/2022 3:58:07 PM, 3/24/2022 3:57:44
PM, 3/24/2022 3:52:58 PM, 3/24/2022 3:49:31 PM, 7/14/1601 10:36:49 PM}
usncreated                      : 3054181
instancetype                    : 4
codepage                        : 0
```

Shares

Shares allow users on a domain to quickly access information relevant to their daily roles and share content with their organization. When set up correctly, domain shares will require a user to be domain joined and required to authenticate when accessing the system. Permissions will also be in place to ensure users can only access and see what is necessary for their daily role. Overly permissive shares can potentially cause accidental disclosure of sensitive information, especially those containing medical, legal, personnel, HR, data, etc. In an attack, gaining control over a standard domain user who can access shares such as the IT/infrastructure shares could lead to the disclosure of sensitive data such as configuration files or authentication files like SSH keys or passwords stored insecurely. We want to identify any issues like these to ensure the customer is not exposing any data to users who do not need to access it for their daily jobs and that they are meeting any legal/regulatory requirements they are subject to (HIPAA, PCI, etc.). We can use PowerView to hunt for shares and then help us dig through them or use various manual commands to hunt for common strings such as files with pass in the name. This can be a tedious process, and we may miss things, especially in large environments. Now, let's take some time to explore the tool Snaffler and see how it can aid us in identifying these issues more accurately and efficiently.

4-Snaffler https://github.com/SnaffCon/Snaffler

Snaffler is a tool that can help us acquire credentials or other sensitive data in an Active Directory environment. Snaffler works by obtaining a list of hosts within the domain and then enumerating those hosts for shares and readable directories. Once that is done, it iterates through any directories readable by our user and hunts for files that could serve to better our position within the assessment. Snaffler requires that it be run from a domain-joined host or in a domain-user context.

To execute Snaffler, we can use the command below:

Snaffler Execution

```
.\Snaffler.exe -s -d inlanefreight.local -o snaffler.log -v data
```

- **-s ---> to print the result on the consloe**
- **-d ---> specific domain**
- **-o ---> store output**

- **-v ---> verbos**

```
PS C:\htb> .\Snaffler.exe  -d INLANEFREIGHT.LOCAL -s -v data

 .::::::.:::.    :::.  :::.    .-::::'.-::::'::::    .,:::::: :::::::..
;;;`    ``;;;;,  `;;;  ;;`;;   ;;;'''' ;;;'''' ;;;    ;;;;'''' ;;;;``;;;;
'[==/[[[[, [[[[[. '[[ ,[[ '[[, [[[,,== [[[,,== [[[     [[cccc   [[[,/[[['
  '''    $ $$$ 'Y$c$$c$$$cc$$$c`$$$'`` `$$$'`` $$'     $$""   $$$$$$c
 88b    dP 888    Y88 888   888,888     888   o88oo,.__888oo,__ 888b '88bo,
  'YMmMY'  MMM     YM YMM   '''` 'MM,    'MM,  ''''YUMMM''''YUMMMMMMM   'W'
                       by l0ss and Sh3r4 - github.com/SnaffCon/Snaffler

2022-03-31 12:16:54 -07:00 [Share] {Black}(\\ACADEMY-EA-
MS01.INLANEFREIGHT.LOCAL\ADMIN$)
2022-03-31 12:16:54 -07:00 [Share] {Black}(\\ACADEMY-EA-
MS01.INLANEFREIGHT.LOCAL\C$)
2022-03-31 12:16:54 -07:00 [Share] {Green}(\\ACADEMY-EA-
MX01.INLANEFREIGHT.LOCAL\address)
2022-03-31 12:16:54 -07:00 [Share] {Green}(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares)
2022-03-31 12:16:54 -07:00 [Share] {Green}(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\User Shares)
2022-03-31 12:16:54 -07:00 [Share] {Green}(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\ZZZ_archive)
2022-03-31 12:17:18 -07:00 [Share] {Green}(\\ACADEMY-EA-
CA01.INLANEFREIGHT.LOCAL\CertEnroll)
2022-03-31 12:17:19 -07:00 [File] {Black}
<KeepExtExactBlack|R|^\.kdb$|289B|3/31/2022 12:09:22 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Infosec\GroupBackup.kdb) .kdb
2022-03-31 12:17:19 -07:00 [File] {Red}
<KeepExtExactRed|R|^\.key$|299B|3/31/2022 12:05:33 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Infosec\ShowReset.key) .key
2022-03-31 12:17:19 -07:00 [Share] {Green}(\\ACADEMY-EA-
FILE.INLANEFREIGHT.LOCAL\UpdateServicesPackages)
2022-03-31 12:17:19 -07:00 [File] {Black}
<KeepExtExactBlack|R|^\.kwallet$|302B|3/31/2022 12:04:45 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Infosec\WriteUse.kwallet)
.kwallet
2022-03-31 12:17:19 -07:00 [File] {Red}
<KeepExtExactRed|R|^\.key$|298B|3/31/2022 12:05:10 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Infosec\ProtectStep.key) .key
2022-03-31 12:17:19 -07:00 [File] {Black}
<KeepExtExactBlack|R|^\.ppk$|275B|3/31/2022 12:04:40 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Infosec\StopTrace.ppk) .ppk
```

```
2022-03-31 12:17:19 -07:00 [File] {Red}
<KeepExtExactRed|R|^\.key$|301B|3/31/2022 12:09:17 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Infosec\WaitClear.key) .key
2022-03-31 12:17:19 -07:00 [File] {Red}
<KeepExtExactRed|R|^\.sqldump$|312B|3/31/2022 12:05:30 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Development\DenyRedo.sqldump)
.sqldump
2022-03-31 12:17:19 -07:00 [File] {Red}
<KeepExtExactRed|R|^\.sqldump$|310B|3/31/2022 12:05:02 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department
Shares\IT\Development\AddPublish.sqldump) .sqldump
2022-03-31 12:17:19 -07:00 [Share] {Green}(\\ACADEMY-EA-
FILE.INLANEFREIGHT.LOCAL\WsusContent)
2022-03-31 12:17:19 -07:00 [File] {Red}
<KeepExtExactRed|R|^\.keychain$|295B|3/31/2022 12:08:42 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Infosec\SetStep.keychain)
.keychain
2022-03-31 12:17:19 -07:00 [File] {Black}
<KeepExtExactBlack|R|^\.tblk$|279B|3/31/2022 12:05:25 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Development\FindConnect.tblk)
.tblk
2022-03-31 12:17:19 -07:00 [File] {Black}
<KeepExtExactBlack|R|^\.psafe3$|301B|3/31/2022 12:09:33 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Development\GetUpdate.psafe3)
.psafe3
2022-03-31 12:17:19 -07:00 [File] {Red}
<KeepExtExactRed|R|^\.keypair$|278B|3/31/2022 12:09:09 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department
Shares\IT\Infosec\UnprotectConvertTo.keypair) .keypair
2022-03-31 12:17:19 -07:00 [File] {Black}
<KeepExtExactBlack|R|^\.tblk$|280B|3/31/2022 12:05:17 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Development\ExportJoin.tblk)
.tblk
2022-03-31 12:17:19 -07:00 [File] {Red}
<KeepExtExactRed|R|^\.mdf$|305B|3/31/2022 12:09:27 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Development\FormatShow.mdf)
.mdf
2022-03-31 12:17:19 -07:00 [File] {Red}
<KeepExtExactRed|R|^\.mdf$|299B|3/31/2022 12:09:14 PM>(\\ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL\Department Shares\IT\Development\LockConfirm.mdf)
.mdf

<SNIP>
```

We may find passwords, SSH keys, configuration files, or other data that can be used to further our access. Snaffler color codes the output for us and provides us with a rundown of the file types found in the shares.

Now that we have a wealth of data about the INLANEFREIGHT.LOCAL domain (and hopefully clear notes and log file output!), we need a way to correlate it and visualize it. Let's dive deeper into `BloodHound` and see how powerful this tool can be during any AD-focused security assessment.

5-BloodHound

As discussed in the previous section, `Bloodhound` is an exceptional open-source tool that can identify attack paths within an AD environment by analyzing the relationships between objects. Both penetration testers and blue teamers can benefit from learning to use BloodHound to visualize relationships in the domain. When used correctly and coupled with custom Cipher queries, BloodHound may find high-impact, but difficult to discover, flaws that have been present in the domain for years.

First, we must authenticate as a domain user from a Windows attack host positioned within the network (but not joined to the domain) or transfer the tool to a domain-joined host. There are many ways to achieve this covered in the File Transfer module. For our purposes, we will work with SharpHound.exe already on the attack host, but it's worth experimenting with transferring the tool to the attack host from Pwnbox or our own VM using methods such as a Python HTTP server, smbserver.py from Impacket, etc.

If we run SharpHound with the `--help` option, we can see the options available to us.

```
PS C:\htb>  .\SharpHound.exe --help

SharpHound 1.0.3
Copyright (C) 2022 SpecterOps

  -c, --collectionmethods    (Default: Default) Collection Methods:
Container, Group, LocalGroup, GPOLocalGroup,
                             Session, LoggedOn, ObjectProps, ACL,
ComputerOnly, Trusts, Default, RDP, DCOM, DCOnly

  -d, --domain               Specify domain to enumerate

  -s, --searchforest         (Default: false) Search all available domains
in the forest

  --stealth                  Stealth Collection (Prefer DCOnly whenever
```

```
possible!)

  -f                        Add an LDAP filter to the pregenerated filter.

  --distinguishedname       Base DistinguishedName to start the LDAP search
at

  --computerfile            Path to file containing computer names to
enumerate

  <SNIP>
```

We'll start by running the SharpHound.exe collector from the MS01 attack host.

```
PS C:\htb> .\SharpHound.exe -c All --zipfilename ILFREIGHT

2022-04-18T13:58:22.1163680-07:00|INFORMATION|Resolved Collection Methods:
Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container,
RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2022-04-18T13:58:22.1163680-07:00|INFORMATION|Initializing SharpHound at
1:58 PM on 4/18/2022
2022-04-18T13:58:22.6788709-07:00|INFORMATION|Flags: Group, LocalAdmin,
GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps,
DCOM, SPNTargets, PSRemote
2022-04-18T13:58:23.0851206-07:00|INFORMATION|Beginning LDAP search for
INLANEFREIGHT.LOCAL
2022-04-18T13:58:53.9132950-07:00|INFORMATION|Status: 0 objects finished (+0
0)/s -- Using 67 MB RAM
2022-04-18T13:59:15.7882419-07:00|INFORMATION|Producer has finished, closing
LDAP channel
2022-04-18T13:59:16.1788930-07:00|INFORMATION|LDAP channel closed, waiting
for consumers
2022-04-18T13:59:23.9288698-07:00|INFORMATION|Status: 3793 objects finished
(+3793 63.21667)/s -- Using 112 MB RAM
2022-04-18T13:59:45.4132561-07:00|INFORMATION|Consumers finished, closing
output channel
Closing writers
2022-04-18T13:59:45.4601086-07:00|INFORMATION|Output channel closed, waiting
for output task to complete
2022-04-18T13:59:45.8663528-07:00|INFORMATION|Status: 3809 objects finished
(+16 46.45122)/s -- Using 110 MB RAM
2022-04-18T13:59:45.8663528-07:00|INFORMATION|Enumeration finished in
00:01:22.7919186
```

```
2022-04-18T13:59:46.3663660-07:00|INFORMATION|SharpHound Enumeration
Completed at 1:59 PM on 4/18/2022! Happy Graphing
```

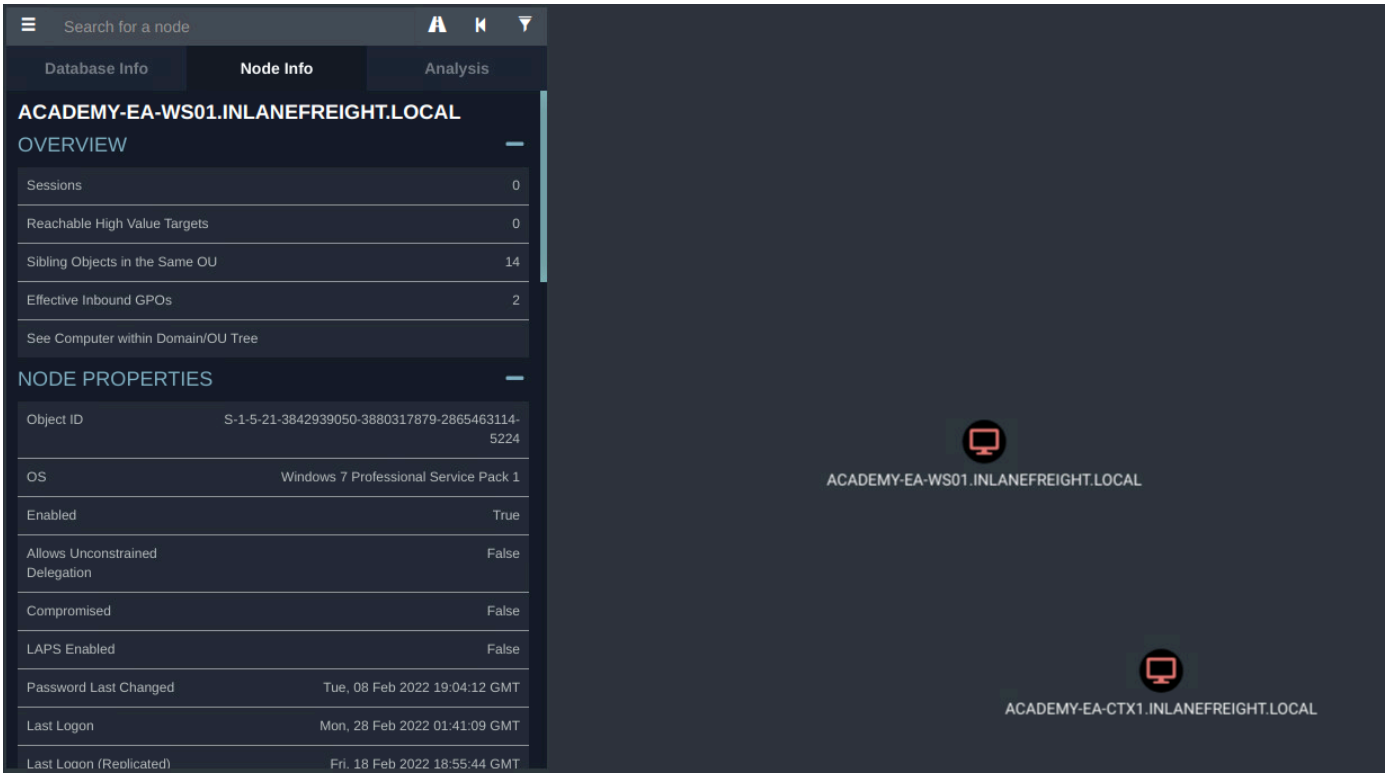
Next, we can exfiltrate the dataset to our own VM or ingest it into the BloodHound GUI tool on MS01. We can do this on MS01 by typing `bloodhound` into a CMD or PowerShell console. The credentials should be saved, but enter `neo4j: HTB_@cademy_stdnt!` if a prompt appears. Next, click on the `Upload Data` button on the right-hand side, select the newly generated zip file, and click `Open`. An `Upload Progress` window will pop up. Once all .json files show 100% complete, click the X at the top of that window.

We can start by typing `domain:` in the search bar on the top left and choosing `INLANEFREIGHT.LOCAL` from the results. Take a moment to browse the node info tab. As we can see, this would be a rather large company with over 550 hosts to target and trusts with two other domains.

Now, let's check out a few pre-built queries in the `Analysis` tab. The query `Find Computers with Unsupported Operating Systems` is great for finding outdated and unsupported operating systems running legacy software. These systems are relatively common to find within enterprise networks (especially older environments), as they often run some product that cannot be updated or replaced as of yet. Keeping these hosts around may save money, but they also can add unnecessary vulnerabilities to the network. Older hosts may be susceptible to older remote code execution vulnerabilities like MS08-067. If we come across these older hosts during an assessment, we should be careful before attacking them (or even check with our client) as they may be fragile and running a critical application or service. We can advise our client to segment these hosts off from the rest of the network as much as possible if they cannot remove them yet, but should also recommend that they start putting together a plan to decommission and replace them.

This query shows two hosts, one running Windows 7 and one running Windows Server 2008 (both of which are not "live" in our lab). Sometimes we will see hosts that are no longer powered on but still appear as records in AD. We should always validate whether they are "live" or not before making recommendations in our reports. We may write up a high-risk finding for Legacy Operating Systems or a best practice recommendation for cleaning up old records in AD.

Unsupported Operating Systems


Search for a node
Database Info
Node Info
Analysis
ACADEMY-EA-WS01.INLANEFREIGHT.LOCAL
OVERVIEW
Sessions 0
Reachable High Value Targets 0
Sibling Objects in the Same OU 14
Effective Inbound GPOs 2
See Computer within Domain/OU Tree
NODE PROPERTIES
Object ID S-1-5-21-3842939050-3880317879-2865463114-5224
OS Windows 7 Professional Service Pack 1
Enabled True
Allows Unconstrained Delegation False
Compromised False
LAPS Enabled False
Password Last Changed Tue, 08 Feb 2022 19:04:12 GMT
Last Logon Mon, 28 Feb 2022 01:41:09 GMT
Last Logon (Replicated) Fri, 18 Feb 2022 18:55:44 GMT
ACADEMY-EA-WS01.INLANEFREIGHT.LOCAL
ACADEMY-EA-CTX1.INLANEFREIGHT.LOCAL