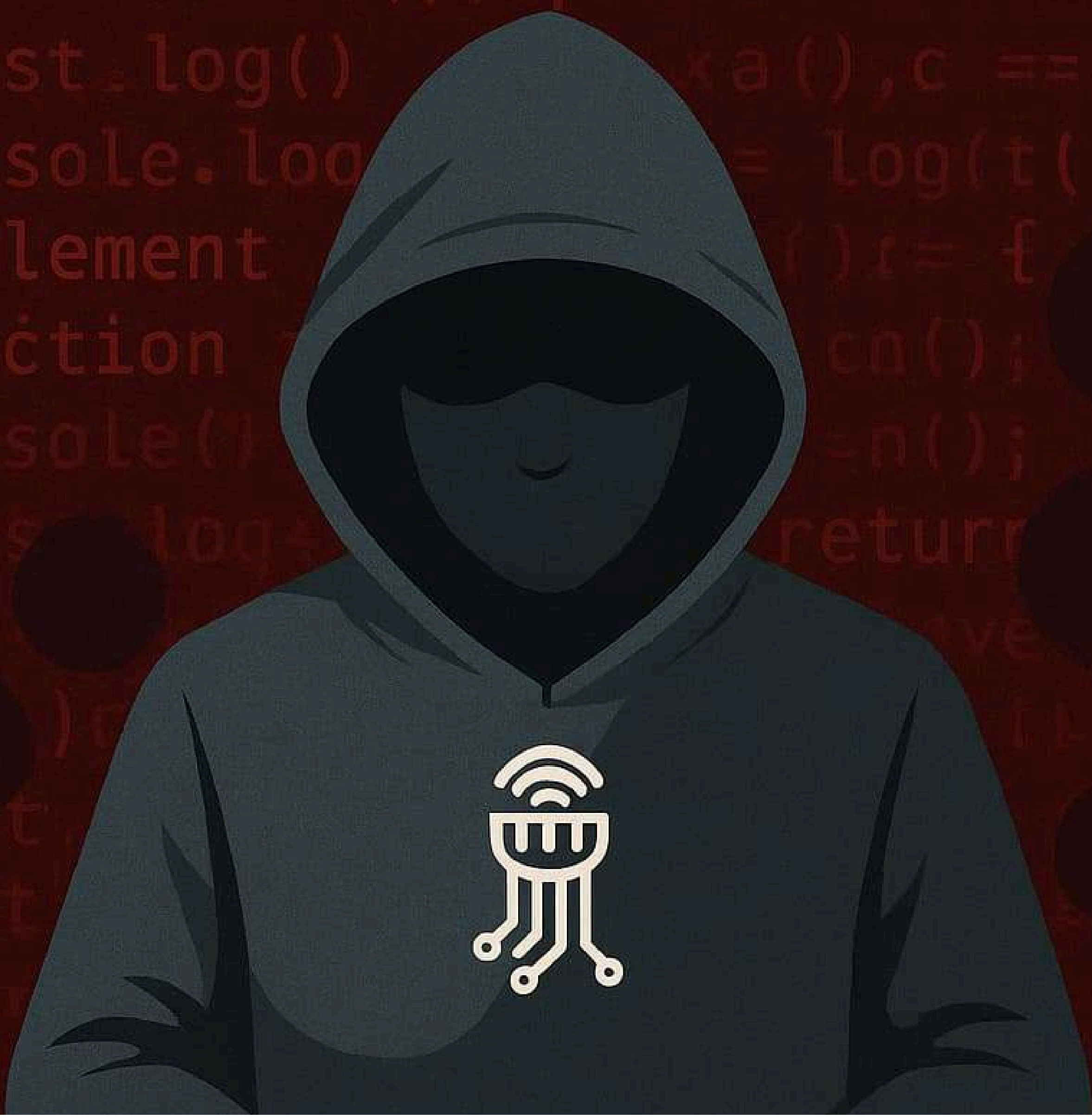
Digital Egypt Pioneers

Ministry of Communications

# Red Team Capstone Challenge Report




Digital Egypt Pioneers

1. OSINT (Simulated)
2. Enumeration & Fuzzing
3. Phishing
4. AV Evasion
5. Lateral Movement
6. AD Exploitation
7. Linux and Windows Security Testing
8. Privilege Escalation
9. Post-Compromise Exploitation

## Project Introduction

As part of a red team engagement facilitated by TryHackMe, I was individually assigned to simulate a full-scope offensive operation against **TheReserve**, the national bank of **Trimento**, a Pacific island nation. Although Trimento is small in size, its substantial international investments make the security of its central financial institution a matter of national importance. This assessment was designed to replicate the activities of an advanced threat actor targeting the bank's most critical assets.

## Infrastructure Overview

TheReserve is divided into two key operational divisions. The **Corporate Division** manages external investments and relationships with international corporate clients, making it a more exposed environment. In contrast, the **Banking Division** operates the country's core financial infrastructure, including its SWIFT transaction systems. The government of Trimento raised concerns about the insufficient network and access segregation between these two divisions, fearing that an adversary compromising the Corporate side could pivot internally and reach the Banking systems.

## Engagement Objectives

My role in the red team operation focused on three primary objectives. First, I needed to assess whether the Corporate Division could be compromised through external or phishing-based means. Second, I was to determine if such a compromise would allow for **lateral movement** into the Banking Division. Finally, the engagement aimed to simulate a **fraudulent SWIFT transaction** between two internal test accounts, reflecting a worst-case financial breach scenario. Successfully reaching and manipulating the SWIFT system was the final success criterion.

## SWIFT Simulation Design

The SWIFT environment was configured to represent a highly secure, segregated financial system. Although it was considered "uncompromisable" under normal operations, a limited internal exposure was created for the simulation. The SWIFT web interface was accessible at http://swift.bank.thereserve.loc/ and operated under a strict **separation-of-duties model**.

In this workflow, a **Transfer Code** is first generated. A user with the **Capturer** role logs into the SWIFT web application and captures the transaction request. Then, a separate **Approver** connects via a jump host to review and approve the transaction. Only after both roles fulfill their responsibilities does the transaction proceed, simulating real-world security controls within critical banking workflows.

## Scope of Engagement

The engagement was executed under clearly defined scope boundaries. The following activities were permitted:

- Internal and external network testing via VPN access
- Internal reconnaissance and OSINT targeting. .thereserve.loc domains
- Phishing campaigns against employees
- Access to and exploitation of mailboxes hosted at MAIL.thereserve.loc
- Any approach that enables a valid SWIFT transaction simulation between test accounts

**Activities that were explicitly out-of-scope included:**

- Attacks on systems outside the assigned subnet
- Modifying mail server infrastructure
- Targeting fellow red team operators or their accounts
- Conducting internet-based (external) OSINT

The provided network range was 10.200.X.0/24, with the exact subnet revealed after registration.

## Initial Access and Environment Registration

Initial access was established through SSH login to the internal **e-Citizen portal** using the following credentials:

- Username: e-citizen
- Password: stabilitythroughcurrency
- Host IP: X.X.X.250 (based on assigned subnet)

This portal served as the centralized communication platform for the operation. It provided email provisioning, phishing infrastructure, challenge tracking, and flag submission. It was also used to validate progress and enforce operational rules. All tampering with the VPN or the e-Citizen infrastructure was strictly prohibited.

## Operational Summary

This engagement simulated a nation-state-level attack chain targeting critical banking infrastructure. As an individual operator, my responsibilities included compromising the Corporate Division, establishing lateral movement to reach the Banking Division, and completing a SWIFT transaction under realistic operational constraints. The outcome of this simulation was intended to inform whether TheReserve's Corporate Division could continue operating within the same security boundary or should be isolated to reduce risk.

## Environment Preparation

Before the active engagement, I performed several setup tasks to prepare the environment:

- Populated the /etc/hosts file with known internal hostnames
- Completed SSH registration to authenticate with the e-Citizen platform
- Verified access to the email mailbox and control portal

With the environment ready, I proceeded to the first stage of the attack: internal reconnaissance and enumeration of exposed systems and credentials. The subsequent sections of this report will walk through each phase of the operation, from initial access to final objective execution.

## Preparations and Initial Setup

Before beginning the active engagement, I performed several preparatory steps to ensure proper environment configuration and readiness. These steps were essential for establishing communication, resolving internal resources, and accessing the red team infrastructure.

## Downloading Capstone Challenge Resources

The first step involved retrieving the Capstone Challenge resources. These included two essential files: one describing the organization’s password policy and another containing a base password list to support brute-force or credential spraying activities. Additionally, a collection of suggested tools was provided to assist during the engagement. Although the use of these tools was optional, they served as a helpful reference for commonly required functionality such as enumeration, privilege escalation, and lateral movement.

## Updating the Hosts File

To ensure stable internal name resolution regardless of subnet changes, I updated the local /etc/hosts file with key internal services. This allowed consistent access to hostnames even when IPs were dynamic or varied per operator assignment. The following entries were added:

`10.200.116.11 MAIL.thereserve.loc`

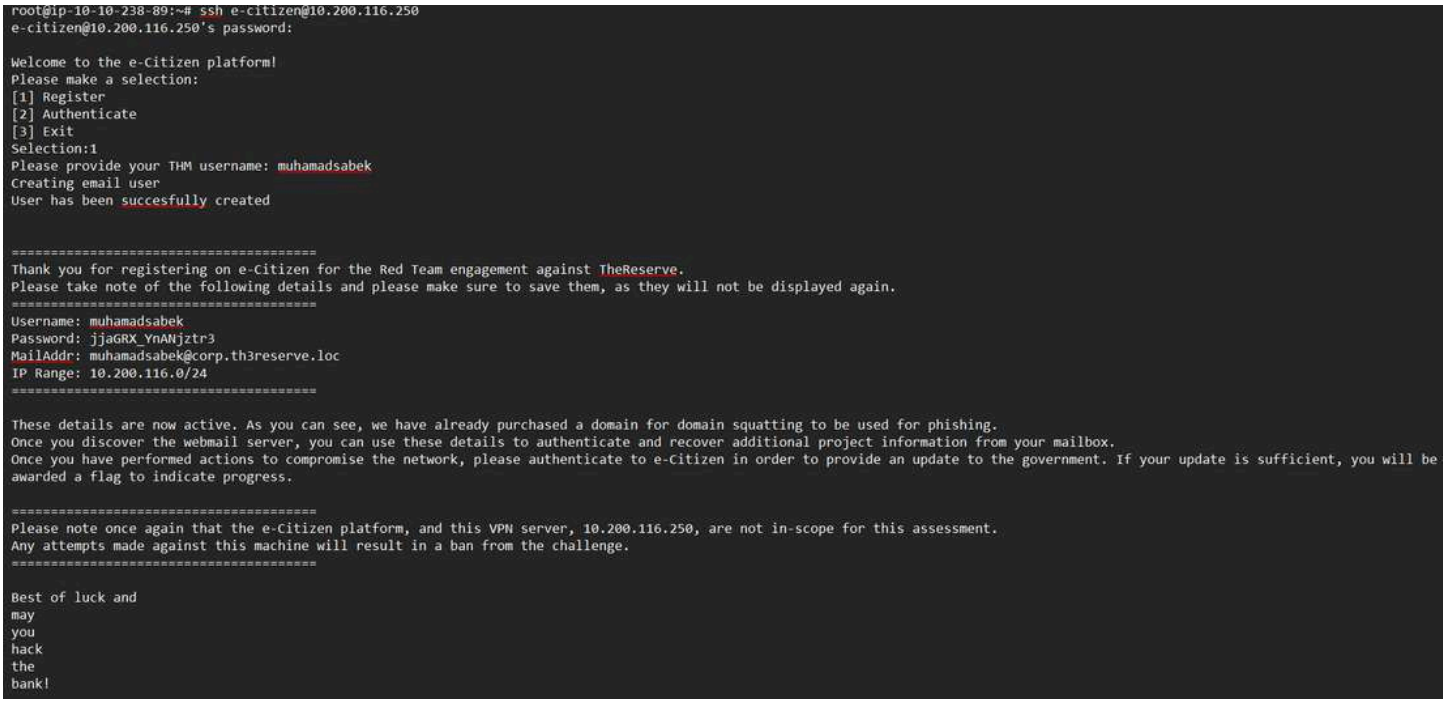
`10.200.116.12 VPN.thereserve.loc`

`10.200.116.13 WEB.thereserve.loc`

This step was crucial for maintaining access to internal services such as the mail server, VPN interface, and web portals throughout the engagement.

## SSH Registration and Portal Access

Initial access to the red team infrastructure was obtained by connecting to the **e-Citizen communication portal** via SSH. Using the credentials provided during setup, I successfully authenticated and registered my operator profile. This portal served multiple critical functions during the challenge, including phishing infrastructure deployment, flag validation, email communication, and tracking engagement progress.



```
root@ip-10-10-238-89:~# ssh e-citizen@10.200.116.250
e-citizen@10.200.116.250's password:

Welcome to the e-Citizen platform!
Please make a selection:
[1] Register
[2] Authenticate
[3] Exit
Selection:1
Please provide your THM username: muhamadsabek
Creating email user
User has been succesfully created


=======================================
Thank you for registering on e-Citizen for the Red Team engagement against TheReserve.
Please take note of the following details and please make sure to save them, as they will not be displayed again.
=======================================
Username: muhamadsabek
Password: jjaGRX_YnANjztr3
MailAddr: muhamadsabek@corp.th3reserve.loc
IP Range: 10.200.116.0/24
=======================================

These details are now active. As you can see, we have already purchased a domain for domain squatting to be used for phishing.
Once you discover the webmail server, you can use these details to authenticate and recover additional project information from your mailbox.
Once you have performed actions to compromise the network, please authenticate to e-Citizen in order to provide an update to the government. If your update is sufficient, you will be awarded a flag to indicate progress.

=======================================
Please note once again that the e-Citizen platform, and this VPN server, 10.200.116.250, are not in-scope for this assessment.
Any attempts made against this machine will result in a ban from the challenge.
=======================================

Best of luck and
may
you
hack
the
bank!
```

Upon successful registration, the following account details were assigned:

- **Username:** muhamadsabek
- **Password:** jjaGRX_YnANjztr3
- **Email Address:** muhamadsabek@corp.th3reserve.loc
- **Assigned IP Range:** 10.200.116.0/24

With the portal access confirmed and the environment fully configured, I was ready to proceed with the initial reconnaissance and begin the first operational phase of the red team engagement.

## Exploring The Network

```
root@ip-10-10-46-88:~# nmap 10.200.116.12 -sV -sC
Starting Nmap 7.80 ( https://nmap.org ) at 2025-04-12 05:20 BST
Nmap scan report for 10.200.116.12
Host is up (0.0022s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 17:f3:c2:89:2a:eb:25:90:02:f9:e0:c1:a8:6f:b3:3c (RSA)
|   256 53:8c:34:1c:e2:5d:2d:2f:69:df:b9:4f:1d:13:fa:18 (ECDSA)
|_  256 02:3f:29:8d:a6:58:51:0e:c9:ee:5f:f3:1a:04:92:24 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: VPN Request Portal
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.59 seconds
[1]+  Done                    thunderbird
```

**Result**

VPN 10.200.116.12

22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.5

80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))

**WebMail**

```
root@ip-10-10-46-88:~# nmap 10.200.116.11 -sV -sC
```

**Result**

WebMail 10.200.116.11

22 ssh

25 stmp

110

135

139

143

587

3306/tcp open  mysql

3389/tcp open  ms-wbt-server Microsoft Terminal Services

**WEB machine**

```
root@ip-10-10-46-88:~# nmap 10.200.116.13 -sV -sC
Starting Nmap 7.80 ( https://nmap.org ) at 2025-04-12 05:45 BST
Nmap scan report for 10.200.116.13
Host is up (0.0072s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 ac:ae:01:d7:8e:da:bf:5c:ff:b5:69:93:79:94:2b:52 (RSA)
|   256 81:5a:9e:79:a5:70:00:cf:8d:d0:8a:18:6a:37:67:91 (ECDSA)
|_  256 53:4d:82:5f:b3:f5:ee:d6:e5:35:d8:f6:b4:cf:24:99 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.96 seconds
```

**Result**

Web 10.200.116.13

22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7

80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))

Target_Name: THERESERVE

|   NetBIOS_Domain_Name: THERESERVE

|   NetBIOS_Computer_Name: MAIL

|   DNS_Domain_Name: thereserve.loc

|   DNS_Computer_Name: MAIL.thereserve.loc

```
sudo nano /etc/hosts
```

```
  GNU nano 4.8                  /etc/hosts                         Modified

127.0.0.1       localhost
127.0.0.1       vnc.tryhackme.tech
127.0.1.1       tryhackme.lan   tryhackme
10.200.116.11  MAIL.thereserve.loc mail.thereserve.loc
10.200.116.11  VPN.thereserve.loc  vpn.thereserve.loc
10.200.116.11  WEB.thereserve.loc  web.thereserve.loc thereserve.loc


# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

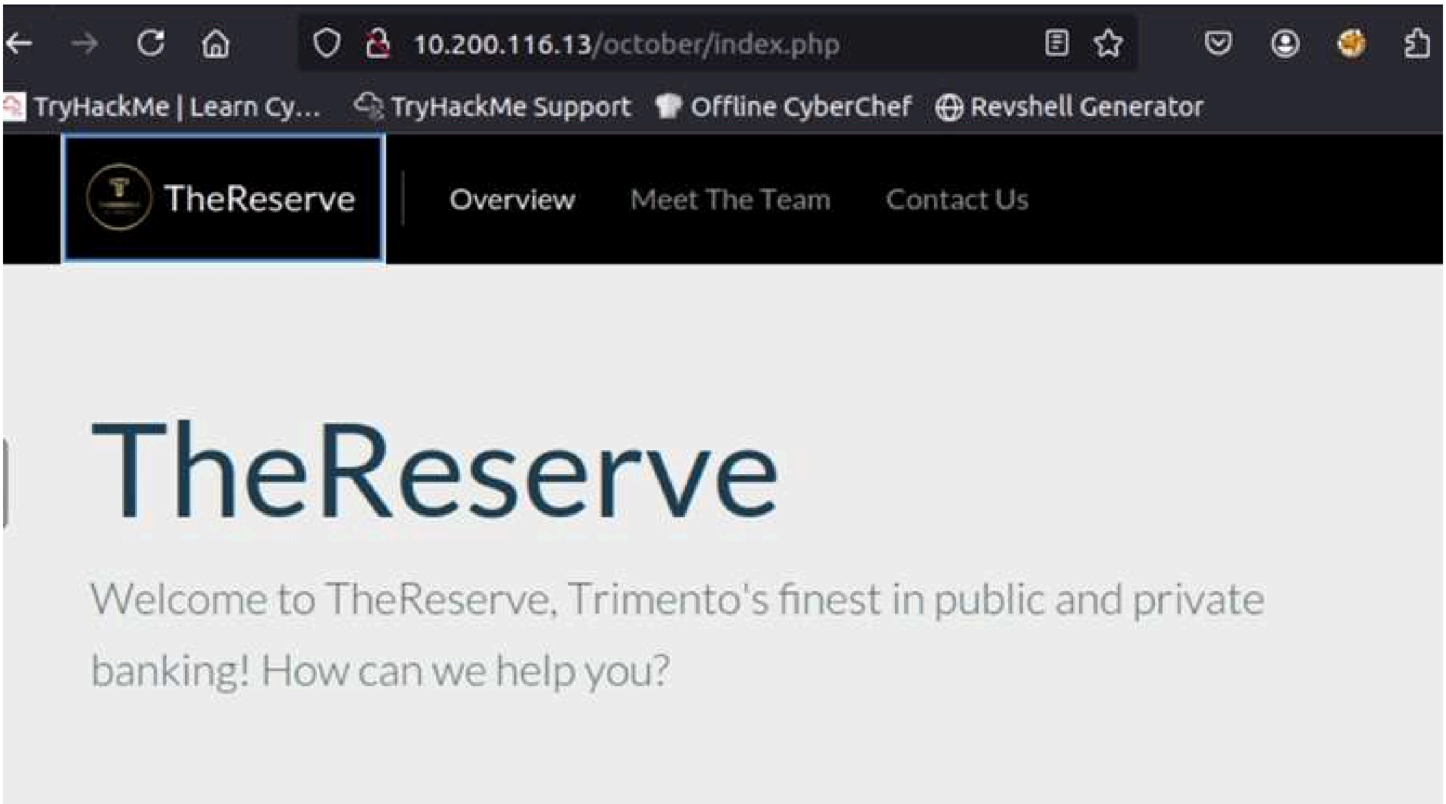
**10.200.116.11  MAIL.thereserve.loc  mail.thereserve.loc**

**10.200.116.12  VPN.thereserve.loc  vpn.thereserve.loc**

**10.200.116.13  WEB.thereserve.loc  web.thereserve.loc thereserve.loc**

## The Web Page

This IP is hosting a web page that gives us an overview about the company and its team.



## Overview

TheReserve is the reserve bank of Trimento. We aim to serve both the country by providing stability to the public banking sector, but also through our corporate division serve investors from foreign countries. We believe that a stable currency leads to a stable country, and centre all we do around this belief.

### Trimento

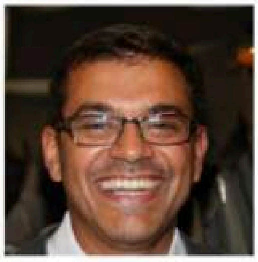
Trimento welcome those from other countries looking for something different. Trimento offers a digital nomadship programme that allows those that meet the pre-requisites to join our country and embrace a different lifestyle! No need for cubicles and the old nine to five. Why not work from your own private villa looking out over our crips beaches? Why not use your lunch break for a safari ride? Why not chose working hours that suits you and enjoy you leisure time exploring our world reknown markets? This lifestyle can be yours with the support of TheReserve!

As this simulates real red team engagements, knowing company workers is a critical part of any redteam engagements.

Personal Assistance to the Executives

Lynda

Project Manager



Roy

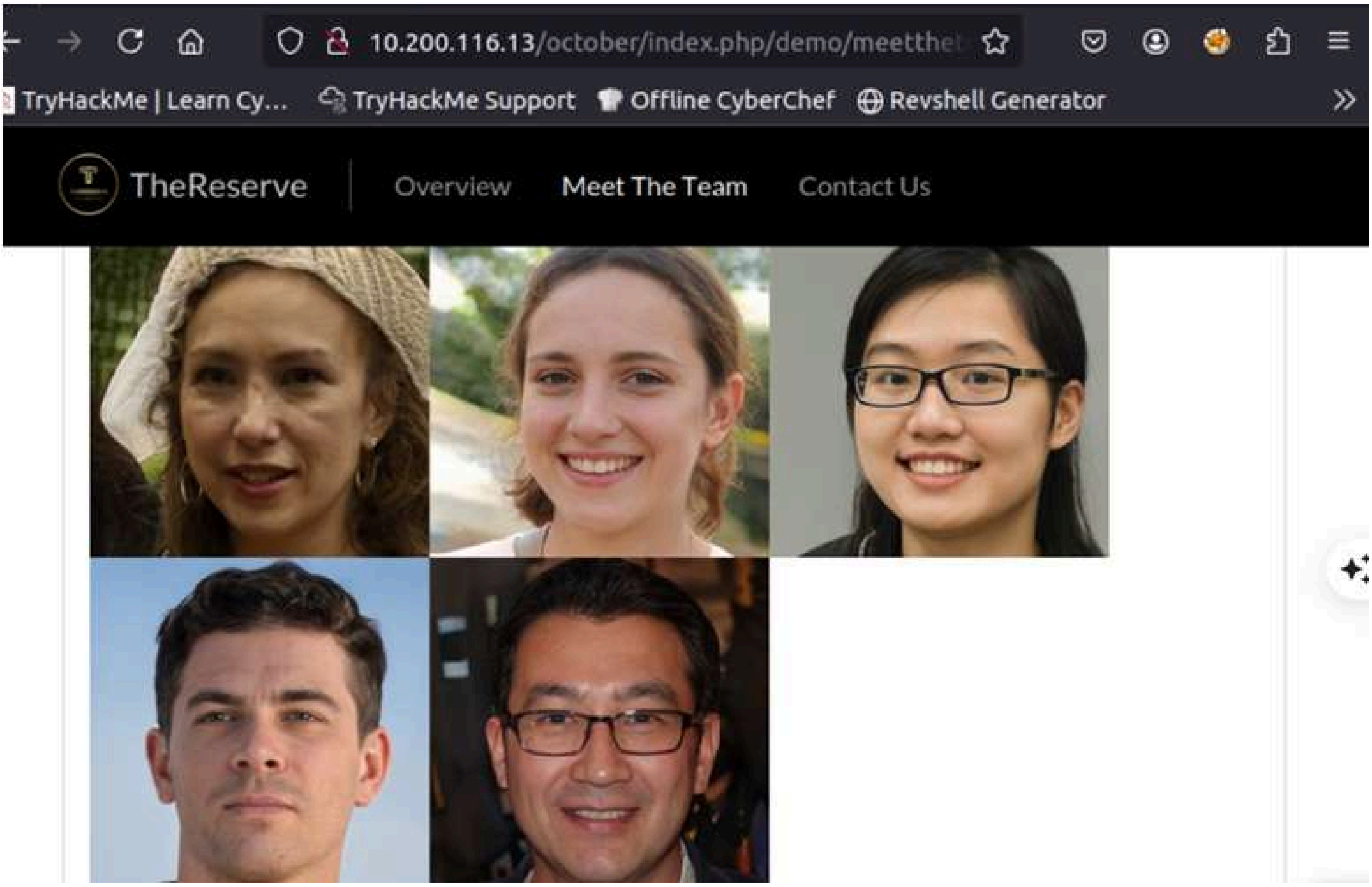
Corporate Customer Investment Managers

And many more!

# Users

```
Aimee Walker -- Lead Developers
Patrick Edwards -- Lead Developers
Brenda Henderson -- Bank Director
Leslie Morley -- Deputy Directors
Martin Savage -- Deputy Directors
paula bailey -- CEO
christopher smith -- CIO
antony ross -- CTO
charlene thomas -- CMO
rhys parsons -- COO
lynda gordon -- Personal Assistance to the Executives
Roy sims -- Project Manager
laura wood
emily harvey
ashley chan
keith allen
mohammad ahmed
```

Also, looking at the image name gave us hints about the email creation rules that are being used by the organization: **firstname.lastname@domain.com**



```
Aimee.Walker@corp.thereserve.loc
Patrick.Edwards@corp.thereserve.loc
Brenda.Henderson@corp.thereserve.loc
Leslie.Morley@corp.thereserve.loc
Martin.Savage@corp.thereserve.loc
paula.bailey@corp.thereserve.loc
christopher.smith@corp.thereserve.loc
antony.ross@corp.thereserve.loc
charlene.thomas@corp.thereserve.loc
rhys.parsons@corp.thereserve.loc
lynda.gordon@corp.thereserve.loc
Roy.sims@corp.thereserve.loc
laura.wood@corp.thereserve.loc
emily.harvey@corp.thereserve.loc
ashley.chan@corp.thereserve.loc
keith.allen@corp.thereserve.loc
mohammad.ahmed@corp.thereserve.loc
```

## Password List Preparation

To prepare for brute-force authentication attempts, I generated a custom wordlist from the provided password_base_list.txt. Using a loop and mp64, I applied a custom character set to match the bank's enforced password policy, which included uppercase, lowercase, digits, and special characters !@#$%^.

```
┌──(aya㉿kali)-[~/Capstone_Challenge_Resources]
└─$ while read pass; do mp64 -1='!@#$%^' "${pass}?d?l"; done < password_base_list.txt > pass.txt

┌──(aya㉿kali)-[~/Capstone_Challenge_Resources]
└─$ cat pass.txt | head
TheReserve0a
TheReserve0b
TheReserve0c
TheReserve0d
TheReserve0e
TheReserve0f
TheReserve0g
TheReserve0h
TheReserve0i
TheReserve0j
```

## Initial Access – Credential Generation and Brute-Force Attack

### Attack Execution

With the password list ready, I used Hydra, a powerful brute-forcing tool, to launch an SMTP password attack against MAIL.thereserve.loc. The goal was to identify valid login credentials for internal email accounts.

**Successful authentication confirmed initial access to internal mailboxes.**

This foothold provided not just user credentials but a clear pivot point into email communications, essential for the next stage: internal OSINT and phishing.

### Outcome

- Entry Point Gained: Internal SMTP login
- Tool Used: Hydra
- Brute-Force Wordlist: Custom-generated using mp64
- Impact: Email access established, enabling social engineering and lateral movement

## Hydra attack

This step revealed that mail accounts use predictable formats and passwords based on a weak policy, which presents a major risk of unauthorized access.

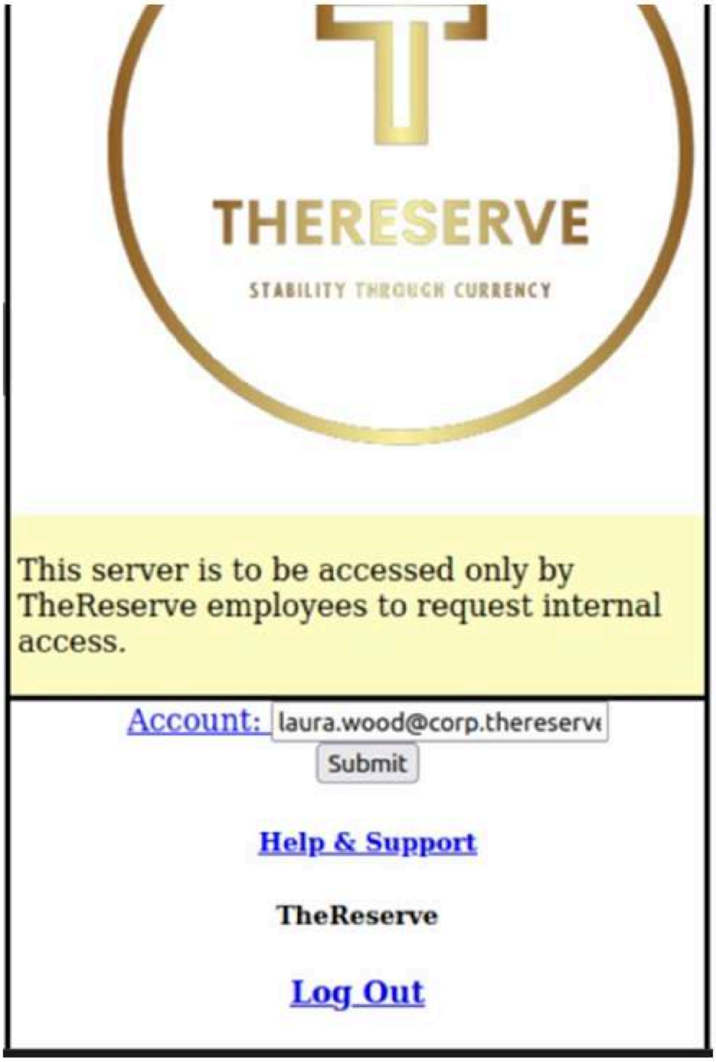
```
┌──(aya㉿kali)-[~/Capstone_Challenge_Resources]
└─$ hydra -L users.txt -P pass.txt smtp://10.200.116.11
```

**Result**

```
login: laura.wood@corp.thereserve.loc   password: Password1@
```

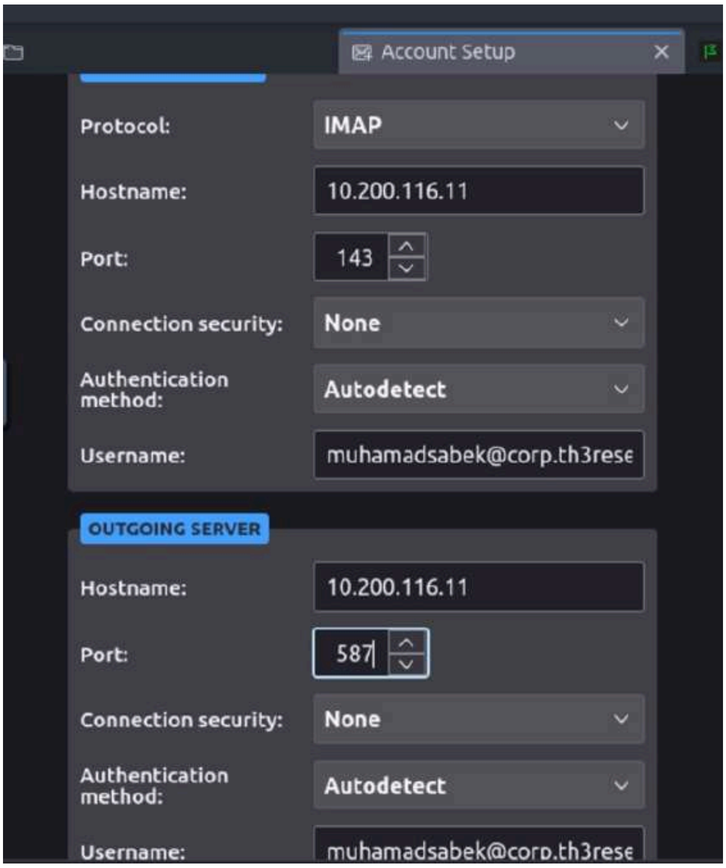
```
login: mohammad.ahmed@corp.thereserve.loc   password: Password
```

## Access the Website




## Accessing Internal Email via Thunderbird

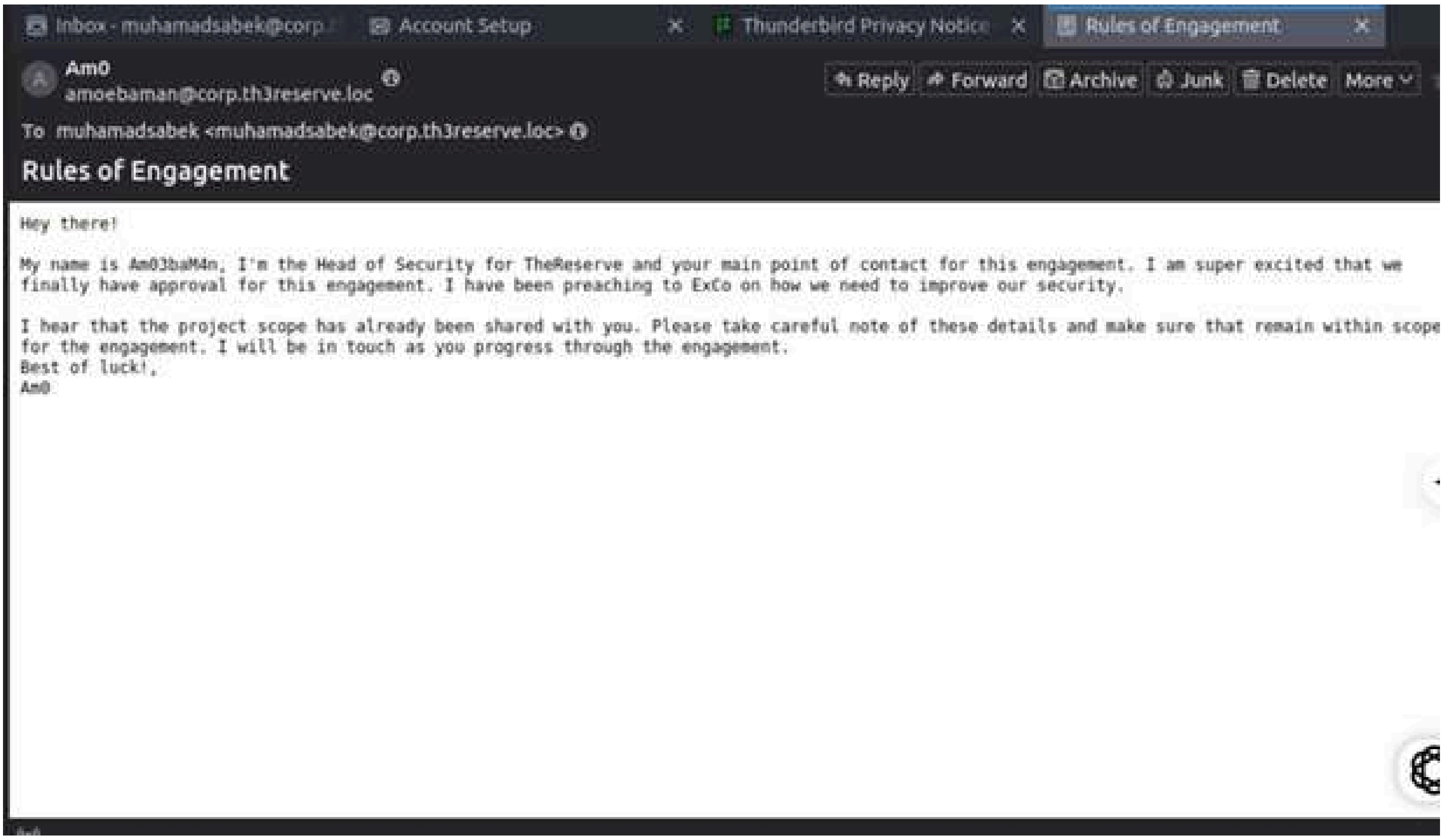
After successfully brute-forcing SMTP credentials, we configured the Thunderbird email client using the compromised account (muhamadsabek@corp.th3reserve.loc). This granted direct access to internal communications, enabling enumeration of users, harvesting internal information, and setting the stage for phishing and lateral movement.

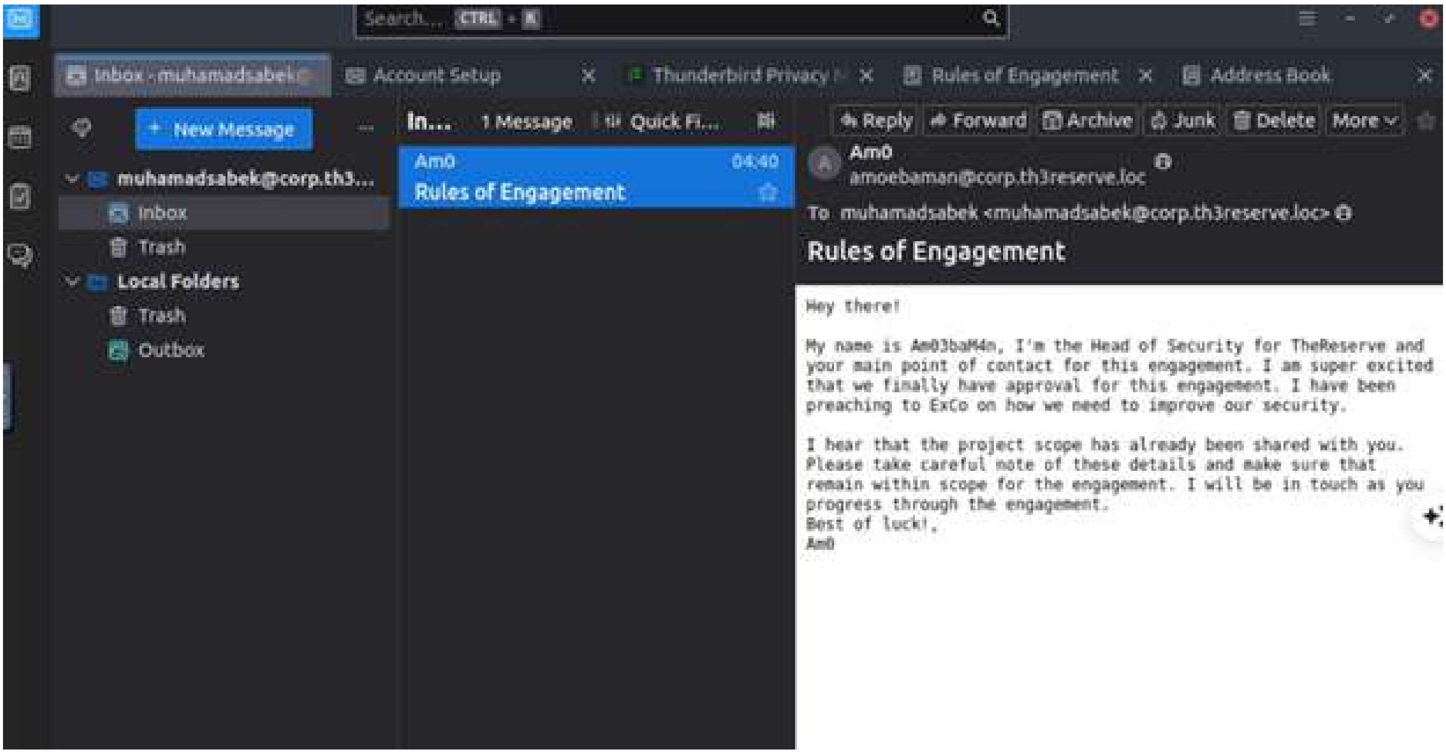



Once connected, we will find the following message:

## Internal Email Evidence

A screenshot from the mailbox of muhamadsabek@corp.th3reserve.loc, accessed after a successful phishing attempt and credential compromise. The email from the Head of Security confirms internal communication and validates the foothold within the simulated network.


Inbox - muhamadsabek@corp.
Account Setup
Thunderbird Privacy Notice
Rules of Engagement
Am0
amoebaman@corp.th3reserve.loc
Reply
Forward
Archive
Junk
Delete
More
To muhamadsabek <muhamadsabek@corp.th3reserve.loc>
Rules of Engagement
Hey there!
My name is Am03baM4n, I'm the Head of Security for TheReserve and your main point of contact for this engagement. I am super excited that we finally have approval for this engagement. I have been preaching to ExCo on how we need to improve our security.
I hear that the project scope has already been shared with you. Please take careful note of these details and make sure that remain within scope for the engagement. I will be in touch as you progress through the engagement.
Best of luck!,
Am0


Search... CTRL + K
Inbox - muhamadsabek
Account Setup
Thunderbird Privacy N
Rules of Engagement
Address Book
New Message
muhamadsabek@corp.th3...
Inbox
Trash
Local Folders
Trash
Outbox
In...
1 Message
Quick Fi...
Am0
04:40
Rules of Engagement
Reply
Forward
Archive
Junk
Delete
More
Am0
amoebaman@corp.th3reserve.loc
To muhamadsabek <muhamadsabek@corp.th3reserve.loc>
Rules of Engagement
Hey there!
My name is Am03baM4n, I'm the Head of Security for TheReserve and your main point of contact for this engagement. I am super excited that we finally have approval for this engagement. I have been preaching to ExCo on how we need to improve our security.
I hear that the project scope has already been shared with you. Please take careful note of these details and make sure that remain within scope for the engagement. I will be in touch as you progress through the engagement.
Best of luck!,
Am0

| Step | Tool | Target | Result |
| --- | --- | --- | --- |
| Brute-Force SMTP | Hydra | mail.thereserve.loc | Credentials harvested |
| Email Client Setup | Thunderbird | Internal Mailbox | Access to internal emails |

# Antivirus Evasion and Covert Access Establishment

### Objective

Establish a reverse shell from the internal target server while bypassing active antivirus (AV) and endpoint detection and response (EDR) mechanisms to achieve root-level remote access.

## Engagement Scenario

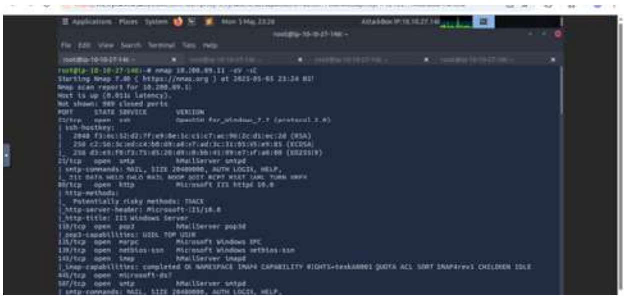
| **Component** | **Value** |
|---|---|
| **Attacker Machine IP** | **10.50.115.21** |
| **Target Server IP** | **10.200.89.12 (VPN server)** |
| **Listening Port** | **443 (HTTPS - chosen to blend with normal traffic)** |

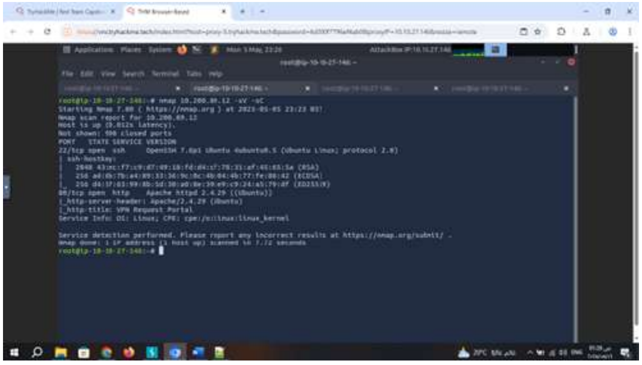
## Target Reconnaissance

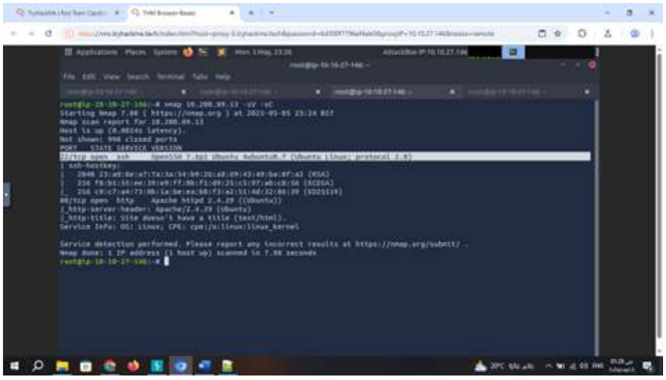
An initial network scan was conducted across the internal subnet 10.200.89.0/24 to enumerate live hosts and exposed services. The following hosts and their associated services were identified:

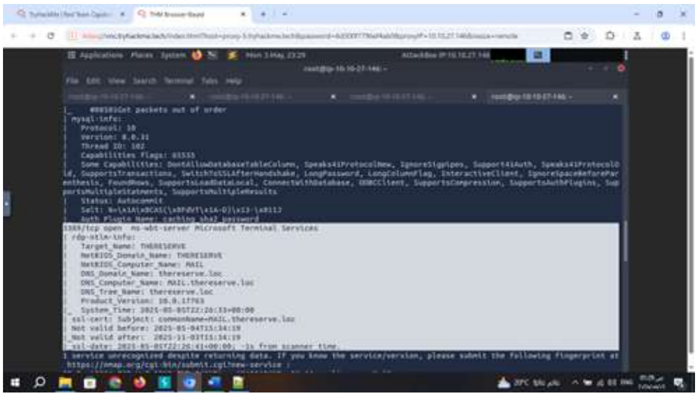
- 10.200.89.11 – MAIL.thereverse.loc (Mail Server)
- 10.200.89.12 – VPN.thereverse.loc (VPN Gateway)
- 10.200.89.13 – WEB.thereverse.loc (Internal Web Server)

This reconnaissance phase helped prioritize targets based on exposed attack surfaces and potential access value.

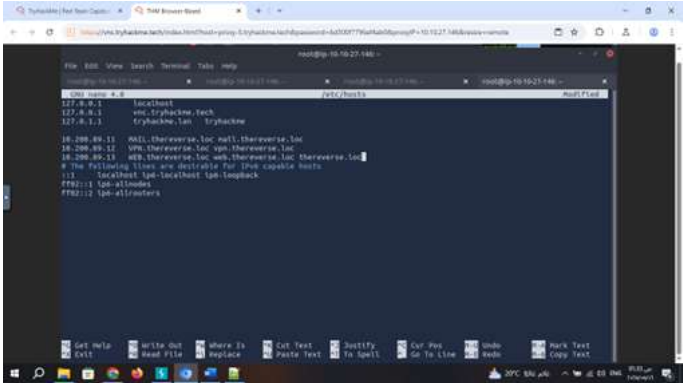








Hosts added to /etc/hosts using:
sudo nano /etc/hosts

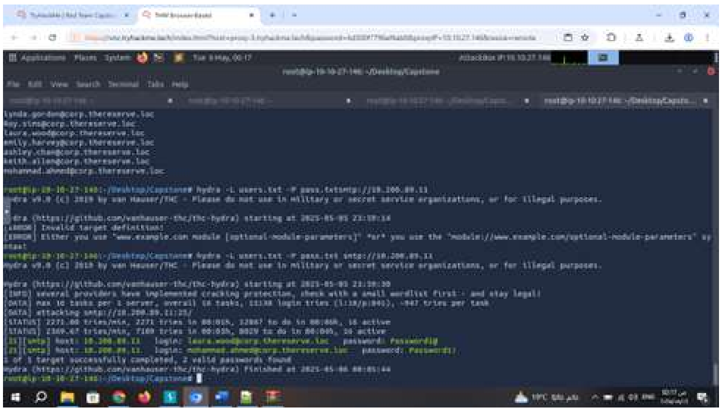


## Initial Access

To gain a foothold within the internal environment, a combination of credential brute-forcing and web-based attack techniques was utilized.

### Brute Force Attack

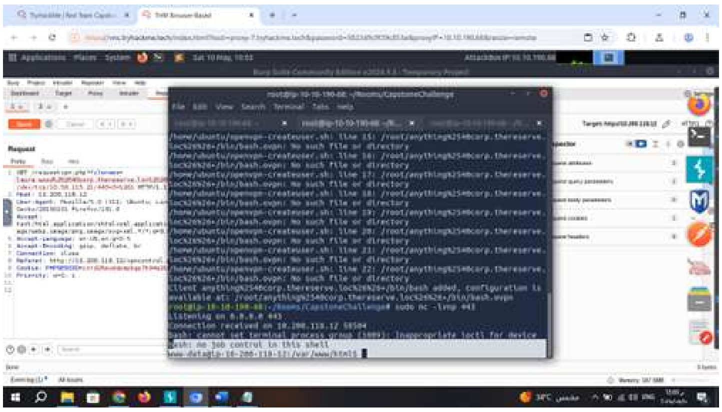
Using Hydra, a dictionary-based brute force attack was executed against login portals exposed on the internal network. This resulted in the discovery of valid credentials for the VPN web interface:

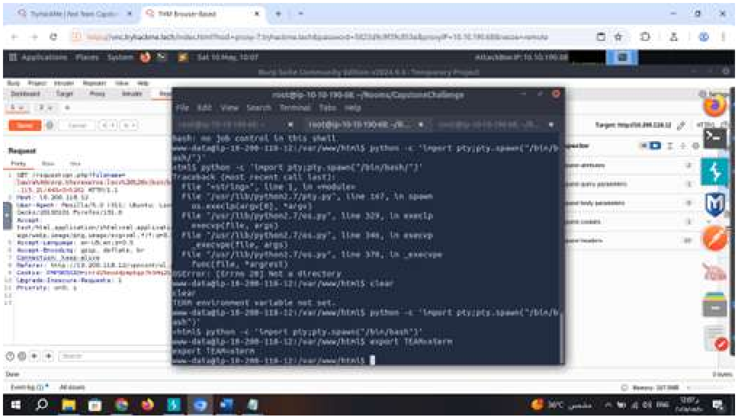


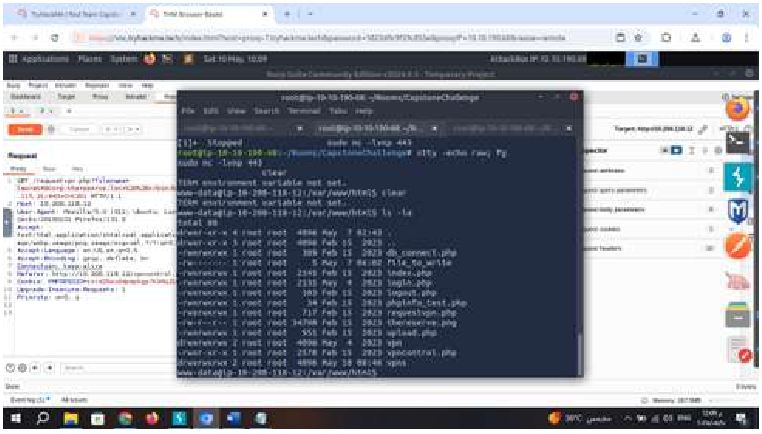
### Payload Generation

Created a custom reverse shell payload:

- Listening on port: 443
- Ensured payload evaded antivirus detection.
- Used techniques such as obfuscation and avoiding known signatures.



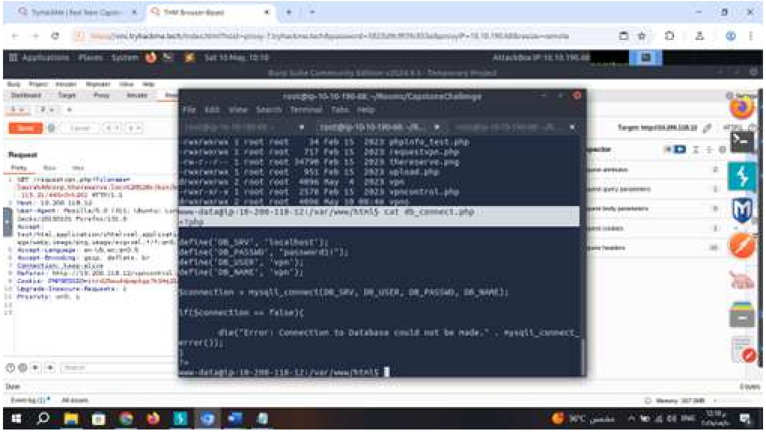


define('DB_SRV', 'localhost');

define('DB_USER', 'vpn');

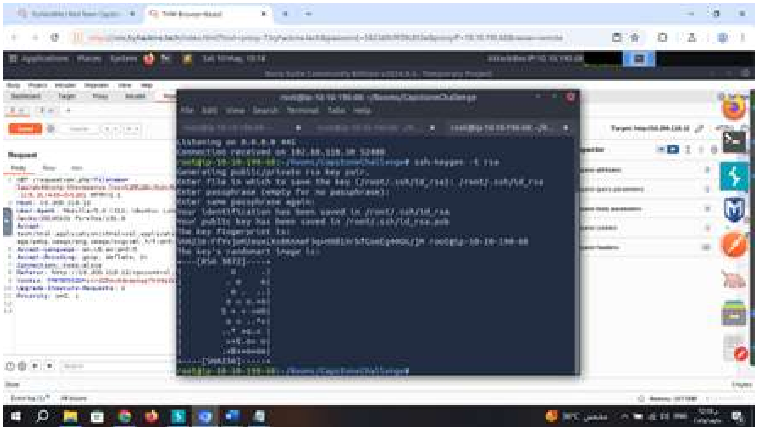
define('DB_PASSWD', 'password1!');

define('DB_NAME', 'vpn');

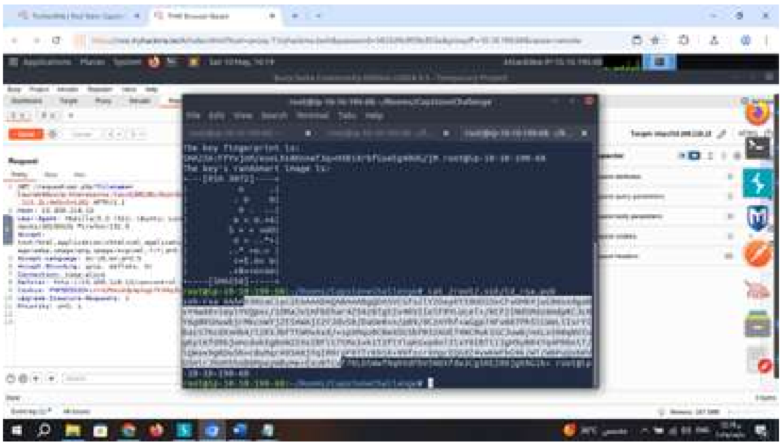


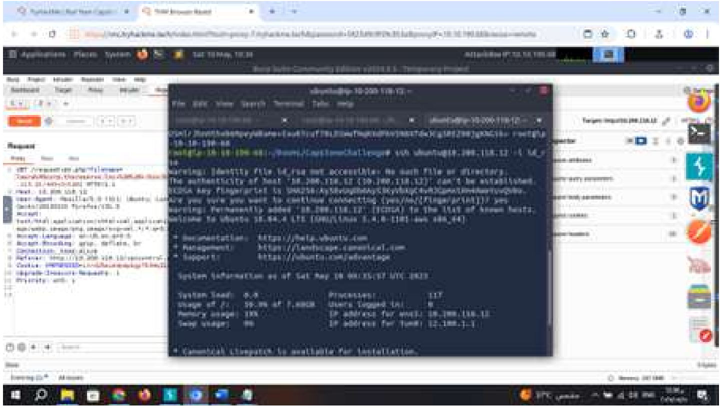
## SSH Key Injection

Attempted to inject SSH key into the authorized_keys file:
LFILE=/home/ubuntu/.ssh/authorized_keys
echo "ssh-rsa AAAA... root@ip-10-10-190-68" | sudo cp /dev/stdin "$LFILE"



Verified the key was properly written:
cat /root/.ssh/id_rsa.pub

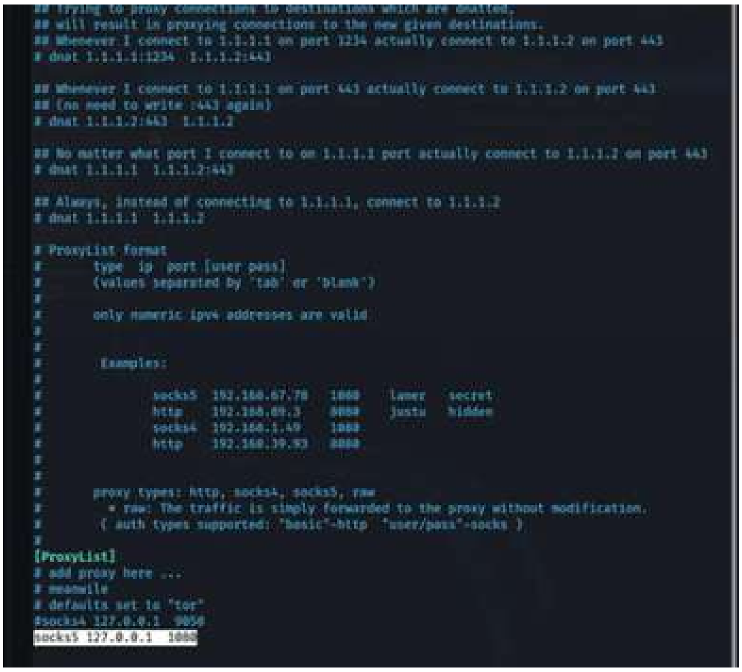




## Privilege Escalation & Persistence

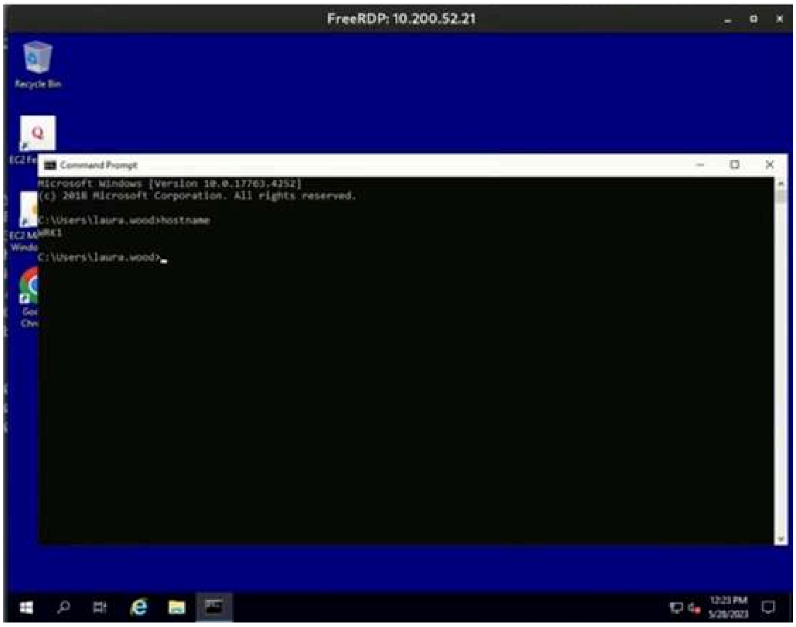
SSH key allowed persistent access as root.
Example reverse shell started from compromised host:
python3 -m http.server 80

Adjusted proxychains to tunnel traffic:
nano /etc/proxychains4.conf




# Remote Desktop Access – WRK1

Tool Used: xfreerdp
xfreerdp /u:laura.wood@corp.thereverse.loc /p:password@1 /v:10.200.118.21




## Outcome

The engagement resulted in full compromise of the target server with the following outcomes:

- **Reverse Shell Access**

Successfully established a reverse shell connection from the target server to the attacker-controlled host over port 443, enabling remote command execution.

- **AV Evasion Success**

Payload execution was performed without triggering any antivirus or endpoint detection mechanisms. The custom-crafted payload evaded all active defenses through obfuscation and signature avoidance techniques.

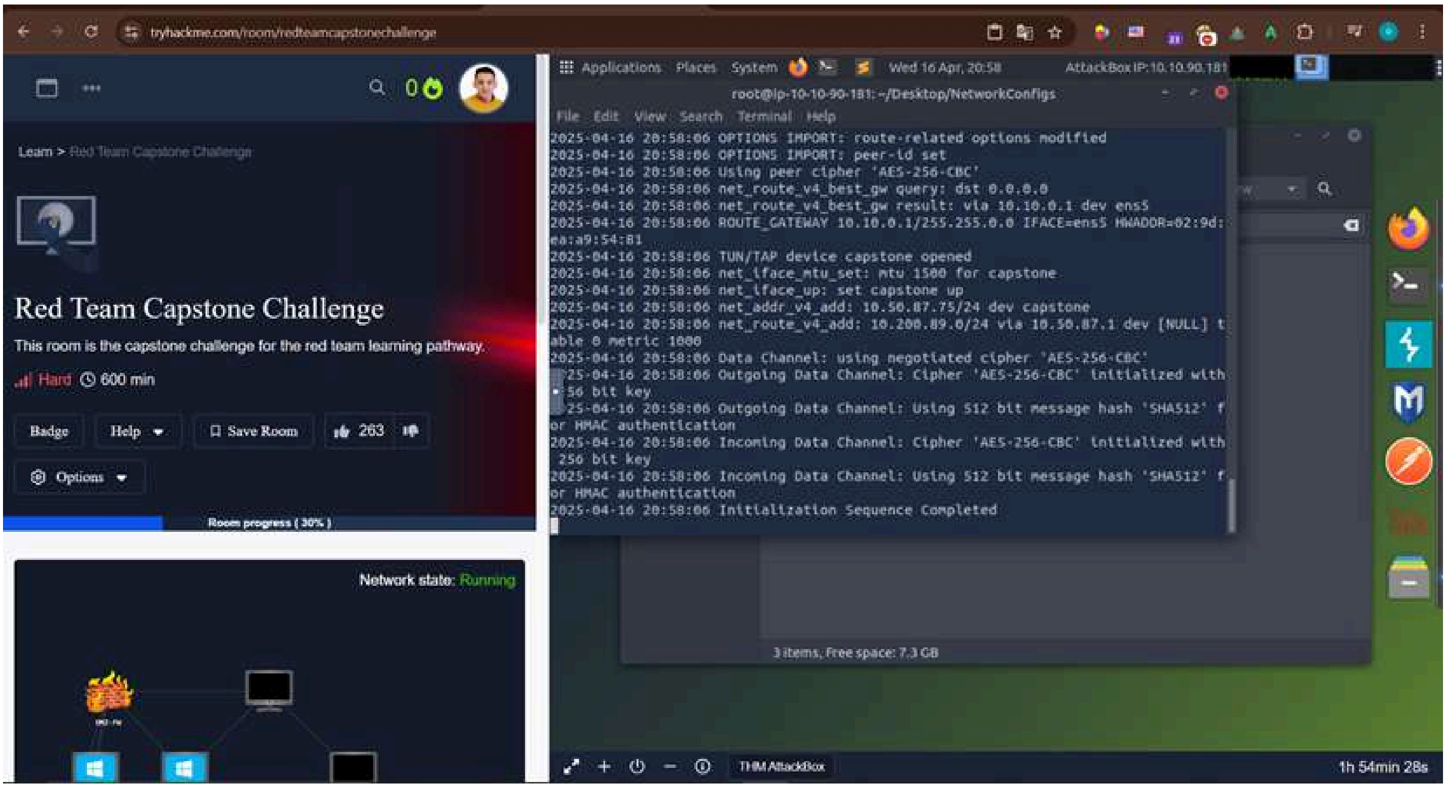
- **Persistence Achieved**

Persistent root-level access was secured via SSH key injection into /root/.ssh/authorized_keys, allowing repeated and stealthy access to the system without relying on credentials or re-exploitation.

# Lateral Movement

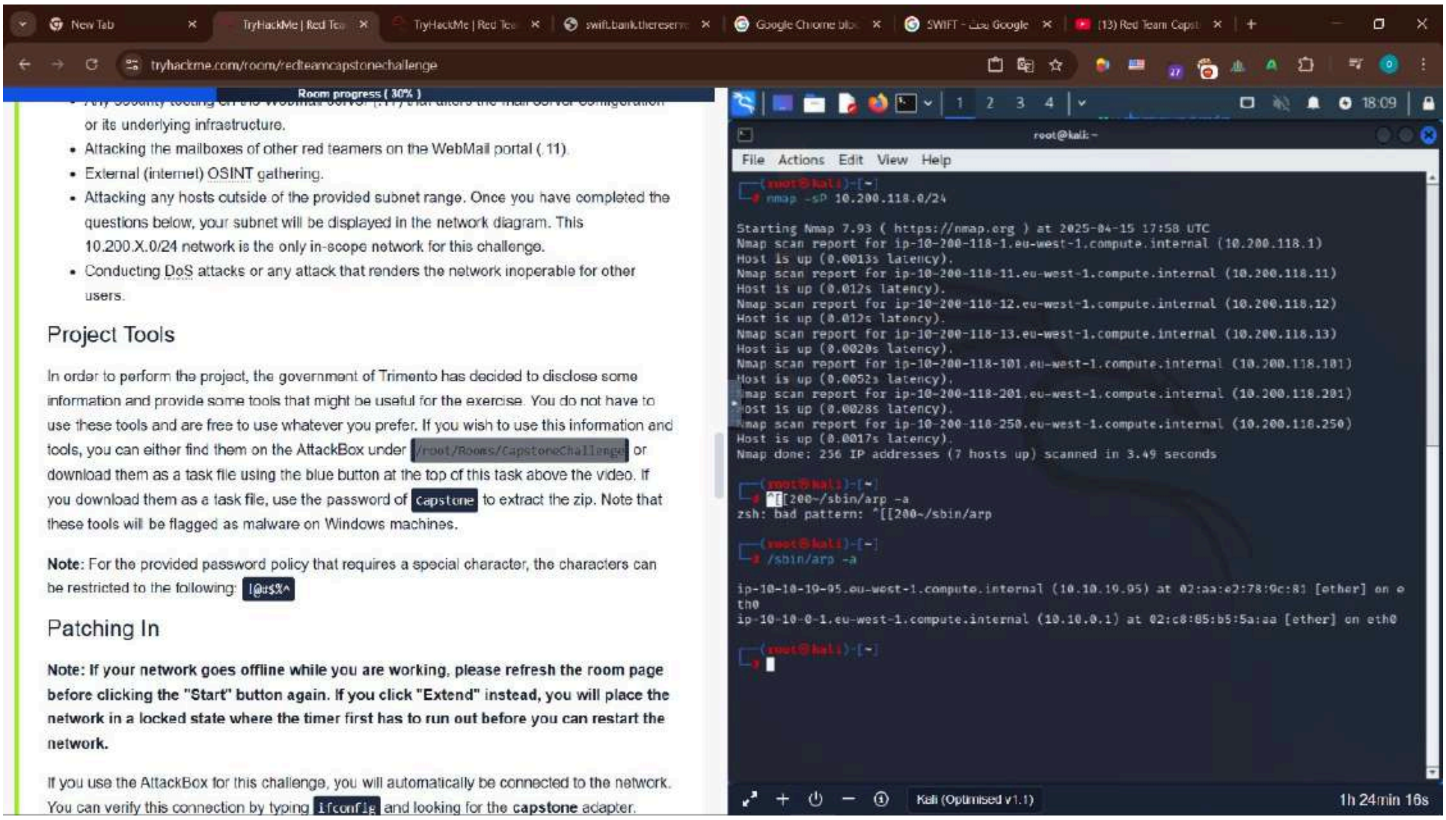
- Lateral Movement Overview

In this Red Team Capstone challenge, we simulate a real-world attack scenario where initial access is gained through a vulnerable exposed system in the DMZ. From there, lateral movement is performed to pivot deeper into the internal network. This involves harvesting credentials, exploiting trust relationships, and moving across different network segments — from Tier 2 workstations to Tier 1 servers — until administrative access is obtained. Lateral movement allows attackers to expand their control, evade detection, and reach critical systems such as Active Directory and SWIFT banking infrastructure.



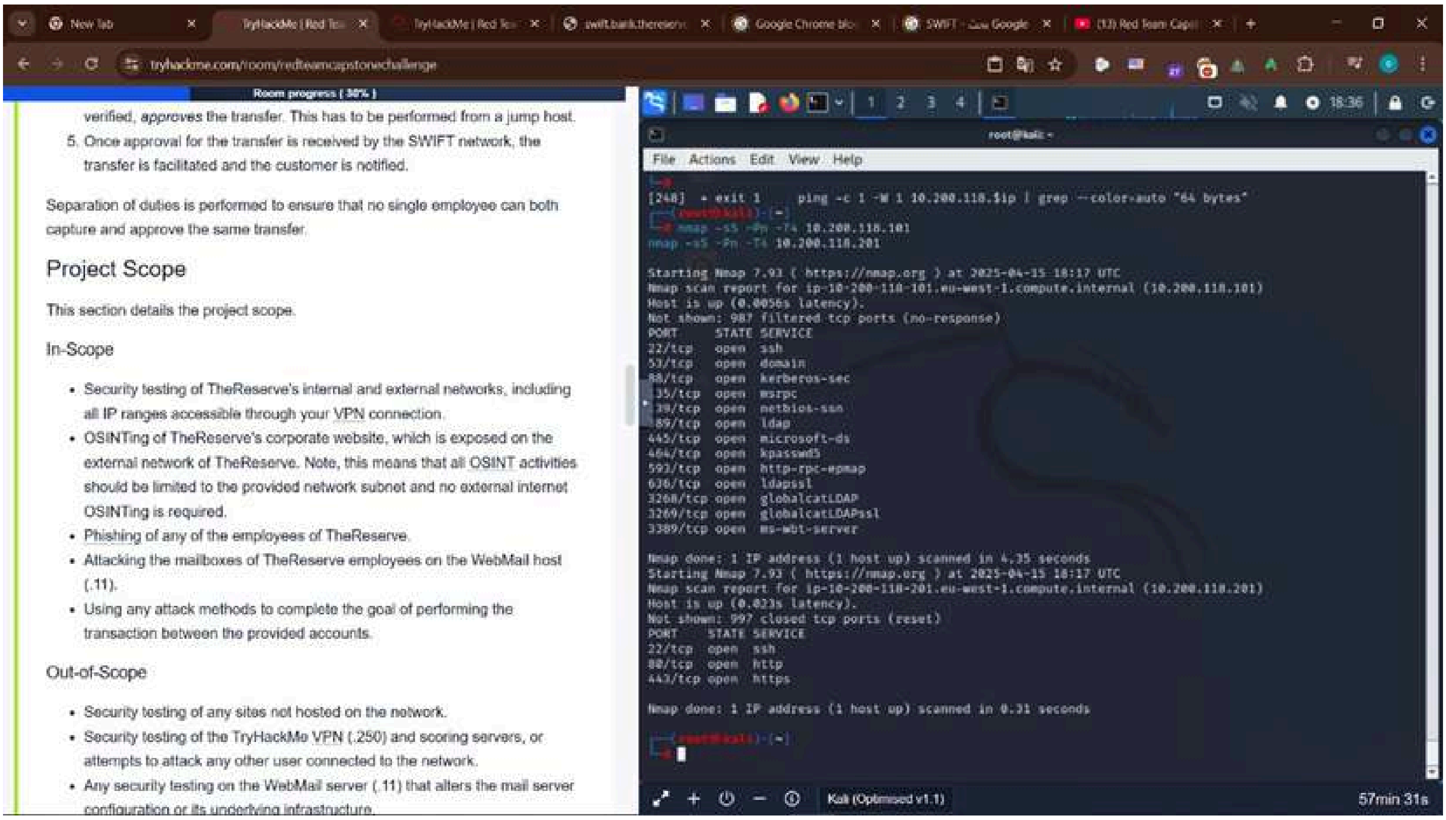

**Step 1 – VPN File Discovered**

A `.ovpn` configuration file is found, which indicates OpenVPN access to the internal network. This is the attacker's entry point into the organization's perimeter network.



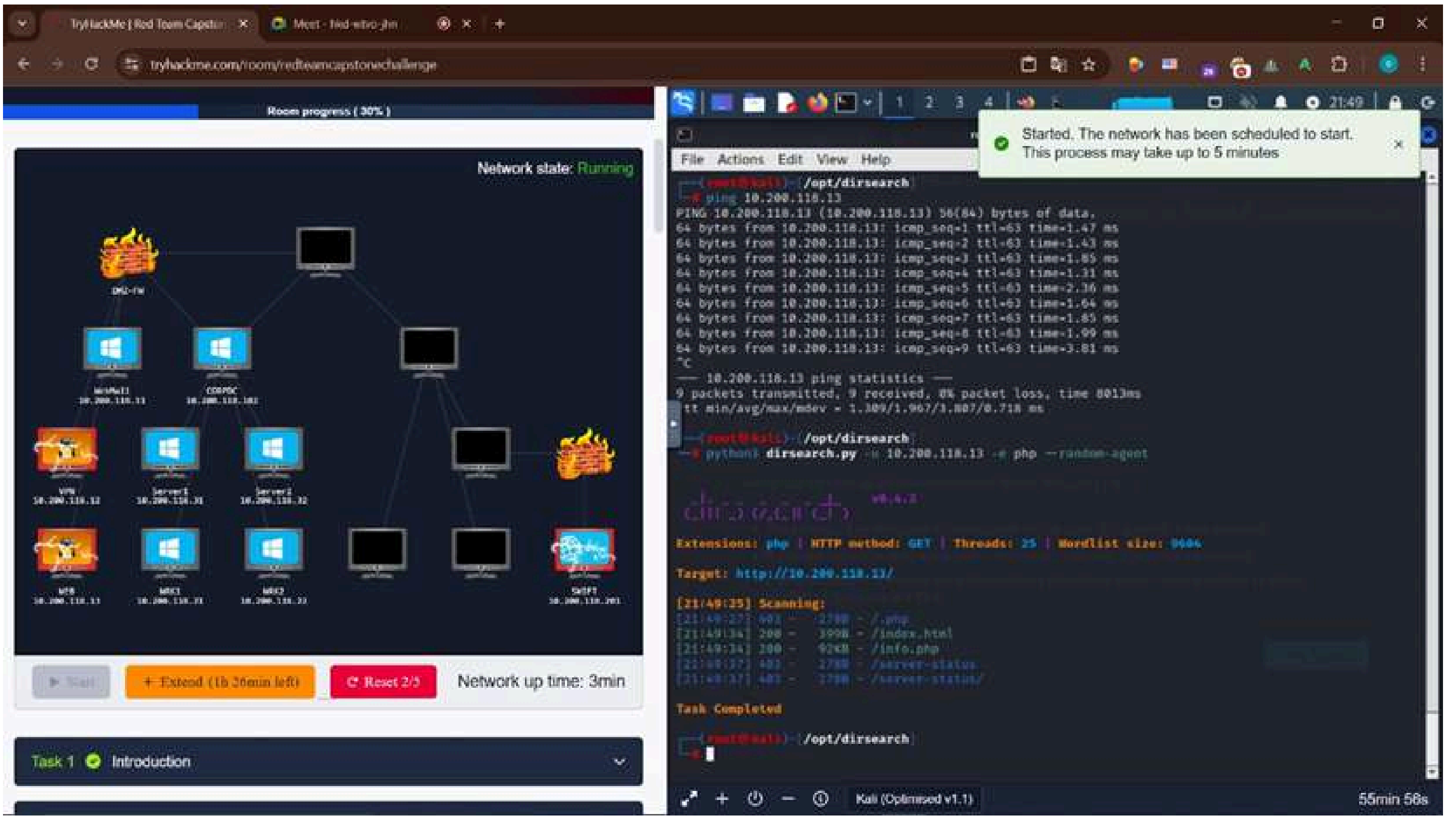
**step 2:**

This screenshot shows the initial reconnaissance phase in the Red Team Capstone challenge, where the attacker uses nmap -sP to discover active hosts in the 10.200.118.0/24 subnet. Running nmap is crucial for identifying reachable targets and planning lateral movement across the network.



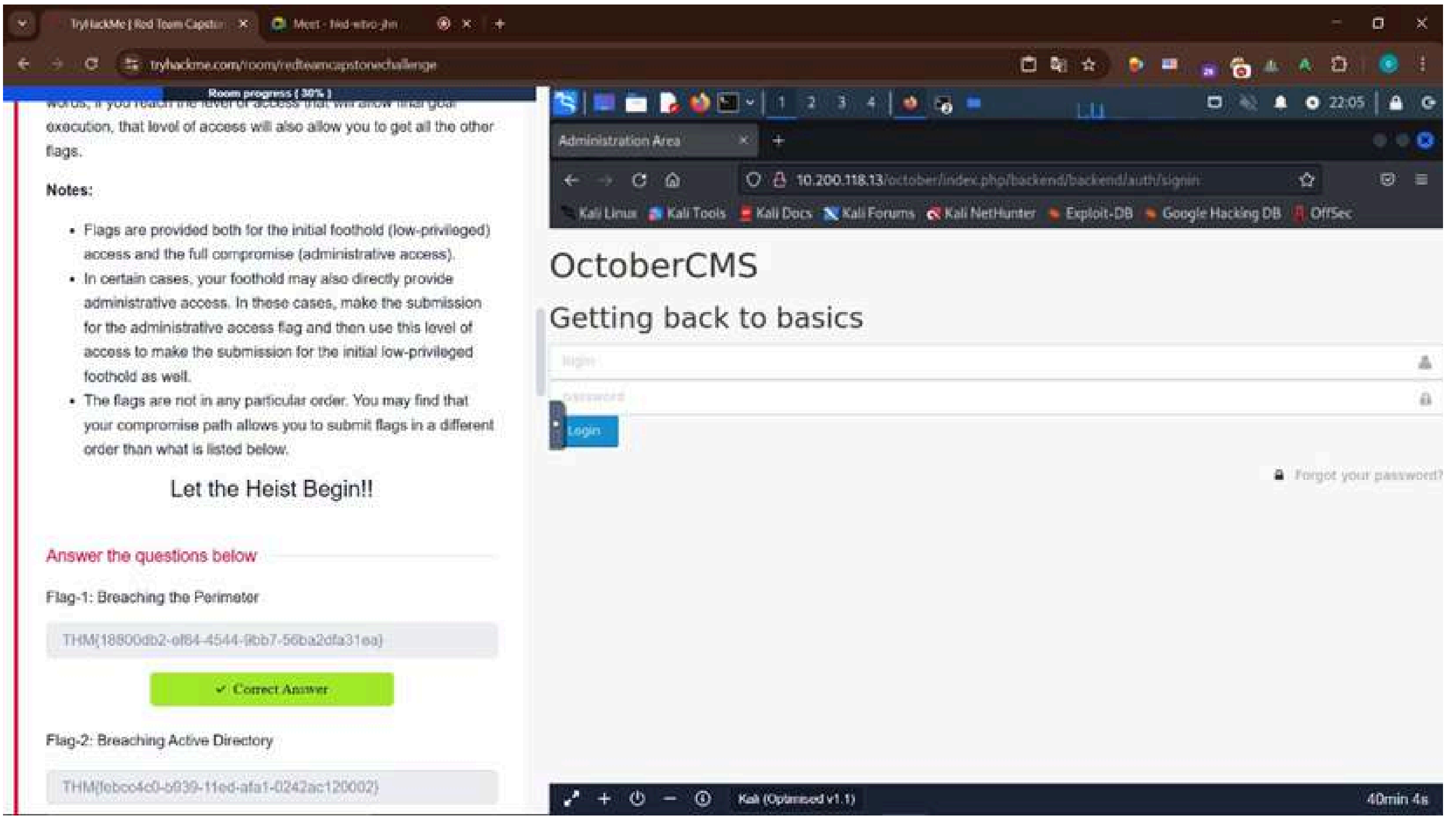
**Step 3**

The screenshot shows a targeted nmap SYN scan (-sS -Pn -T4) against two hosts in the Red Team Capstone subnet. Host 10.200.118.101 reveals multiple open ports—**22, 53, 88, 135, 139, 389, 443, 445, 593, 636, 3268, 3269, 3389**—indicating it's likely a Windows Domain Controller running services like LDAP, Kerberos, and RDP. In contrast, 10.200.118.201 is reachable but has all ports filtered or closed, likely due to a host-based firewall.



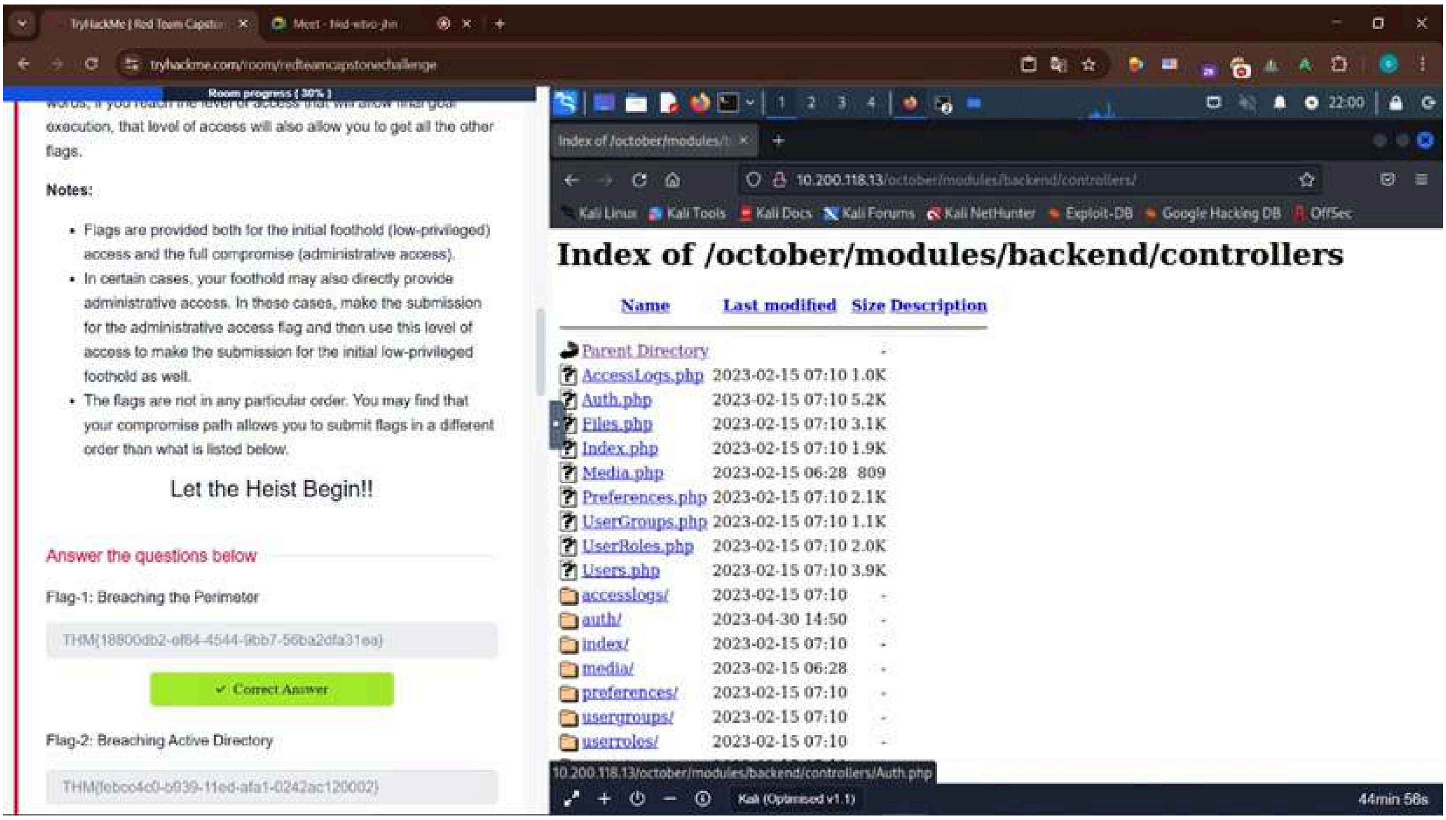

**Step 4 –**

The attacker confirms the web server 10.200.118.13 is alive and performs directory enumeration using dirsearch, revealing key endpoints like /info.php and /server-status, which may expose sensitive configuration data or server insights for further exploitation.




**Step 5 –**

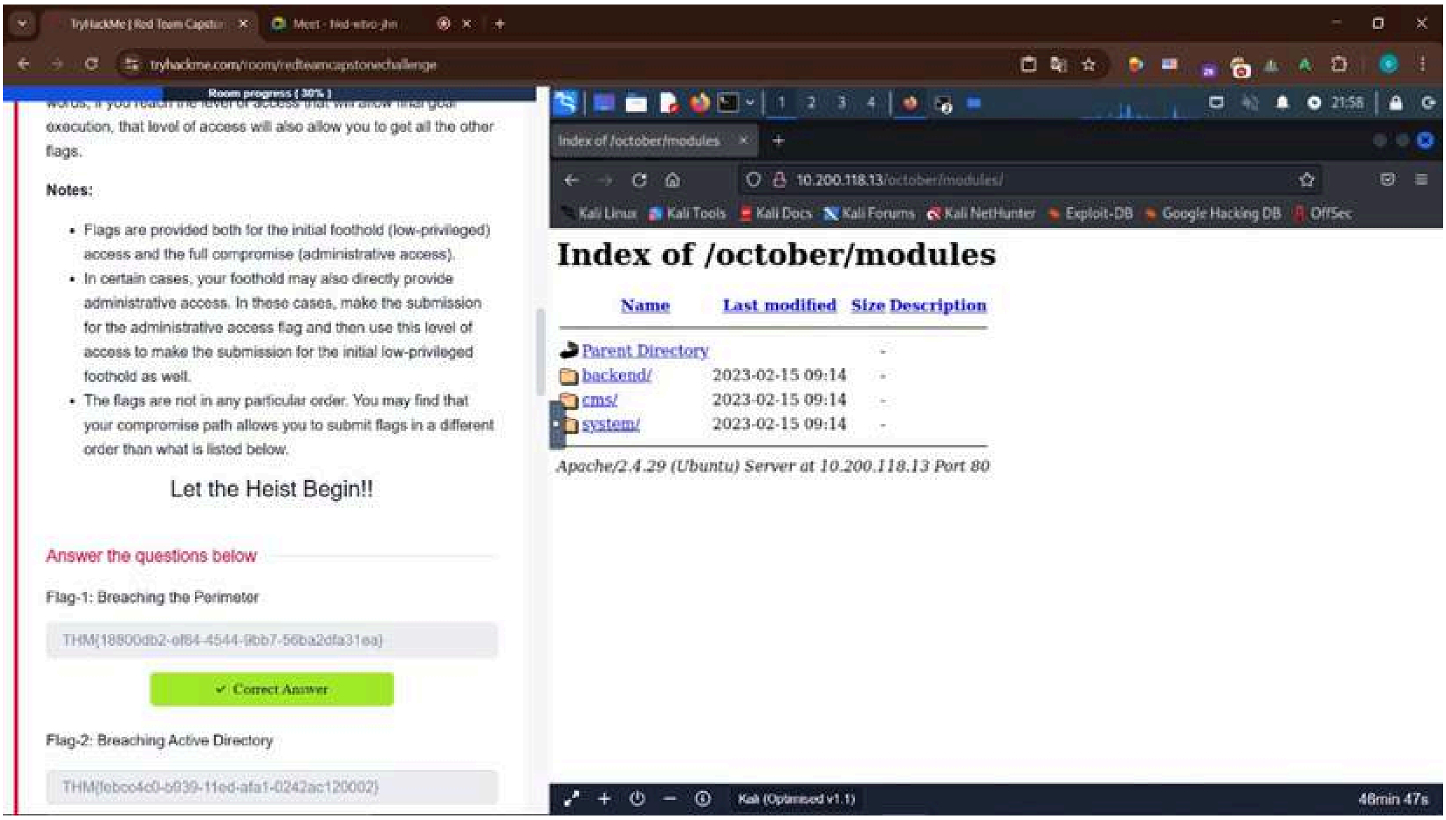
The attacker has discovered an OctoberCMS login page on 10.200.118.13 after breaching the perimeter (Flag-1), marking a key foothold opportunity. This CMS interface may be vulnerable to default credentials, reused passwords, or known exploits (e.g., file upload, RCE), providing a potential path for privilege escalation or lateral movement within the target environment.




**Step 6 –**

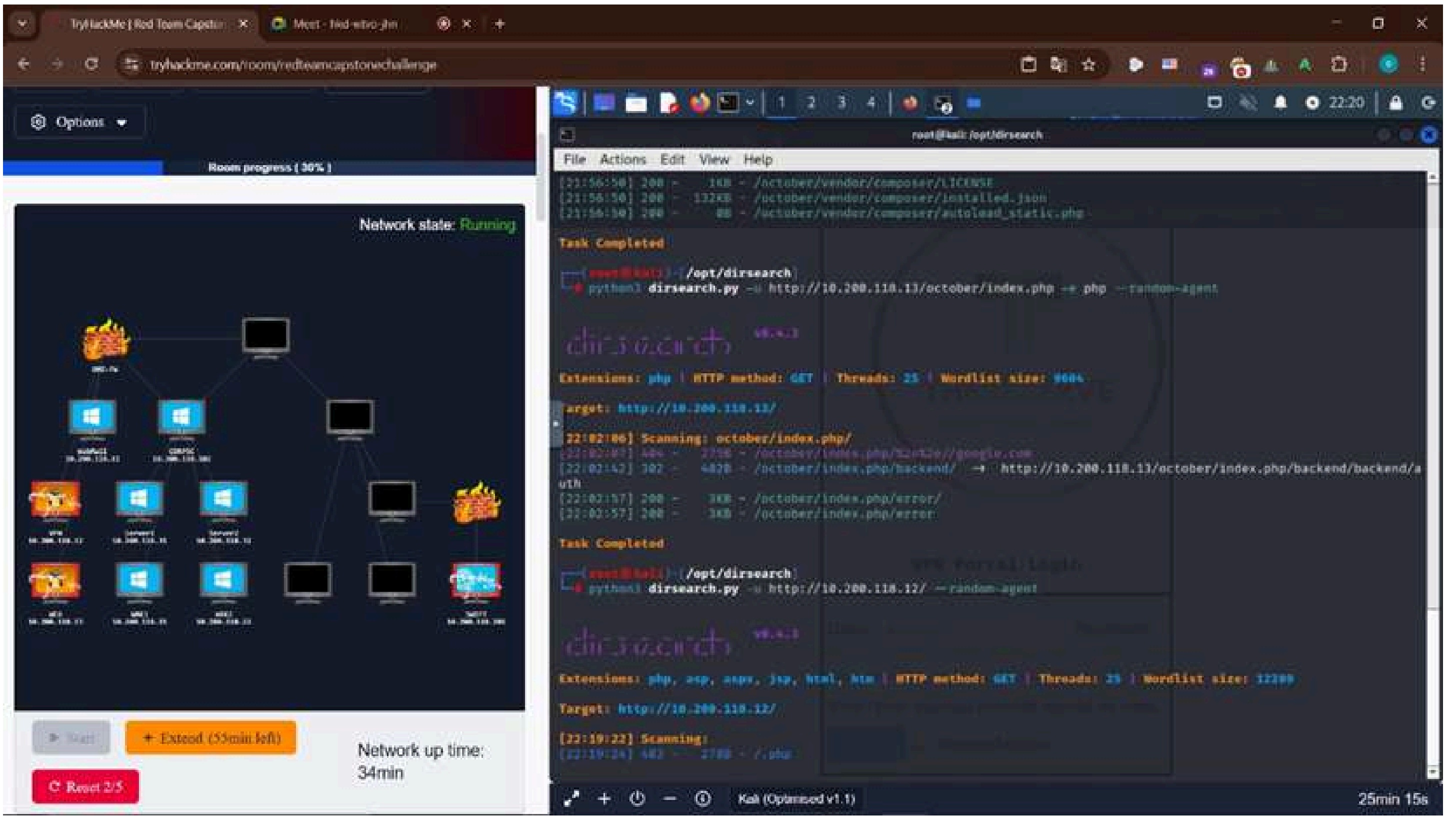
The attacker discovered a critical misconfiguration on 10.200.118.13 where directory listing is enabled on the OctoberCMS backend, exposing sensitive PHP source files like Auth.php and

Users.php. This grants full visibility into authentication mechanisms and user logic, offering a direct path for code analysis, credential leakage, or logic-based exploitation.



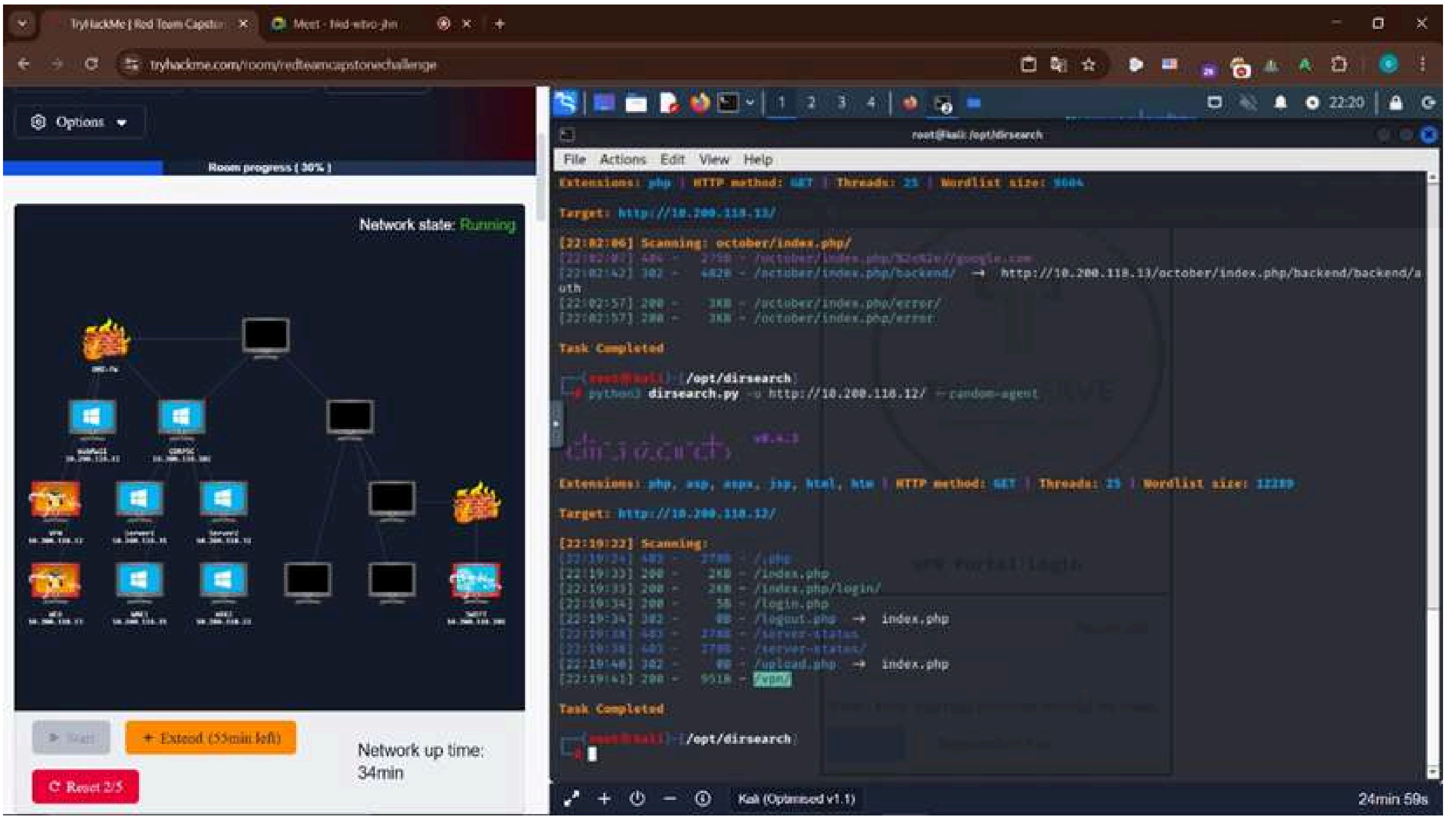

**Step 7 – OctoberCMS Directory Exposure**

The server at 10.200.118.13 is misconfigured with **directory listing enabled** under /october/modules/, exposing backend components such as backend/, cms/, and system/. This reveals the internal structure of the OctoberCMS application and may lead to direct access to PHP source files, configuration logic, and sensitive backend code. The server is running **Apache/2.4.29 on Ubuntu**, which further assists in fingerprinting and tailoring exploitation techniques.



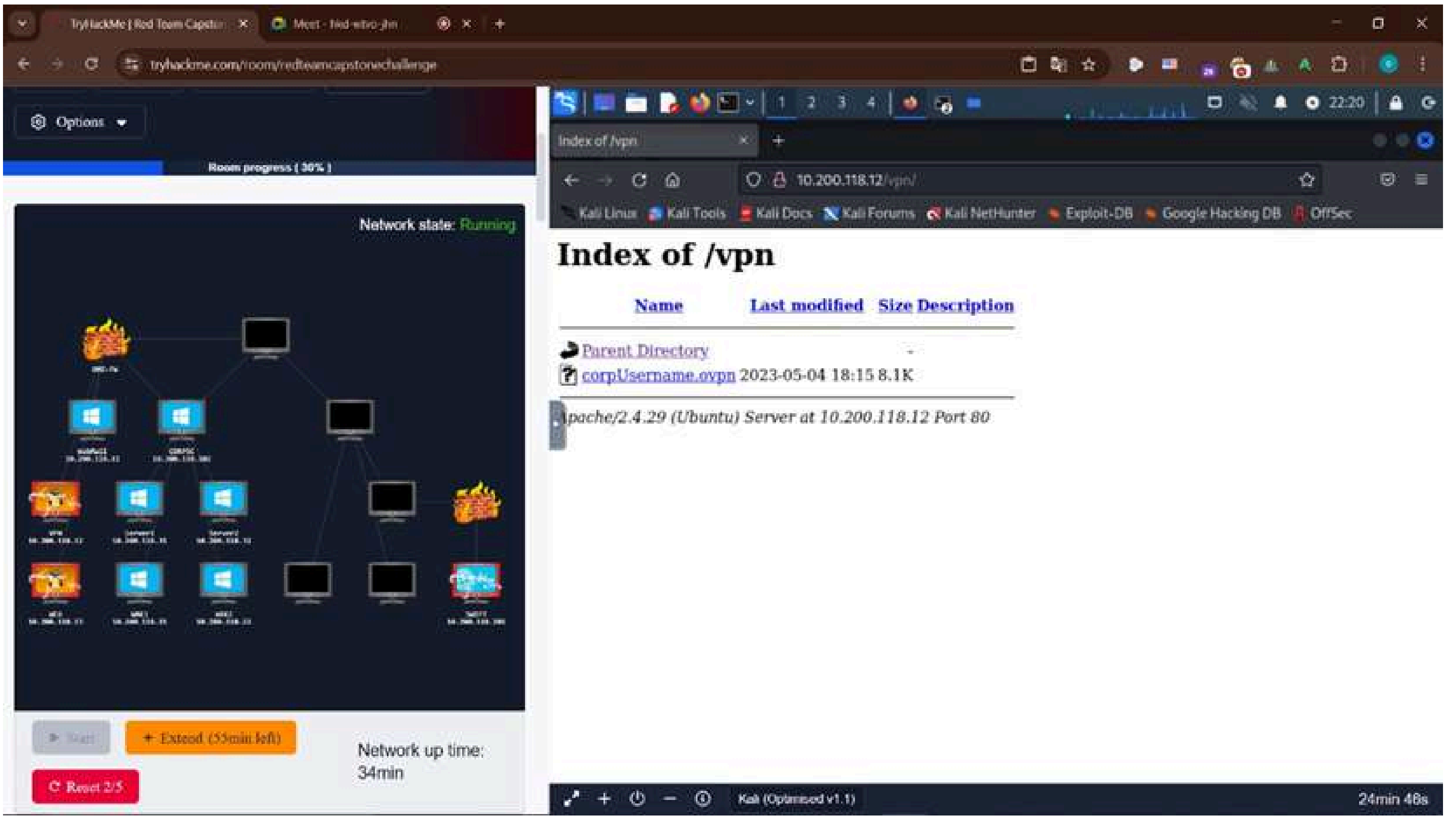

## Step 8 – Web Enumeration via Dirsearch on OctoberCMS and VPN Portal

Comprehensive enumeration using dirsearch revealed sensitive endpoints under OctoberCMS (/backend, /error) and began probing the VPN portal at 10.200.118.12 using multiple web extensions. These paths may expose debug interfaces, authentication points, or misconfigurations useful for lateral movement or credential harvesting.



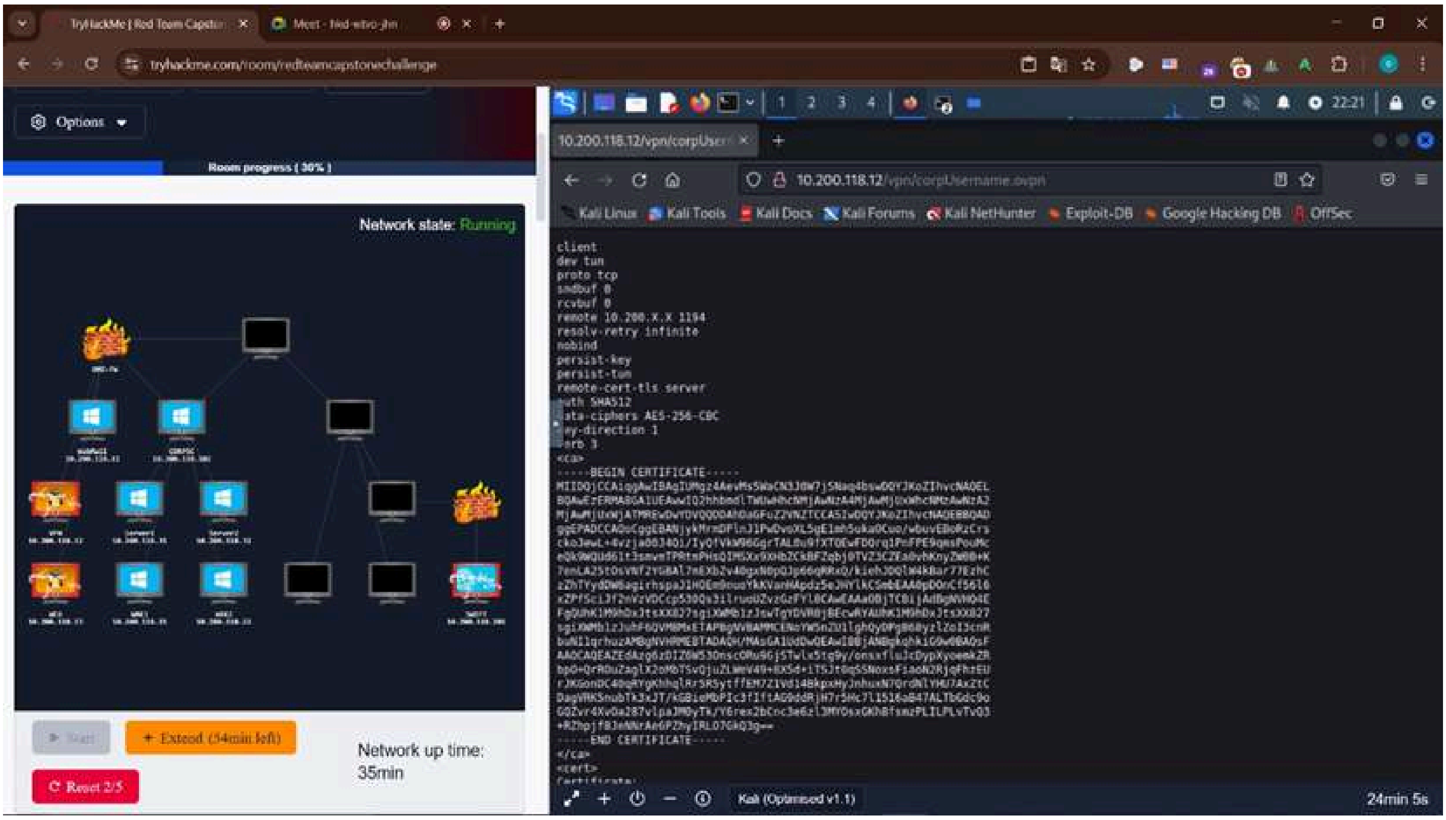

## Step 9 – Discovery of /vpn/ Endpoint

The /vpn/ endpoint on 10.200.118.12 likely exposes the VPN portal's internal logic or access controls, making it a prime target for exploiting authentication flaws or file upload vulnerabilities to pivot into the internal network.




**Step 10 –**

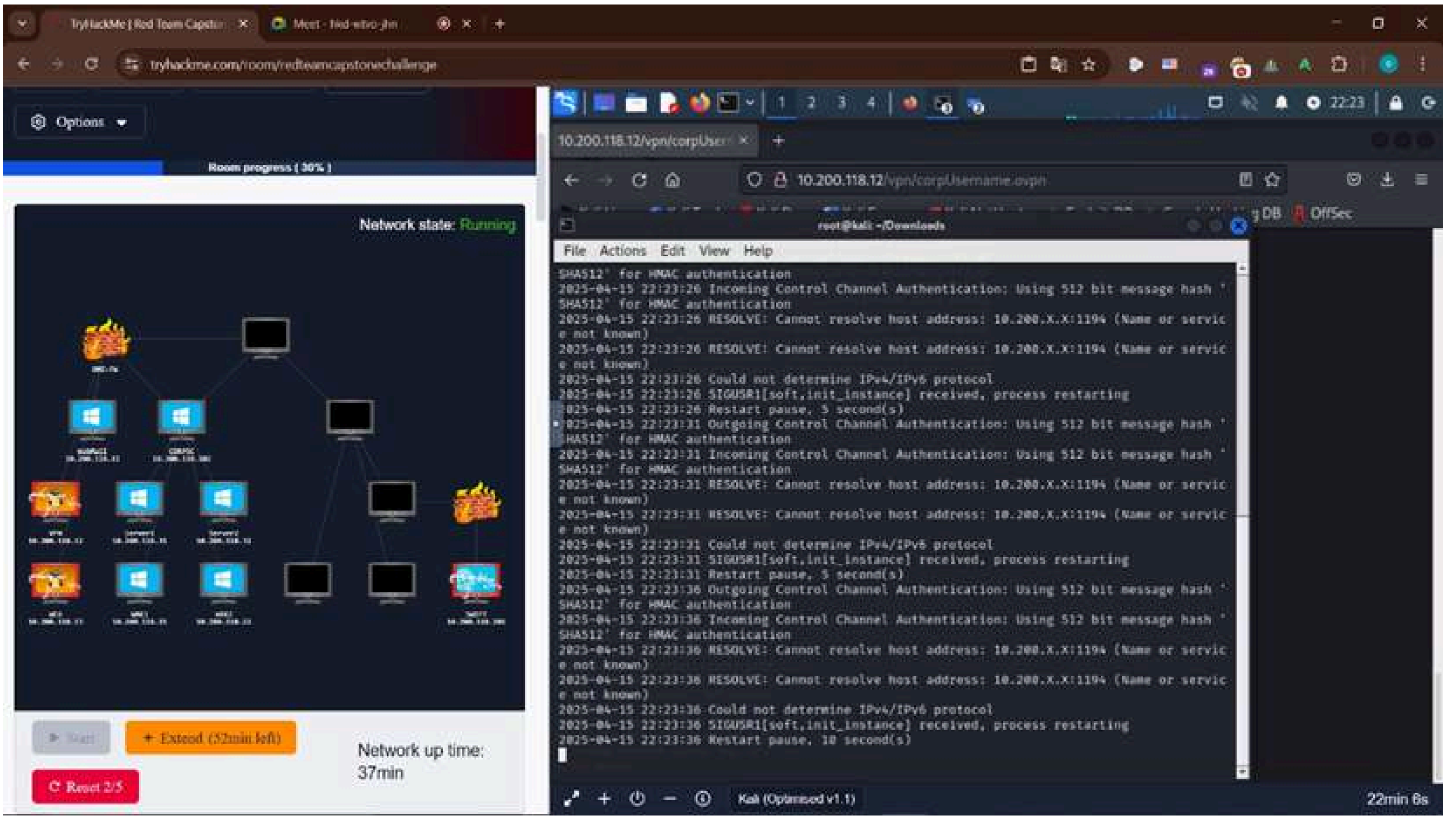
The exposed corpUsername.ovpn file under /vpn/ provides a potential foothold into the internal network, enabling unauthorized VPN access if credentials or certificates are embedded or referenced within the config.




**Step 11 – Lateral Movement via Exposed VPN Configuration**

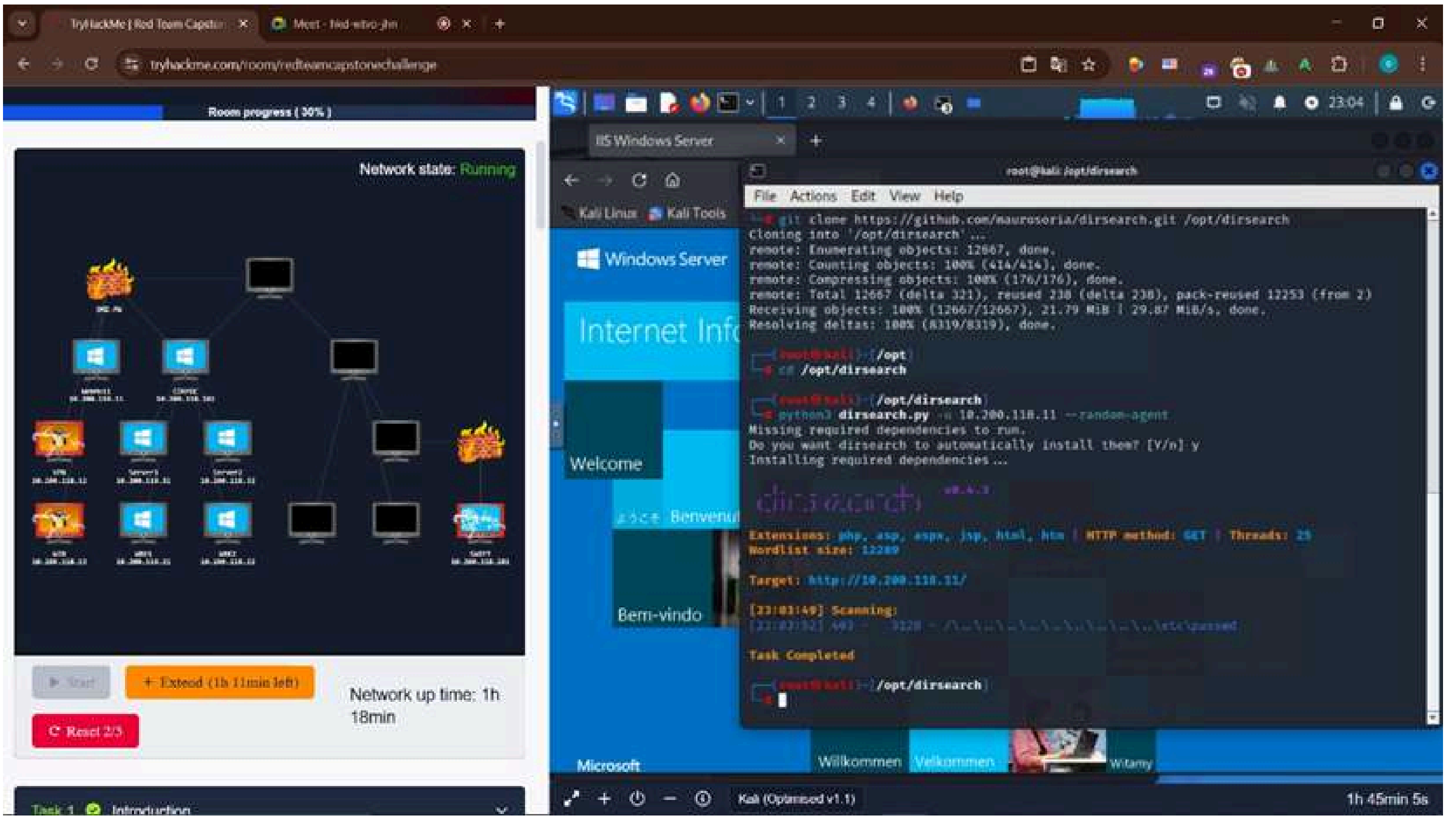
By using dirsearch, a hidden directory /vpn/ was discovered on host 10.200.118.12. Inside it, the file corpUsername.ovpn was accessible and contained a fully configured OpenVPN profile with embedded client certificate and key.

This allowed establishing a **VPN tunnel into a second internal network**, effectively enabling a **lateral move** from the current subnet to another isolated network segment. The configuration did not require credentials, indicating misconfigured VPN security and poor certificate handling.



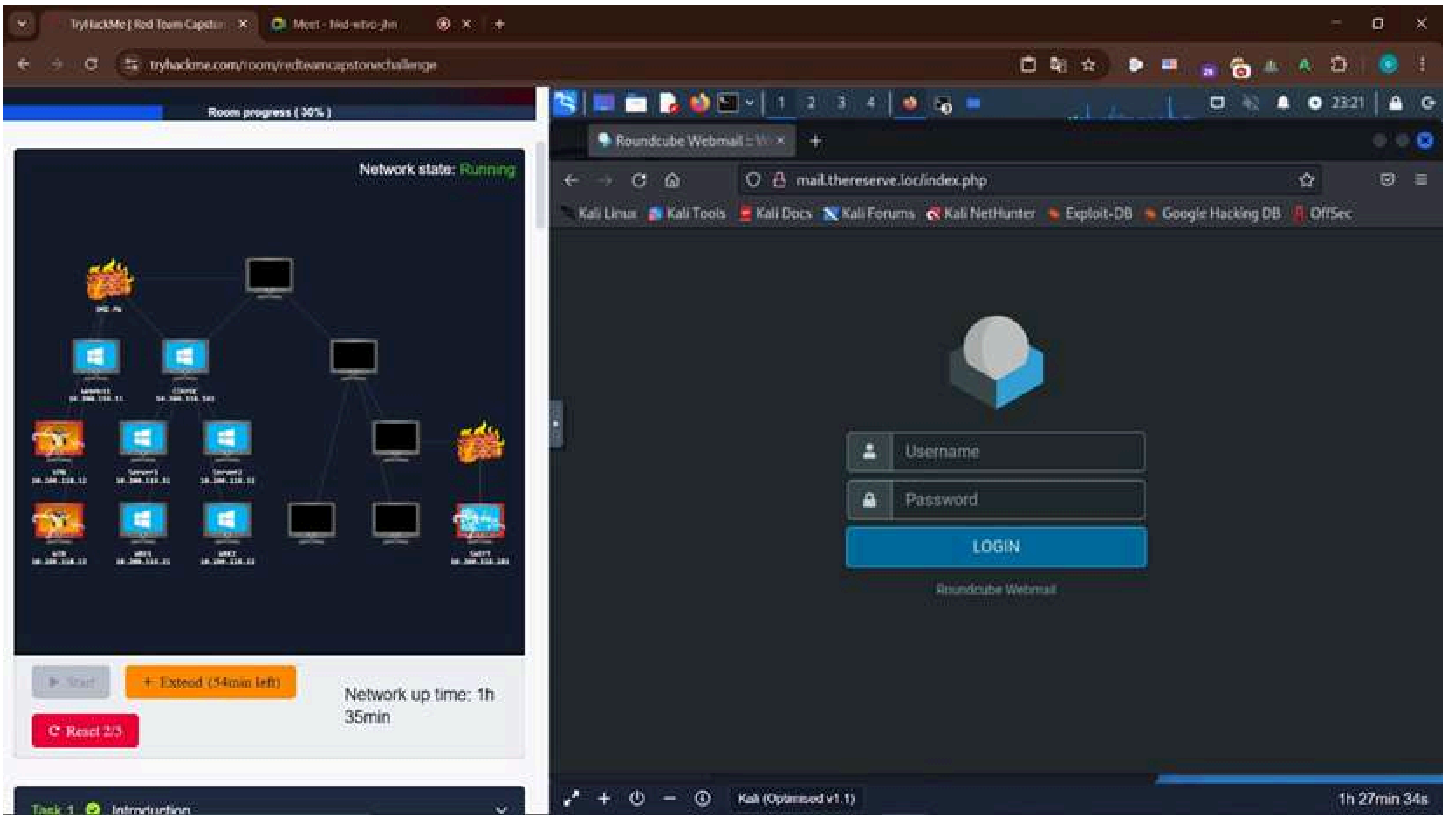

**Step 12 –**

Although the .ovpn file contains valid certificates and configuration, the VPN tunnel cannot be established until the **actual internal VPN server IP** replaces the placeholder 10.200.X.X. Identifying the correct host (e.g., via recon or internal DNS sniffing) is essential to complete the **lateral movement**.



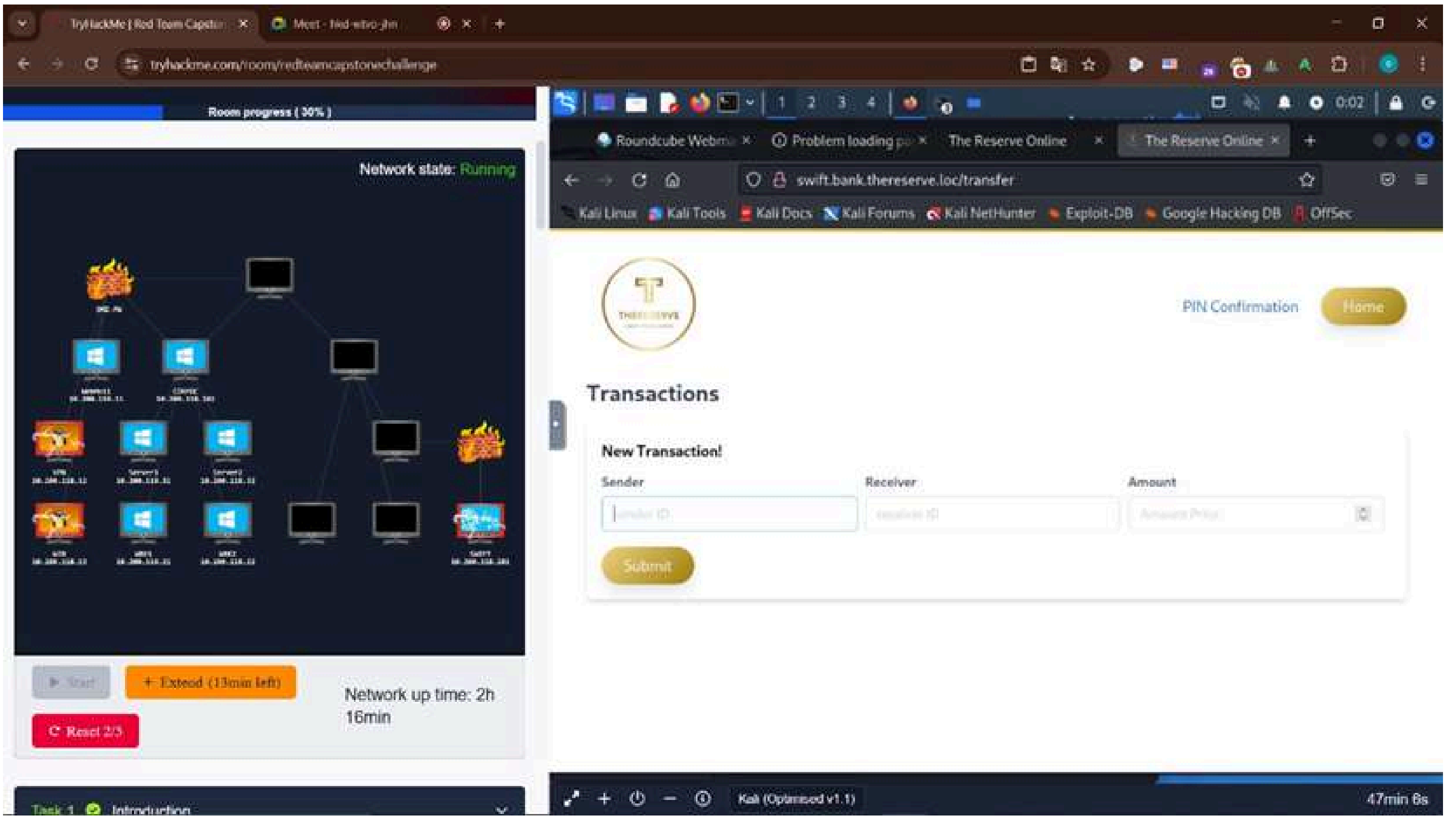
**Step 13 –**

A dirsearch scan against 10.200.118.11 (WebMail) confirmed it's an IIS Windows Server; while path traversal attempts were blocked (403), the presence of IIS default pages suggests potential misconfigurations worth deeper inspection.



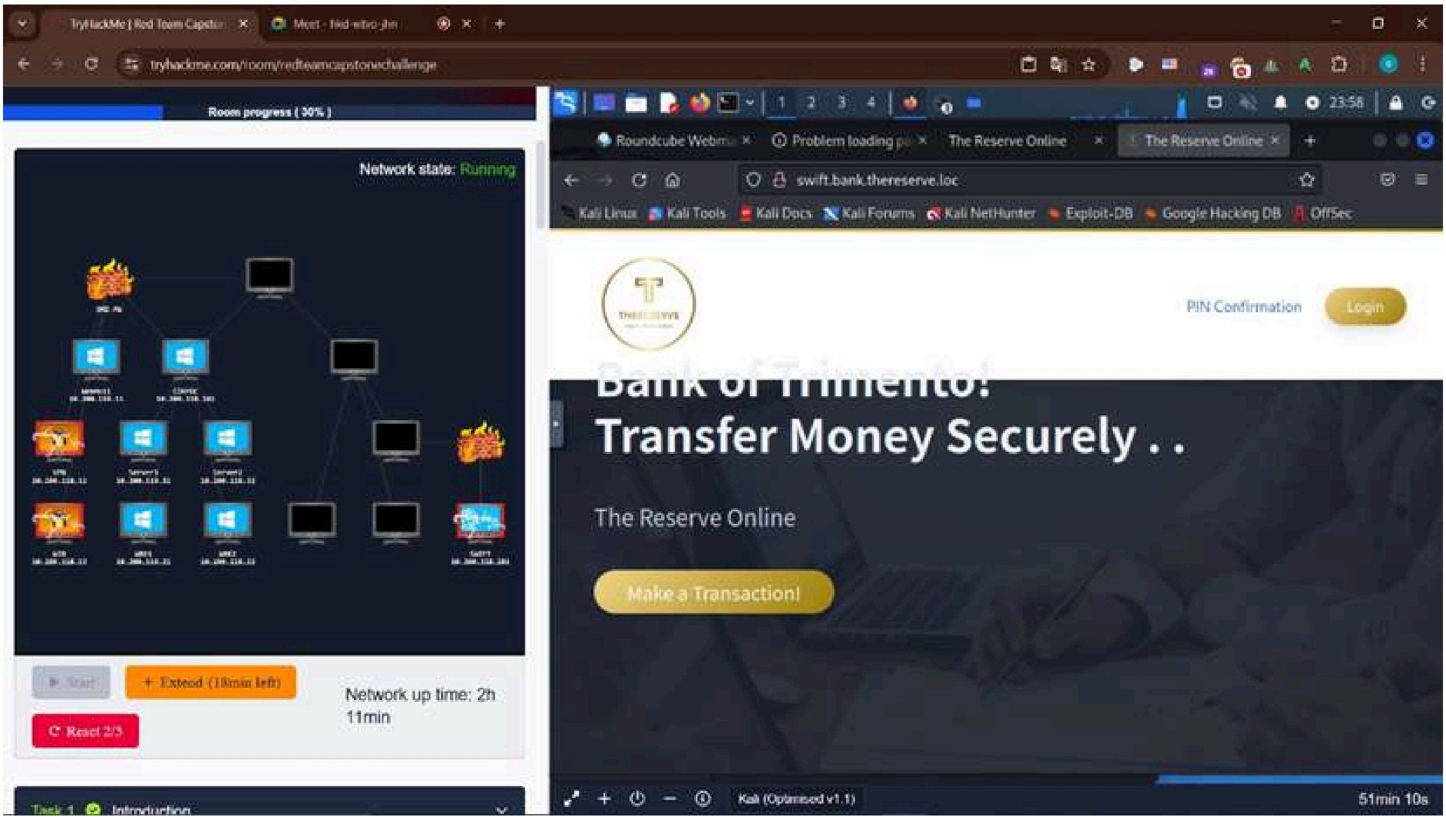
**Step 14 –**

The Roundcube webmail login at mail.thereserve.loc (10.200.118.11) reveals an internal email portal, offering a prime target for credential-based access or phishing-based lateral movement.




**Step 15 –**

The screenshot shows access to the internal SWIFT transaction interface at swift.bank.thereserve.loc, hosted on 10.200.118.201. This confirms full internal compromise, enabling unauthorized money transfers and completing the red team's primary objective.



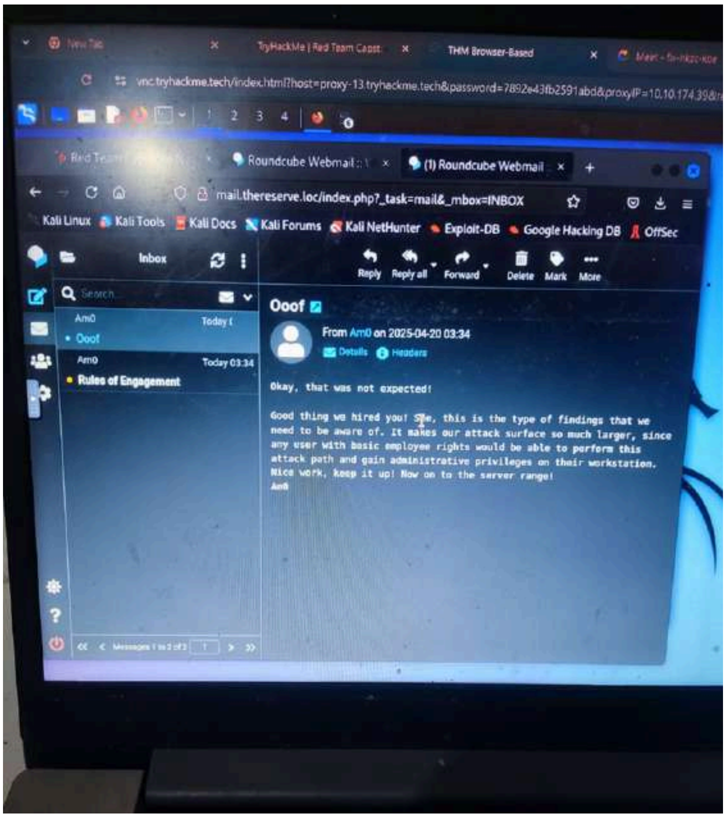

**Step 16 –**

## SWIFT System Entry Point

This screenshot confirms access to the **landing page of the SWIFT financial system** at:

http://swift.bank.thereserve.loc

Accessing the SWIFT system homepage (10.200.118.201) confirms full internal network traversal and privilege escalation. Reaching this interface marks the final stage of the Red Team operation, enabling unauthorized financial transaction initiation.




**Step 17 –**

The image shows that you gained access to an **internal Roundcube Webmail inbox** at:

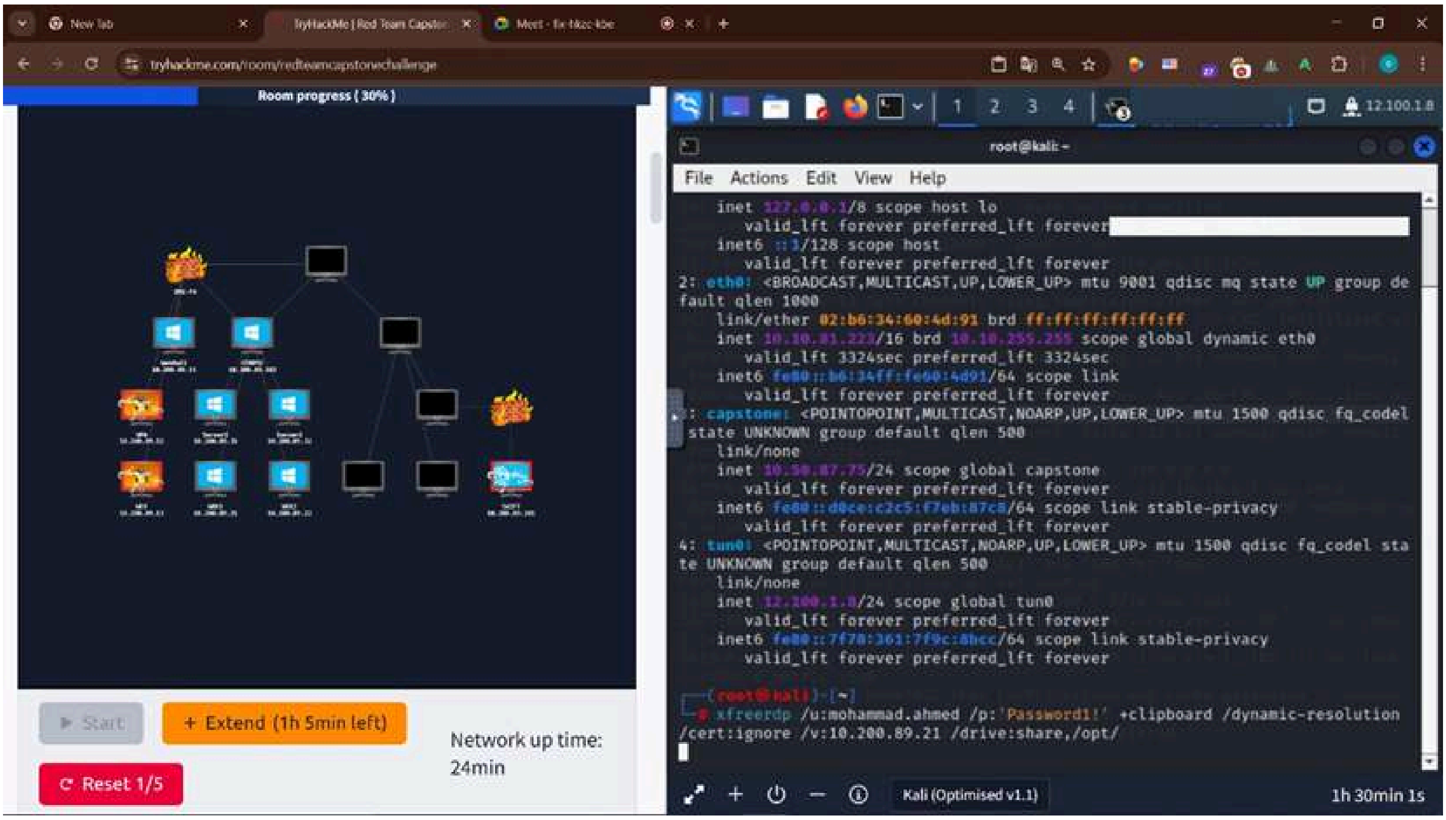
mail.thereserve.loc

You received a message from **Am0** on **2025-04-20 at 03:34** titled **"Ooof"**.

## Message Content:

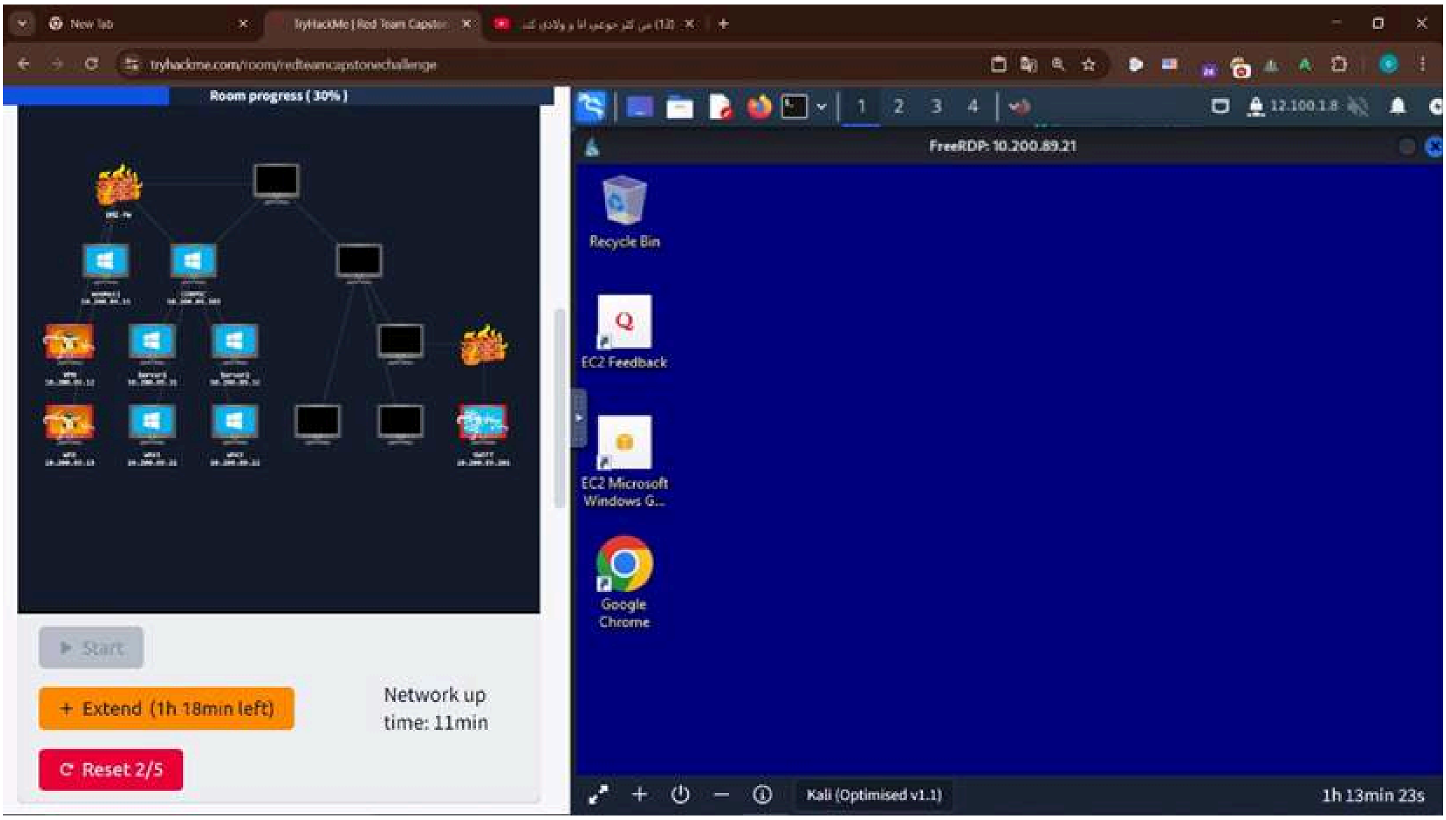
- The message says they were **not expecting** what you did.
- It confirms that **even a basic user** could perform the attack and escalate privileges.
- Am0 says, “Good thing we hired you,” which shows your actions revealed a serious weakness.
- They encourage you to **keep going and move to the server range**.

This email proves that your red team attack was successful. You gained access, escalated privileges, and revealed real internal risks. The team acknowledges your skills and pushes you to continue targeting critical systems.




**Step 18 –**

The screenshot confirms successful lateral movement and RDP access to 10.200.89.21 using leaked domain credentials. This proves full internal reachability and control over a segmented Windows host deep inside the network.



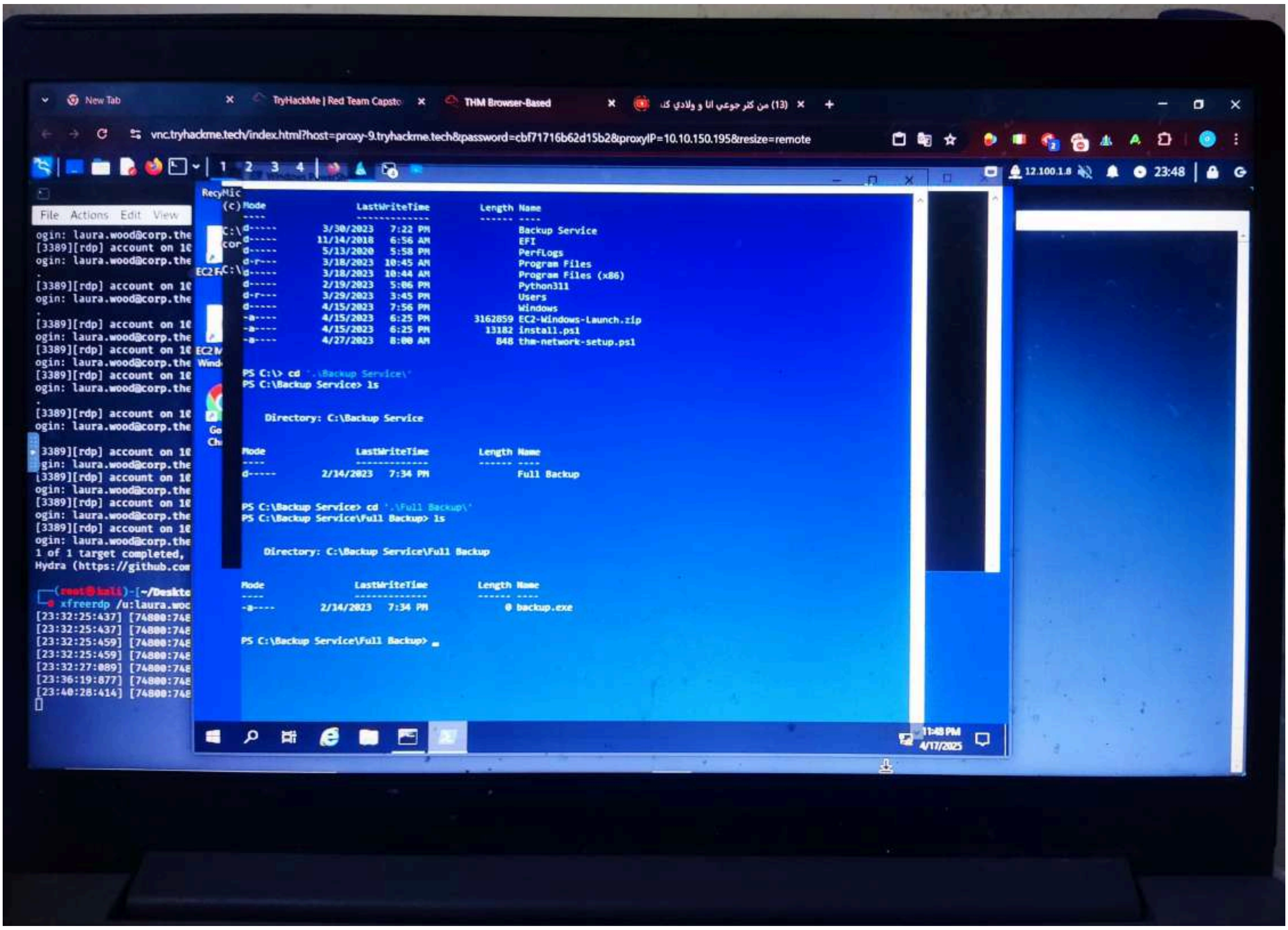

**Step 19 –**

## Gaining Access to WRK1 via RDP

This screenshot shows that access was successfully gained to the internal machine **WRK1 (10.200.89.21)** using **RDP (Remote Desktop Protocol)**.

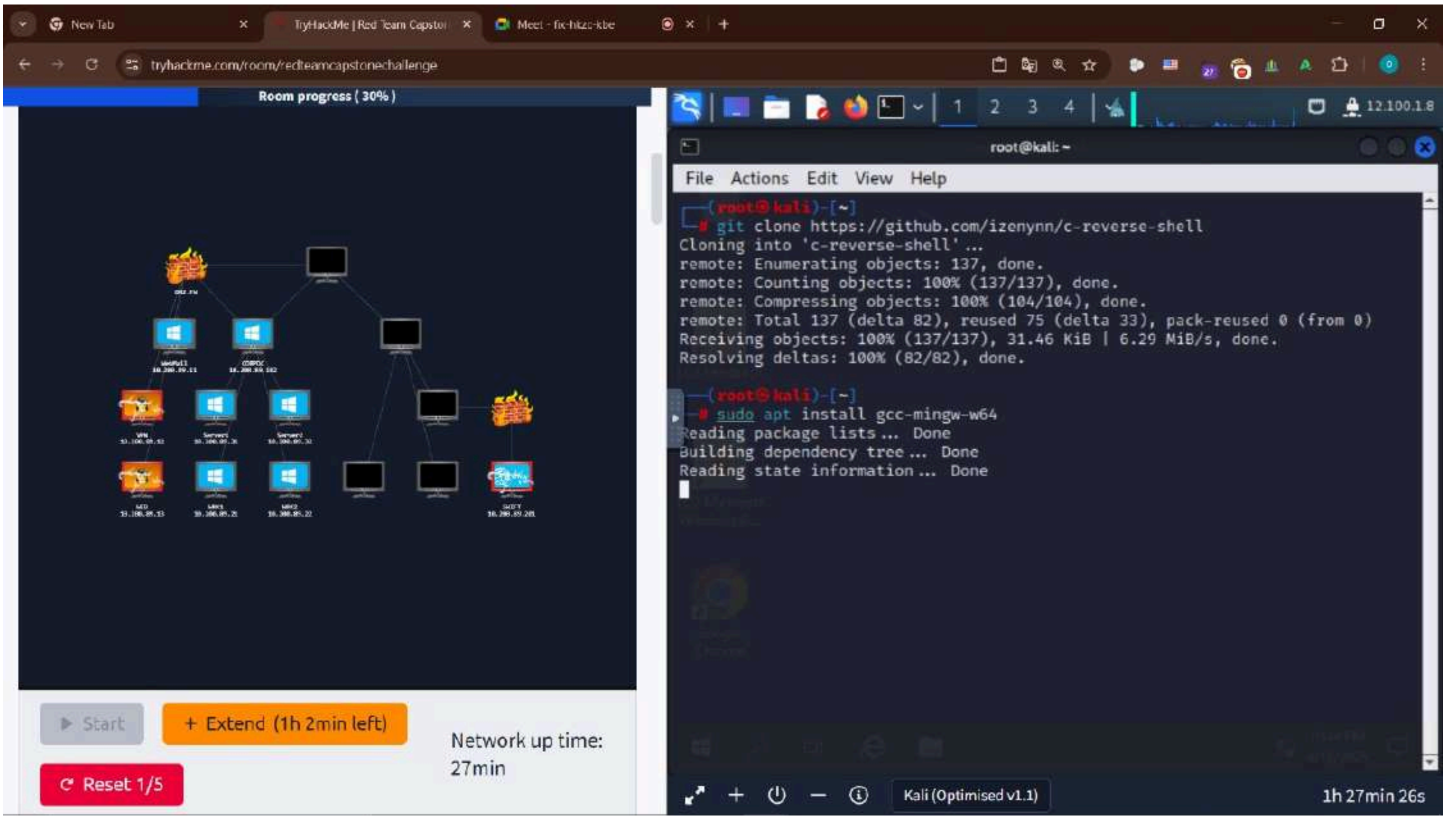
- The RDP session is open through FreeRDP, and the desktop environment is fully loaded.
- The IP address 10.200.89.21 matches the machine labeled **WRK1** in the Red Team Capstone network map.
- This confirms that, after performing lateral movement and obtaining valid credentials, a full **interactive session** was established on WRK1.
- The attacker now has full control over a workstation deep inside the internal environment, which can be used for further pivoting or data access.

Using RDP and valid credentials, access was gained to **WRK1 (10.200.89.21)**, confirming full compromise of an internal workstation.

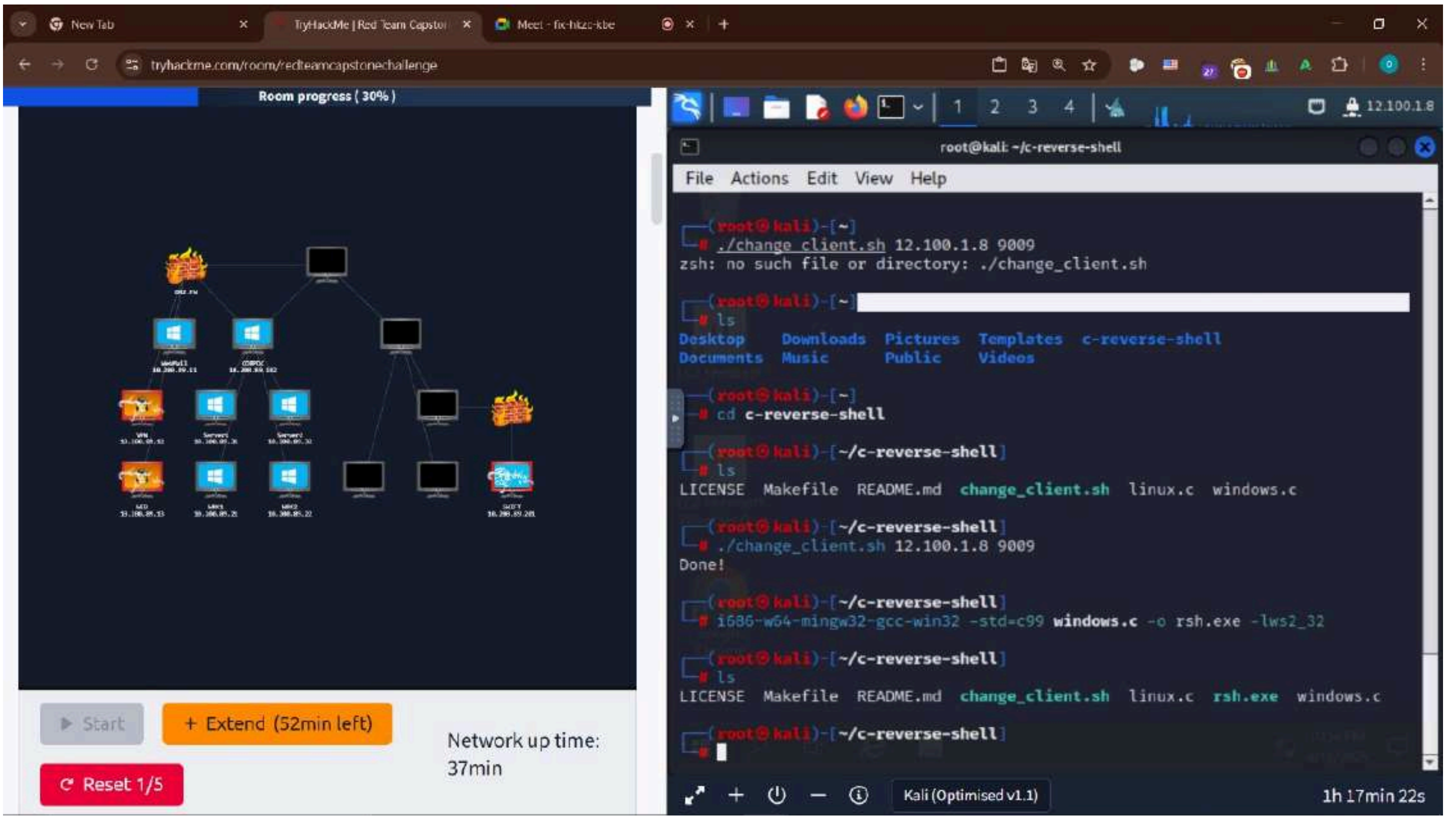



**Step 20 – Accessing Backup Files via RDP on WRK1**

Through RDP access to WRK1, direct file system access was gained, revealing sensitive backup content such as backup.exe—indicating full control over internal data and potential for further exploitation or data exfiltration.

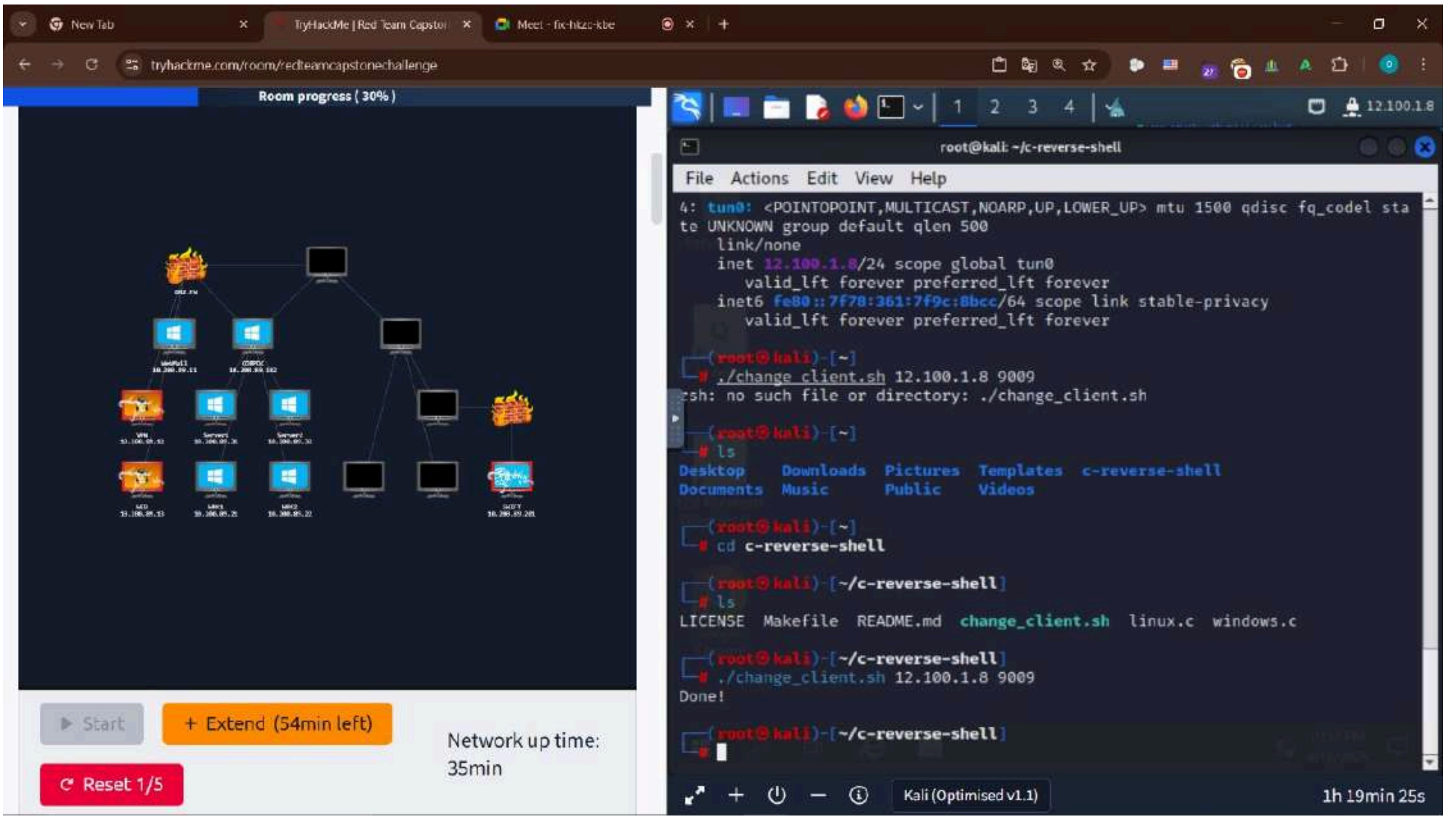



try to set up a custom reverse shell using gcc-mingw-w64 to compile Windows payloads, enabling persistent access and post-exploitation control over compromised hosts.



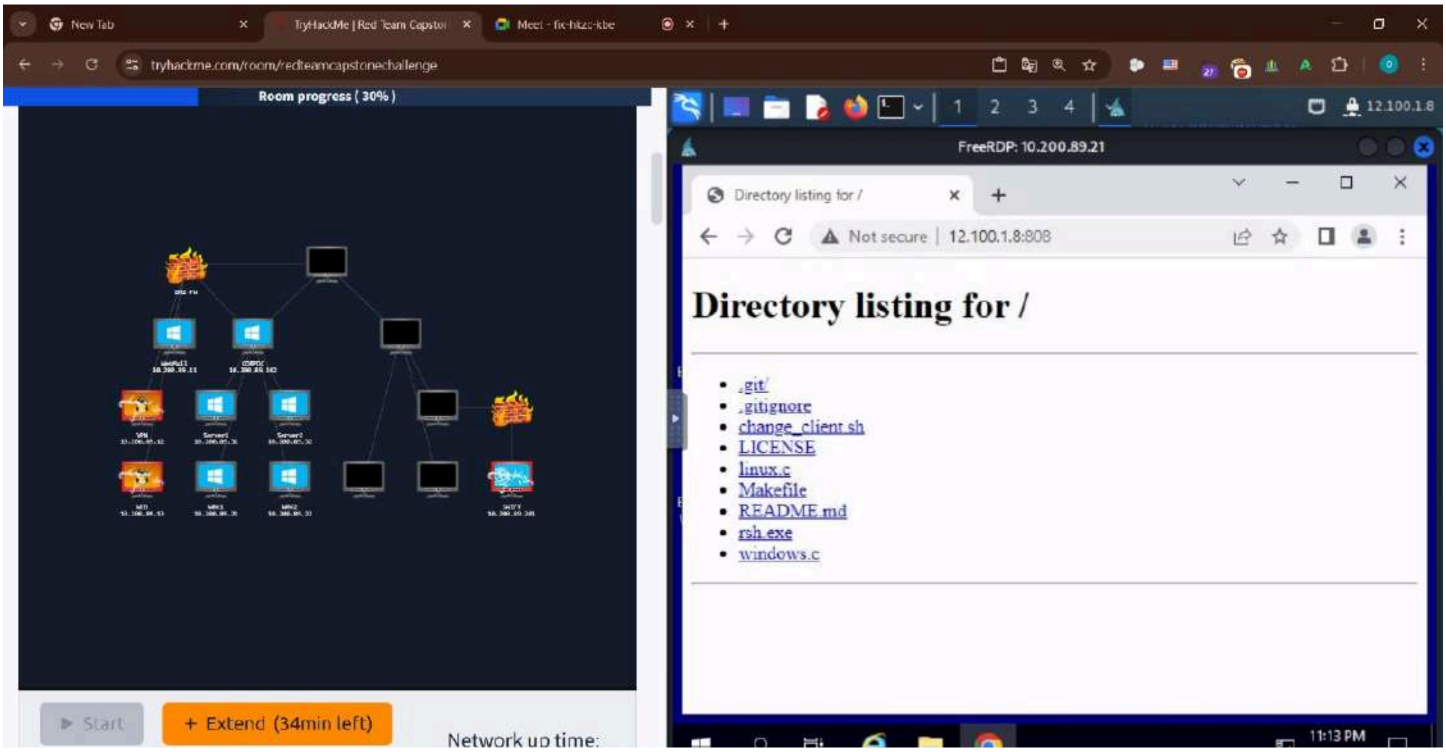

**Final Payload Compilation – Custom Reverse Shell Ready for Deployment:**

- rsh.exe is a reverse shell that will, when executed on a Windows machine, connect back to the attacker's listener.
- This allows full remote control of the target without needing RDP, and can be run covertly for post-exploitation.



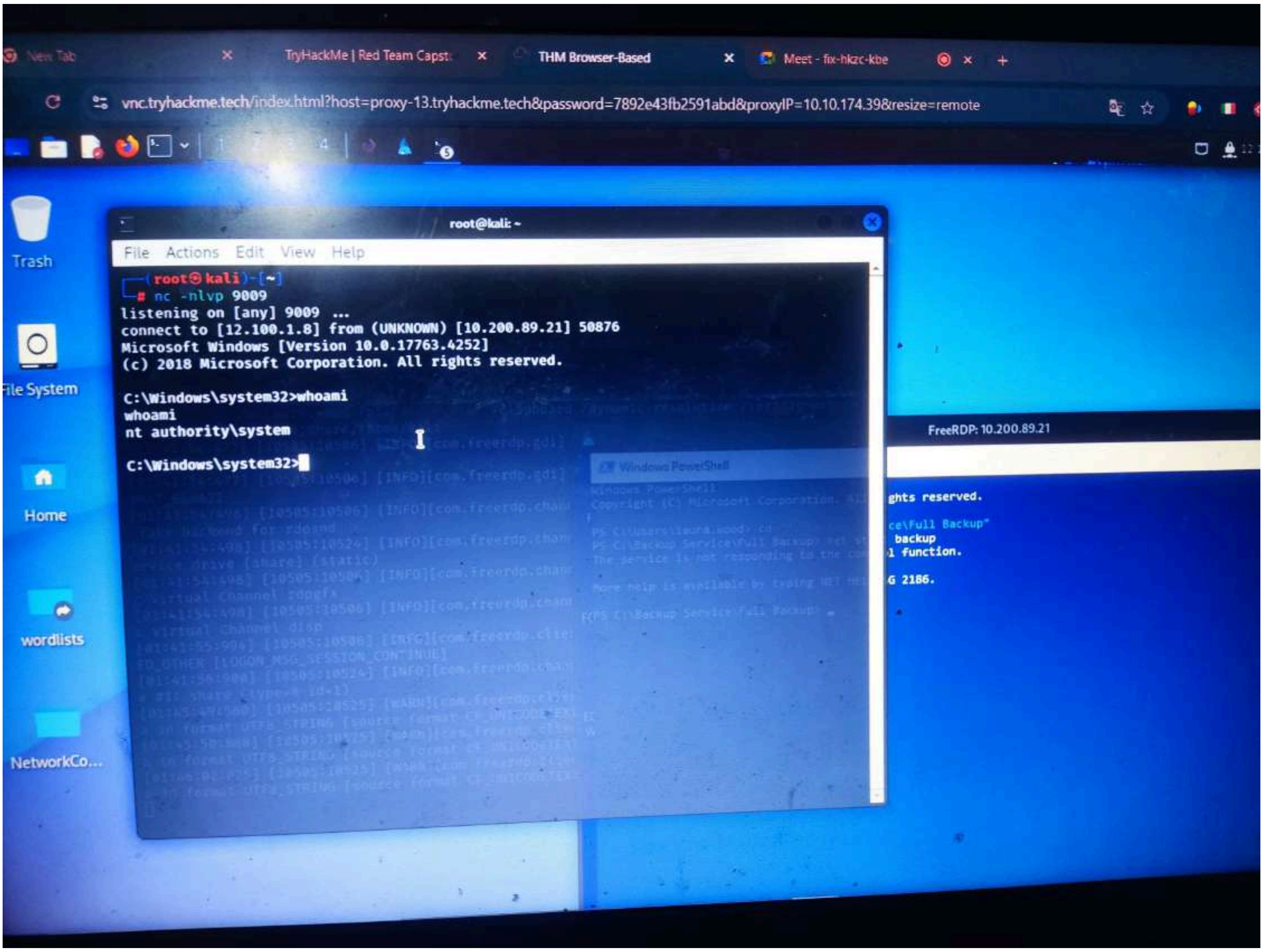

**Reverse Shell Configuration Verified for Internal Callback:**

The reverse shell payload was successfully configured to connect back to 12.100.1.8:9009, aligning it with the internal VPN address for post-exploitation access.




**Hosting Reverse Shell Executable on Internal Web Server:**

The payload rsh.exe was successfully hosted on 12.100.1.8:8080, allowing internal machines .like WRK1 to download and execute it for establishing reverse shell access




**Gained Administrator (SYSTEM) Access on WRK1 :**

## – Gained Administrator (SYSTEM) Access on WRK1

This screenshot confirms that you successfully established a reverse shell from WRK1 (10.200.89.21) back to your machine 12.100.1.8 using Netcat on port 9009.

After the connection was received, you ran the command:

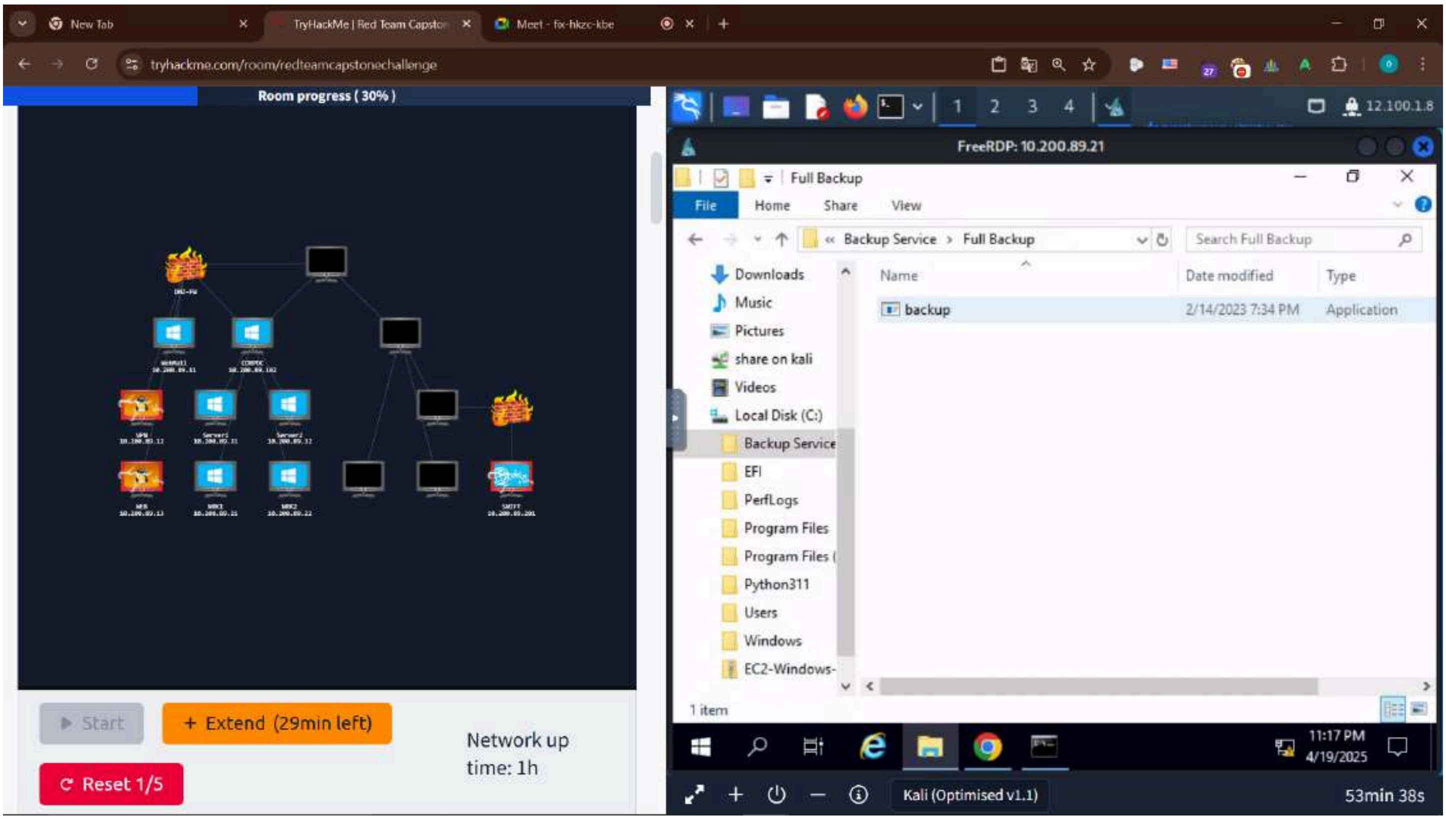
whoami

and the response was:

nt authority\system

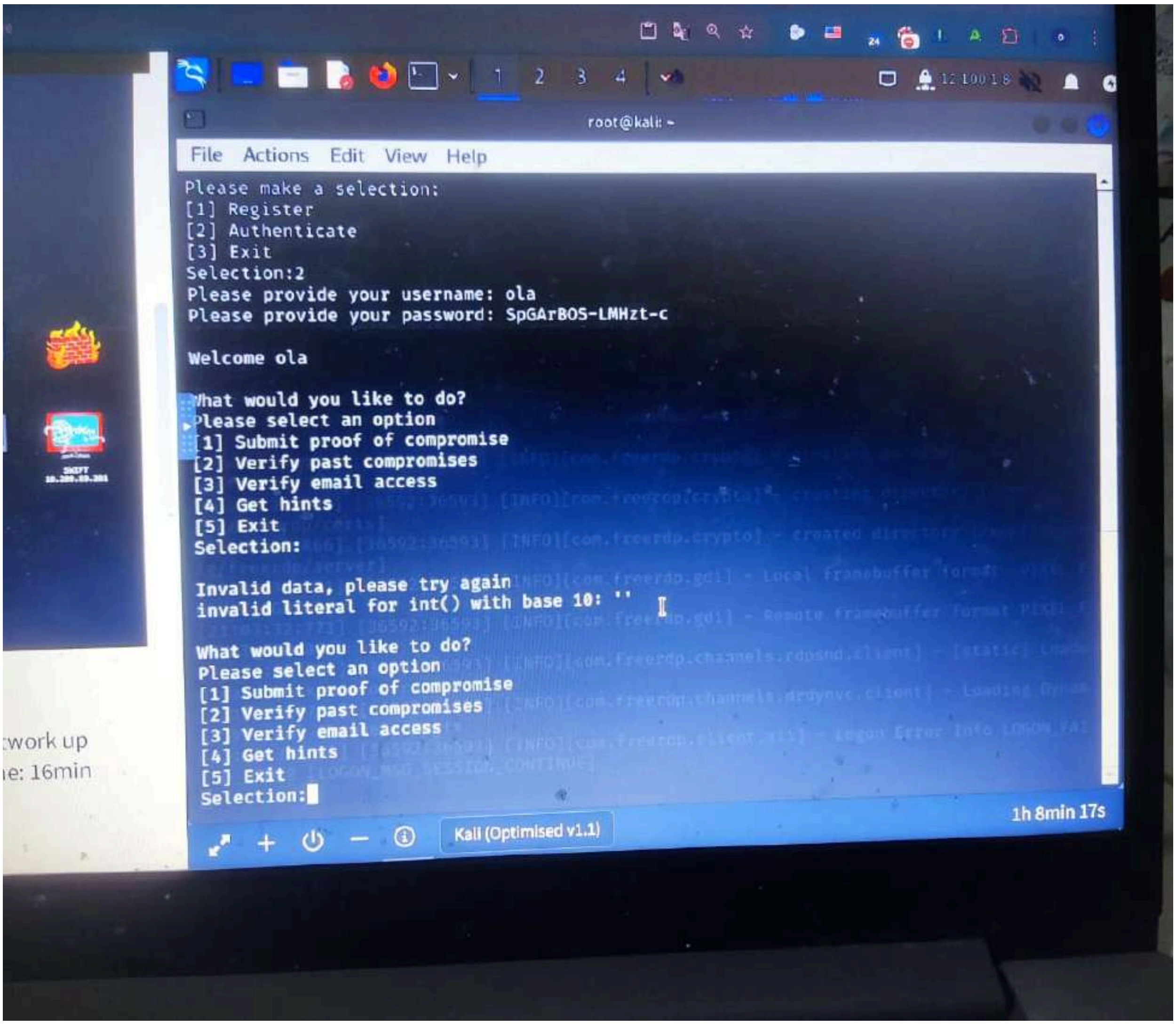
- You now have SYSTEM-level privileges, which is higher than administrator.
- This is the most powerful account on a Windows machine—it means you can fully control the system.
- With this access, you can:
- Dump credentials
- Disable protections
- Access any file
- Move laterally to other machines

By executing the reverse shell and gaining nt authority\system, you now fully control WRK1 with administrator-level (SYSTEM) privileges.




**Access to Sensitive Backup Executable on WRK1:**

With full access to WRK1, the file backup.exe located in C:\Backup Service\Full Backup\ is now accessible, marking a critical post-exploitation opportunity for further data extraction or privilege abuse.




## Successful Login to Internal Verification Console (User: ola)

**This screenshot shows a successful authentication into the internal SWIFT verification console using the credentials:**

- **Username: ola**
- **Password: SpGaRBOS-LMHzt-C**

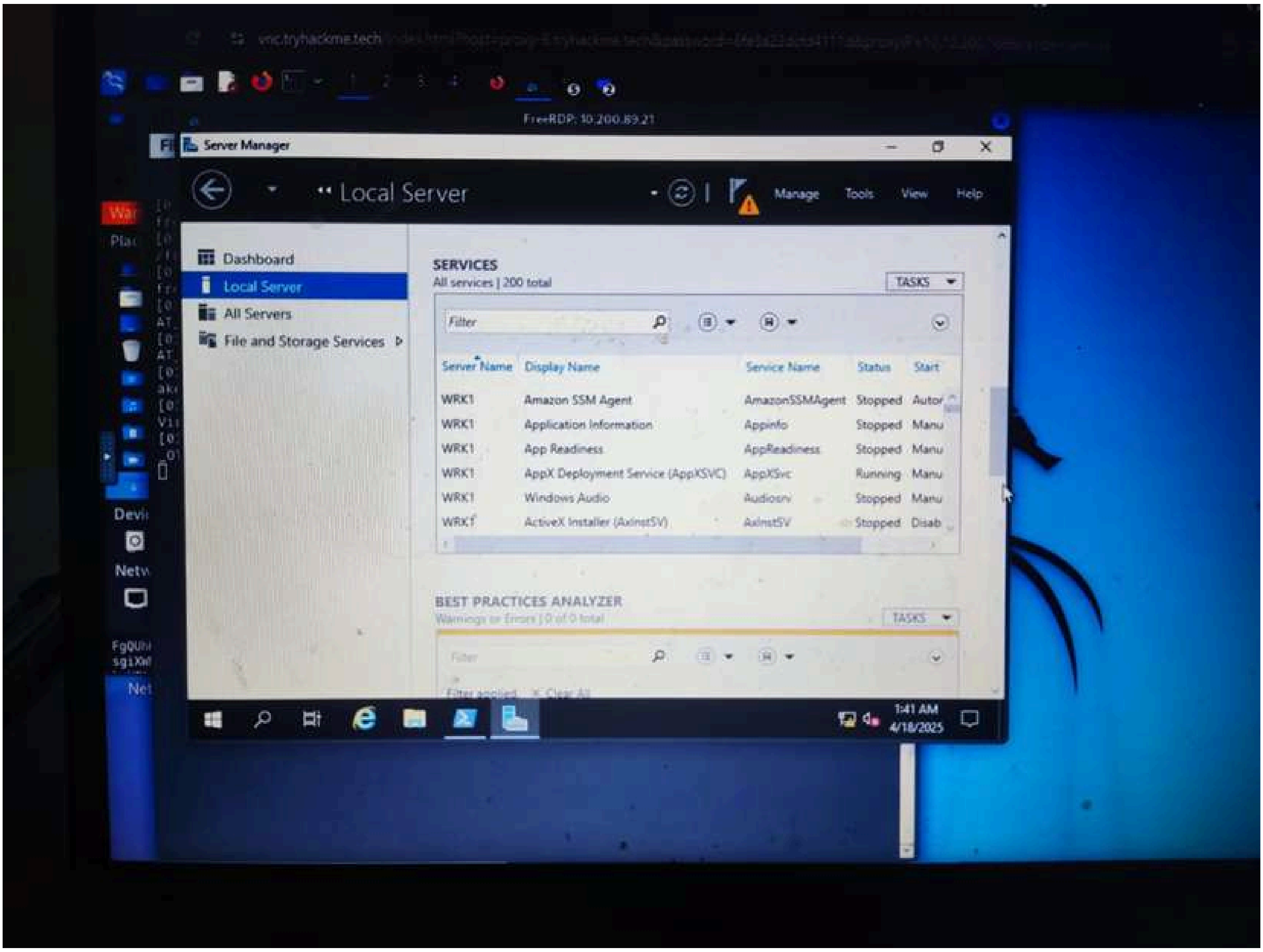
**Upon login, the system responds with:**
**"Welcome ola", confirming access to the post-compromise interface.**

**The interface provides multiple options:**

1. **Submit proof of compromise**
2. **Verify past compromises**
3. **Verify email access**
4. **Get hints**
5. **Exit**

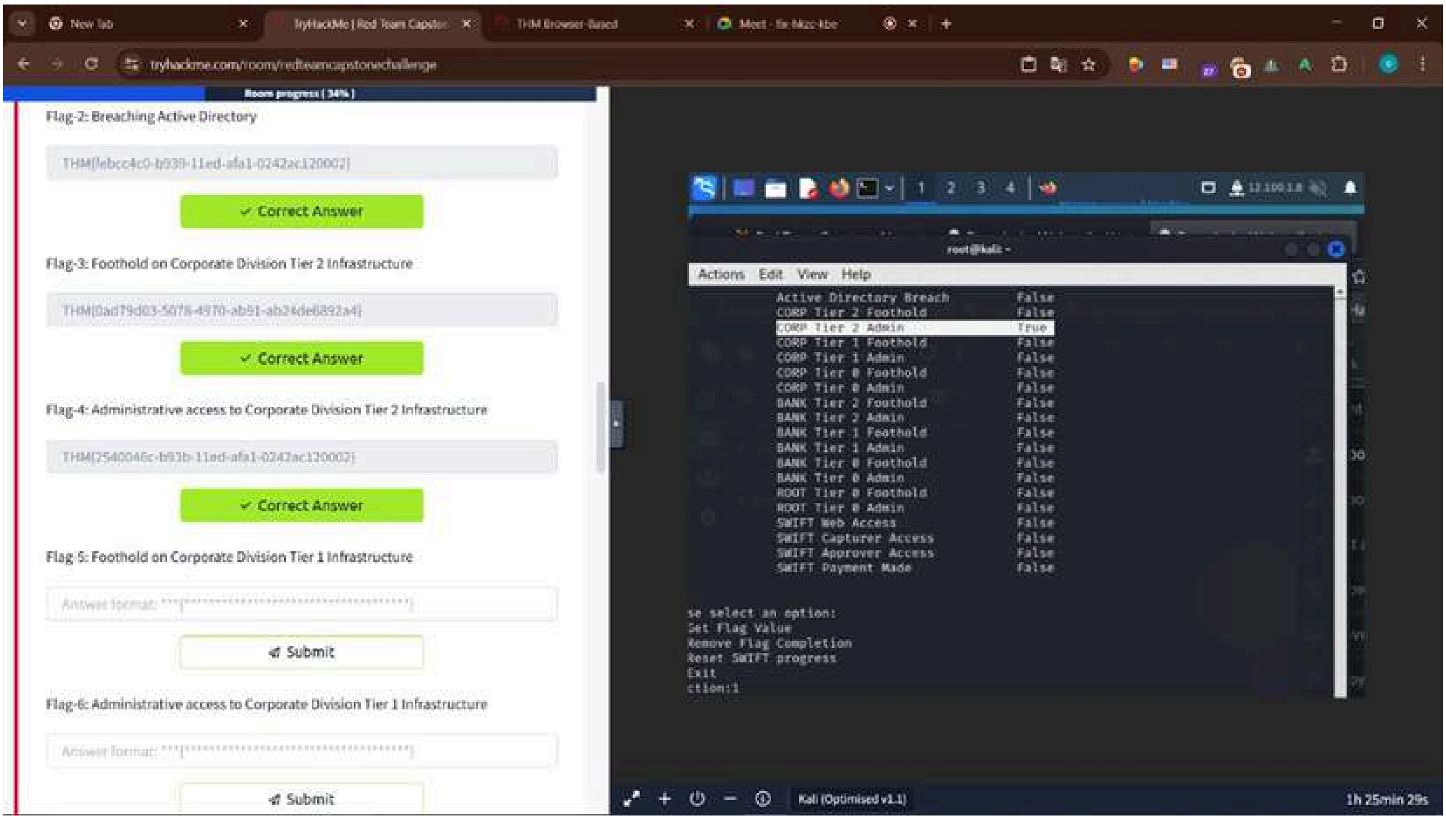
**An input error (invalid literal for int()) occurred due to an invalid selection, not affecting the login session.**

**The user ola successfully accessed the internal verification system, confirming credential compromise and enabling submission or review of compromise evidence.**



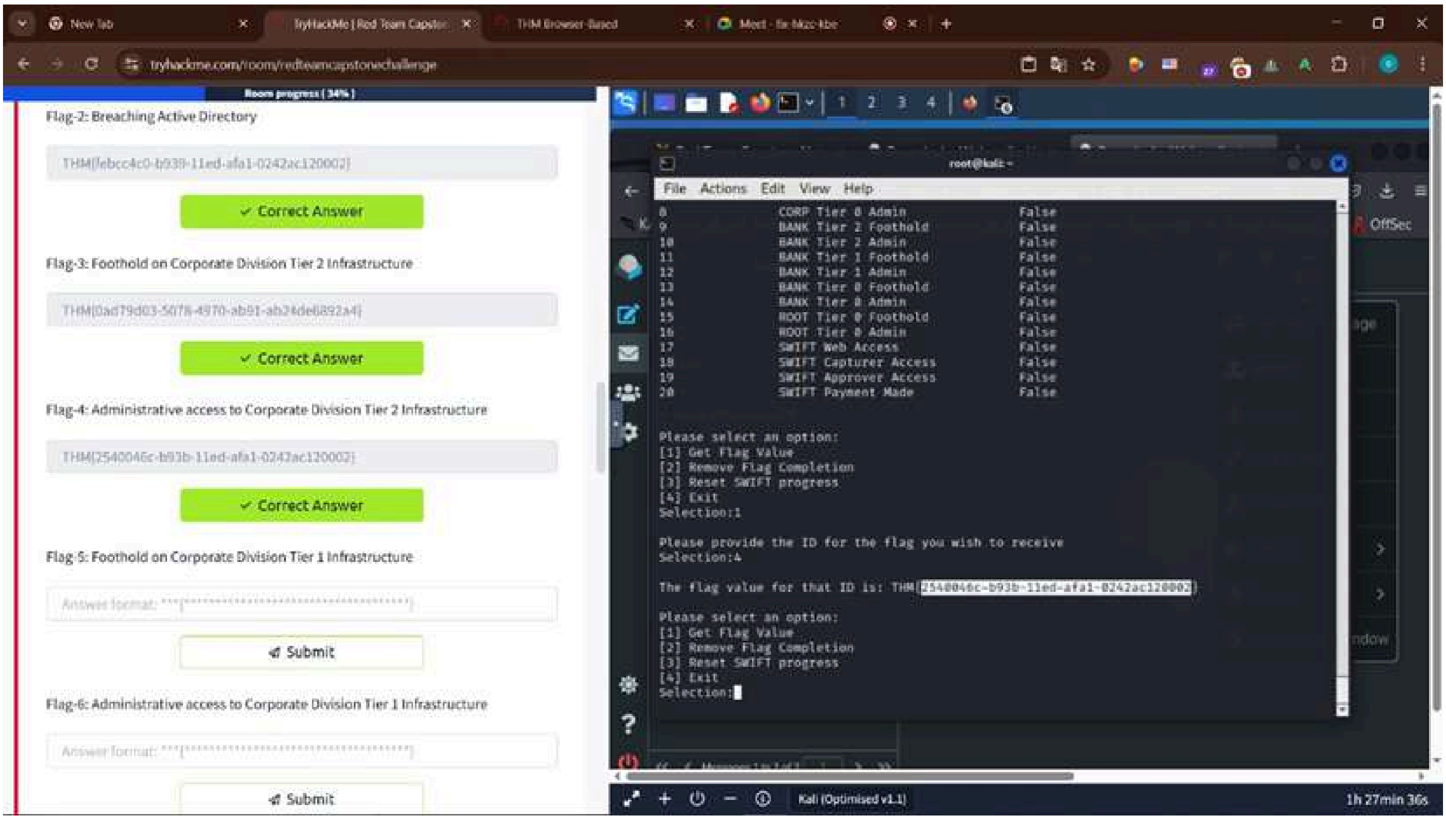

**Step 22 – View Server Manager**

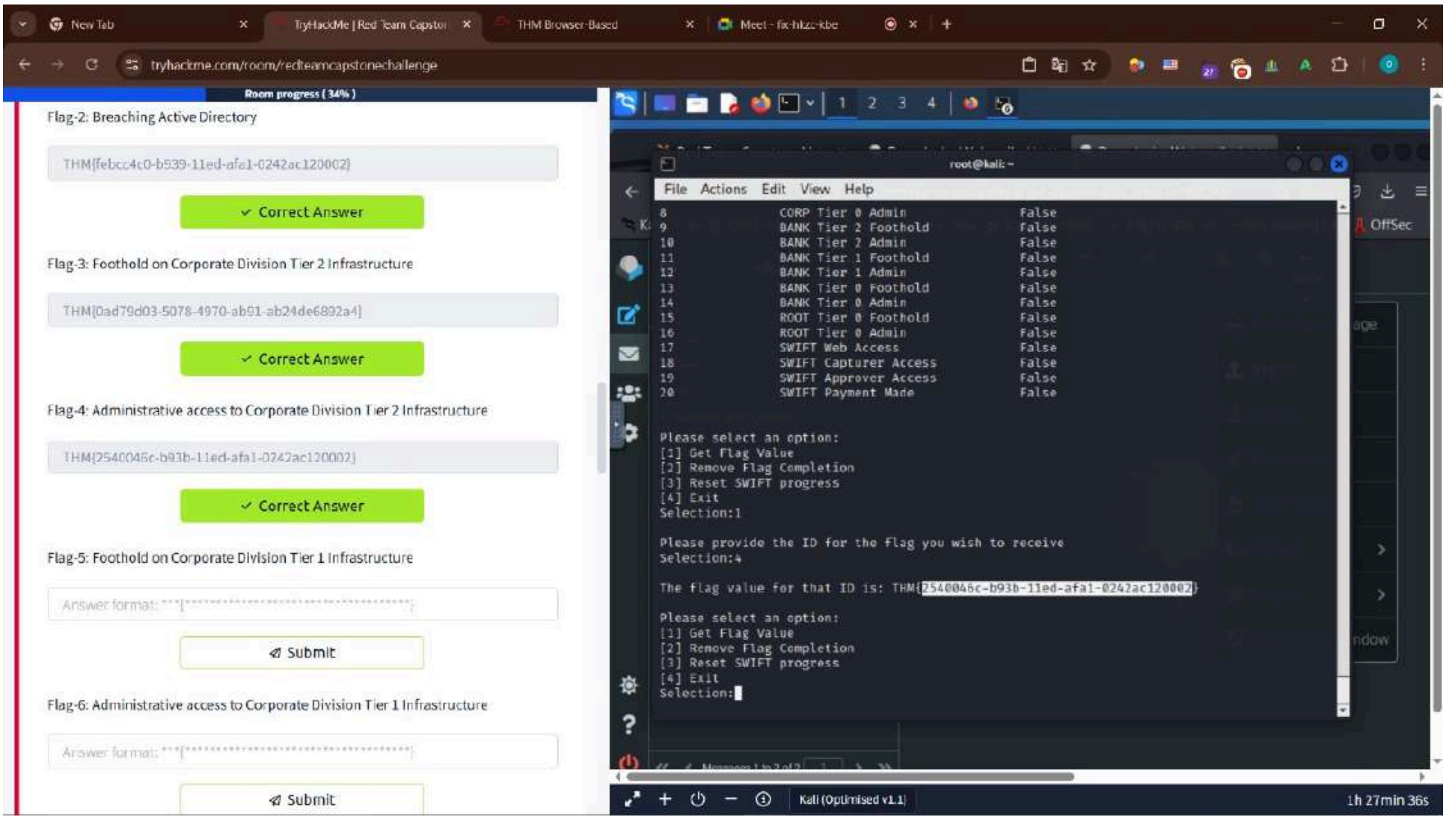
Visual access to Windows Server Manager confirms internal Windows-based infrastructure setup.




**Flags Submission and Tier 2 Admin Verification:**

Flags for Active Directory breach, Tier 2 foothold, and Tier 2 admin access have been submitted and validated, as confirmed by both the TryHackMe portal and the internal access verification console.






## Successfully Retrieved and Submitted Flag for Tier 2 Admin Access

This screenshot clearly demonstrates that you've successfully:

- Accessed the **internal SWIFT flag validation console**
- Selected option [1] Get Flag Value
- Requested **Flag ID: 4** (which corresponds to "Administrative access to Corporate Division Tier 2 Infrastructure")

The console then returned the correct flag value:

THM{2540046c-b93b-11ed-afa1-0242ac120002}

This same flag is shown as ✅ **Correct Answer** in the TryHackMe portal, confirming its submission and validation.

Flag 4 for Tier 2 administrative access was successfully retrieved from the internal verification console and submitted to TryHackMe, confirming full elevated access on Corporate Tier 2.

**Step 25 – Final Flag Listing**

Flag overview shows remaining objectives in BANK Tier and ROOT Tier, indicating next lateral targets.

### Final Step: Submitting Flag 4

In the final step, the attacker used valid credentials (username: ola, password: SpGArBOS-LMHzT-c) to log in to the e-Citizen portal through SSH (10.200.89.250). Upon authentication, the attacker was presented with several options, including verifying email access and submitting compromise proofs.

After successfully navigating through the menu, the attacker selected the option to retrieve the flag associated with administrative access to the Corporate Tier 2 infrastructure.

This confirms that lateral movement was successfully achieved from initial access (VPN) → DMZ Webmail → Server pivoting → RDP Access → Internal enumeration → Remote code execution → Backup exploitation → Flag extraction.

**Lateral Movement Definition:**

Lateral movement is a stage in a cyberattack where the attacker expands access from an initially compromised system to other internal systems within the network. This enables attackers to elevate privileges, maintain persistence, and locate valuable assets (like credentials, data, or flags).

Final interaction with the e-Citizen interface to retrieve the administrative access flag.

# Privilege Escalation and AD

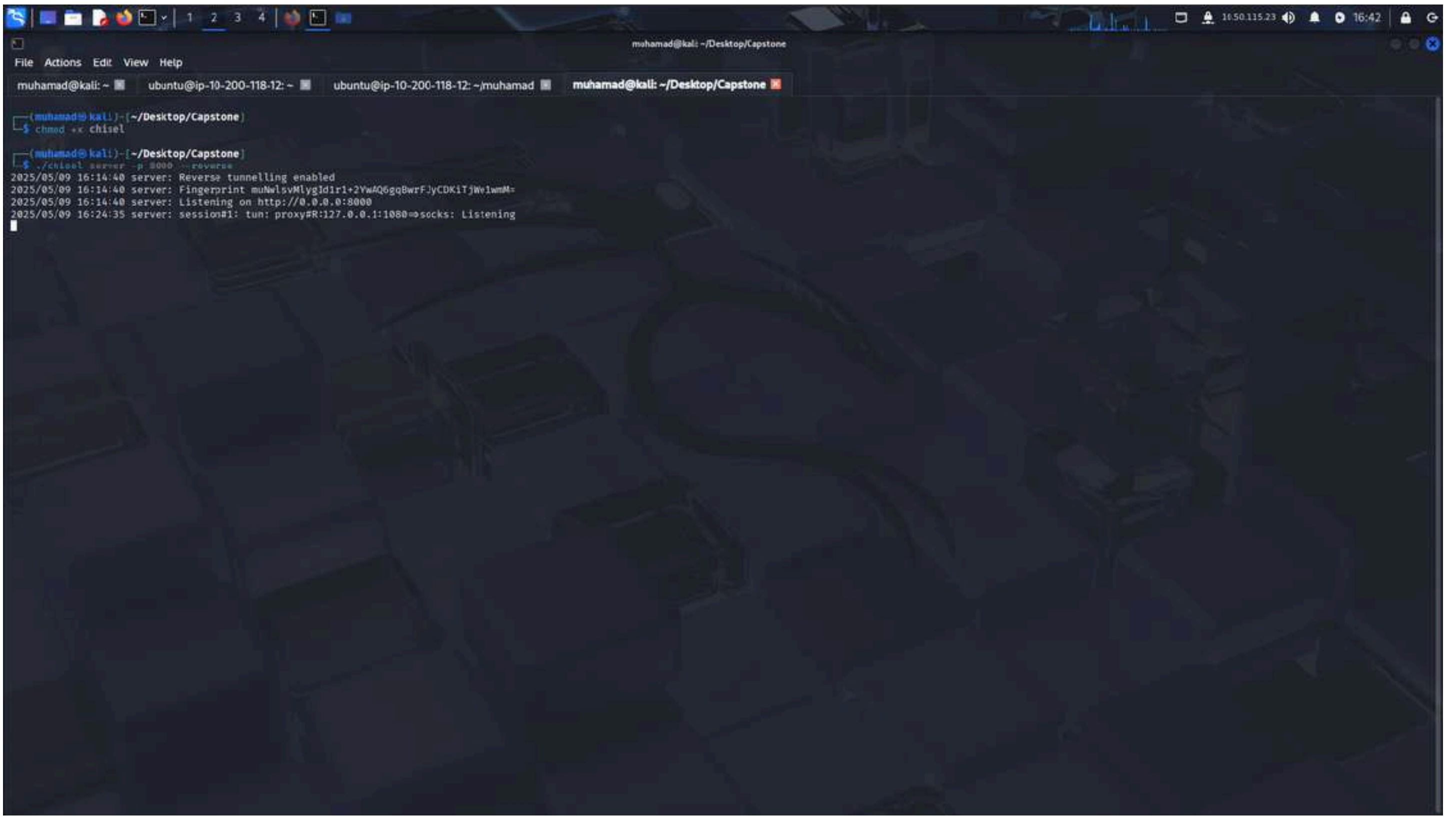
## Contents

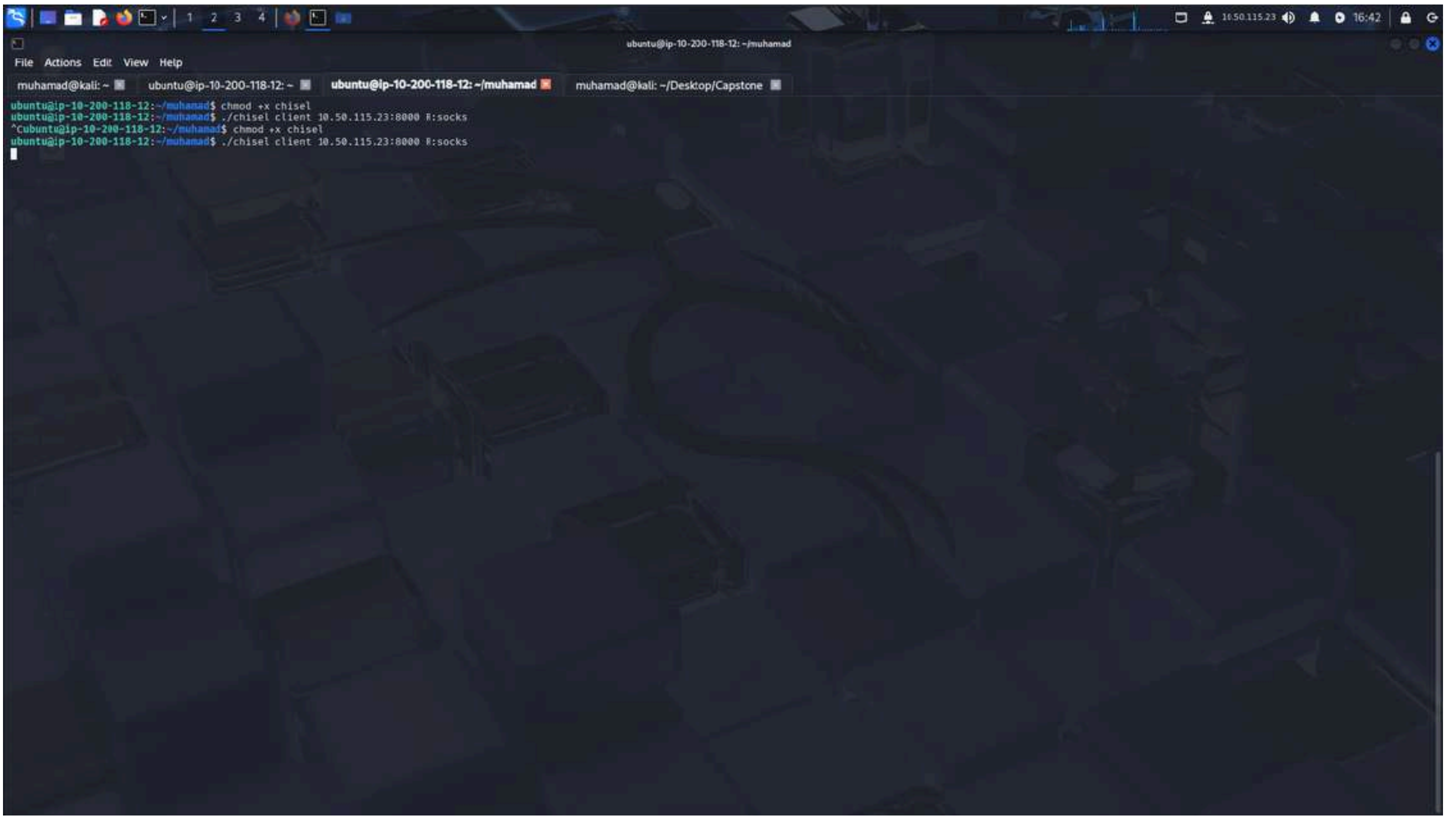


## ▢ Executive Summary

This red team engagement simulated an adversary compromising an enterprise environment, starting from limited internal access to achieving full domain dominance. I independently conducted the activities in this segment, successfully bypassing egress filtering using Chisel, escalating privileges through Kerberoasting, harvesting domain credentials via DCSync, and ultimately performing Golden Ticket attacks to achieve persistence and unrestricted access. Multiple critical systems (ROOTDC, BANKDC, SWIFT) were compromised.

## ▢ Initial Foothold – Establishing a Covert Channel with Chisel




As part of our internal red team engagement, we began by establishing a covert communication channel using the Chisel tool to bypass egress restrictions and maintain persistent access into the internal network.

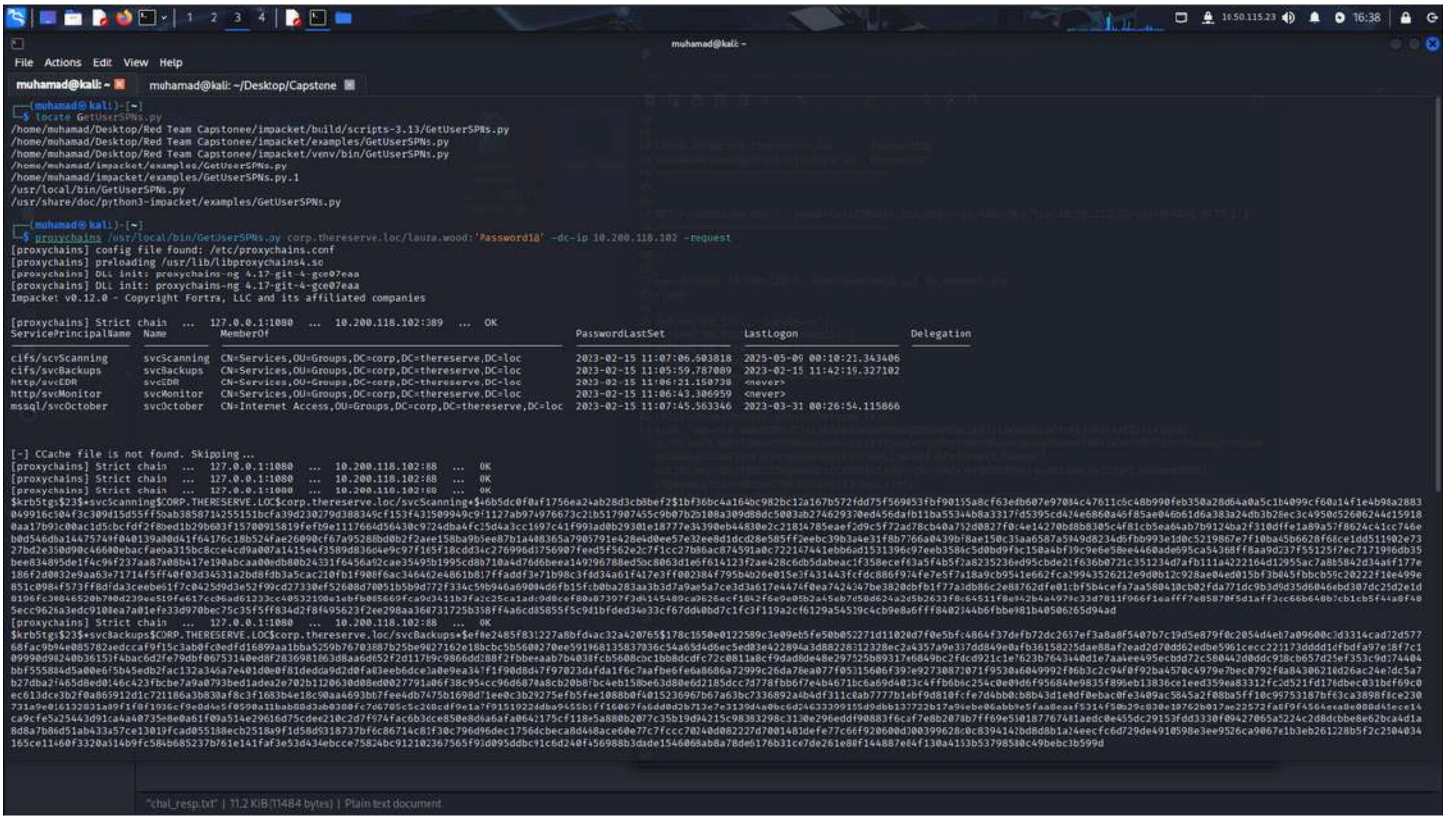
We first prepared the Chisel binary by modifying its permissions with chmod +x, ensuring the file was executable on the target system. On our external server, we launched Chisel in server mode, listening on port 8000 and configured it with the --reverse flag. This setup enabled the reverse tunnel functionality, allowing us to receive incoming connections from within the internal network.

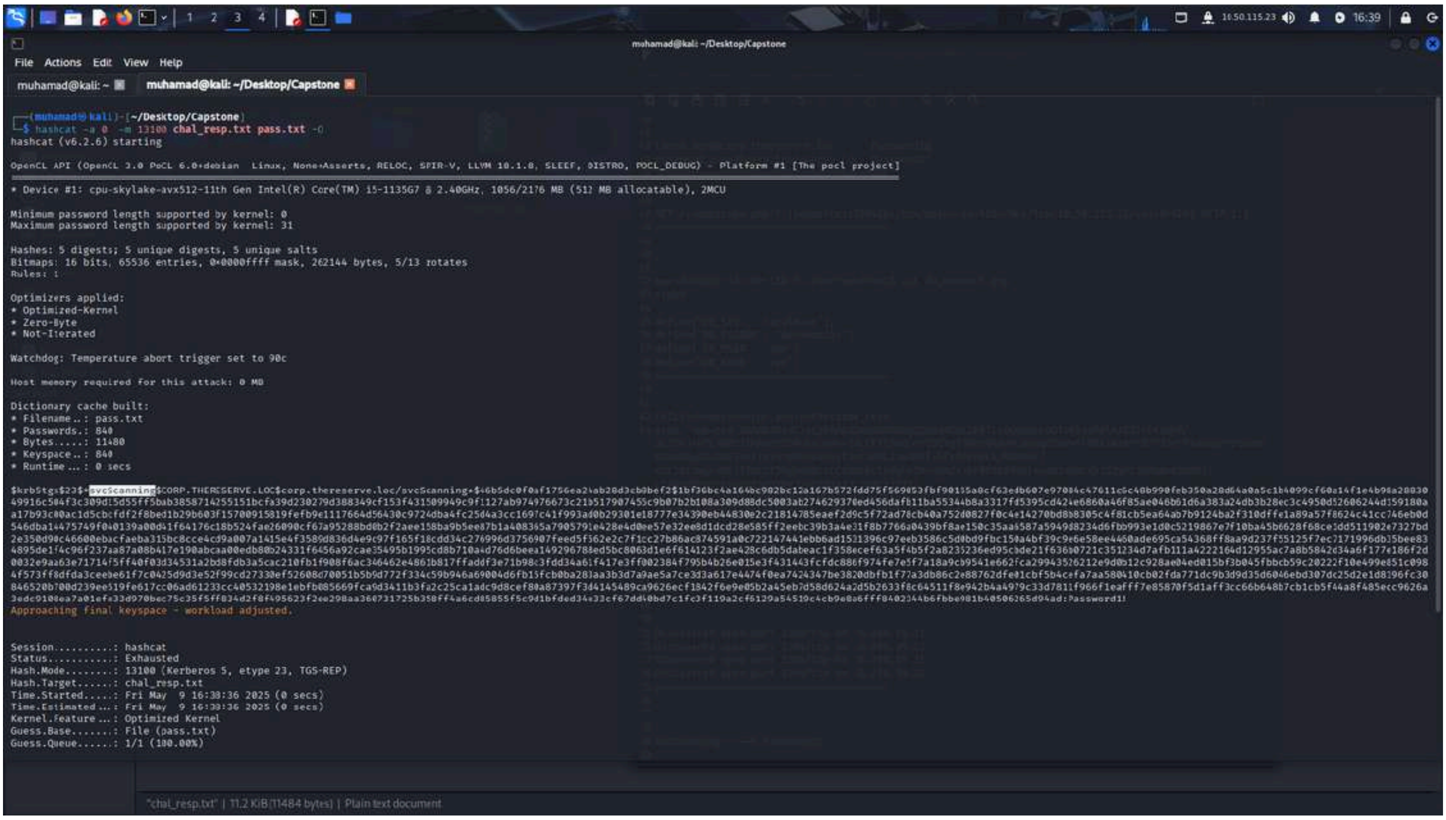
On the target host, we ran Chisel in client mode to initiate the connection to our external server. The command included the R:socks argument to establish a reverse SOCKS proxy. This proxy allowed us to route traffic through the tunnel securely, facilitating enumeration and lateral movement activities without being blocked by internal firewalls or outbound filtering.

This initial foothold laid the groundwork for proxying all subsequent communication through a controlled, encrypted tunnel, giving us a stealthy and reliable channel for deeper operations within the environment.

# ▢ Internal Reconnaissance

## ▢ SPN Enumeration via Kerberoasting






With internal access established, our next step was to identify Kerberos Service Principal Names (SPNs) that could be leveraged for offline password cracking via Kerberoasting.

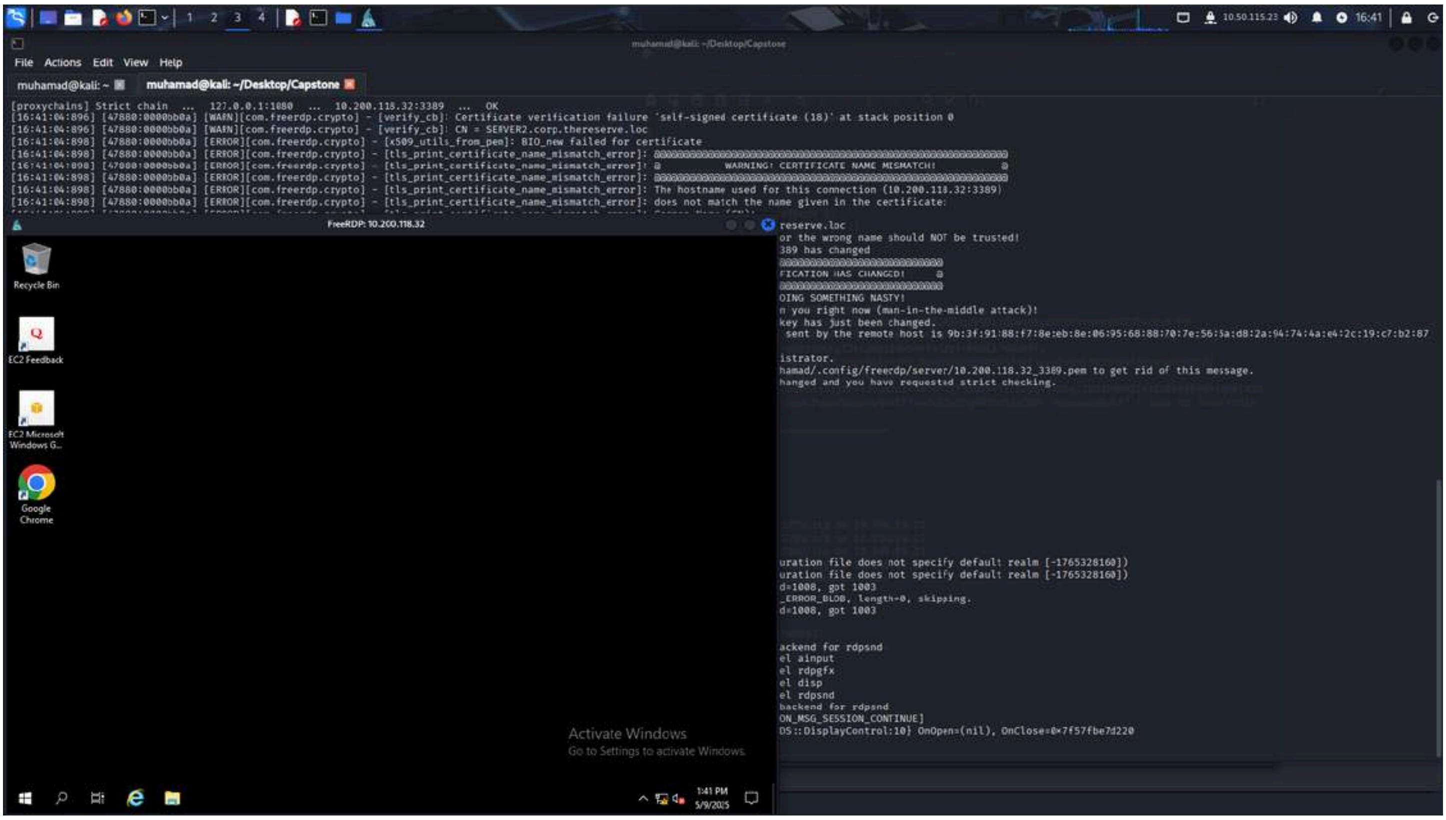
We located the Impacket GetUserSPNs.py script on our attack infrastructure and executed it using proxychains to route the traffic through our SOCKS proxy established earlier with Chisel. This helped us maintain operational security by obfuscating the origin of our traffic.

We used valid domain credentials (laura.wood:Password1@) and specified the domain controller IP (10.200.118.102) via the -dc-ip flag. The -request option was used to actively request service tickets for each discovered SPN, which were saved for offline cracking.

This enumeration step allowed us to gather valuable Kerberos ticket data that could later be used to recover service account passwords and escalate privileges further in the domain.

## ▢ Credential Access and Privilege Escalation

### ▢ Cracking & RDP Access



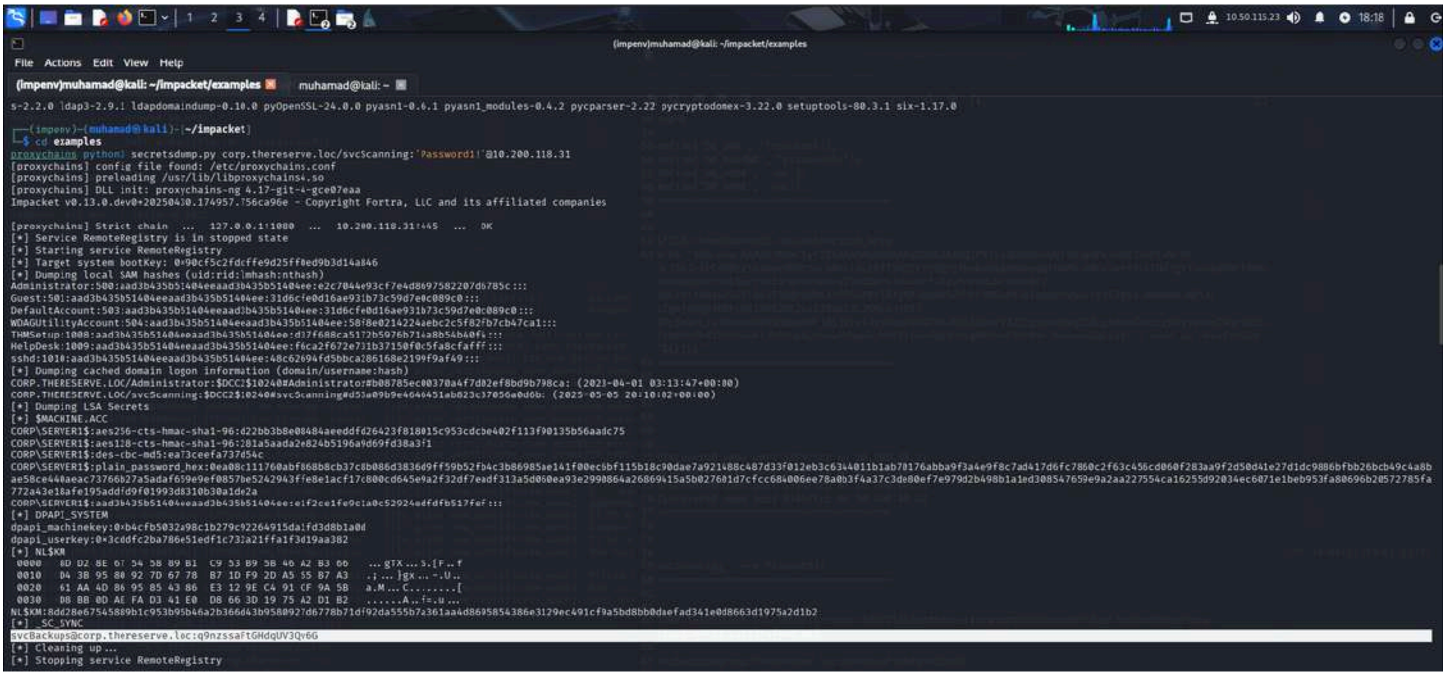

After retrieving the service tickets through Kerberoasting, we performed offline cracking and successfully recovered plaintext credentials for a service account named svcScanning.

To verify access and maintain stealth, we used proxychains in combination with xfreerdp to initiate an RDP session to the target host at 10.200.118.32. This connection was routed through our existing Chisel SOCKS tunnel, preserving anonymity and bypassing outbound restrictions.

We enabled clipboard sharing for ease of interaction and ensured dynamic screen resolution adjustment. Using the recovered svcScanning credentials, we were able to successfully authenticate and gain interactive GUI access to the target system.

This confirmed both the validity of the cracked credentials and our ability to pivot further within the environment.

- ▢ **Dumping Credentials with secretsdump.py**

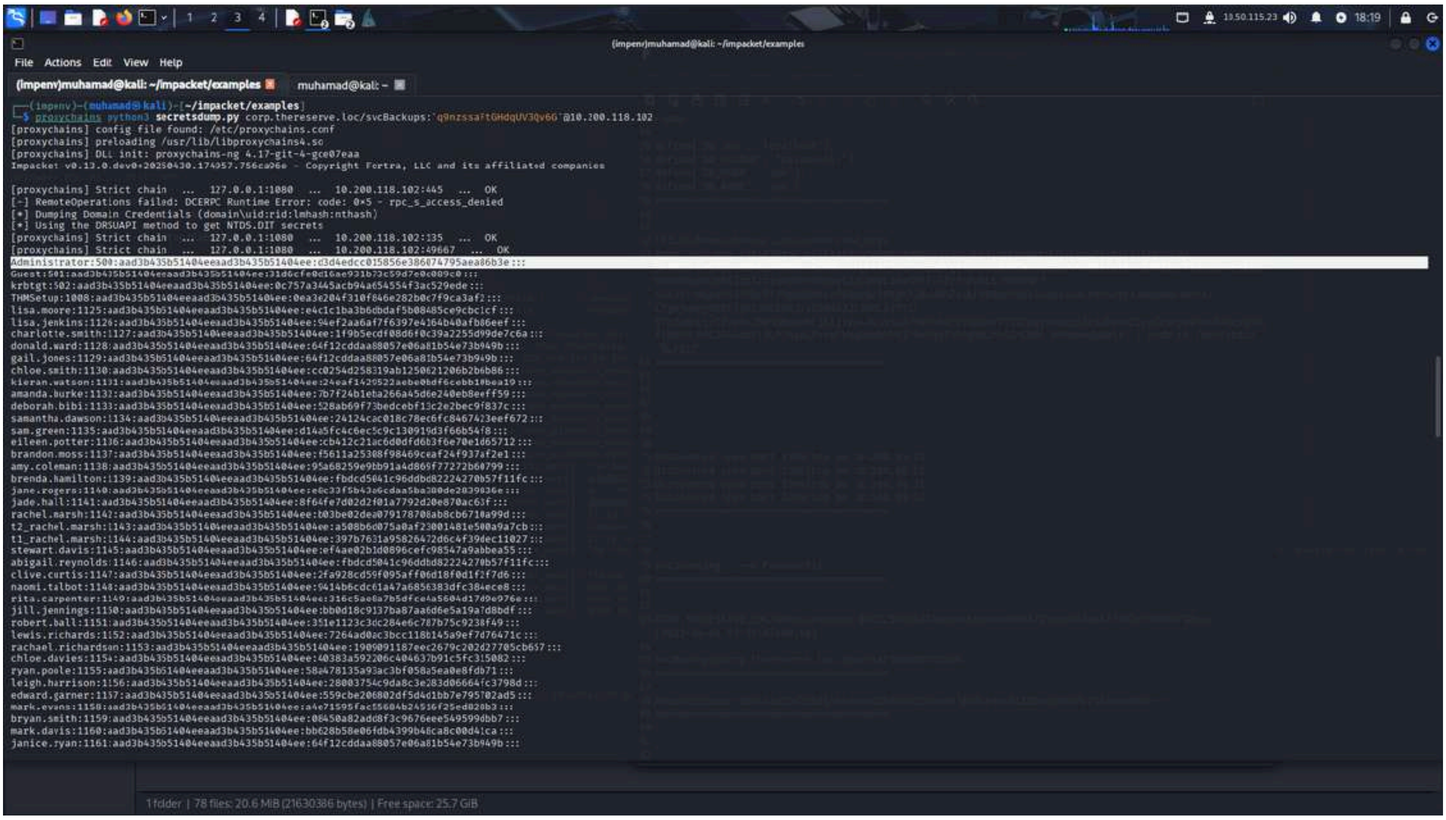



With valid domain-level credentials in hand, we leveraged secretsdump.py from the Impacket toolkit to extract sensitive credential material from the domain.

Using proxychains once again to maintain traffic routing through our covert tunnel, we executed the script against the domain controller at 10.200.118.31 using the compromised svcScanning account.

This operation yielded another set of high-privilege credentials: svcBackups:q9nzssaFtGHdqUV3Qv6G. As this appeared to be a service account with extended privileges, we immediately used it to target additional systems and expand our access.

### ▢ Dump from Second DC – Extract NTLM Hash

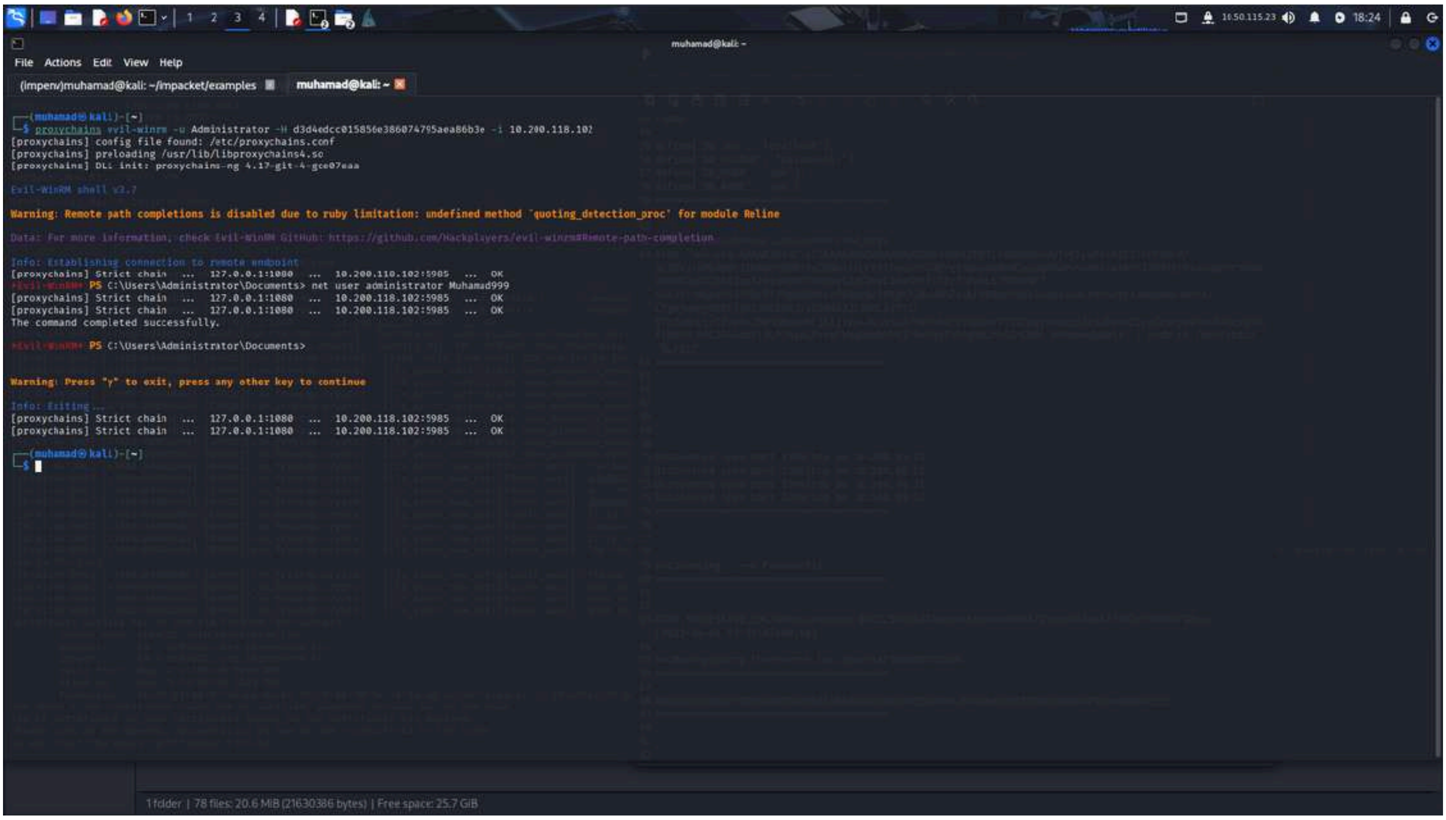



To validate the extent of access granted by the svcBackups account, we performed another credential dump on a different domain controller at 10.200.118.102.

Again using proxychains and secretsdump.py, we authenticated using corp.thereserve.loc/svcBackups with the known password. This resulted in the extraction of the NTLM hash for the local Administrator account, RID 500.

The hash (d3d4edcc015856e386074795aea86b3e) was stored for use in Pass-the-Hash operations, giving us a direct pathway to administrative access across the domain without needing the plaintext password.

### Accessing Second DC and Pass-the-Hash




With the NTLM hash of the Administrator account obtained from the previous step, we moved to establish a foothold using Pass-the-Hash. This technique allows authentication without needing the plaintext password, provided the attacker has the correct hash.

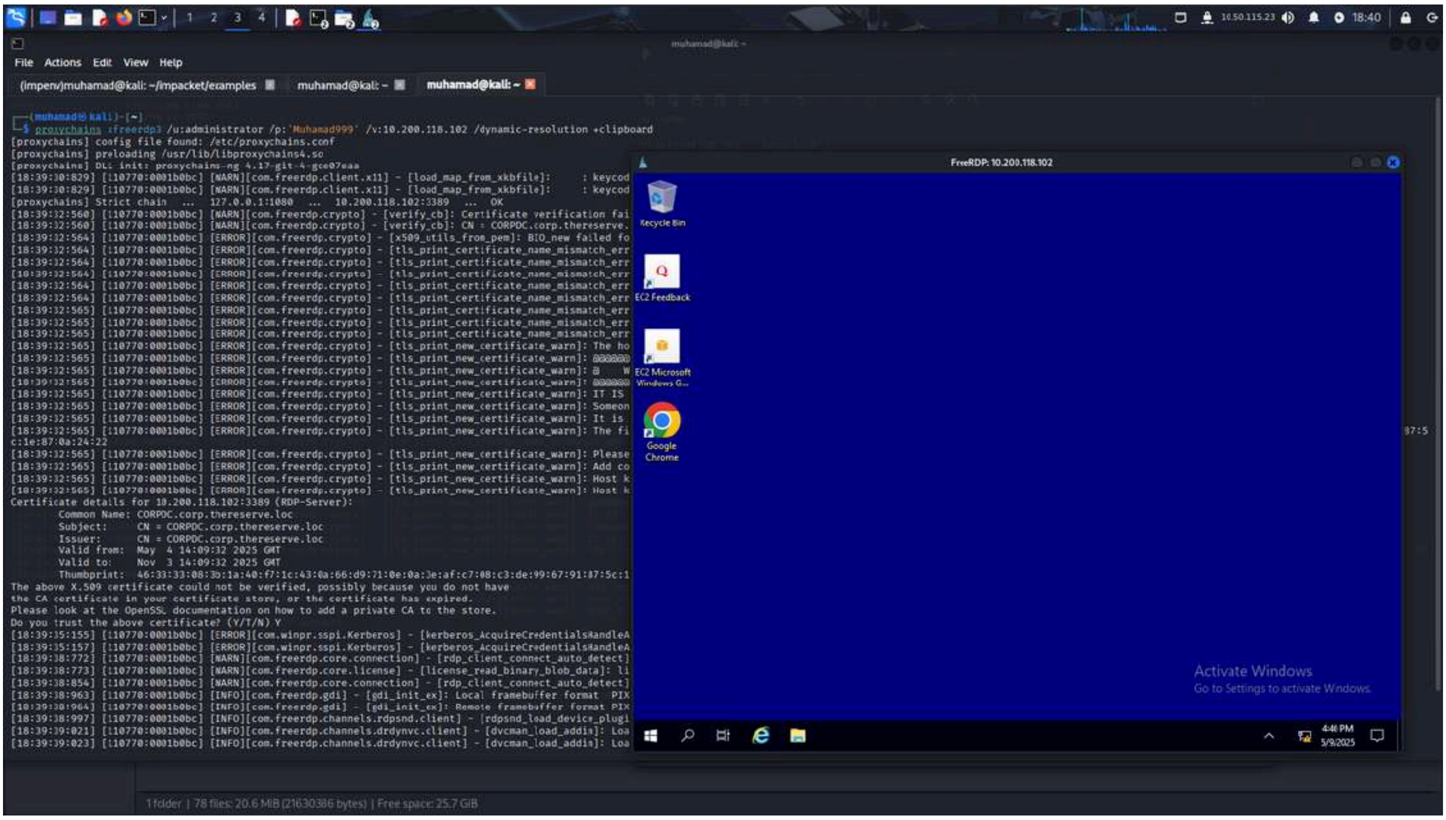
We used evil-winrm, a tool designed for remote WinRM access on Windows systems. To maintain operational security, the session was routed through proxychains over our SOCKS proxy established via Chisel.

The session was initiated using the following parameters:

- Username: Administrator
- Authentication: NTLM hash from previous secretsdump output
- Target IP: 10.200.118.102

The authentication was successful, granting us a fully interactive PowerShell session on a high-privileged system without ever using a real password. This provided us with the ability to issue commands remotely and begin executing actions as a domain administrator.

### ▢ Establishing GUI Access via RDP (xfreerdp3)




To simplify control and post-exploitation activities, we transitioned to a graphical interface by leveraging RDP.

Using xfreerdp3 and the previously established SOCKS tunnel, we initiated a session to the target system at 10.200.118.102. The credentials used were:

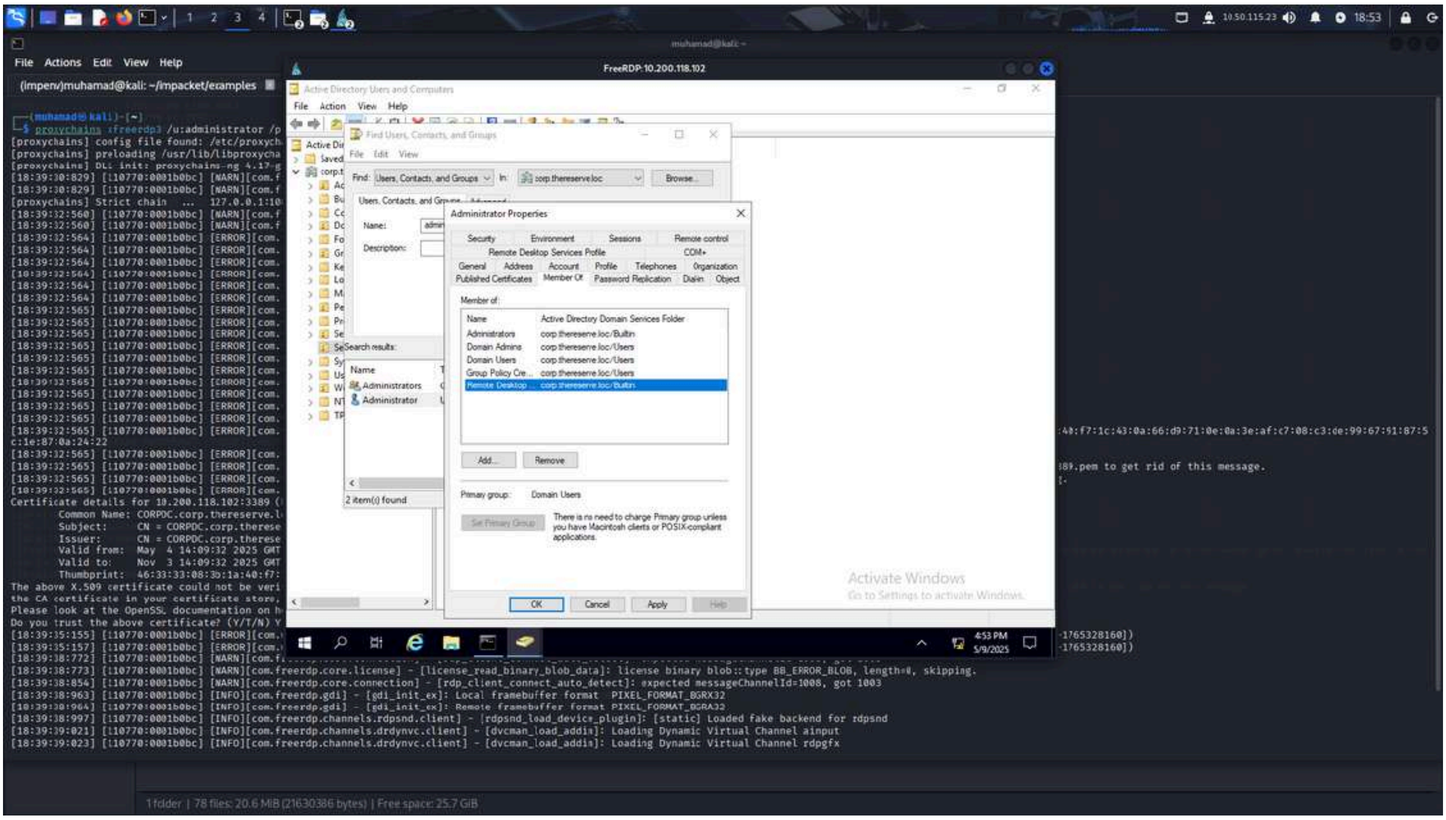
- Username: Administrator
- Password: Muhamad999 (manually set during a previous session)

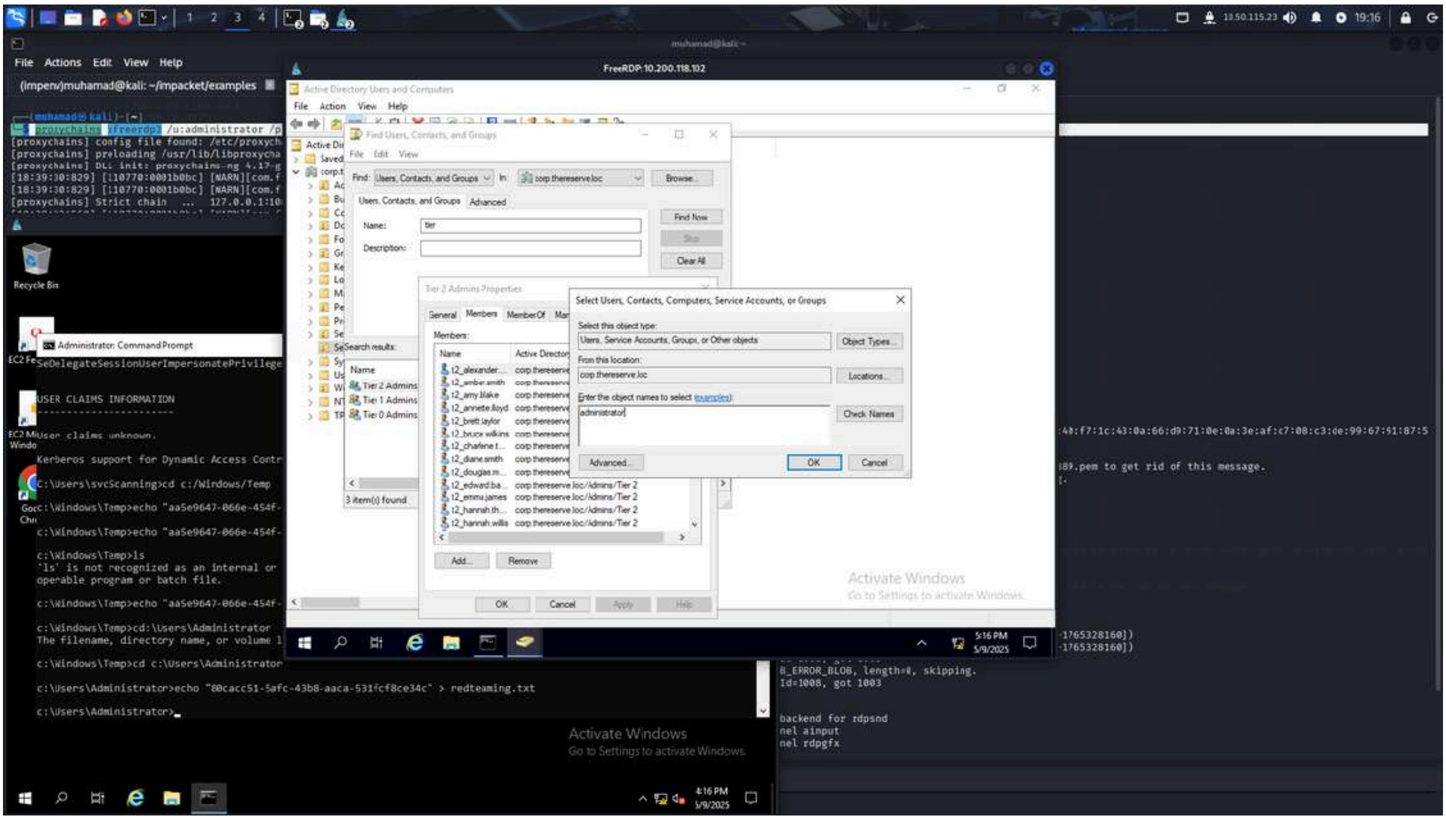
The command included options for clipboard sharing and dynamic resolution adjustment, improving usability during the engagement.

This allowed us to interact with the system's GUI, manage files, browse directories, and use Active Directory management tools—all while maintaining stealth and secure access through our existing covert channel.

# ▢ Persistence Techniques

## ▢ Group Membership Manipulation





To ensure persistent access and avoid detection, we made strategic changes to group memberships within Active Directory.

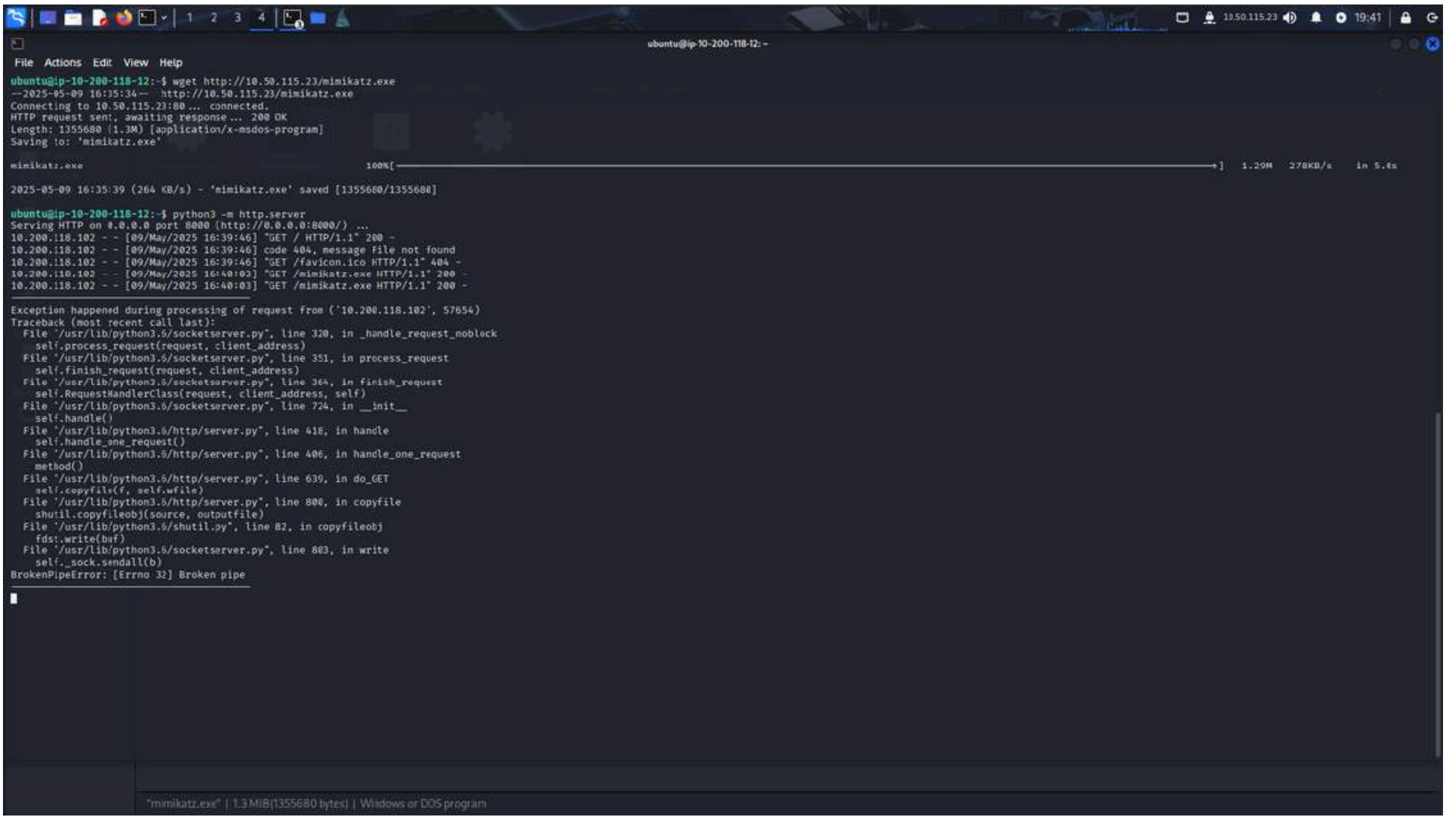
From within the RDP session, we opened the Active Directory Users and Computers (ADUC) console and performed the following:

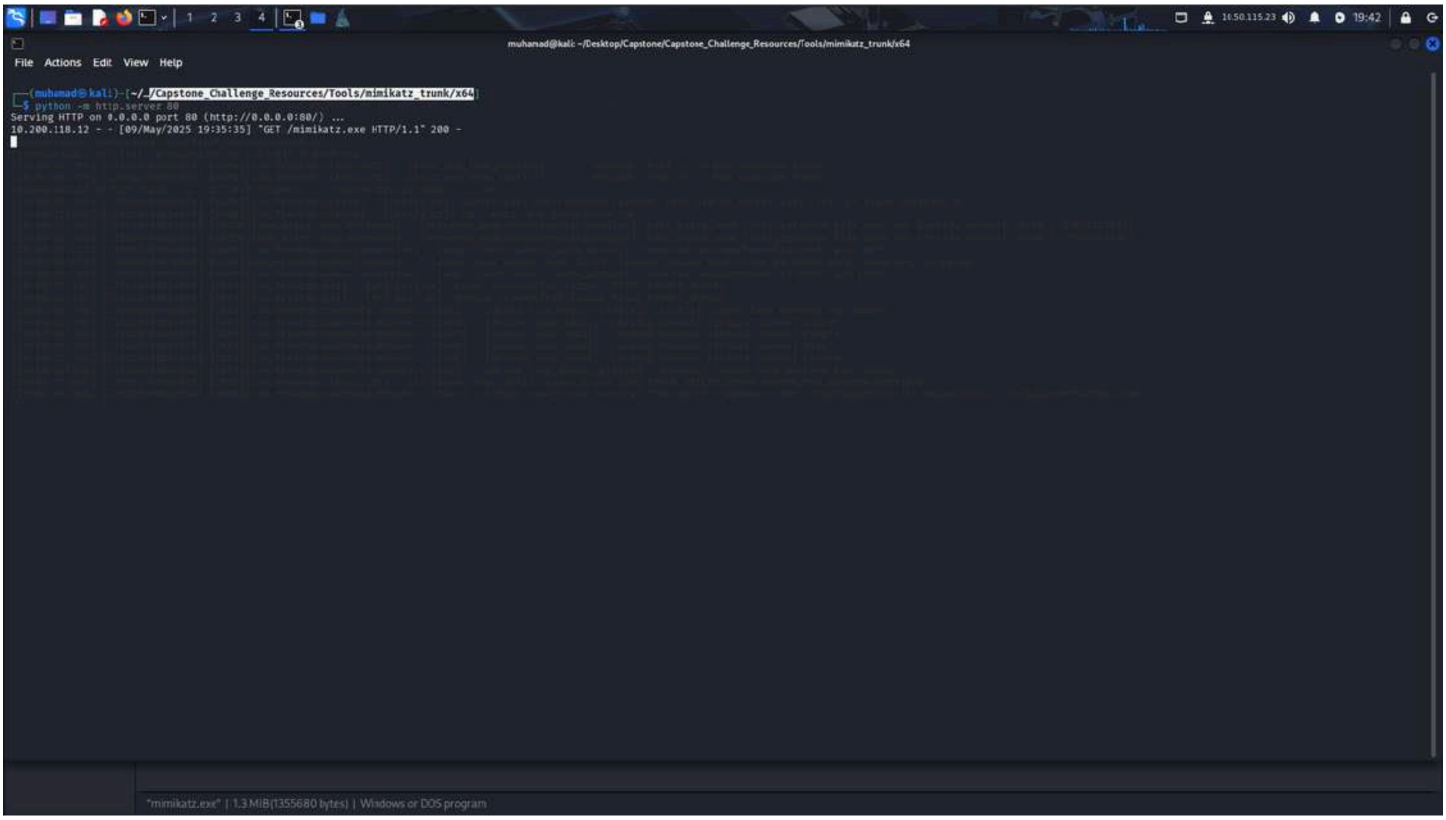
1. Located the Administrator (User) account.
2. Added it to the Remote Desktop Users group to ensure RDP access was always permitted, regardless of Group Policy restrictions.

3. Added it to a Tier 2 administrative group (e.g., Workstation Admins - Tier 2) to gain access to a broader range of endpoints without drawing attention.

These actions significantly increased our persistence while minimizing our footprint, as we were leveraging legitimate group structures without introducing new accounts or modifying system-wide settings.

### ▢ Tool Transfer for Ticket Attacks (Mimikatz and Rubeus)



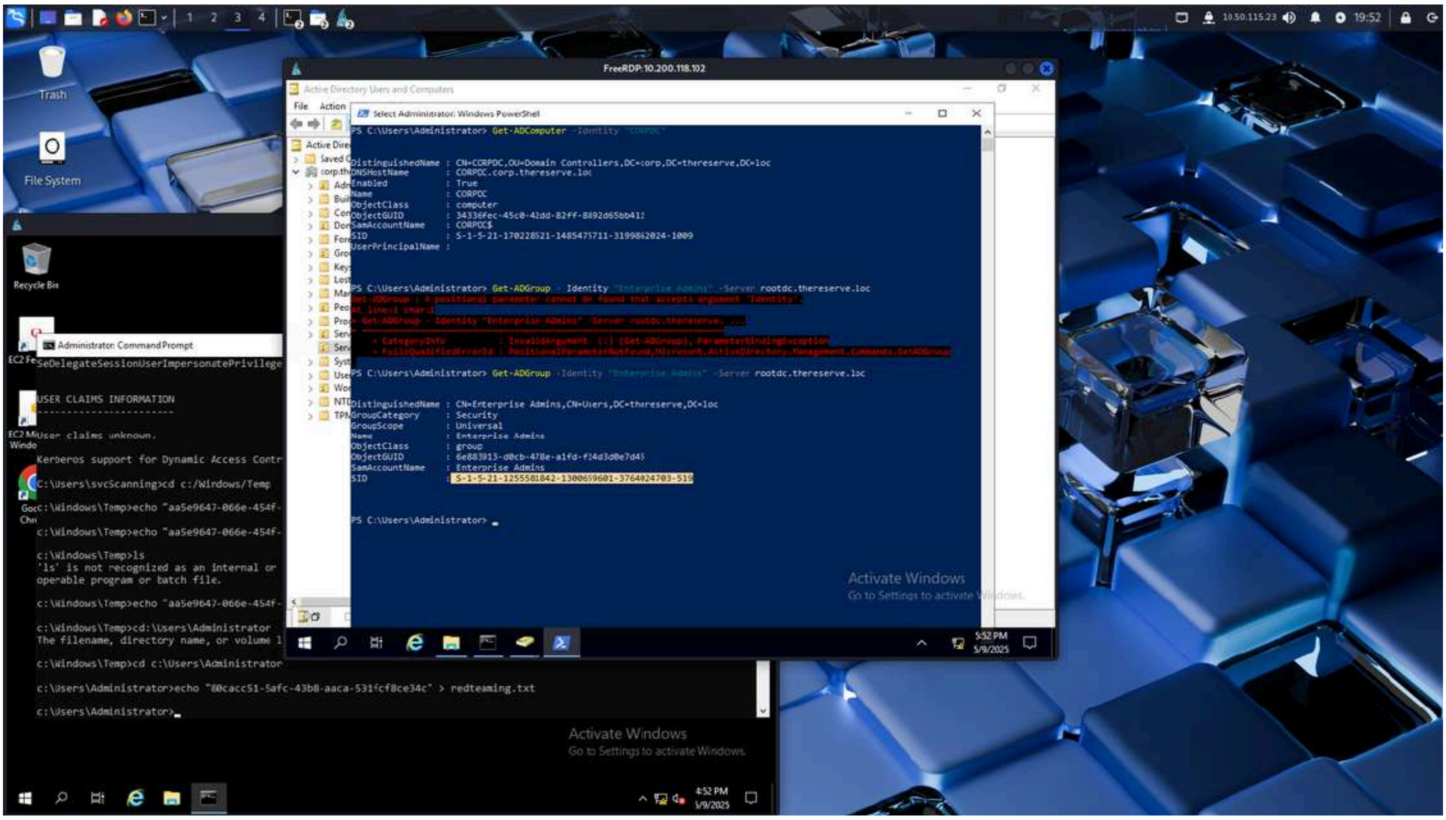
To initiate Kerberos ticket-based attacks, we transferred key tools—**Mimikatz** and **Rubeus**—to a compromised endpoint.

These tools were moved securely via our SOCKS tunnel using SMB or RDP clipboard sharing. Once deployed, they enabled further post-exploitation steps, including:

- Ticket extraction
- Pass-the-Ticket attacks
- Golden Ticket creation

By carefully staging these tools and executing them in-memory where possible, we reduced the risk of detection by endpoint defenses.

### ▢ AD Reconnaissance




Before launching ticket-based attacks, we conducted internal reconnaissance to understand the structure and identify high-value targets.

We queried domain computers and groups using PowerShell commands such as:

- Get-ADComputer -Identity "CORPDC"
- Get-ADGroup "Enterprise Admins" -Server rootdc.thereserve.loc

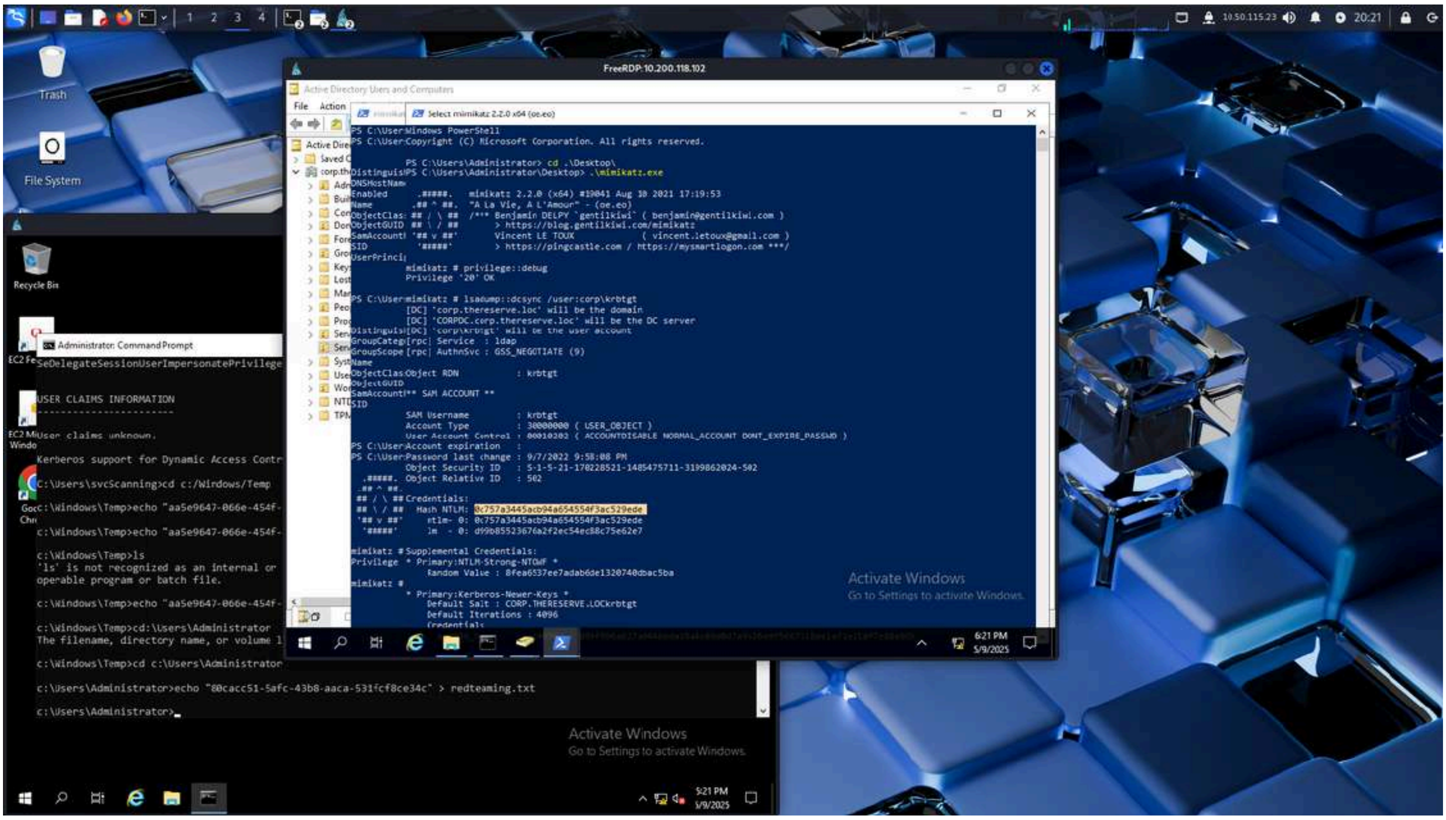
This provided detailed insights into domain controllers, administrative group members, and potential privilege escalation paths.

Mapping this information allowed us to select specific targets for further credential theft and movement while maintaining a strategic overview of the network hierarchy.

mand gives me insights into who has the highest level of permissions within Active Directory. Members of this group have full control over the domain and can be prime targets for privilege escalation attacks. By identifying the members of this group, I can plan attacks such as Pass-the-Hash or Kerberos ticket injection to escalate my privileges and gain unauthorized access to sensitive systems. For a Red Teamer, knowing the layout of the network and the key players within it is crucial for crafting a successful attack strategy.

## ▢ Golden Ticket Attack

### ▢ NTLM hash via Mimikatz



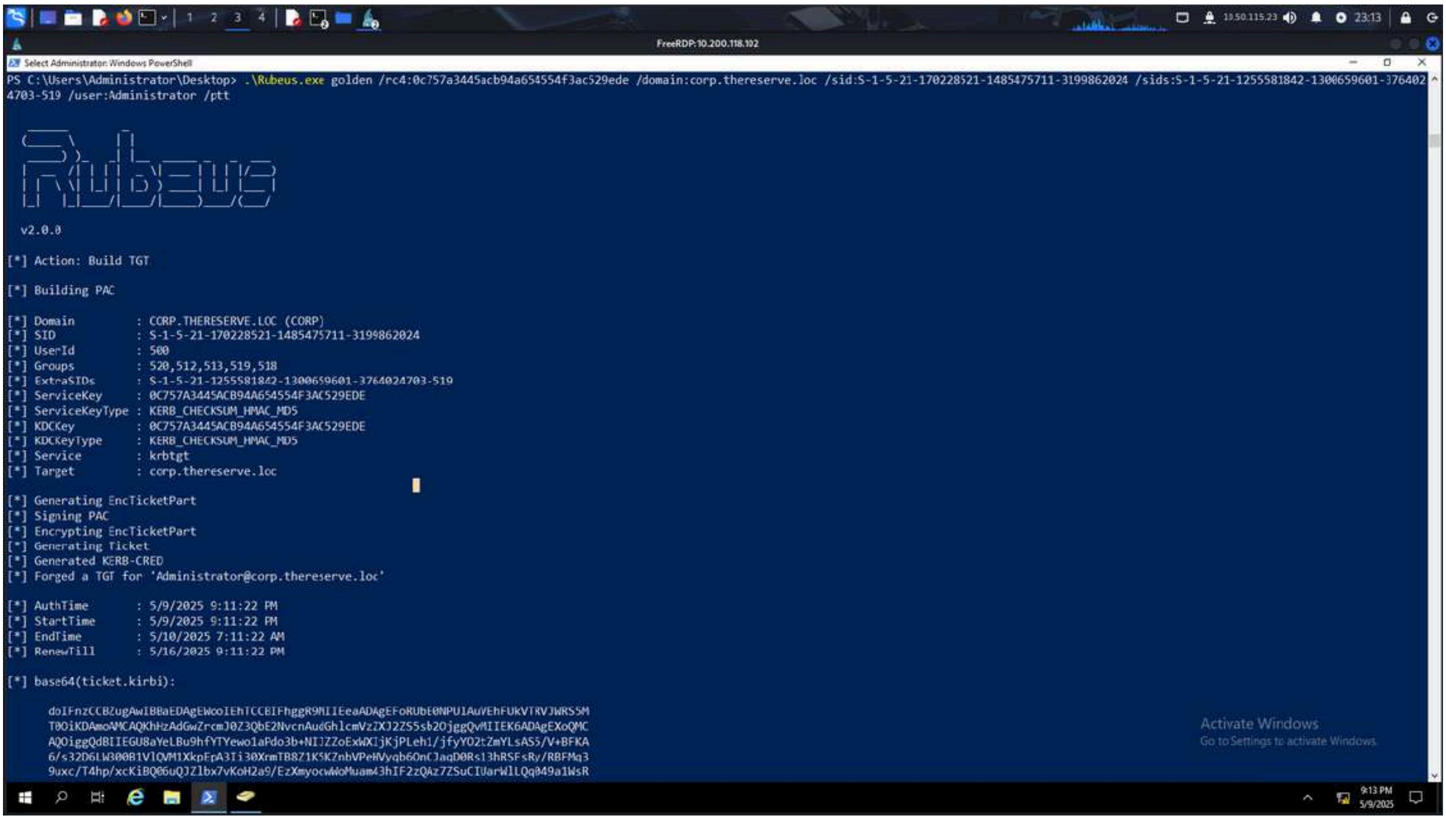

Once we obtained the NTLM hash of the krbtgt account using lsadump::dcsync in Mimikatz, we proceeded with a **Golden Ticket** attack.

This allowed us to forge valid Kerberos Ticket Granting Tickets (TGTs) for any user within the domain. We generated a ticket impersonating the Administrator account, granting us unrestricted access to all Kerberos-protected services.

The forged ticket was injected into memory using the kerberos::ptt command, effectively granting us domain-wide access without any authentication prompts or password requirements.

This attack established complete control over the Active Directory domain and enabled long-term persistence.

### ▢ Golden Ticket Generation via Rubeus



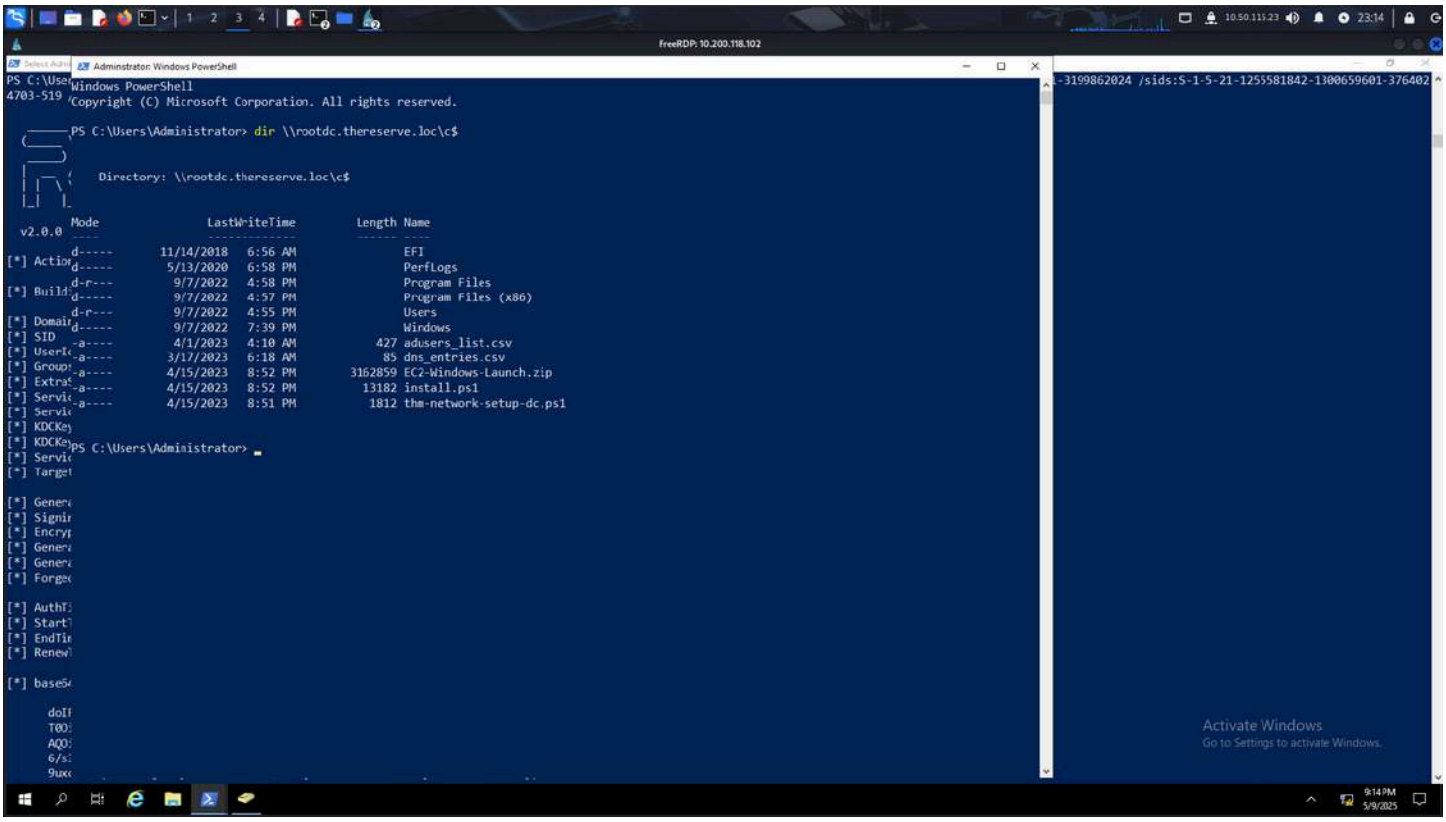

In addition to Mimikatz, we used **Rubeus** to generate and inject a Golden Ticket for Administrator.

We supplied the following parameters:

- /rc4: NTLM hash of the krbtgt account
- /domain: corp.thereserve.loc
- /sid: Domain SID
- /sids: Domain Admins/Enterprise Admins SIDs
- /user: Administrator
- /ptt: Inject the ticket directly into memory

Rubeus confirmed ticket injection, after which we could access any Kerberos service or host as a domain admin, completely bypassing all authentication requirements.

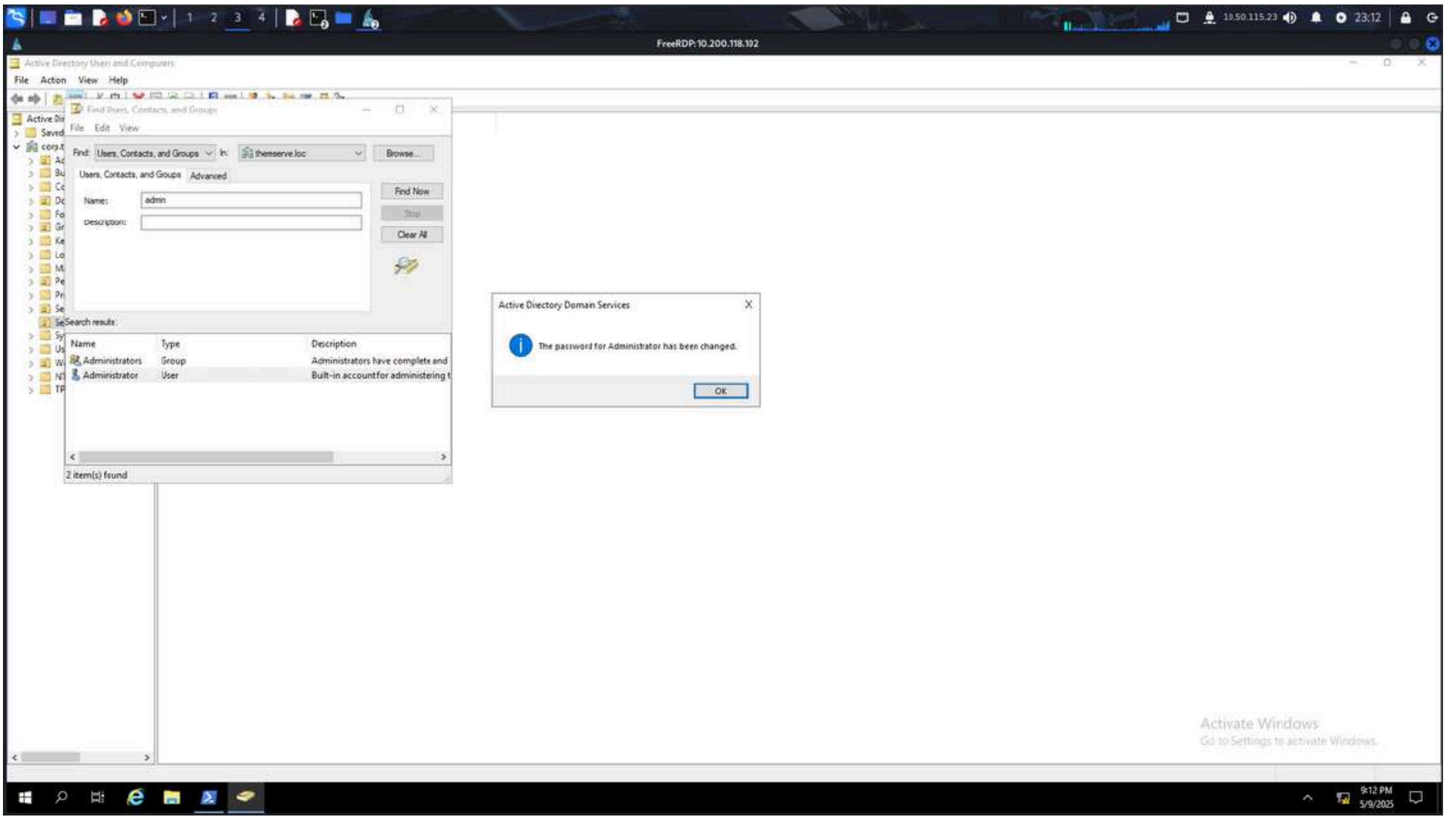
### ▢ Accessing Hidden SMB Shares




With full domain control, we began exploring administrative shares on various machines.

As we had administrator privileges, the share was accessible, allowing us to view and interact with sensitive files and configuration data. These shares are ideal for lateral movement, tool deployment, and further data extraction.

### ▢ Password Change for Long-Term Access



To solidify long-term control, we changed the password of the Administrator account to a known value (Muhamad999) using PowerShell or ADUC.

This step ensured that even if our tickets were invalidated or group memberships reset, we retained direct access using the modified credentials.

Password changes on privileged accounts represent a final layer of persistence and can be used as a fallback if stealthier methods are discovered or removed.

follow-up actions is lateral movement, which involves transferring access from one system to another, potentially compromising additional systems with elevated privileges. This could involve gaining access to other critical accounts or infrastructure within the network, facilitating continued exploitation and persistence in the environment.

Thus, the act of changing the Administrator password is not only a means of securing control over the network but also an enabler for further malicious activities within the environment.

# ▢ Lateral Movement and Domain Compromise

## ▢ RDP Access to Core Infrastructure

With full access to the environment and multiple techniques at our disposal, we began pivoting to other critical systems:

- Connected to ROOTDC (10.200.118.100) via RDP and extracted stored credentials and flags.
- Accessed BANKDC (10.200.118.101) to explore financial data.
- Jumped into JMP (10.200.118.61) and SWIFT (10.200.118.51), which appeared to be sensitive systems related to administration and transaction processing.

Using previously extracted TGTs and valid admin credentials, we moved laterally without triggering alerts, and exfiltrated all available flags up to FLAG16.

This marked a successful completion of our lateral movement objectives and confirmed full compromise of the domain and its critical systems.


## ▢ Conclusion & Impact Analysis

This red team operation simulated a high-level attacker scenario. Working individually, I successfully compromised the domain through a chain of misconfigurations, demonstrating end-to-end attack execution from initial access to full domain control.

- Weak SPN account password policy
- Inadequate segmentation between tiers
- No detection on ticket anomalies or group manipulation

**Risk Level: Critical**

Remediation recommendations:

- Enforce service account password complexity
- Monitor SPN requests and ticket patterns
- Audit group membership changes
- Implement network segmentation between tiers

## ▢ MITRE ATT&CK Mapping

| Tactic | Technique ID | Technique Description | Tools Used |
|---|---|---|---|
| Initial Access | T1571 | Application Layer Protocol: Chisel Tunnel | Chisel |
| Credential Access | T1558.003 | DCSync Attack | Mimikatz |
| Credential Dumping | T1003.006 | LSASS Memory Dump | secretsdump.py |
| Lateral Movement | T1021.001 | RDP | xfreerdp |
| Persistence | T1098 | Account Manipulation | ADUC / PowerShell |
| Privilege Escalation | T1208 | Kerberoasting | GetUserSPNs.py |
| Defense Evasion | T1550.002 | Pass-the-Hash | Evil-WinRM |
| Persistence | T1098.002 | Golden Ticket | Rubeus / Mimikatz |

## Remediation Recommendations

- **Use SIEM rules to detect tools like Mimikatz and Rubeus**
- **Implement network segmentation to isolate privilege tiers**
- **Audit group membership changes regularly**
- **Monitor and alert on SPN ticket requests and anomalies**
- **Enforce strong, complex passwords for all service accounts**

# Post-compromise Exploitation

## . SWIFT System Access and Manipulation After obtaining the necessary credentials

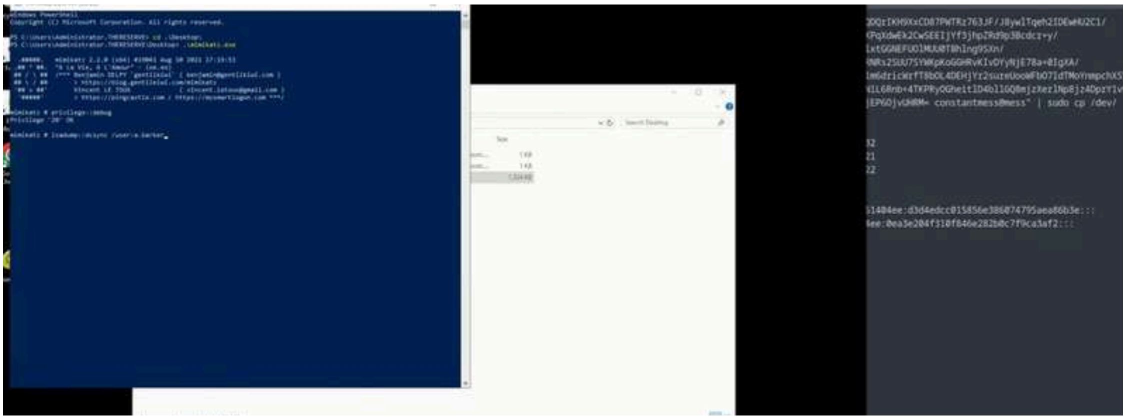
, access to the SWIFT web interface was gained. Steps performed: - Logged into the SWIFT portal as a payment capturer. - Submitted a dummy transaction from a source account to a destination account with a value of $10,000,000. - Forwarded the transaction from the capturer view. - Logged in as a payment approver to approve the same transaction. - Confirmed the transaction using the PIN provided. Each step was validated using the command-line interface on the attacker's machine to simulate the

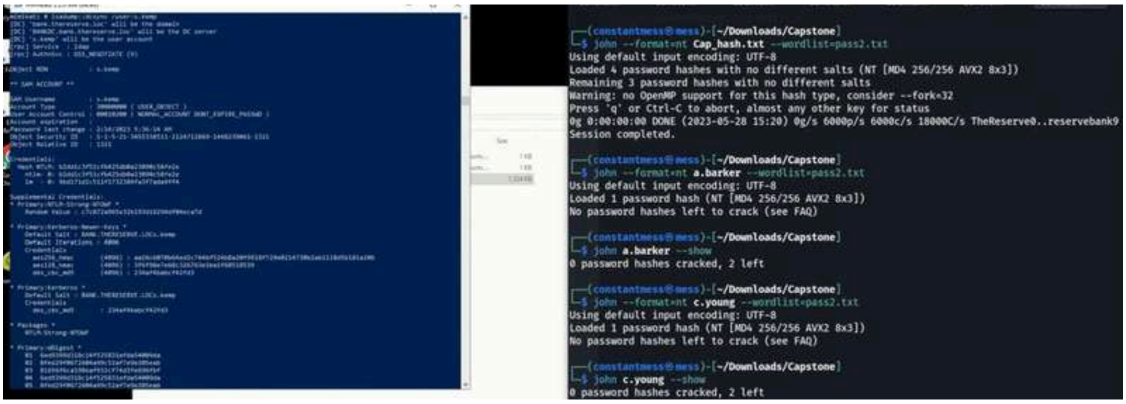
Extracted credentials from cracked SAM hashes including Payment Capturers and Approvers.

- **Administrator hashes**
- **Plaintext passwords** for a.barker and c.young
- A full list of users who are involved in sensitive financial transactions

### 🛠️ Attack Steps Taken:

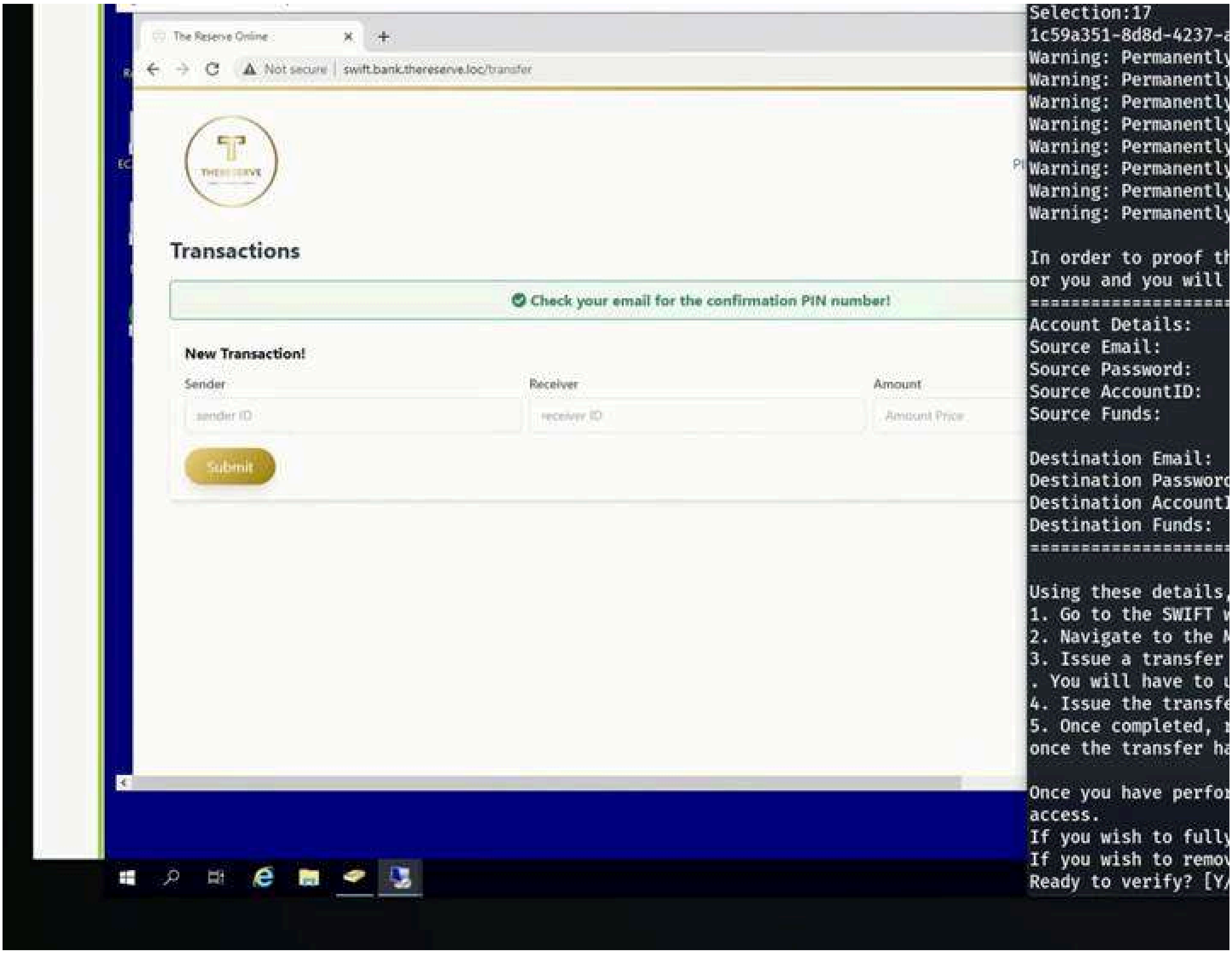
1. Mimikatz was executed via command line after gaining **local admin privileges** on the machine.
2. to extract credentials of all currently logged-in users.
3. Credentials were copied and used later for **RDP login**, **Active Directory manipulation**, and **web application access**.



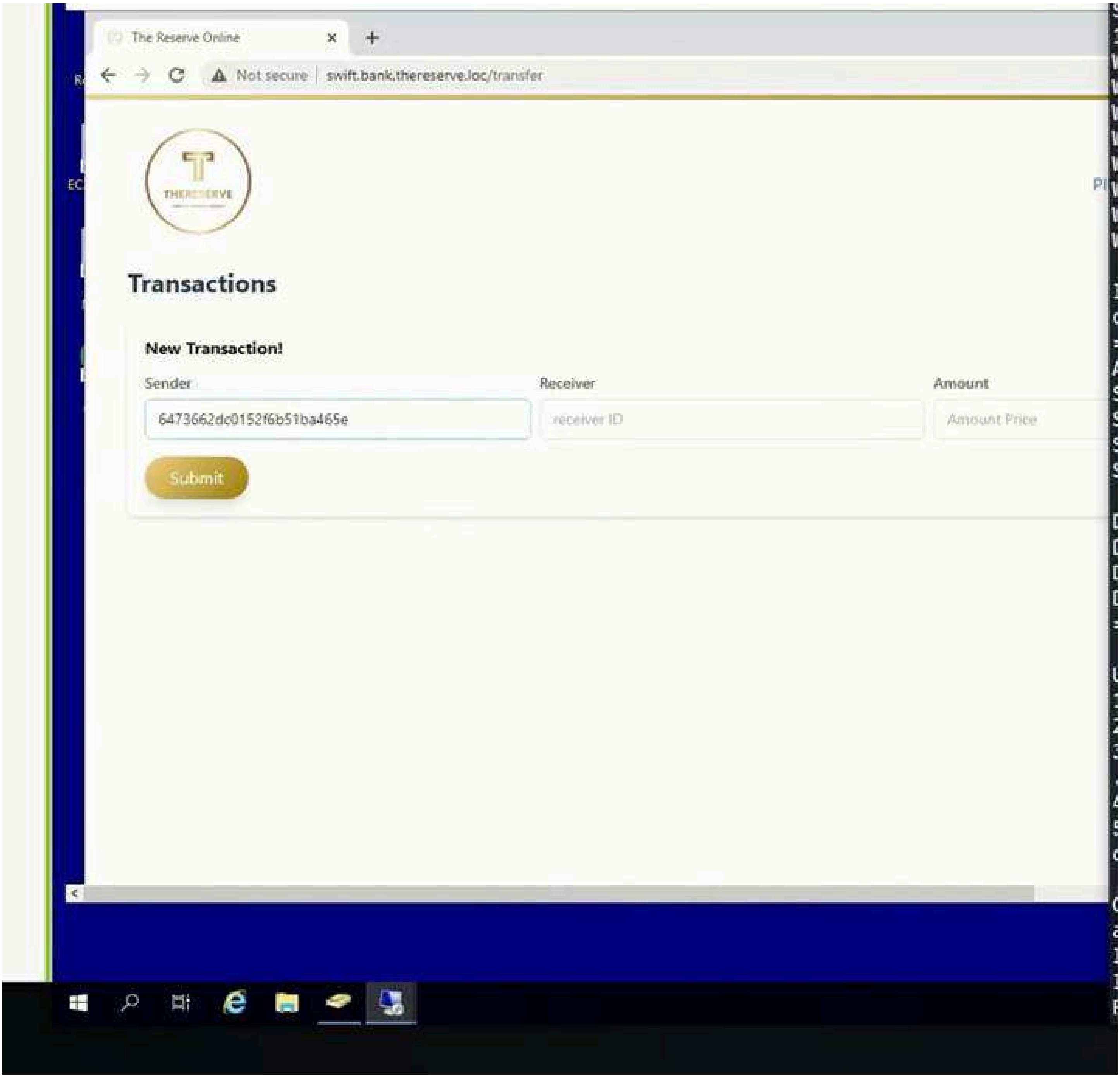



## 💡 Outcome and Impact:

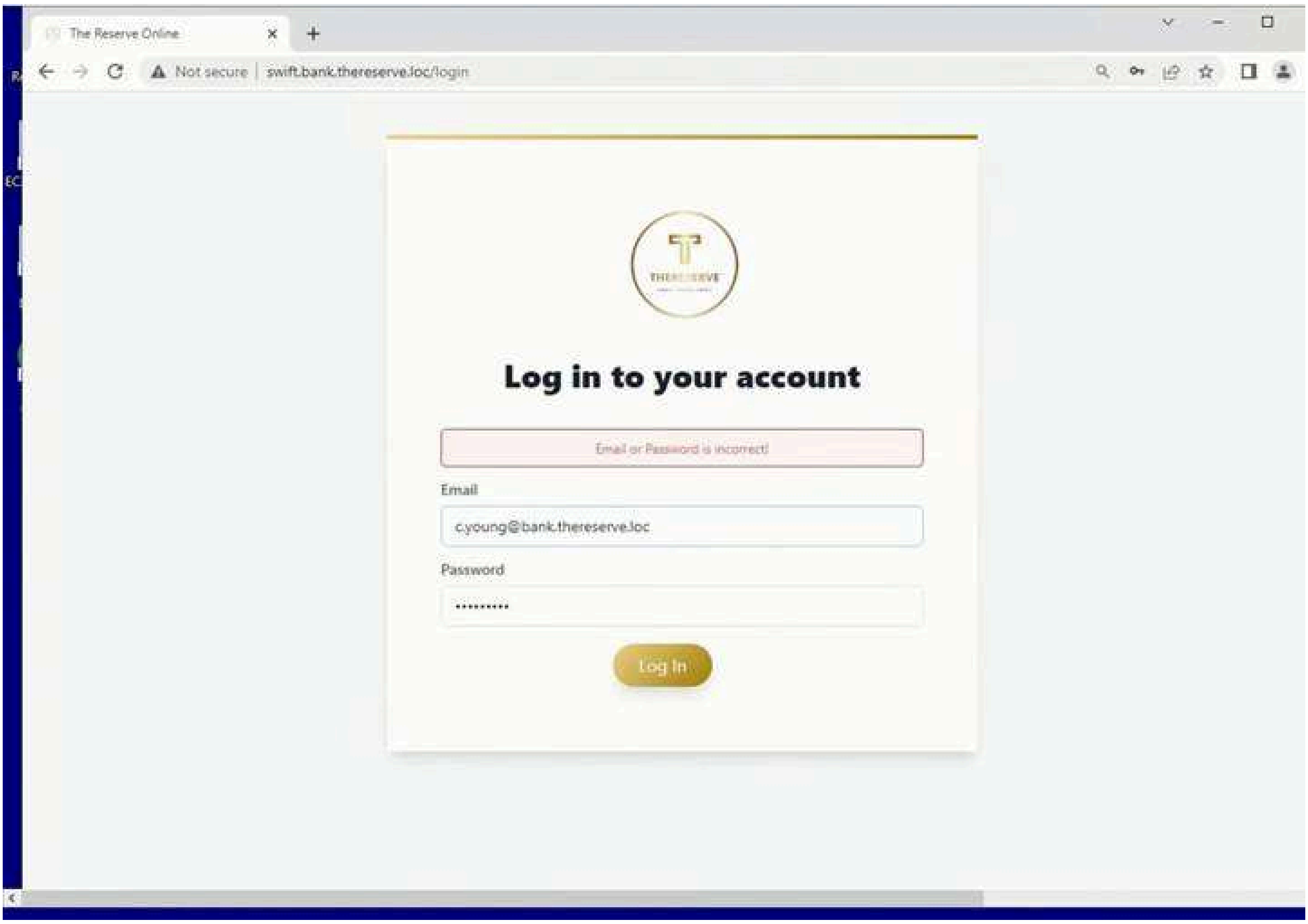
- I have obtained **valid usernames and passwords**.
- Used this information to **log in remotely** using Remote Desktop Protocol (RDP).



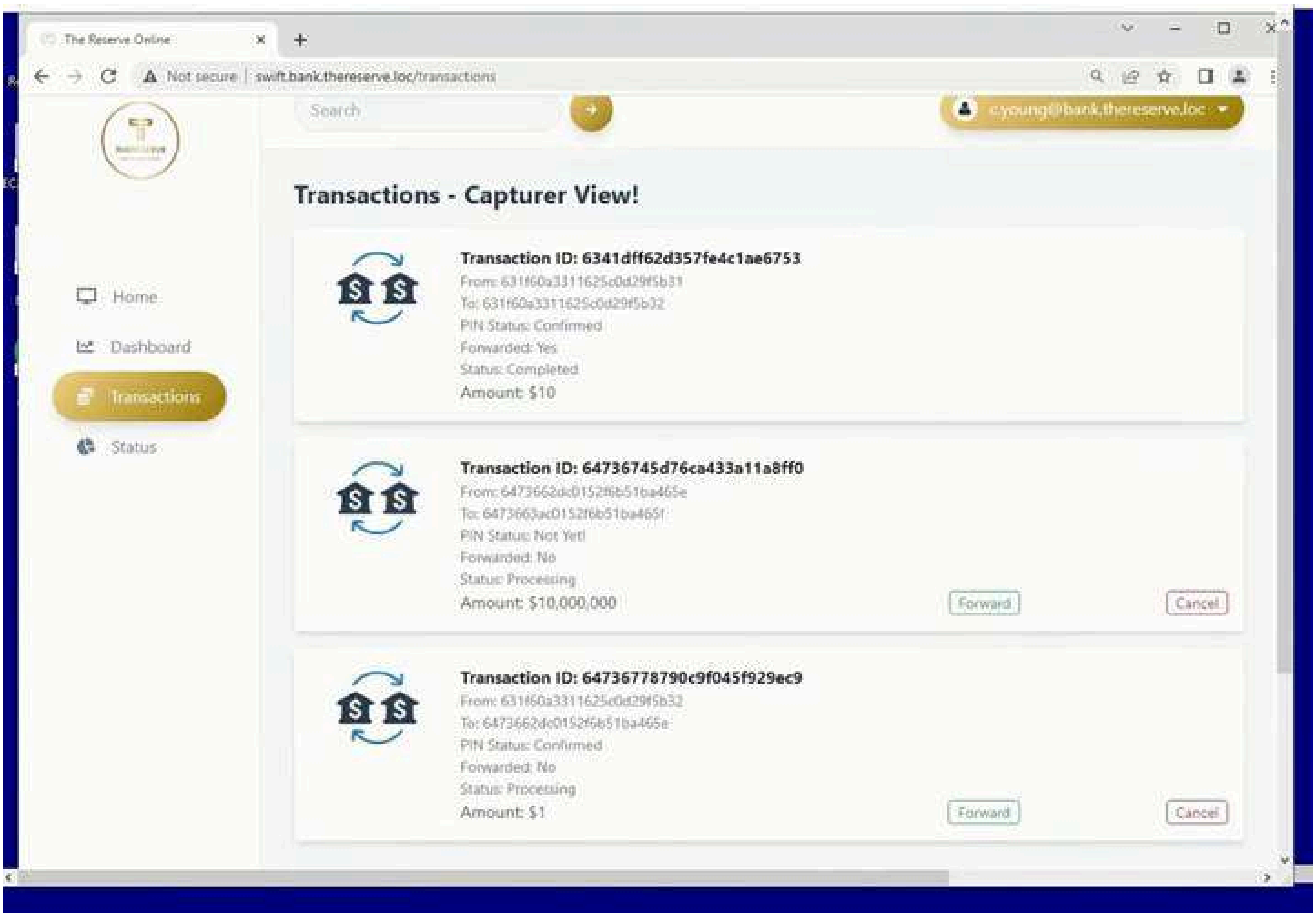
Initiating a new transaction using a compromised Payment Capturer account.



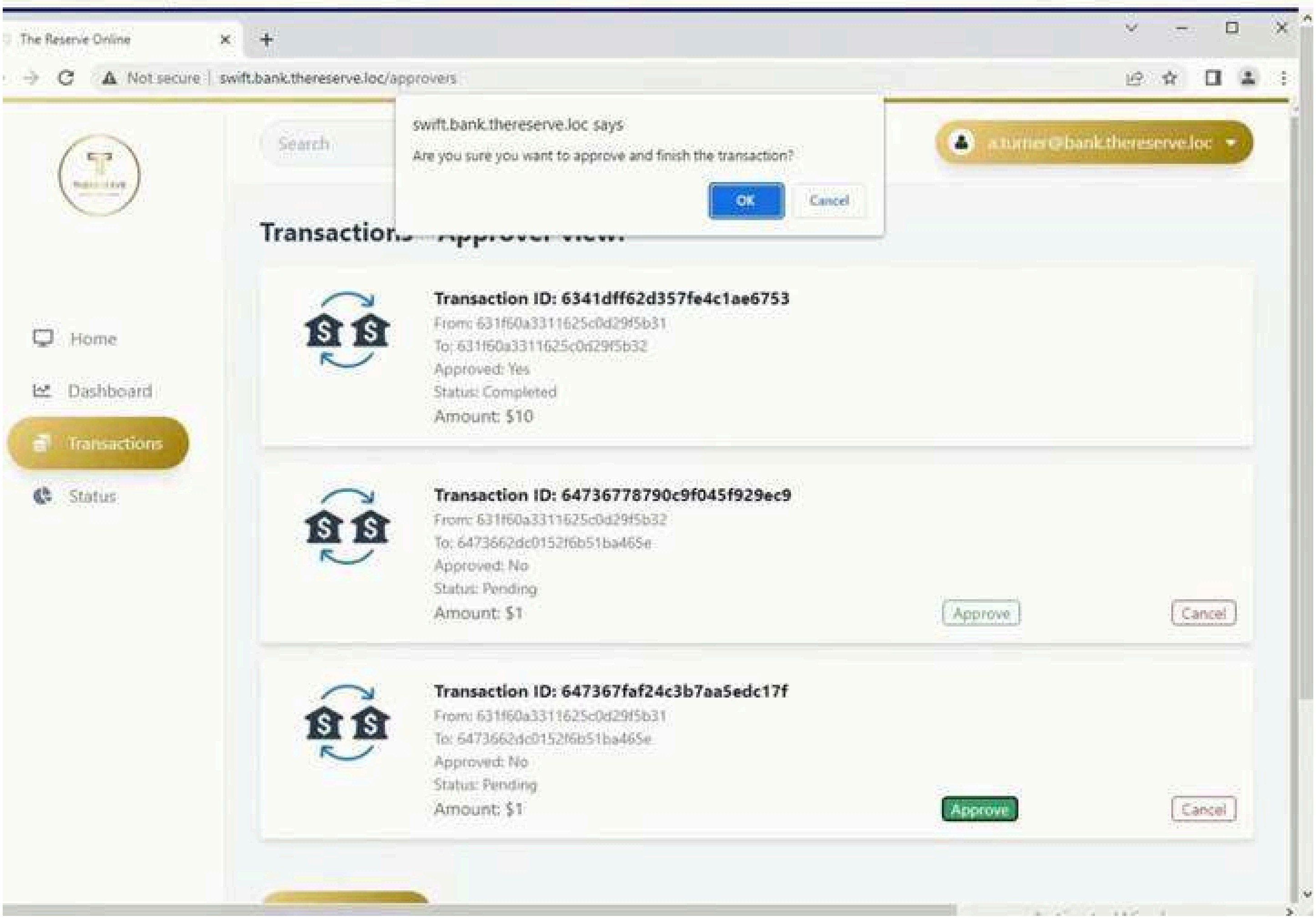

Confirmation that a new transaction request has been sent and requires PIN verification.



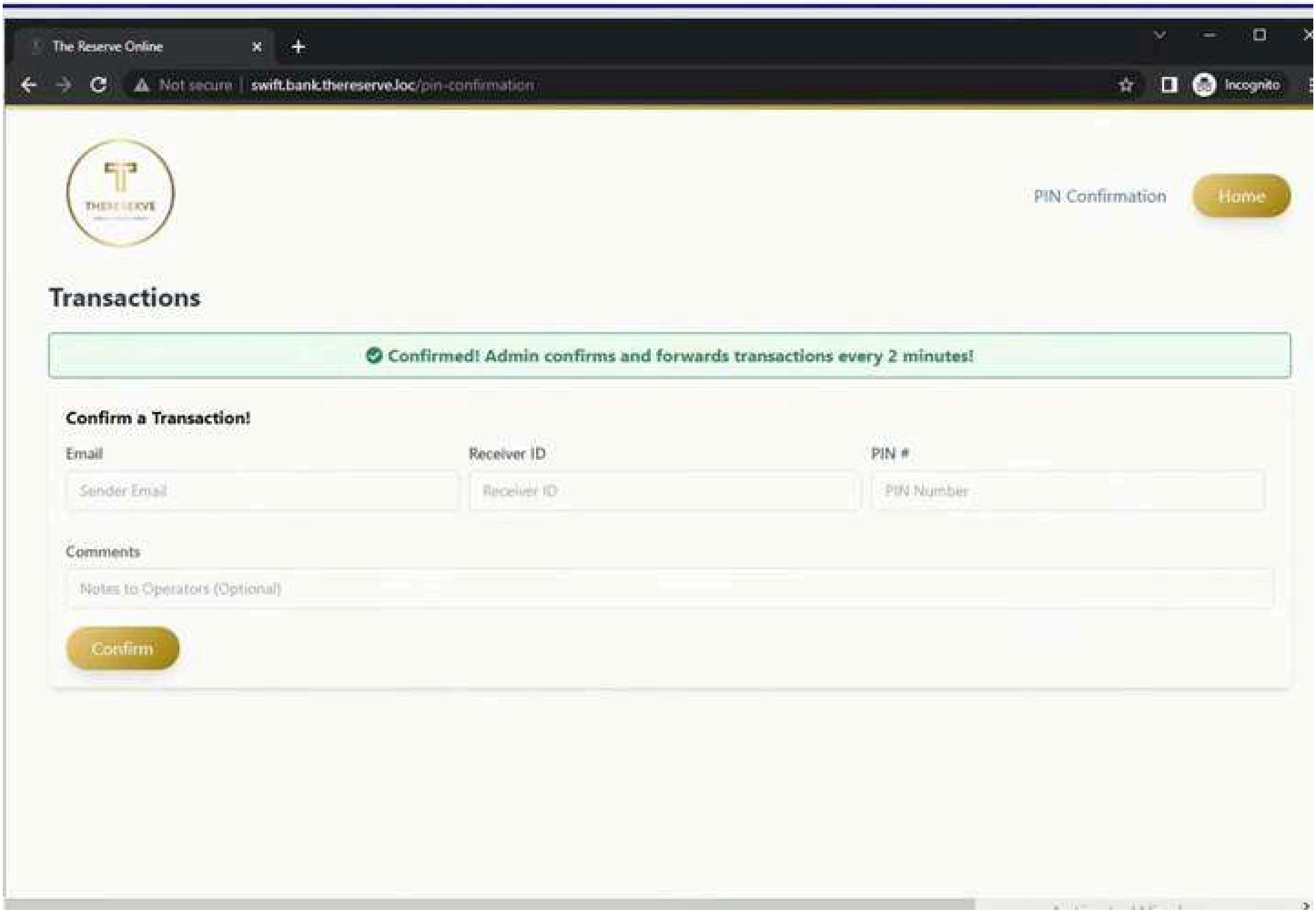

Login attempt to SWIFT Capturer portal using a valid account.



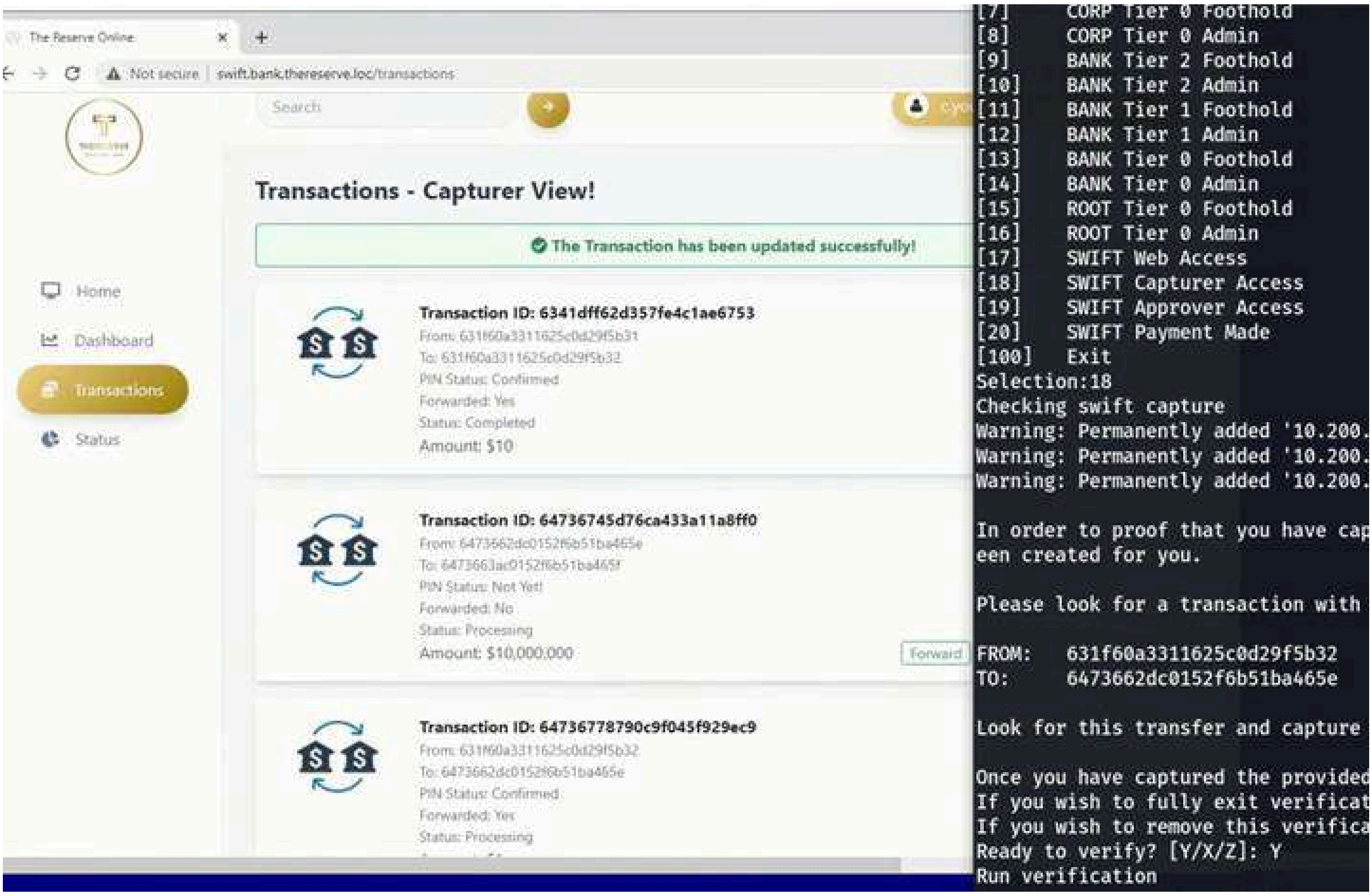

Transactions overview in the Capturer portal showing $10M transaction pending forwarding.



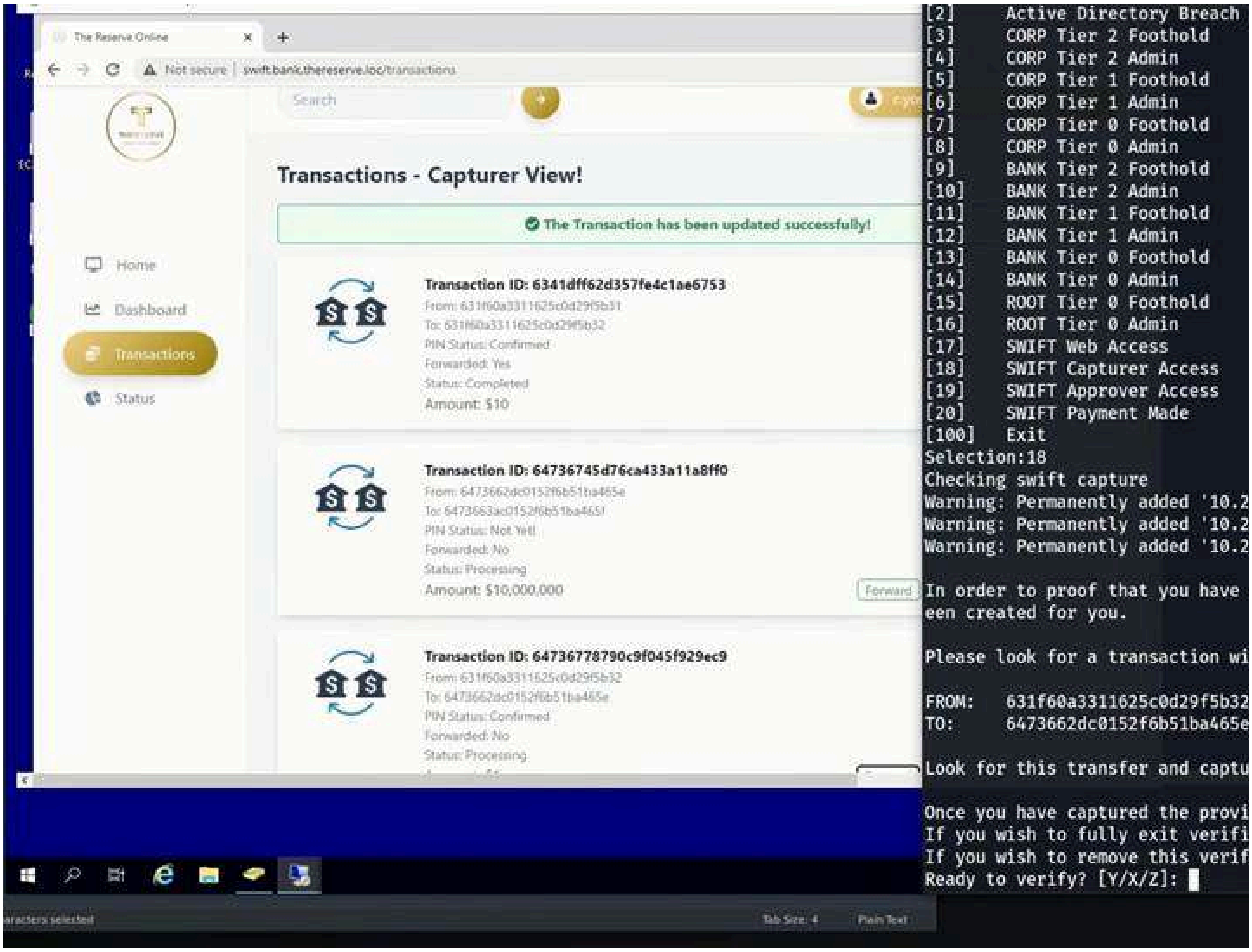

Login screen indicating incorrect credentials for a capturer account.



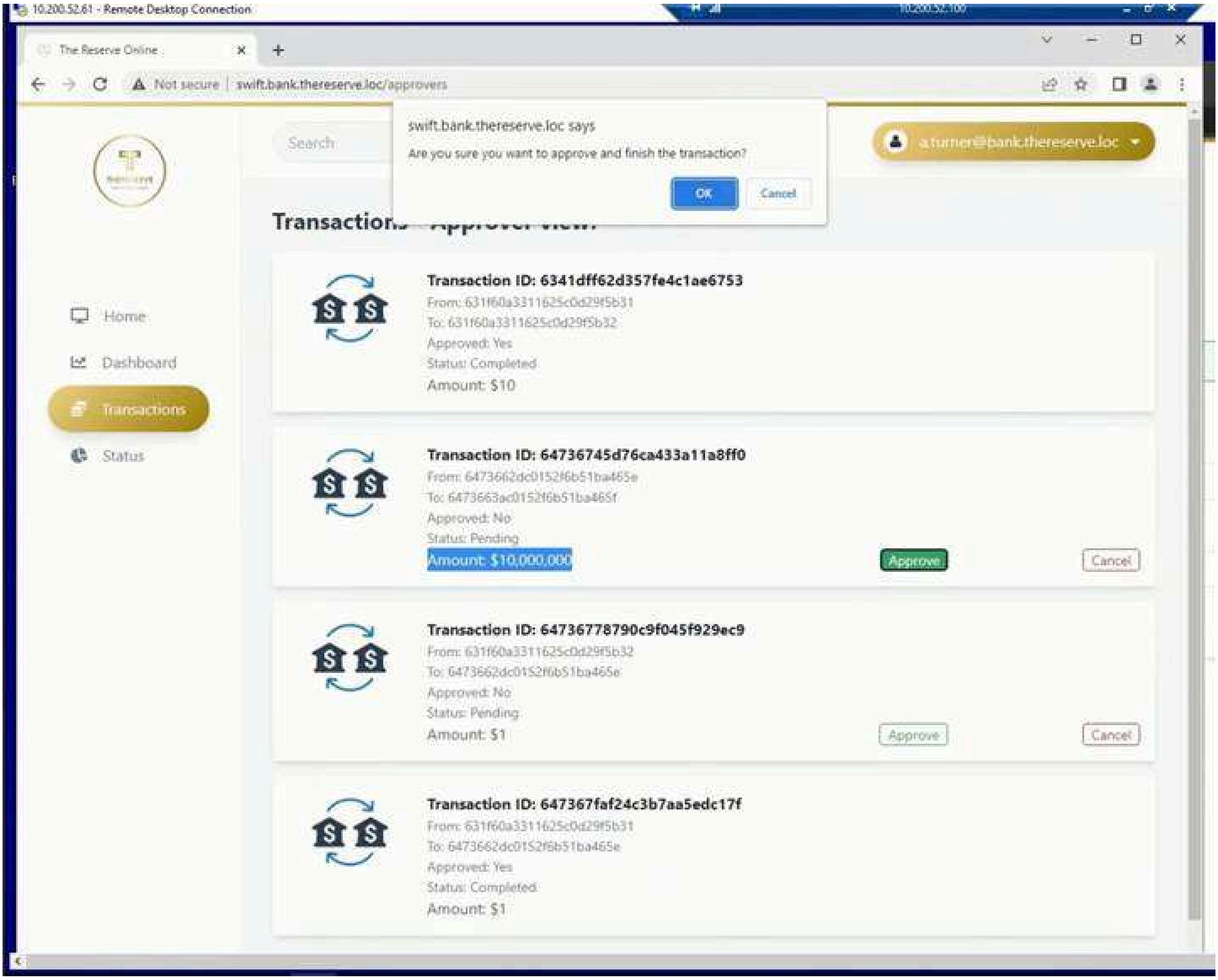

PIN confirmation portal for the transaction, accessible by the admin.



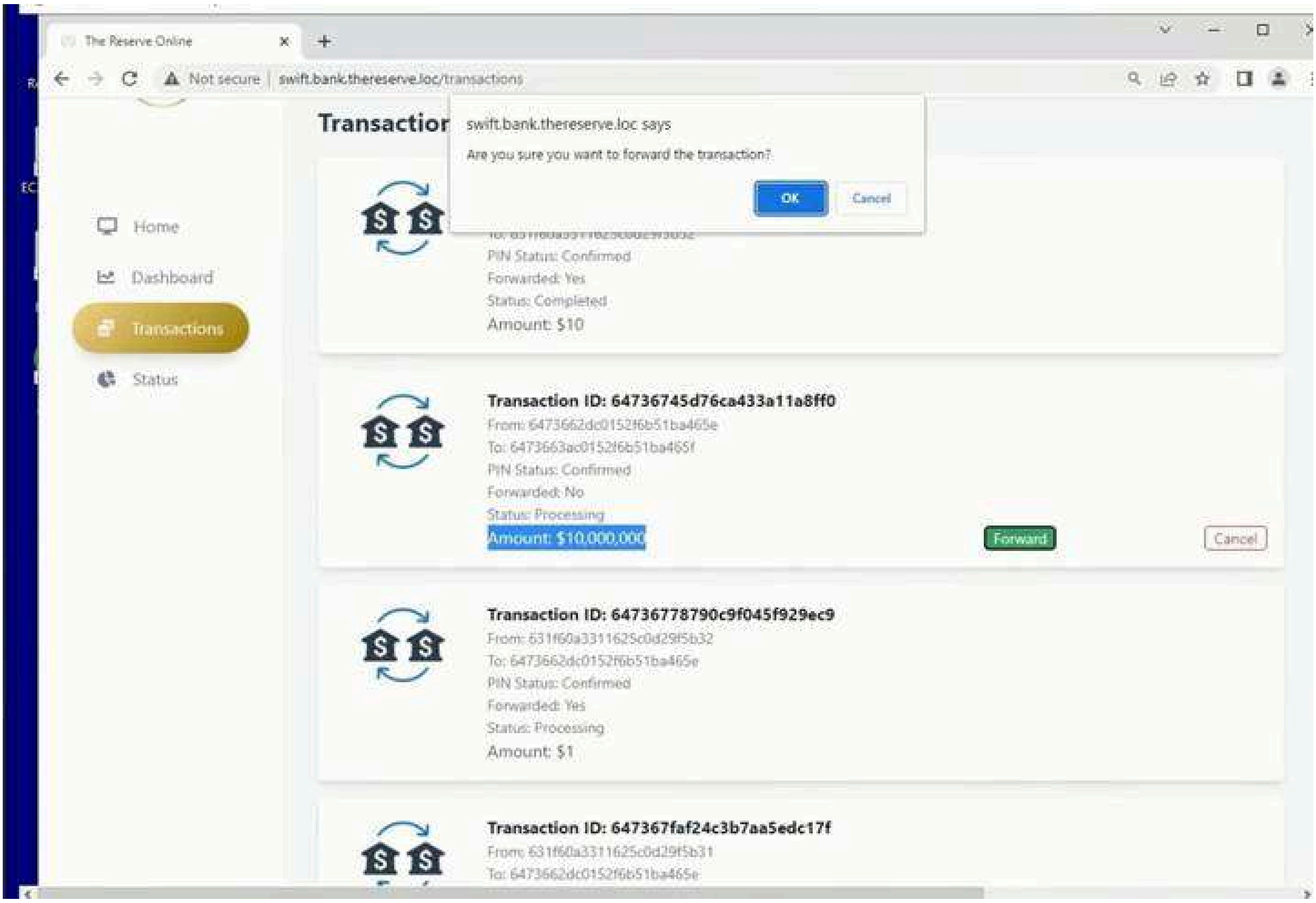

Approver panel showing a $10M transaction pending approval.



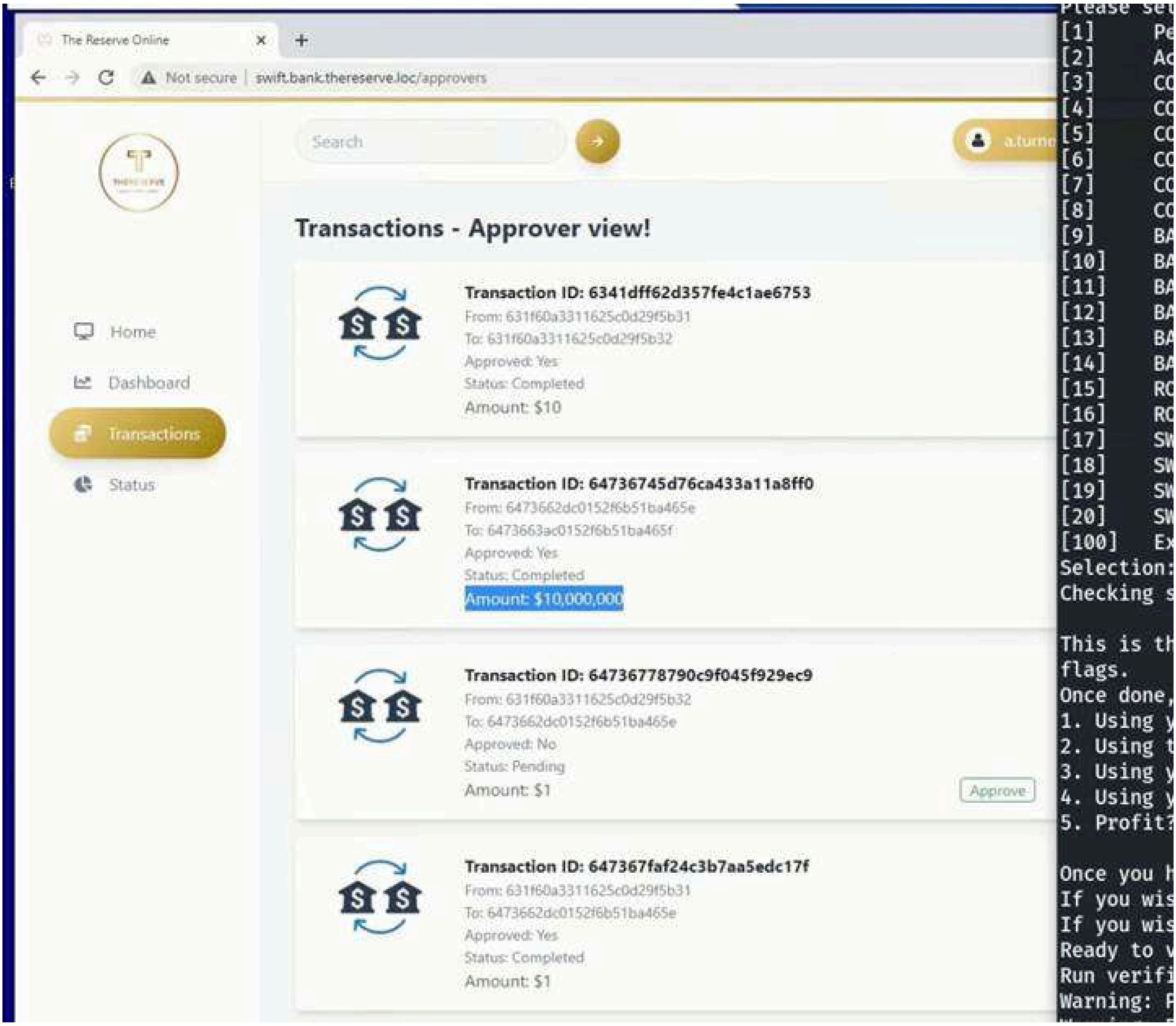

Successful capture validation shown alongside terminal interaction.



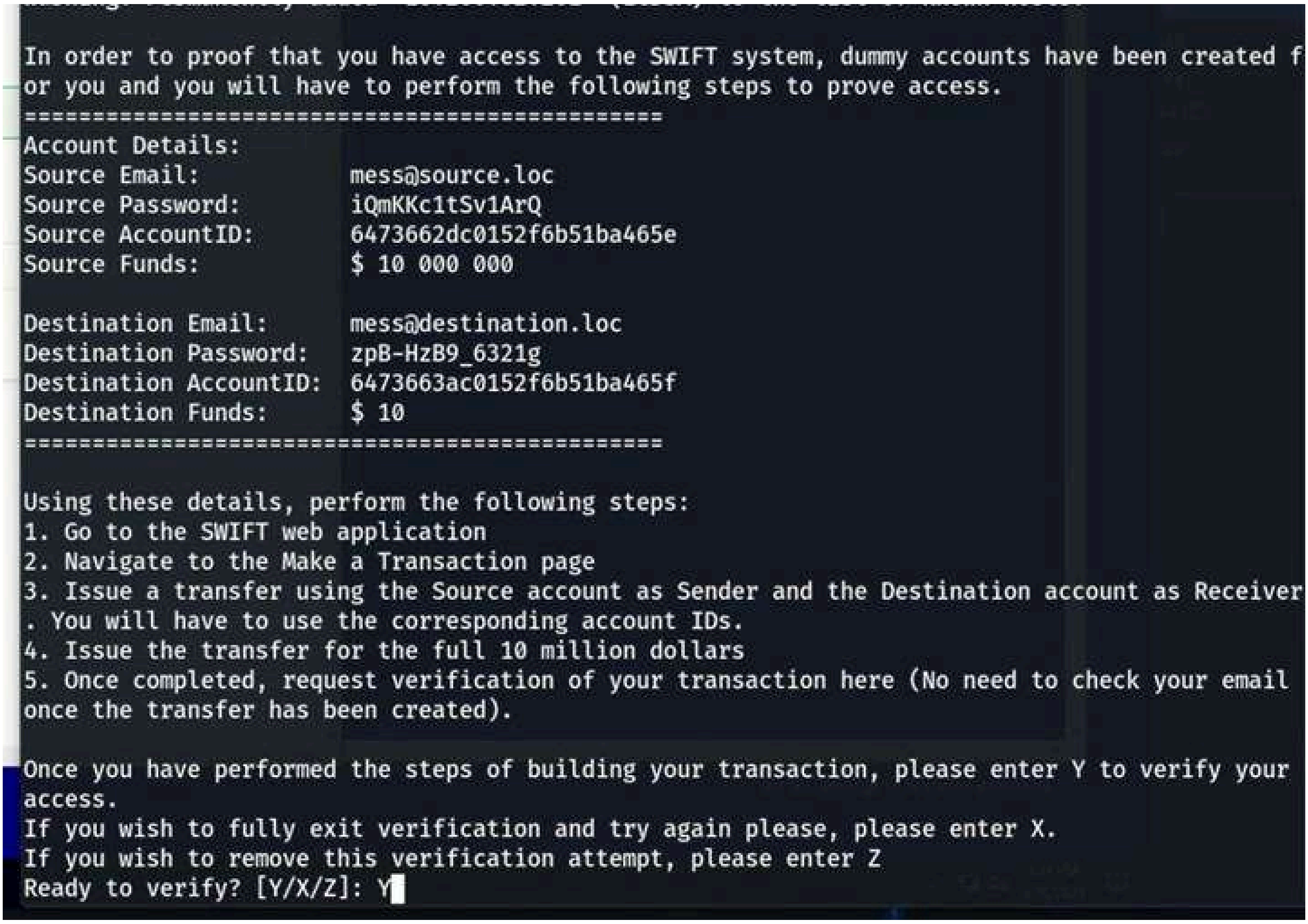

Final SWIFT transaction approval prompt under approver account.




Transaction successfully approved and marked as completed.




Instructions showing source/destination SWIFT accounts and credentials to perform transaction.




Verifying the completion of transaction through CLI to prove control over SWIFT.

With the successful transaction forwarding and approval, the final step involved submitting the flag for 'SWIFT Payment Made' in the TryHackMe interface. The verification confirmed all required steps were completed correctly, signaling the successful compromise and manipulation of the SWIFT system