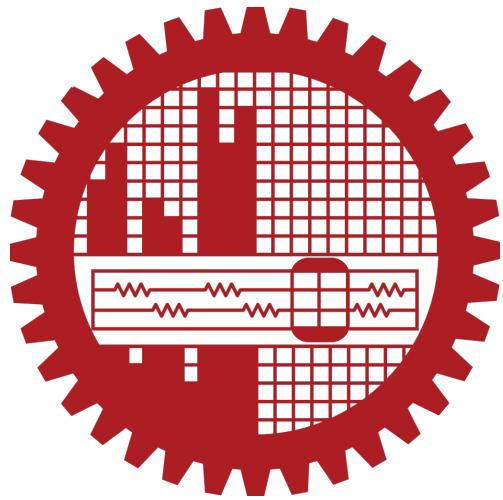


CSE 406

Term Project:

Port Scanner Tool



Md. Shanjinur Islam
Level 4 Term 1

Student ID: 1505066

Group: 03

Tool Introduction

I have implemented **Port Scanner Tool** using Python Programming Language. To run it, you will need only one file which is **main.py**. I have used positional and optional arguments for better utilization of the tool. In the following section, I will provide each features that can be done with my Port Scanner Tool.

Steps of using the tool

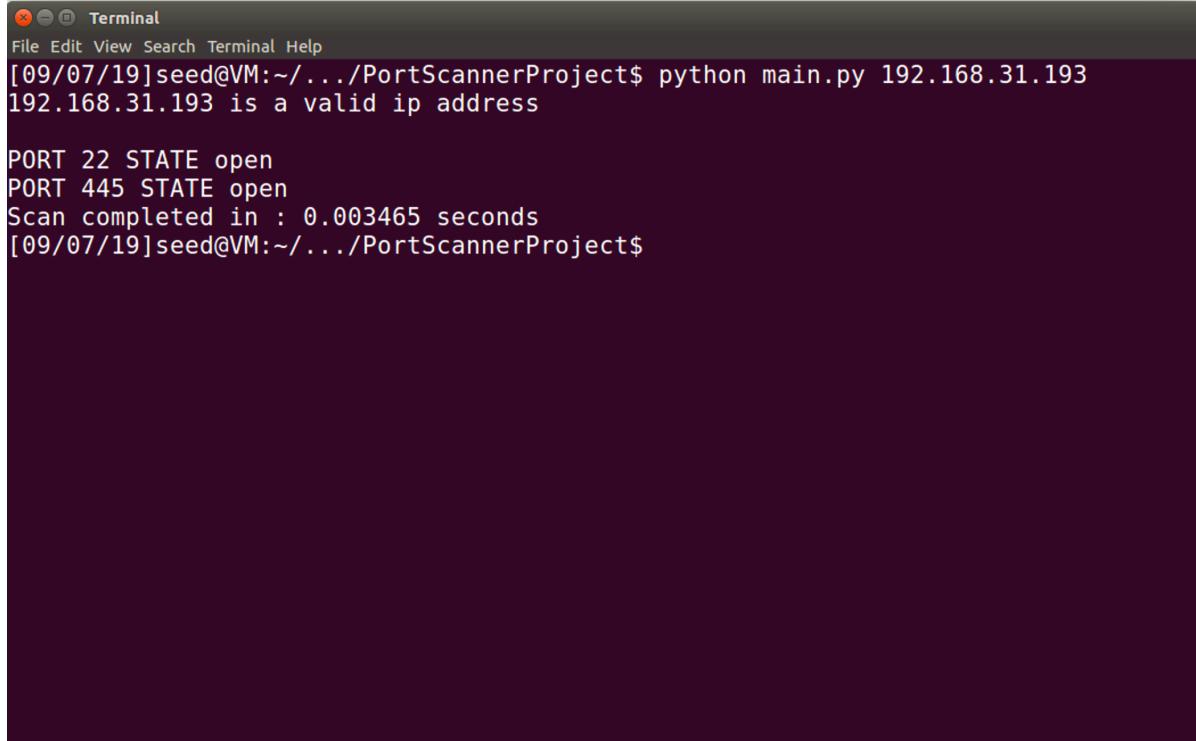
The tool is very easy to operate. The basic command is to:

```
python main.py {ip_address/url}
```

When no other optional arguments is provided it basically checks for open ports in usual Common port which are 1, 5, 7, 18, 20, 21, 22, 23, 25, 29, 37, 42, 43, 49, 53, 69, 70, 79, 80, 103, 108, 109, 110, 115, 118, 119, 137, 139, 143, 150, 156, 161, 179, 190, 194, 197, 389, 396, 443, 444, 445, 458, 546, 547, 563, 569, 1080

For IP Address :

```
python main.py 192.168.31.193 (IP Address of a PC in my network)
```



A screenshot of a terminal window titled "Terminal". The window shows the command `python main.py 192.168.31.193` being run. The output indicates that the IP address is valid and provides a scan summary:

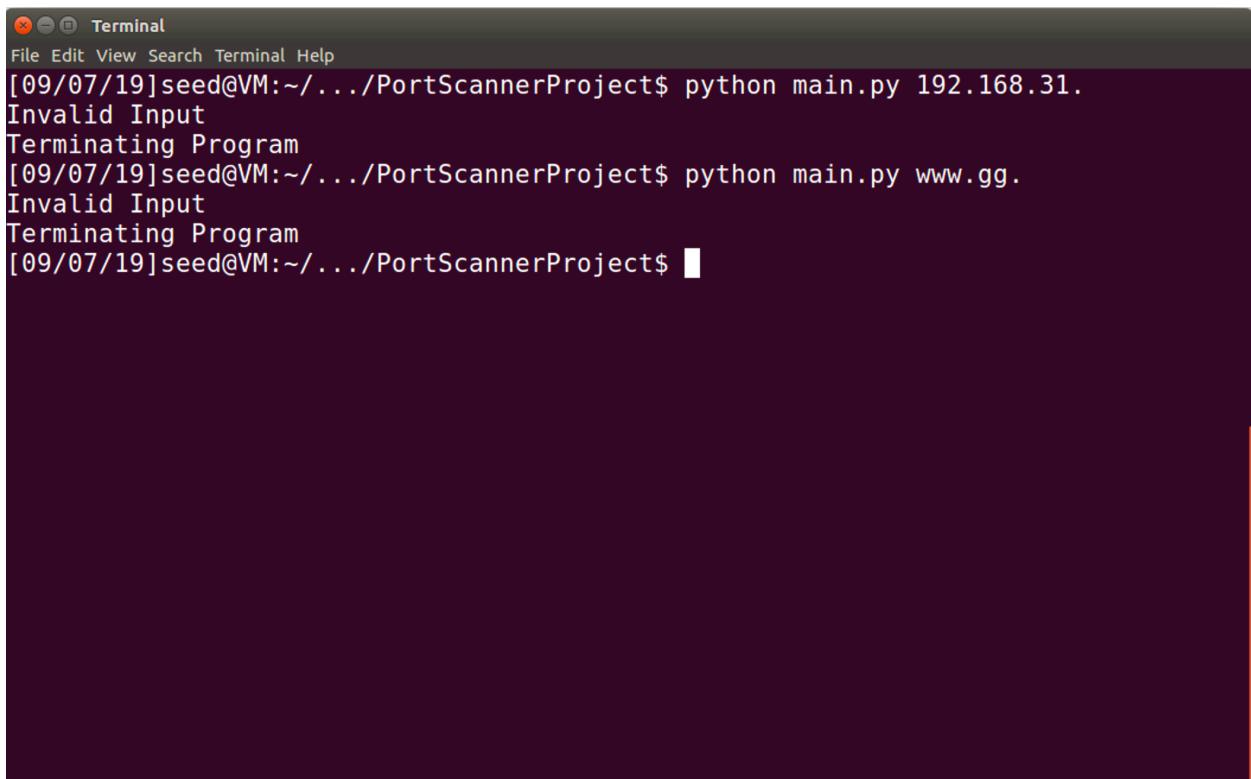
```
Terminal
File Edit View Search Terminal Help
[09/07/19]seed@VM:~/.../PortScannerProject$ python main.py 192.168.31.193
192.168.31.193 is a valid ip address

PORT 22 STATE open
PORT 445 STATE open
Scan completed in : 0.003465 seconds
[09/07/19]seed@VM:~/.../PortScannerProject$
```

Here, the positional argument I used is target IP address or URL. There is a validity checker whether the provided IP or URL is correct or not.

Here I provided two inputs where in first case invalid url and in second case invalid url.

In both cases it returns invalid input and terminates the program.



A screenshot of a terminal window titled "Terminal". The window has a dark background and a light-colored title bar. The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The main area of the terminal shows the following text:

```
[09/07/19]seed@VM:~/.../PortScannerProject$ python main.py 192.168.31.  
Invalid Input  
Terminating Program  
[09/07/19]seed@VM:~/.../PortScannerProject$ python main.py www.gg.  
Invalid Input  
Terminating Program  
[09/07/19]seed@VM:~/.../PortScannerProject$ █
```

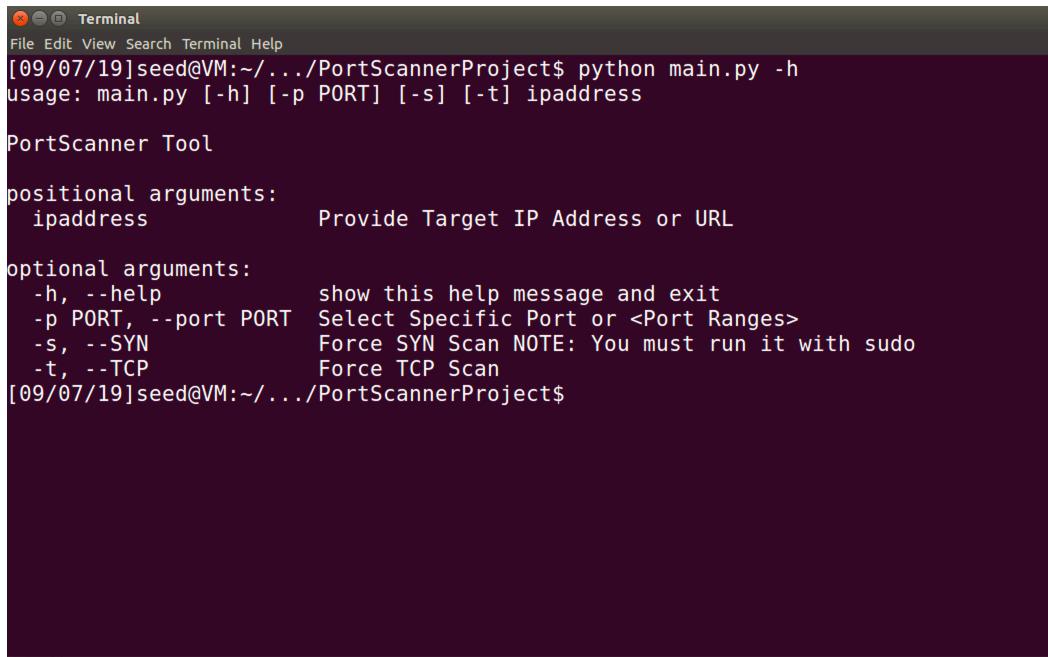
Scan Types:

Here I implemented two versions of Port Scanning as I mentioned in design report. They are:

- TCP Scanning
- SYN Scanning

For better user friendly experience I have provided a **help** screen to understand how to provide inputs.

In terminal write : **python main.py -h**



```
[09/07/19]seed@VM:~/.../PortScannerProject$ python main.py -h
usage: main.py [-h] [-p PORT] [-s] [-t] ipaddress

PortScanner Tool

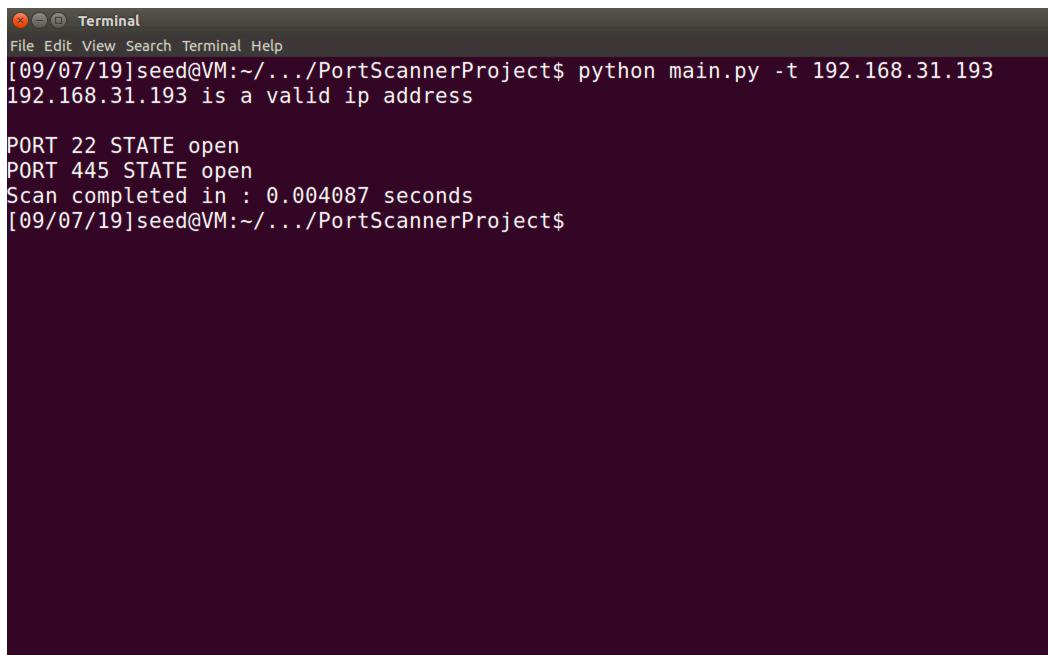
positional arguments:
  ipaddress            Provide Target IP Address or URL

optional arguments:
  -h, --help           show this help message and exit
  -p PORT, --port PORT Select Specific Port or <Port Ranges>
  -s, --SYN            Force SYN Scan NOTE: You must run it with sudo
  -t, --TCP            Force TCP Scan
[09/07/19]seed@VM:~/.../PortScannerProject$
```

TCPScan

Although default Scan type is set TCP Scanning but you can force it by following the command.

In terminal write : **python main.py -t 192.168.31.193**



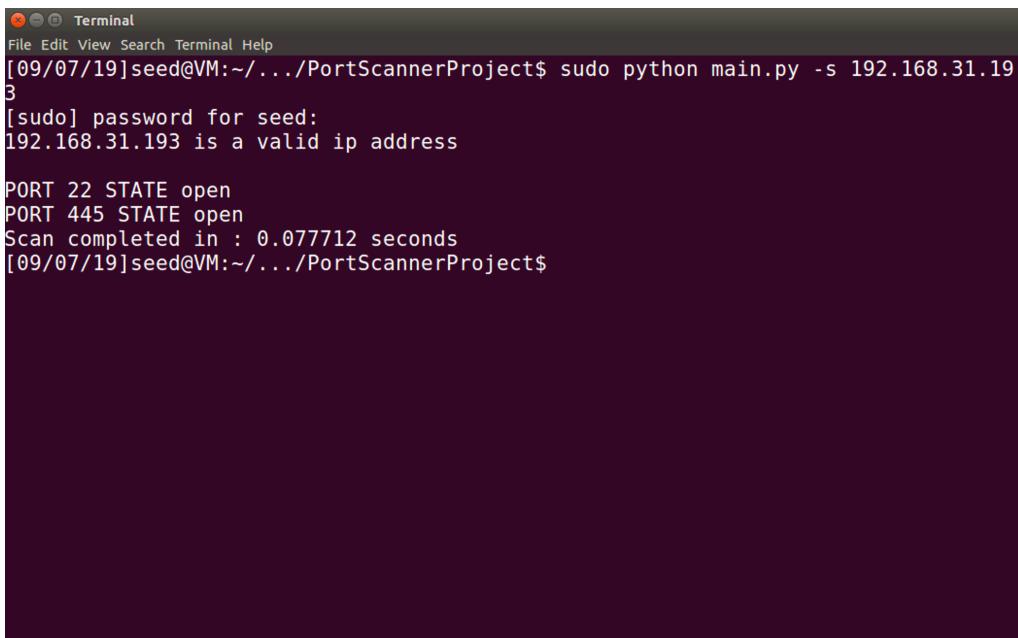
```
[09/07/19]seed@VM:~/.../PortScannerProject$ python main.py -t 192.168.31.193
192.168.31.193 is a valid ip address

PORT 22 STATE open
PORT 445 STATE open
Scan completed in : 0.004087 seconds
[09/07/19]seed@VM:~/.../PortScannerProject$
```

SYN Scan

Although default Scan type is set TCP Scanning but you can force it by following the command.

In terminal write : **sudo python main.py -s 192.168.31.193**



A screenshot of a terminal window titled "Terminal". The window shows the following command and its output:

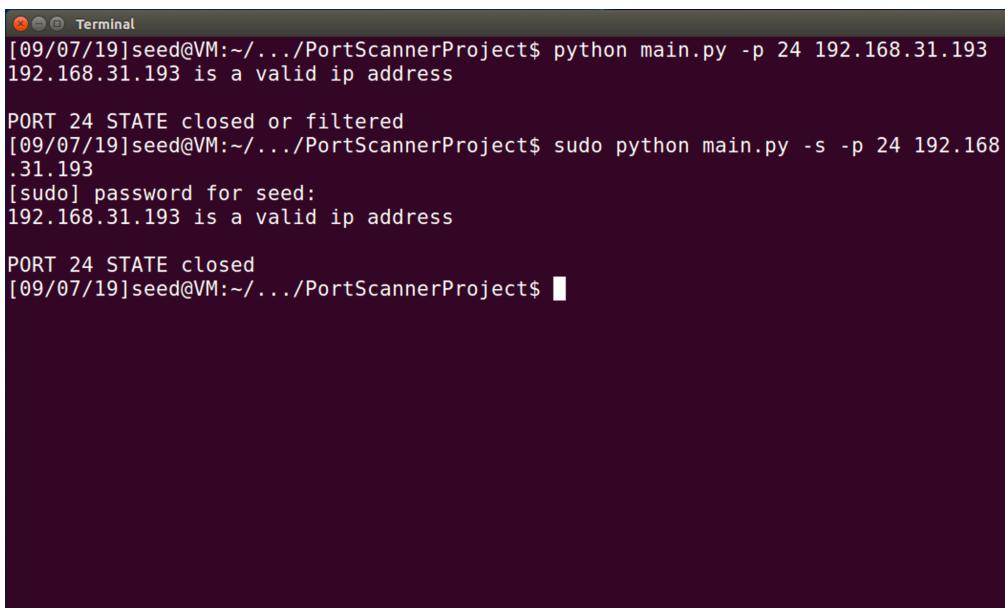
```
[09/07/19]seed@VM:~/.../PortScannerProject$ sudo python main.py -s 192.168.31.193
[sudo] password for seed:
192.168.31.193 is a valid ip address

PORT 22 STATE open
PORT 445 STATE open
Scan completed in : 0.077712 seconds
[09/07/19]seed@VM:~/.../PortScannerProject$
```

Specific Port or Range of Ports Scan

To scan a specific port write **python main.py -p 22 192.168.31.193**.

It will show whether the port is opened or closed.



A screenshot of a terminal window titled "Terminal". The window shows the following command and its output:

```
[09/07/19]seed@VM:~/.../PortScannerProject$ python main.py -p 22 192.168.31.193
192.168.31.193 is a valid ip address

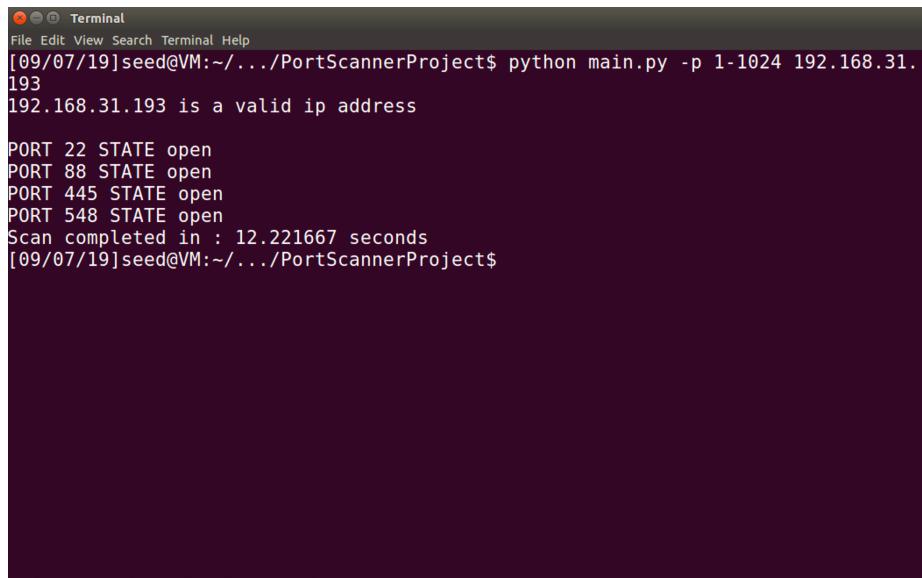
PORT 22 STATE open

[09/07/19]seed@VM:~/.../PortScannerProject$ sudo python main.py -s -p 24 192.168.31.193
[sudo] password for seed:
192.168.31.193 is a valid ip address

PORT 24 STATE closed
[09/07/19]seed@VM:~/.../PortScannerProject$
```

To scan range of ports write **python main.py -p 1-1024 192.168.31.193**

Here 1-1024 means range from 1-1024



```
Terminal
File Edit View Search Terminal Help
[09/07/19]seed@VM:~/.../PortScannerProject$ python main.py -p 1-1024 192.168.31.193
192.168.31.193 is a valid ip address

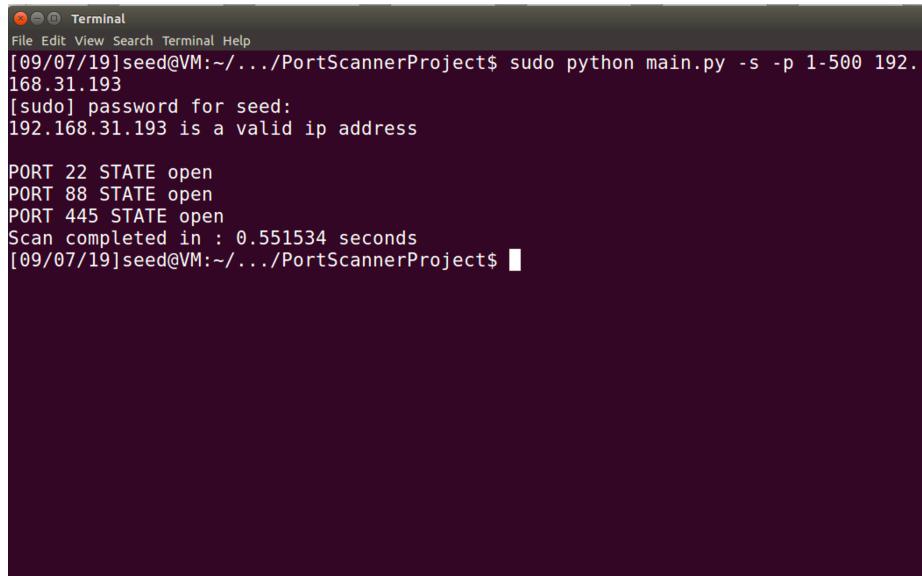
PORT 22 STATE open
PORT 88 STATE open
PORT 445 STATE open
PORT 548 STATE open
Scan completed in : 12.221667 seconds
[09/07/19]seed@VM:~/.../PortScannerProject$
```

Using multiple options

You can use multiple options with a single command. For example:

You want to do SYN Scanning over a range of ip from 1-500. So to perform this scan you have to write in terminal:

sudo python main.py -s -p 1-500 192.168.31.193



```
Terminal
File Edit View Search Terminal Help
[09/07/19]seed@VM:~/.../PortScannerProject$ sudo python main.py -s -p 1-500 192.168.31.193
[sudo] password for seed:
192.168.31.193 is a valid ip address

PORT 22 STATE open
PORT 88 STATE open
PORT 445 STATE open
Scan completed in : 0.551534 seconds
[09/07/19]seed@VM:~/.../PortScannerProject$
```

Wireshark Observations

For a single port TCP scan , **python main.py -p 22 192.168.31.193**, Wireshark observation is given below:

The screenshot shows the Wireshark interface with the title bar "Wireshark · Packet 17 · wireshark_enp0s5_20190907211545_p3x912". The packet details pane displays the following information for Frame 17:

- Frame 17: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface
- Ethernet II, Src: Parallel_c6:14:83 (00:1c:42:c6:14:83), Dst: 88:e9:fe:67:d4:3a
- Internet Protocol Version 4, Src: 192.168.31.66, Dst: 192.168.31.193
- Transmission Control Protocol, Src Port: 43400, Dst Port: 22, Seq: 144925259, Ac

Source Port: 43400
Destination Port: 22
[Stream index: 2]
[TCP Segment Len: 0]
Sequence number: 144925259
Acknowledgment number: 1402229071
Header Length: 32 bytes
Flags: 0x010 (ACK)
Window size value: 229
[Calculated window size: 29312]
[Window size scaling factor: 128]
Checksum: 0xc07a [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]

The hex dump pane shows the following bytes:

| | | | |
|------|-------------------------|-------------------------|-------------------------|
| 0000 | 88 e9 fe 67 d4 3a 00 1c | 42 c6 14 83 08 00 45 00 | ...g.:.. B.....E. |
| 0010 | 00 34 26 1f 40 00 40 06 | 54 51 c0 a8 1f 42 c0 a8 | .4&.@. TQ...B.. |
| 0020 | 1f c1 a9 88 00 16 08 a3 | 62 4b 53 94 51 4f 80 10 | bKS.Q0.. |
| 0030 | 00 e5 c0 7a 00 00 01 01 | 08 0a 00 16 44 54 2f c8 | ...z....DT/. kw |
| 0040 | 6b 77 | | |

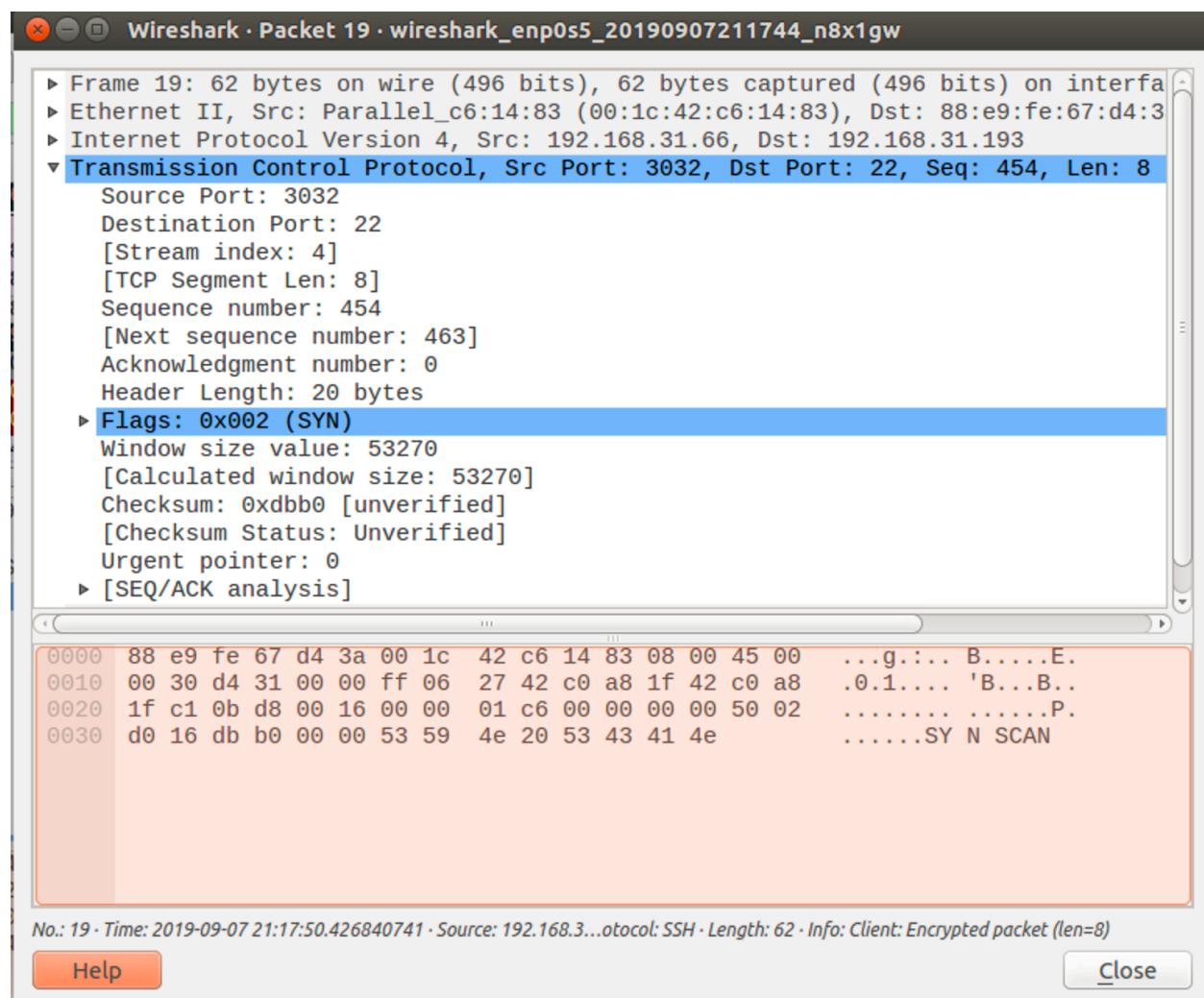
No.: 17 · Time: 2019-09-07 21:15:48.371894518 · Source: 192...ck=1402229071 Win=29312 Len=0 TSval=1459284 TSecr=801663863

[Help](#) [Close](#)

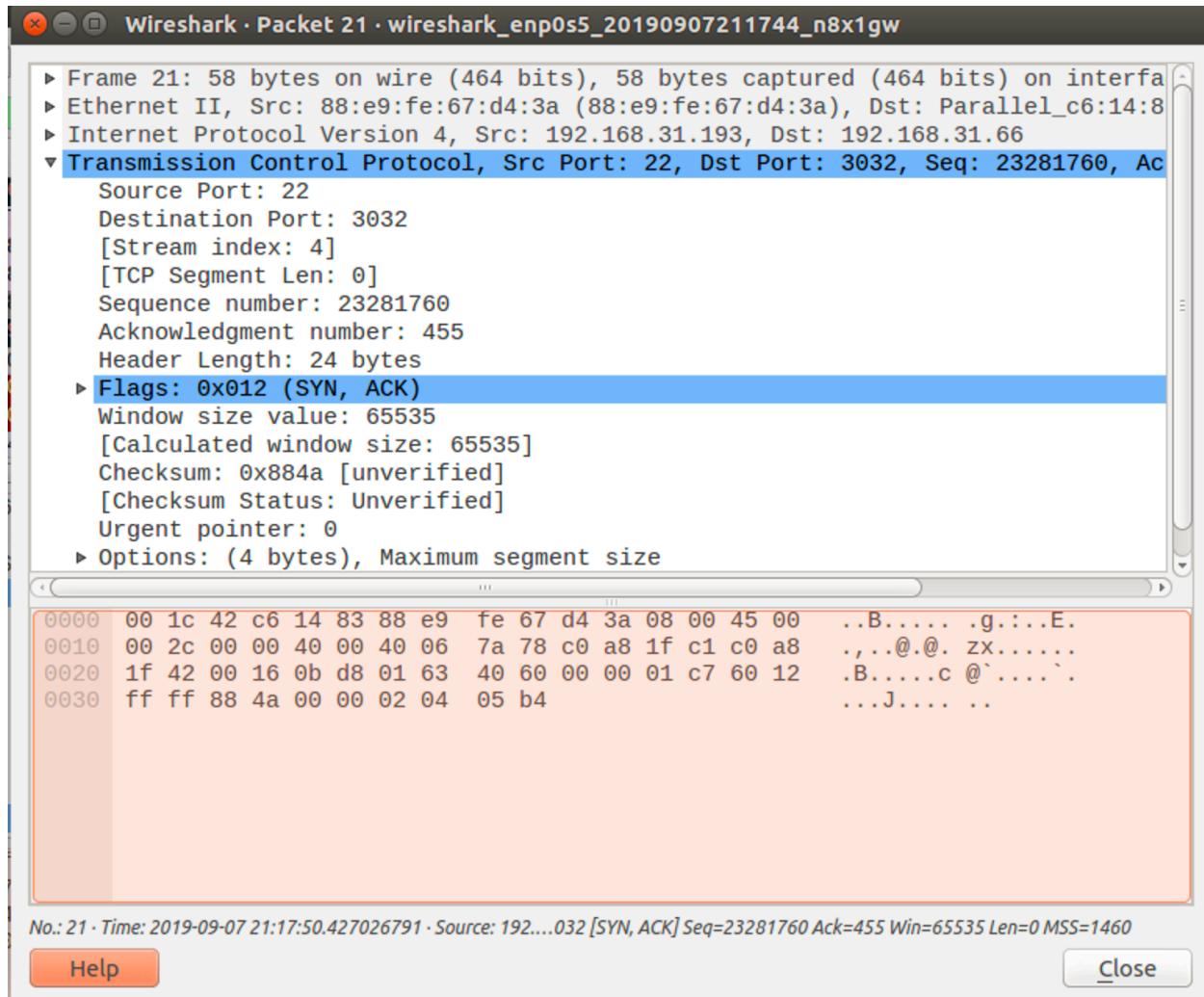
For a single port SYN scan , **sudo python main.py -p 22 192.168.31.193**, Wireshark observation is given below:

SYN Scan is a three step process.

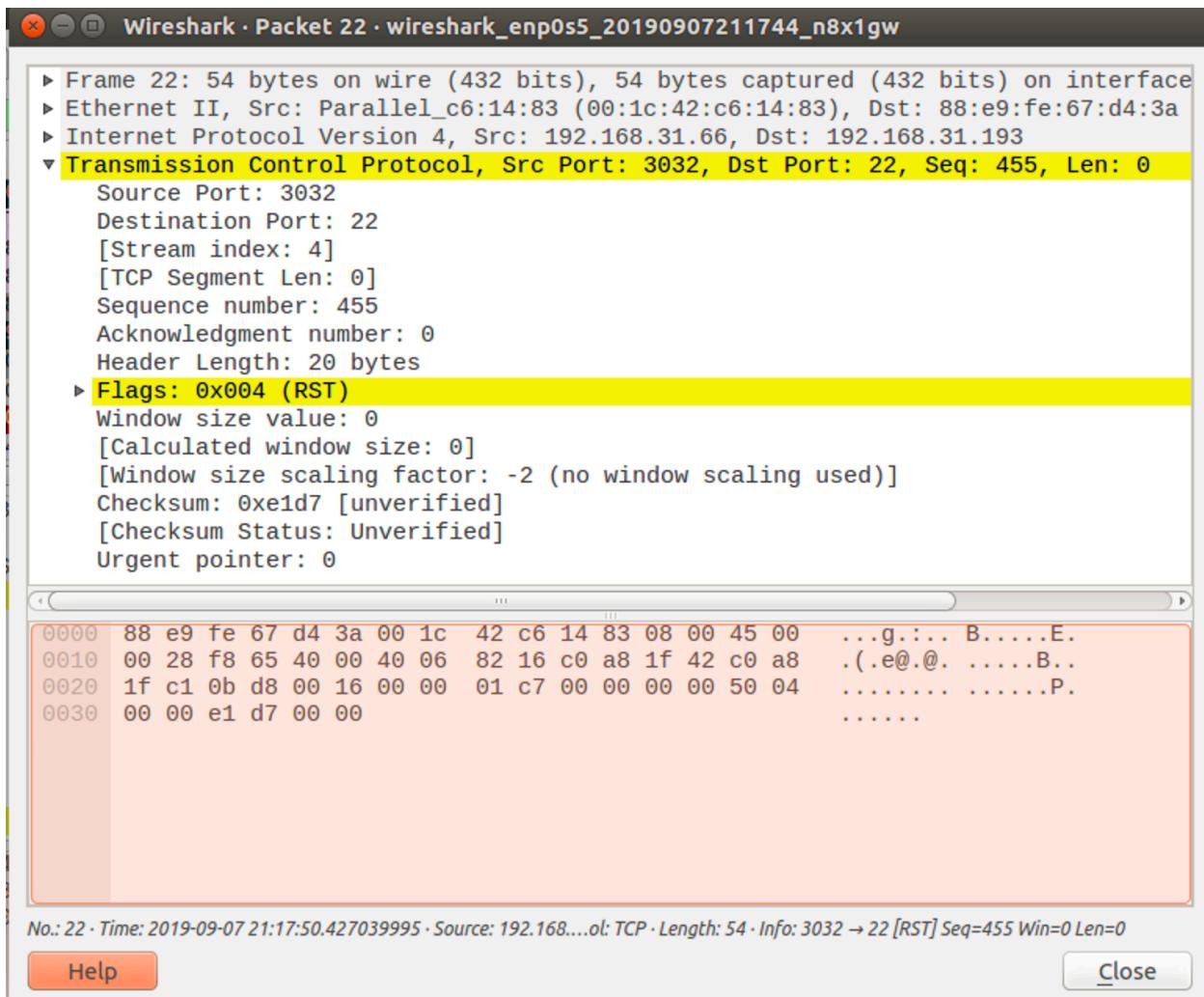
First Step is send a packet with SYN flag on.



Second Step is reception of a packet from destination with SYN and ACK flag on.

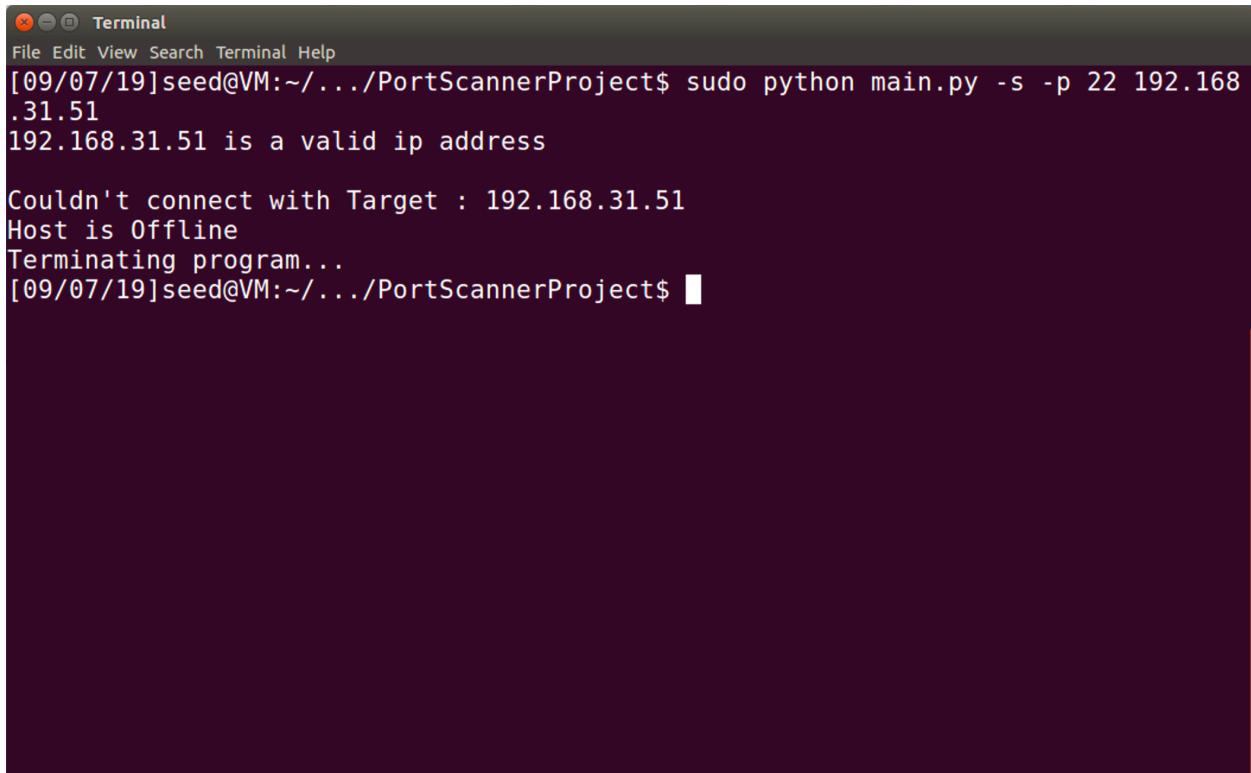


Third and Final Step is send a packet with RST flag on. The Screenshot is provided in next Page



Extra Features

If host is not connected to the network but ip address is provided then port scanner will simply show that the host is down and terminate the program.



A screenshot of a terminal window titled "Terminal". The window has a dark background and light-colored text. The text shows the following command and its output:

```
[09/07/19]seed@VM:~/.../PortScannerProject$ sudo python main.py -s -p 22 192.168.31.51
192.168.31.51 is a valid ip address
Couldn't connect with Target : 192.168.31.51
Host is Offline
Terminating program...
[09/07/19]seed@VM:~/.../PortScannerProject$ █
```

Conclusion

The tool can also be expanded to various features like a range of IP or UDP Scanning which will be implemented in future hopefully.