

CS 3205 – Introduction to Computer Networks: Jan-May 2020

Assignment 5 (Macro Assignment 3)

Playing around with data packets using Scapy

Group assignment (maximum of 2 people)

Assignment Deadline: No penalty 3rd May 2020, 15 % penalty 5th May 2020 30% penalty 7th May 2020 (Final Hard Deadline). Total of 100 marks, and possible 30 bonus marks.

Assignment weightage – 7.5 marks out of 100 for final grading.

Aim of the Assignment: Understand the versatile python based tool Scapy, using which construct ICMP, IP, TCP, UCP, and DNS packets, send and receive the packets. Materials to understand Scapy: Internet has abundant material on Scapy. The necessary and minimal needs that should help you accomplish the goal of the assignment are placed below along with the materials uploaded in the moodle.

Section 1: Scapy Reference Resources:

Video Tutorial: Youtube video titled “Create Packets from Scratch with Scapy” by NullByte at <https://www.youtube.com/watch?v=yD8qrP8sCDs> is a short video to get started to create, send and receive packets..

The Youtube video titled “Introduction to Scapy” by Kurtis Schlienger at <https://www.youtube.com/watch?v=KzeD3GCZGdI> is also a very detailed source. Covers additional aspects. You can view the above directly or the same videos are uploaded to the moodle, can be downloaded and viewed. These Tutorials will provide the necessary background to construct packets, to send and receive them.

Philippe BIONDI is the inventor/author of the Scapy tool, and today Scapy comes across a great tools for performing network security related actions. The PDF with title Network packet forgery with Scapy – 2005 dated, is a good resource. Please refer in this PDF slides – 30 to 56 under Quick overview.

Couple of cheat sheets with scapy commands to construct, send and receive packets are also placed in moodle.

Section 2: Scapy Sources:

Scapy source (Refer Figure 1): The primary source is <https://scapy.net/index>.



Figure 1. Scapy home page

There are several options to download scapy, and please use the method that you would feel convenient (Refer Figure 2).

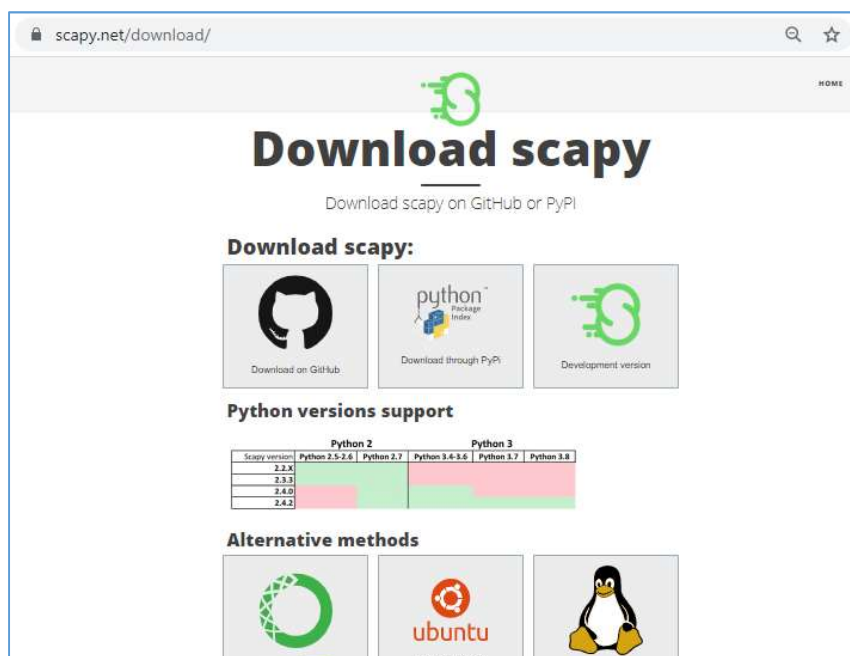


Figure 2. Scapy Download options

There are several options to download scapy, and please use the method that you would feel convenient (Refer Figure 2).

The latest Scapy documentation is available at <https://scapy.readthedocs.io/en/latest/>

Section 3: Assignment Steps:

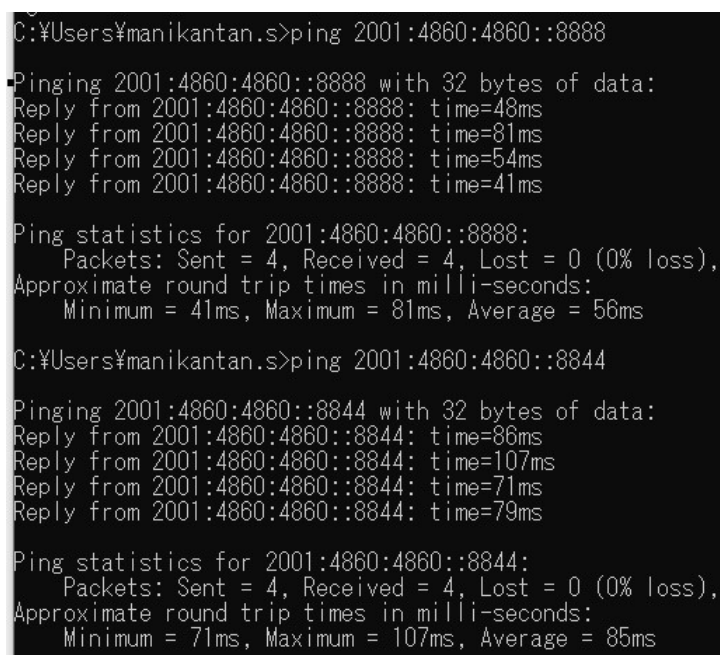
This assignment is to enable you to construct ICMP (Echo Request, Echo Reply) packets, IP Packets, TCP and UDP Packets.

In the entire assignment, the exchange is done across two systems – your's and your partners. You can determine the Public IP address of your system by

Section 3.1: Step 1: Determine the Public IP address of your system, and your Partner's system. You can determine the unique Public IP address associated with your system by using "What is my IP" in google search. <https://whatismyipaddress.com/> and <https://www.whatismyip.com/> are two example sources to get your Public IP addresses.

Please note, your Public IP address can be an IPv4 address, or an IPv6 address. (If you are connected via 4G Mobile data cards, you can possibly be assigned IPv6 address).

If you see a IPv6 Public IP address assigned, to become familiarized you can execute PING (ICMPv6) to the well known GOOGLE DNS Server IPv6 Address at 2001:4860:4860::8888. Or 2001:4860:4860::8844. (Refer Figure 3). Sample Capture of the packets on Wireshark for the IPv6 Google DNS Server is uploaded to moodle as well for packet structure reference.



```
C:\Users\manikantan.s>ping 2001:4860:4860::8888

Pinging 2001:4860:4860::8888 with 32 bytes of data:
Reply from 2001:4860:4860::8888: time=48ms
Reply from 2001:4860:4860::8888: time=81ms
Reply from 2001:4860:4860::8888: time=54ms
Reply from 2001:4860:4860::8888: time=41ms

Ping statistics for 2001:4860:4860::8888:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 41ms, Maximum = 81ms, Average = 56ms

C:\Users\manikantan.s>ping 2001:4860:4860::8844

Pinging 2001:4860:4860::8844 with 32 bytes of data:
Reply from 2001:4860:4860::8844: time=86ms
Reply from 2001:4860:4860::8844: time=107ms
Reply from 2001:4860:4860::8844: time=71ms
Reply from 2001:4860:4860::8844: time=79ms

Ping statistics for 2001:4860:4860::8844:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 71ms, Maximum = 107ms, Average = 85ms
```

Figure 3. Ping (ICMPv6) To Well Known Google DNS Servers - IPv6 Address

Output expected: Screen shot at PS1 and Screen Shot at PS2, indicating the respective Public IP Address.

Credits: 10 Marks

Section 3.2: Step 2: ICMP echo request sent to the partner systems.

From Partner System 1 (PS1) construct Ping (ICMP echo request) message destined to Partner System 2 (PS2), and send 5 Ping messages. In the normal scenario, the Partner system 2's OS will respond if the ICMP echo request packet is correctly constructed. Similarly Send from PS2 to PS1.

Output expected: The scapy command used to construct the packets. PS1-Screenshot for sending and receive, PS2-Screen shot for sending and receiving. Two Wireshark Packets with only the 10 ICMP packets, at the respective ends. (Each Wireshark to have only the ICMP filtered packets).

Credits: 20 Marks (3 marks for each scapy packet construction command (list the text command or screen shot), 3 marks for each screen showing the PING exchange, 4 marks for each Wireshark file.) Please place suitable caption or section header to know the output for the section.

Section 3.3: Step 3: ICMP echo request and reply sent to the partner systems.

From Partner System 1 (PS1) construct Ping (ICMP echo request) message destined to Partner System 2 (PS2), and send 5 Ping messages. In the normal scenario, the Partner system 2's OS will respond if the ICMP echo request packet is correctly constructed. Similarly Send from PS2 to PS1.

Now enable reception of the packets at the scapy, i.e. sniff or receive. From the received ICMP echo request, construct the ICMP Echo response using scapy and send response back. For self generated ICMP response, in the payload, place a marker "<ROLL No 1> <ROLL No 2>" (In Wireshark, you should see two responses for the same request, one auto generated by the system and another generated by you using scapy).

Output expected: PS1-Screenshot for send and receive, PS2-Screen shot for send and receive. Two Wireshark Packets with only the 15 ICMP packets, at the respective ends. (Each Wireshark to have only the ICMP filtered packets).

Credits: 20 Marks (1 marks for each scapy packet (ICMP Request) construction command, 4 marks each for the scapy packet (ICMP Response) construction (list the text command or screen shot), 1 marks for each screen showing the PING exchange, 4 marks for each Wireshark file.) Please place suitable caption or section header to know the output for the section.

Additional Ref: <https://www.hackingarticles.in/understanding-guide-icmp-protocol-wireshark/>

Section 3.4: Step 4: UDP DNS Exchange with the Partner System.

First a PS1 assume PS2 is a DNS Server. Send a DNS Query for a "A" Record associated with www.google.com. At PS2 receive the DNS Query and construct a suitable DNS response for the requested DNS query from PS1 and send response back. (Repeat this from PS2 to PS1. In this case let the DNS query be made w.r.to www.cse.iitm.ac.in)

For reference, you could use nslookup from your system invoke the nslookup for the A record query to the system, obtain the response, capture them via Wireshark. Use the packet contents as sample to construct responses from PS2 to PS1 and vice versa.

Output expected: PS1-Screenshot for normal nslookup, and for sending the DNS query packet after construction using scapy send, and reception. Similarly at PS-2. Wireshark captures for the exchanges, 4 packets at each end. One set (2 packets) will be a query and response with normal (associated DNS server) and one set (2 packets) will be a query to and response from the partner system

Credits: 20 Marks (3 marks for each scapy packet DNS Request and DNS Response construction command, 1 marks for each screen showing the PING exchange, 3 marks for each Wireshark file.) Please place suitable caption or section header to know the output for the section.

Section 3.5: Step 5: TCP Based File Exchange with the Partner System.

<BACKGROUND for Step5>

To emulate a TCP based data exchange between systems we first utilize the help of “iperf” or “iperf3” a tool that enables to understand the bandwidth capacity between systems.

A TCP server is initiated at one host and then a TCP client is initiated to connect to the server. Data flows from Client to the server. At the system (with IP as “SERVER-IP”) acting as server, and waiting on tcp port 5003 (for example) by using the “iperf3 -s -p 5003” command the server is initiated. At the system acting as a Client, to reach the sever waiting on tcp port 5003 (for example) using the command “iperf3 -c <SERVER-IP> -p 5003” Client is initiated, and it transmits data, by default for 10 seconds.

For reference, iperf3 server was enabled in an Amazon Instance, with Public IP address – 3.89.3.34. The local system (at my home, having a LAN IP – 192.168.1.7 mapped to Public IP 117.193.184.155 by Network Address Translation also known as NAT) hosted the iperf3 client. Ref

```
-----
Server listening on 5003
-----
Accepted connection from 117.193.184.155, port 41894
[ 5] local 172.31.37.240 port 5003 connected to 117.193.184.155 port 41895
[ ID] Interval           Transfer     Bandwidth
[ 5]  0.00-1.00       sec   22.0 KBytes    180 Kbits/sec
[ 5]  1.00-1.45       sec   33.0 KBytes    603 Kbits/sec
-----
[ ID] Interval           Transfer     Bandwidth
[ 5]  0.00-1.45       sec    0.00 Bytes    0.00 bits/sec           sender
[ 5]  0.00-1.45       sec   55.0 KBytes   311 Kbits/sec           receiver
-----
Server listening on 5003
-----
```

Figure 4. Iperf3 server instantiation and reception

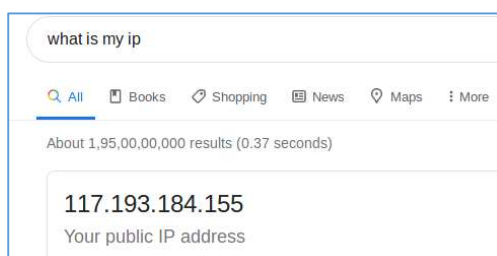


Figure 5. Public IP of the iperf3 client

```
manikantan@manikantan-Vostro-1550:~$ iperf3 -c 3.89.3.34 -p 5003 -t 1
Connecting to host 3.89.3.34, port 5003
[ 4] local 192.168.1.7 port 41895 connected to 3.89.3.34 port 5003
[ ID] Interval           Transfer     Bandwidth   Retr  Cwnd
[ 4]  0.00-1.00       sec   89.4 KBytes    732 Kbits/sec    0   34.4 KBytes
-----
[ ID] Interval           Transfer     Bandwidth   Retr
[ 4]  0.00-1.00       sec   89.4 KBytes    732 Kbits/sec    0
[ 4]  0.00-1.00       sec   55.0 KBytes    450 Kbits/sec
-----
iperf Done.
```

Figure 6. iperf3 client instantiation and transmission

The exchange of the data packets between the iperf3 server and iperf3 client is captured as a pcap file (iperf3_192_168_1_7_3_89_3_34_OneSecond.pcapng) is placed in moodle for reference.

Iperf / iperf3 is a tool/application to determine the maximum bandwidth possible between systems. One can observe (with the help of captured reference) first a connection (Flow 1) is established between the host (192.168.1.7) and server (3.89.3.34). The port at the server end is 5003. Client side port is random and it takes a value – 41894. The first connection (TCP Flow 1) is established to ensure iperf3 application is prepared for measuring the bandwidth.

Subsequently a new flow (Flow 2) is established between Client and Server with Client port – 41895, and Server port 5003. The transmission of data at higher rate is carried out over the Flow2.

In the observed capture, one can notice Server RST (Resets) the Flow 2 after the specified time (probably the application knows the transmission is expected for one second?). The Flow1 is closed gracefully with both ends transmitting TCP segments with FIN bits and acknowledgements received.

<Actions for Step5>

You can use the shared pcap file or any pcap file that has a complete TCP flow. i.e, the 3 way handshake between a client and server, exchange of data, and formal closure of the connection.

Assume Your system is client and your partner's system is server. Let the Partner System (server) wait at Port number (5000 + Roll No). Establish a TCP flow between the two systems and send 2000 bytes of data in two data segments (1000 bytes each) and then close the connection (as shown in Figure 7).

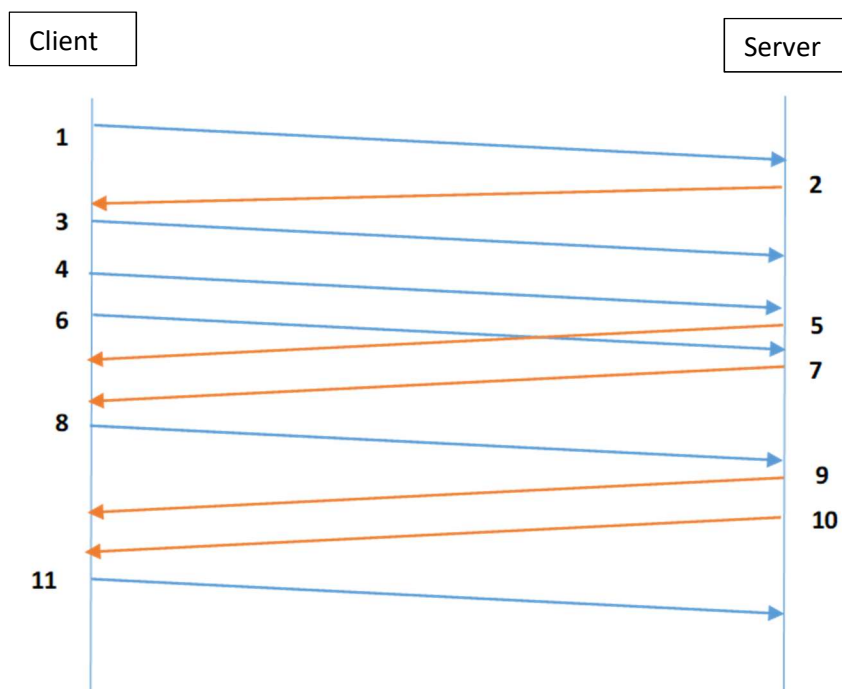


Figure 7. A TCP Exchange

Messages 1, 2 and 3 for the three way hand shake. Messages 2 and 3 are to be generated only upon receipt of messages 1 and 2, respectively at the server and client side. Messages 4, and 6 are data transfer from client, and messages 5 and 7 are ACK segments from server. ACKs are to be generated only upon receipt of the data messages. 4 and 6 are to be spaced apart. Message 8 is FIN

message from client end while Message 10 is FIN message from Server side. Messages 9 and 11 are ACKs for message 8 and 10, respectively generated after the receipt of the messages at the respective ends.

Output expected: The scapy commands (in a script / python file) to generate the Client side and Server side messages, to achieve the flow of packets, and wire shark capture of the 11 frames exchanged.

. 3 points for each message. (sub total of 33 points). Wireshark capture of the exchanged 11 packets at the client end. (27 points) – total of 60 Points.

Bonus Points: If the information and capture is provided for both sets, 40 Bonus points.

Credits: 30 Marks (2 marks for each scapy packet construction syntax – 11 packets, sub total 22 points.) Wireshark capture of 11 packets – 8 points. This is for one set. (Any one of the partner system is Client and Any one of the partner system is Server)

BONUS Points – 30 marks – If information for the reverse direction (with port number suitably taken care, Client and Server reversed) is submitted.