Scapy Network Analysis Lab

This document contains the practical Scapy exercises performed during the lab session, including packet sniffing, packet inspection, and ICMP/TCP/DNS traffic analysis. All commands and outputs are extracted directly from the learning environment.

1. Running Scapy as a Privileged User

Scapy requires elevated permissions for sending or sniffing packets

Switch to root:

Command - *sudo su*

2. Inspecting The IP Packet Header

Scapy allows inspection of protocol headers using the ls(0 function.

Command - *ls(IP)*

Key IP Header Fields Displayed

Field	**Meaning**
version	IP version
ihl	Header length
tos	Type of service
len	Total length
id	Packet ID
flags	Fragmentation flags
frag	Fragment offset
ttl	Time to live
proto	Upper-layer protocol
chksum	Header checksum
src	Source IP
dst	Destination IP
options	Optional fields

➔ This helps to understand how IP packets are structured internally.

3. Packet Sniffing (DNS + ICMP Traffic)

Two terminals were used:

- **Terminal 1 (root):** Runs Sacpy's sniffing
- **Terminal 2 (normal user):** Executes network activity(e.g ping)

Start sniffing
Command - *sniff()*

➔ This locks the terminal until stopped

Generate traffic from normal user
Command - *ping -c 6 google.com*

Stop sniffing (CTRL +C)

Sniffed summary
<Sniffed: TCP:0 UDP:28 ICMP:24 Other:4>

Store and view the results:
paro = _
paro.summary()

4. Traffic Sniffing on br-internal Interface (TCP Traffic)

To capture traffic on an internal bridge interface;

Command - *sniff(iface="br-internal")*

Sniffed Packets:
<Sniffed: TCP:153 UDP:0 ICMP:0 Other:0>

Summary (TCP Three-Way Handshake & Data)

Ether / IP / TCP 10.6.6.1:41410 > 10.6.6.23:http S
Ether / IP / TCP 10.6.6.23:http > 10.6.6.1:41410 SA
Ether / IP / TCP 10.6.6.1:41410 > 10.6.6.23:http A
Ether / IP / TCP 10.6.6.1:41410 > 10.6.6.23:http PA / Raw
Ether / IP / TCP 10.6.6.23:http > 10.6.6.1:41410 A
Ether / IP / TCP 10.6.6.23:http > 10.6.6.1:41410 PA / Raw
Ether / IP / TCP 10.6.6.1:41410 > 10.6.6.23:http A
Ether / IP / TCP 10.6.6.23:http > 10.6.6.1:41410 PA / Raw
Ether / IP / TCP 10.6.6.1:41410 > 10.6.6.23:http A
Ether / IP / TCP 10.6.6.23:http > 10.6.6.1:41410 PA / Raw

Example captured sequence:

- SYN: 10.6.6.1:41410 > 10.6.6.23:http
- SYN/ACK: 10.6.6.23:http > 10.6.6.1:41410
- ACK: handshake complete
- PA / Raw: HTTP data exchange
- Multiple connections using ports 41410, 41414, 41424

This represents:

- TCP handshake
- HTTP data transfer
- Multiple session attempts

5. ICMP Packet Analysis (Echo Requests & Replies)

Capture ICMP packets only;

Command - *sniff(iface="br-internal", filter="icmp", count=10)*

Captured summary:

<Sniffed: TCP:0 UDP:0 ICMP:10 Other:0>

Cleaned ICMP Traffic:

- Echo-request: 10.6.6.1 > 10.6.6.23
- Echo-reply: 10.6.6.23 > 10.6.6.1

This repeats for all 10 packets, indicating a successful ICMP exchange.

Indexed Packet View:

Command - *paro3.nsummary()*

0000 Ether / IP / ICMP 10.6.6.1 > 10.6.6.23 echo-request 0 / Raw
0001 Ether / IP / ICMP 10.6.6.23 > 10.6.6.1 echo-reply 0 / Raw
0002 Ether / IP / ICMP 10.6.6.1 > 10.6.6.23 echo-request 0 / Raw
0003 Ether / IP / ICMP 10.6.6.23 > 10.6.6.1 echo-reply 0 / Raw
0004 Ether / IP / ICMP 10.6.6.1 > 10.6.6.23 echo-request 0 / Raw
0005 Ether / IP / ICMP 10.6.6.23 > 10.6.6.1 echo-reply 0 / Raw
0006 Ether / IP / ICMP 10.6.6.1 > 10.6.6.23 echo-request 0 / Raw
0007 Ether / IP / ICMP 10.6.6.23 > 10.6.6.1 echo-reply 0 / Raw

0008 Ether / IP / ICMP 10.6.6.1 > 10.6.6.23 echo-request 0 / Raw

ICMP Echo Request

Command - *paro3[4]*

<Ether dst=02:42:0a:06:06:17 src=02:42:0a:75:37:07 type=IPv4 |<IP version=4 ihl=5 tos=0x0 len=84 id=24634 flags=DF frag=0 ttl=64 proto=icmp chksum=0xba4b src=10.6.6.1 dst=10.6.6.23 |<ICMP type=echo-request code=0 chksum=0xc097 id=0xe44d seq=0x3 unused='' |<Raw load='\\x81\\xb9:i\x00\x00\x00\x00\\xcb!\r\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17 \x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()+,-./01234567' |>>>>*