CS 528 Network Security Lab1

Zhanfu Yang

yang1676@purdue.edu

Content

Task 1

-------Task 1.a 3
-------Task 1.b 4
-------Task 1.c 7

Task 2

-------Task 2.a 9
-------Task 2.b 10
-------Task 2.c 10
-------Question 4 11
-------Question 5 11
-------Question 6 11
-------Question 7 11

Task 3

-------Task 3 12

Task 1

Task 1.a:

Problem 1:

(1) **pcap_compile(handel, &fp, filter_exp, 0, net)**; //compile the filter program.
(2) **pcap_close(handle)**; // Close the 'handle' session on the device.
(3) **pcap_freecode(&fp)**; // Free the memory
(4) **pcap_loop(handle, num_packets, got_packet, NULL)**; // set the callback function for new packet sniffed.
(5) **pcap_setfilter(handle, &fp)**; // Set the compilter filter for the expression
(6) **pcap_datalinkk(handle)**; // Determine the link-layer type is on an Ethernet device.
(7) **pcap_open_live(dev, SNAP_LEN, 1, 1000, errbuf)**; // set up promiscuous mode to open capture device.
(8) **pcap_lookupdev(errbuf)**; // Find a capture device which is suitable for listning on.
(9) **pcap_lookupnet(dev, &net, &mask, errbuf)**; // get network number and mask related to the capture device.

Problem 2:

We need the root priviledge to run the sniffex because the program directly accessed the lower leverl interface of the OS system in the machine. If it does not have root priviledge, the program will fail when it calls **pcap_lookupdev()** because it has no any device name as commeand argument. What's more, it will fail when call **pcap_open_live()** because the security issue.

Problem 3:

Change the 'promisc' argument in the function of **pcap_open_live():**

When promisc is 1: This mode can be versified by setting the filter expression as 'icmp' and ping a host no matter it is running the sniffer or not, likes www.google.com. The output shows that it can capture the icmp packets.
When promisc is 0: This mode can be versified by setting the filter expression as ’icmp’ and ping a host which is not running the sniffer likes www.google.com, the output will not show the icmp packets. While it ping those IP which is running the sniffer, it will capture the 'icmp' packets.

Task 1.b:

Capture ICMP:
IP of the sniffing machine: 192.168.15.5
Host: 192.168.15.4, 192.168.15.5
Filter expression: "(icmp) and ((net 192.168.15.5) or (net 192.168.15.4))"

```
^C[02/14/2019 06:47] cs528user@cs528vm:~$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: (icmp) and ((net 192.168.15.5) or (net 192.168.15.4))

Packet number 1:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: ICMP

Packet number 2:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: ICMP

Packet number 3:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: ICMP

Packet number 4:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: ICMP

Packet number 5:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: ICMP

Packet number 6:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: ICMP

Packet number 7:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: ICMP

Packet number 8:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: ICMP

Packet number 9:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: ICMP

Packet number 10:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: ICMP

Capture complete.
[02/14/2019 06:48] cs528user@cs528vm:~$
```

Figure 1 '.5' ping '.4'

```
[02/14/2019 06:55] cs528user@cs528vm:~$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: (icmp) and ((net 192.168.15.5) or (net 192.168.15.4))

Packet number 1:
       From: 192.168.15.4
         To: 172.217.5.4
   Protocol: ICMP

Packet number 2:
       From: 172.217.5.4
         To: 192.168.15.4
   Protocol: ICMP

Packet number 3:
       From: 192.168.15.4
         To: 172.217.5.4
   Protocol: ICMP

Packet number 4:
       From: 172.217.5.4
         To: 192.168.15.4
   Protocol: ICMP

Packet number 5:
       From: 192.168.15.4
         To: 172.217.5.4
   Protocol: ICMP

Packet number 6:
       From: 172.217.5.4
         To: 192.168.15.4
   Protocol: ICMP

Packet number 7:
       From: 192.168.15.4
         To: 172.217.5.4
   Protocol: ICMP

Packet number 8:
       From: 172.217.5.4
         To: 192.168.15.4
   Protocol: ICMP

Packet number 9:
       From: 192.168.15.4
         To: 172.217.5.4
   Protocol: ICMP

Packet number 10:
       From: 172.217.5.4
         To: 192.168.15.4
   Protocol: ICMP

Capture complete.
```

Figure 2 '.4' ping www.google.com

```
[02/13/2019 22:43] cs528user@cs528vm:~$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: (icmp) and ((net 192.168.15.5) or (net 192.168.15.4))

Packet number 1:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: ICMP

Packet number 2:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: ICMP

Packet number 3:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: ICMP

Packet number 4:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: ICMP

Packet number 5:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: ICMP

Packet number 6:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: ICMP

Packet number 7:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: ICMP

Packet number 8:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: ICMP

Packet number 9:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: ICMP

Packet number 10:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: ICMP

Capture complete.
[02/13/2019 22:43] cs528user@cs528vm:~$
```

Figure 3. '.4' ping '.5'

Capture the TCP packets that have a destination port range from to port 50 - 100:

IP of the sniffing machine: 192.168.15.5, Host: 192.168.15.5, 192.168.15.4.

Filter expression: "(tcp) and (dst portrange 50-100)"

Just shows like the Figure 4 and Figure 5 which indicates the capture of the packets from '.4' to '.5' with nc command in different ports. (The **nc** (or **netcat**) utility is used for just about anything under the sun involving TCP or UDP.)

```
Device: eth14
Number of packets: 10
Filter expression: (tcp) and (dst portrange 50-100)

Packet number 1:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 44146
   Dst port: 50

Packet number 2:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 60554
   Dst port: 100

Packet number 3:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 50420
   Dst port: 60

Packet number 4:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 44060
   Dst port: 80

Packet number 5:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 44060
   Dst port: 80

Packet number 6:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 44060
   Dst port: 80

Packet number 7:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 44060
   Dst port: 80
```

Figure 4 '.4' ping '.5' with different ports

```
[02/14/2019 12:38] cs528user@cs528vm:~$ nc 192.168.15.5 50
[02/14/2019 12:38] cs528user@cs528vm:~$ nc 192.168.15.5 100
[02/14/2019 12:39] cs528user@cs528vm:~$ nc 192.168.15.5 60
[02/14/2019 12:39] cs528user@cs528vm:~$ nc 192.168.15.5 40
[02/14/2019 12:39] cs528user@cs528vm:~$ nc 192.168.15.5 110
[02/14/2019 12:39] cs528user@cs528vm:~$ nc 192.168.15.5 80
[02/14/2019 12:39] cs528user@cs528vm:~$
```

Figure 5 '.4' ping '.5' with different ports in nc command

Task 1.c:

Telnet server running on 192.168.15.5
Telnet Client: 192.168.15.4
Filter expression: "port 23"(which telnet will use)

```
cs528user@cs528vm:~$ telnet 192.168.15.5
Trying 192.168.15.5...
Connected to 192.168.15.5.
Escape character is '^]'.
Ubuntu 12.04.2 LTS
cs528vm login: cs528user
Password:
Last login: Thu Feb 14 17:35:22 PST 2019 from cs528vm.local on pts/3
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

[02/14/2019 17:37] cs528user@cs528vm:~$
```

Figure 6 Client

```
   Payload (12 bytes):
00000   0d 0a 50 61 73 73 77 6f  72 64 3a 20               ..Password:

Packet number 6:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 43996
   Dst port: 23

Packet number 7:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 43996
   Dst port: 23
   Payload (1 bytes):
00000   63                                                c

Packet number 8:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: TCP
   Src port: 23
   Dst port: 43996

Packet number 9:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 43996
   Dst port: 23
   Payload (1 bytes):
00000   73                                                s

Packet number 10:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: TCP
   Src port: 23
   Dst port: 43996
```

Figure 7 Server Password part 1

```
 Packet number 1:
        From: 192.168.15.4
          To: 192.168.15.5
    Protocol: TCP
    Src port: 43996
    Dst port: 23
    Payload (1 bytes):
 00000   35                                                5

 Packet number 2:
        From: 192.168.15.5
          To: 192.168.15.4
    Protocol: TCP
    Src port: 23
    Dst port: 43996

 Packet number 3:
        From: 192.168.15.4
          To: 192.168.15.5
    Protocol: TCP
    Src port: 43996
    Dst port: 23
    Payload (1 bytes):
 00000   32                                                2

 Packet number 4:
        From: 192.168.15.5
          To: 192.168.15.4
    Protocol: TCP
    Src port: 23
    Dst port: 43996

 Packet number 5:
        From: 192.168.15.4
          To: 192.168.15.5
    Protocol: TCP
    Src port: 43996
    Dst port: 23
    Payload (1 bytes):
 00000   38                                                8

 Packet number 6:
        From: 192.168.15.5
          To: 192.168.15.4
    Protocol: TCP
    Src port: 23
    Dst port: 43996
```

Figure 8 Server Password part 2

```
   Payload (1 bytes):
00000   70                                                p

Packet number 8:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: TCP
   Src port: 23
   Dst port: 43996

Packet number 9:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 43996
   Dst port: 23
   Payload (1 bytes):
00000   61                                                a

Packet number 10:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: TCP
   Src port: 23
   Dst port: 43996

Capture complete.
[02/14/2019 17:36] cs528user@cs528vm:~$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: port 23

Packet number 1:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 43996
   Dst port: 23
   Payload (1 bytes):
00000   73                                                s

Packet number 2:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: TCP
   Src port: 23
   Dst port: 43996

Packet number 3:
       From: 192.168.15.4
         To: 192.168.15.5
   Protocol: TCP
   Src port: 43996
   Dst port: 23
   Payload (1 bytes):
00000   73                                                s

Packet number 4:
       From: 192.168.15.5
         To: 192.168.15.4
   Protocol: TCP
   Src port: 23
   Dst port: 43996
```

Figure 9 Server Password part 3

The result seems like the Figure 6(Client), 7, 8, 9 (Server).

Task 2

Task 2.a:

The program names spoofer.c. When you input 1, you will jump to ICMP spoofer and when you input 2, you will jump to Ethernet spoofer.

Compile:

```
gcc -o spoofer spoofer.c
```

Run:

Sudo ./spoofer 1 //ICMP mode

Sudo ./spoofer 2 //Ethernet mode

Task 2.b:

Spoofing machine: 192.168.15.5

Spoofed machine: 192.168.15.6
Destination machine: 172.69.90.11

tcpdump machine: 192.168.15.4
// "`sudo tcpdump -XX icmp`" inorder to capture the icmp packets.

```
:s528user@cs528vm:~$ sudo tcpdump -XX icmp
[sudo] password for cs528user:
:cpdump: verbose output suppressed, use -v or -vv for full protocol decode
.istening on eth14, link-type EN10MB (Ethernet), capture size 65535 bytes
!3:10:21.102574 IP 192.168.15.6 > 172.69.90.11: ICMP echo request, id 512, seq
 0, length 8
        0x0000:  5254 0012 3500 0800 27d9 aaa9 0800 4583  RT..5...'.....E.
        0x0010:  001c 2659 0000 4001 7e06 c0a8 0f06 ac45  ..&Y..@.~......E
        0x0020:  5a0b 08c4 7bb7 0200 0000 0000 0000 0000  Z...{...........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
!3:10:54.597505 IP 192.168.15.6 > 172.69.90.11: ICMP echo request, id 512, seq
 0, length 8
        0x0000:  5254 0012 3500 0800 27d9 aaa9 0800 4500  RT..5...'.....E.
        0x0010:  001c 2649 0000 4001 7e99 c0a8 0f06 ac45  ..&I..@.~......E
        0x0020:  5a0b 08b4 7db7 0200 0000 0000 0000 0000  Z...}...........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
!3:10:58.115392 IP 192.168.15.6 > 172.69.90.11: ICMP echo request, id 512, seq
 0, length 8
        0x0000:  5254 0012 3500 0800 27d9 aaa9 0800 4500  RT..5...'.....E.
        0x0010:  001c 2619 0000 4001 7ec9 c0a8 0f06 ac45  ..&...@.~......E
        0x0020:  5a0b 0884 6eb7 0200 0000 0000 0000 0000  Z...n...........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
!3:10:58.961365 IP 192.168.15.6 > 172.69.90.11: ICMP echo request, id 512, seq
 0, length 8
        0x0000:  5254 0012 3500 0800 27d9 aaa9 0800 4500  RT..5...'.....E.
        0x0010:  001c 26a9 0000 4001 7e39 c0a8 0f06 ac45  ..&...@.~9.....E
        0x0020:  5a0b 0814 7cb7 0200 0000 0000 0000 0000  Z...|...........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
!3:10:59.638251 IP 192.168.15.6 > 172.69.90.11: ICMP echo request, id 512, seq
 0, length 8
        0x0000:  5254 0012 3500 0800 27d9 aaa9 0800 4500  RT..5...'.....E.
        0x0010:  001c 2609 0000 4001 7ed9 c0a8 0f06 ac45  ..&...@.~......E
        0x0020:  5a0b 0874 7cb7 0200 0000 0000 0000 0000  Z..t|...........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
!3:20:15.017387 IP 192.168.15.6 > 172.69.90.11: ICMP echo request, id 512, seq
 0, length 8
        0x0000:  5254 0012 3500 0800 27d9 aaa9 0800 4500  RT..5...'.....E.
        0x0010:  001c 2639 0000 4001 7ea9 c0a8 0f06 ac45  ..&9..@.~......E
        0x0020:  5a0b 08a4 7db7 0200 0000 0000 0000 0000  Z...}...........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
```

Figure 10 ICMP spoofing

Task 2.c:

Spoofing machine: 192.168.15.5

Spoofed machine: 192.168.15.6
Destination machine: 172.69.90.11

MAC address: 01:02:03:04:05:06

// "`sudo tcpdump -eXX ether host 01:02:03:04:05:06`" inorder to capture the icmp packets.

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth14, link-type EN10MB (Ethernet), capture size 65535 bytes
09:38:04.793756 01:02:03:04:05:06 (oui Unknown) > Broadcast, ethertype IPv4 (0x0800), length 60: truncated-ip - 7122 bytes missing! 192.168.15.6
 > 172.69.90.11: ICMP echo reply, id 0, seq 0, length 7148
        0x0000:  ffff ffff ffff 0102 0304 0506 0800 4500  ..............E.
        0x0010:  1c00 78b7 0000 4001 9474 c0a8 0f06 ac45  ..x...@..t.....E
        0x0020:  5a0b 0000 0000 0000 0000 0000 0000 0000  Z...............
        0x0030:  0000 0000 0000 0000 0000 0000            ............
09:38:12.805812 01:02:03:04:05:06 (oui Unknown) > Broadcast, ethertype IPv4 (0x0800), length 60: truncated-ip - 7122 bytes missing! 192.168.15.6
 > 172.69.90.11: ICMP echo reply, id 0, seq 0, length 7148
        0x0000:  ffff ffff ffff 0102 0304 0506 0800 4500  ..............E.
        0x0010:  1c00 7bb7 0000 4001 9404 c0a8 0f06 ac45  ..{...@........E
        0x0020:  5a0b 0000 0000 0000 0000 0000 0000 0000  Z...............
        0x0030:  0000 0000 0000 0000 0000 0000            ............
```

Figure 11 Ethernet Frame Spoofing

Question 4:
Yes, the IP packet length field can be set to any arbitrary value, regardless of the how big the actual packet.

Question 5:
No, the computer and the system will automatically does this, or rather it fills it in.

Question 6:
If we do not have the root previledge, the program fails when we try to create a new raw socket which will enable us to build a packet we want. This will cause spoofing issue which is the system do not want.

Question 7:

In summary, four necessary library calls in spoofing are
1. Create a raw socket.
2. Set socket opinion.
3. Construct the packet.
4. Send out the packet through the raw socket

In detail:

1.Raw sockets are sockets that allow direct sending of packets by the applications bypassing all applications in network software of operating system. For spoofing the first and most important step is creating raw sockets that would help the program in injecting packets in the network. We may use:

Sock(AF_INET, SOCK_RAW, IPPROTO_RAW); // create new raw socket, which means the ip header will be offered by the user.

For listening to TCP, UDP and ICMP traffic, we have to create 3 separate raw sockets, using **IPPROTO_TCP**,**IPPROTO_UDP** and **IPPROTO_ICMP**

2. To inject our own packets, all we need to know is the structures of the protocols that need to be included. For example, we may use:

Connection.sin_family = AF_INET; // A setting for sending packets out a socket.

3. Construct the packet. Likes **Inet_addr(ip)** // convert an IP address from string to internet address in iphdr struct which deines the header of an IP packet.

4. Send out the packet **:**
Sendto(sd, buffer, ip->tot_len, 0, (struct sockaddr *)&sin, sizeof(connection)) // send content from the buffer to the socket.

Task 3

Machine A: 192.168.15.4
Machine B: 192.168.15.5
Machine X: An Specific IP you want to ping. (Likes www.google.com and 1.0.0.0)

Machine B:

Compile:
```
gcc –Wall –o sniffex–spoofer sniffex–spoofer.c –lpcap
```

Run:
Sudo ./`sniffex–spoofer`

Machine A:

Ping www.google.com
Ping 198.0.0.100 -s 0

```
02/15/2019 12:55] cs528user@cs528vm:~$ sudo ./sniffex-spoofer
evice: eth14
lumber of packets: 1
ilter expression: icmp and src net 192.168.15.4
ent spoofed ICMP packet.
ent spoofed ICMP packet.
ent spoofed ICMP packet.
ent spoofed ICMP packet.
ent spoofed ICMP packet.
ent spoofed ICMP packet.
ent spoofed ICMP packet.
```

Figure 12 Machine B Sniffex and Spoofer

When I ping unknown IP address in the Machine A without the Sniffex and Spoofer running in the Machine B, likes the Figure 13, it will have nothing reply.

```
cs528user@cs528vm:~$ ping 198.0.0.100 -s 0
PING 198.0.0.100 (198.0.0.100) 0(28) bytes of data.
```

Figure 13 Machine A ping unknown IP address without sniffex and spoof

While running the sniffex and spoofer in the Machien B, I ping unknown IP address in the Machine A, likes the Figure 14, it will have fake reply..

```
cs528user@cs528vm:~$ ping 198.0.0.100 -s 0
PING 198.0.0.100 (198.0.0.100) 0(28) bytes of data.
116 bytes from 198.0.0.100: icmp_req=1 ttl=64
116 bytes from 198.0.0.100: icmp_req=2 ttl=64
116 bytes from 198.0.0.100: icmp_req=3 ttl=64
116 bytes from 198.0.0.100: icmp_req=4 ttl=64
116 bytes from 198.0.0.100: icmp_req=5 ttl=64
116 bytes from 198.0.0.100: icmp_req=6 ttl=64
```

Figure 14 Machine A ping unknown IP address

When I ping public address likes www.google.com in the Machine A without running the Sniffex and Spoofer program in the Machine B, likes the Figure 15, it will have one reply.

```
cs528user@cs528vm:~$ ping www.google.com
PING www.google.com (172.217.1.36) 56(84) bytes of data.
64 bytes from ord37s07-in-f4.1e100.net (172.217.1.36): icmp_req=1 ttl=51 time=7.49 ms
64 bytes from ord37s07-in-f4.1e100.net (172.217.1.36): icmp_req=2 ttl=51 time=7.29 ms
64 bytes from ord37s07-in-f4.1e100.net (172.217.1.36): icmp_req=3 ttl=51 time=7.24 ms
64 bytes from ord37s07-in-f4.1e100.net (172.217.1.36): icmp_req=4 ttl=51 time=8.95 ms
```

Figure 15 Machine A ping public IP address

While running the sniffex and spoofer in the Machien B, I ping public address likes www.google.com in the Machine A, likes the Figure 16, it will have fake reply.

```
cs528user@cs528vm:~$ ping www.google.com
PING www.google.com (172.217.1.36) 56(84) bytes of data.
172 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=1 ttl=64 time=0.637 ms
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=1 ttl=51 time=7.45 ms (DUP!)
172 bytes from ord37s07-in-f4.1e100.net (172.217.1.36): icmp_req=2 ttl=64 time=0.789 ms
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=2 ttl=51 time=7.41 ms (DUP!)
172 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=3 ttl=64 time=0.771 ms
64 bytes from ord37s07-in-f4.1e100.net (172.217.1.36): icmp_req=3 ttl=51 time=7.79 ms (DUP!)
172 bytes from ord37s07-in-f4.1e100.net (172.217.1.36): icmp_req=4 ttl=64 time=0.774 ms
64 bytes from ord37s07-in-f4.1e100.net (172.217.1.36): icmp_req=4 ttl=51 time=7.32 ms (DUP!)
172 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=5 ttl=64 time=0.966 ms
64 bytes from ord37s07-in-f4.1e100.net (172.217.1.36): icmp_req=5 ttl=51 time=7.51 ms (DUP!)
172 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=6 ttl=64 time=0.684 ms
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=6 ttl=51 time=10.1 ms (DUP!)
172 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=7 ttl=64 time=0.773 ms
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=7 ttl=51 time=7.44 ms (DUP!)
172 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=8 ttl=64 time=0.715 ms
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_req=8 ttl=51 time=7.71 ms (DUP!)
```

Figure 16 Machine A ping www.google.com