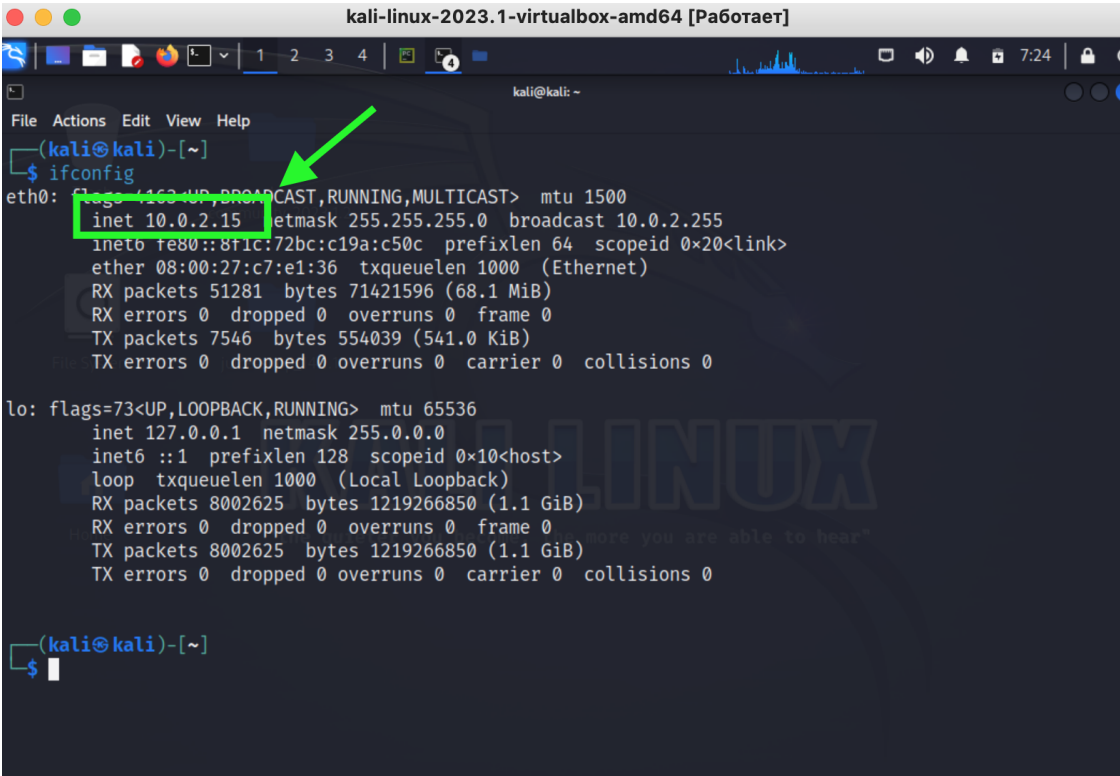


ИНСТРУКЦИЯ

1. Для того, чтобы реализовать скрипт для предотвращения ARP-спуфинга в локальной сети, нужно написать программу, которая будет “сбирать” все MAC-адреса устройств в указанной локальной сети и формировать файл, который при случае можно было бы использовать для проверки корректности MAC-адресов. (Либо же, в дальнейшем, например, сделать выполнение этой программы регулярным и сравнивать, не изменились ли MAC-адреса.)
2. Предварительным действием будет выполнение хорошо знакомой команды `$ifconfig`, чтобы определиться, какую подсеть мы будем сканировать.



```
kali-linux-2023.1-virtualbox-amd64 [Работает]
(kali@kali)~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::8f1c:72bc:c19a:c50c prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:c7:e1:36 txqueuelen 1000 (Ethernet)
    RX packets 51281 bytes 71421596 (68.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7546 bytes 554039 (541.0 KiB)
    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 8002625 bytes 1219266850 (1.1 GiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8002625 bytes 1219266850 (1.1 GiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali@kali)~$
```

Также, если было корректно выполнено Практическое задание 2.5, то в нашей “локальной сети” будет дополнительная виртуальная машина со своим IP и MAC-адресом и мы увидим эти адреса в результатах работы программы. Однако, для успешного выполнения работы присутствие нескольких виртуальных машин не является обязательным.

3. Итак, для написания программы нужно воспользоваться уже известной по Практическому занятию 2.4 библиотекой Scapy, чтобы выполнять ARP запросы. Скрипт имеет следующий код:

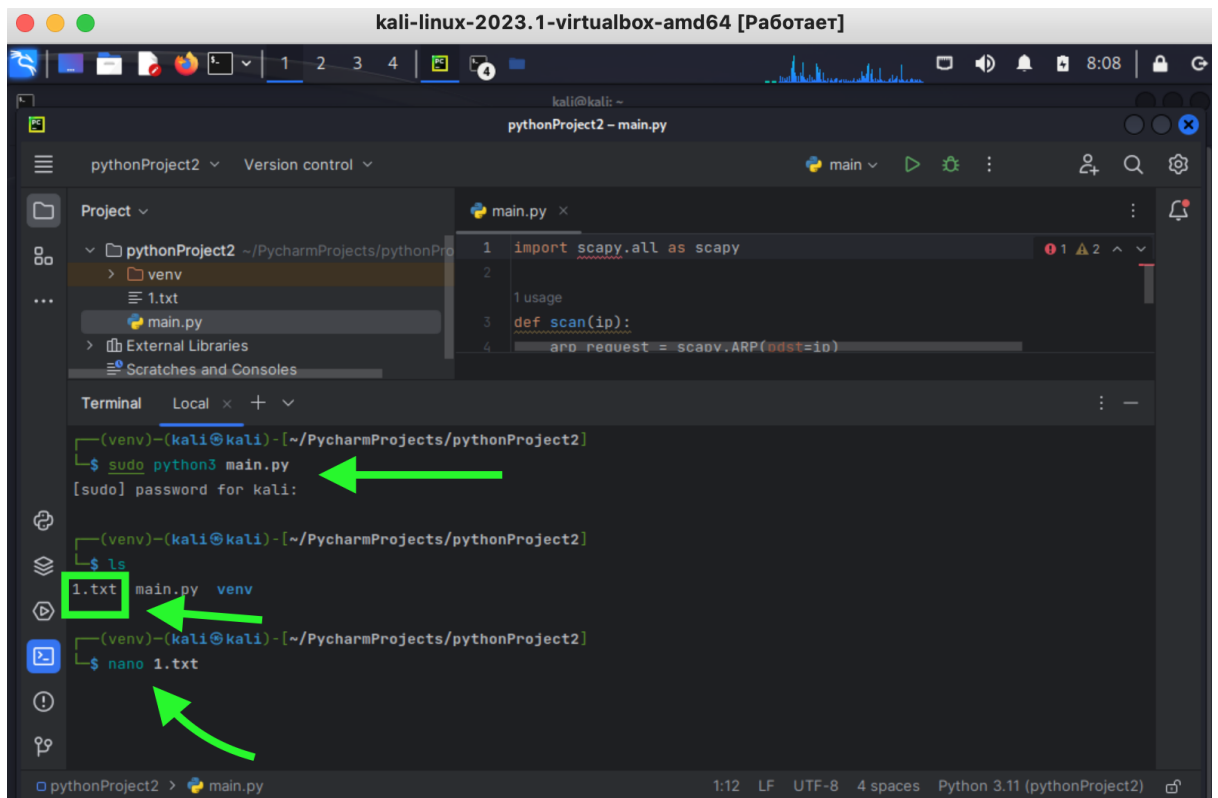
```
import scapy.all as scapy # импорт библиотеки

# функция сканирования - отправка ARP запросов
# отправка осуществляется на широковещательный адрес
# указанной подсети и запись ответов
def scan(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_request_broadcast = broadcast / arp_request
    answered_list = scapy.srp(arp_request_broadcast,
timeout=1, verbose=False)[0]
    clients_list = []
    # далее - формирование списка полученных адресов IP+MAC
    for i in answered_list:
        clients_dict = {"ip": i[1].psrc, "mac":
i[1].hwsrc}
        clients_list.append(clients_dict)
    return clients_list
# запуск сканирования указанной подсети (указать свою,
возможно, такую же, в формате x.x.x.1/24)
scan_result = scan("10.0.2.1/24")
# запись вывода программы в txt файл
scan_to_file = ''.join(map(str, scan_result)) #
добавление каждого найденного адреса в строку
f = open('1.txt', 'w') # создание файла 1.txt (или любое
другое название) в режиме w - write (запись)
f.write(scan_to_file) # запись результатов сканирования в
созданный файл
```

(Также прикрепляем программу в формате снимка экрана, чтобы можно было перепроверить отступы):

```
main.py x
1 import scapy.all as scapy
2
3 1 usage
4 def scan(ip):
5     arp_request = scapy.ARP(pdst=ip)
6     broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
7     arp_request_broadcast = broadcast / arp_request
8     answered_list = scapy.srp(arp_request_broadcast, timeout=1, verbose=False)[0]
9     clients_list = []
10
11     for i in answered_list:
12         clients_dict = {"ip": i[1].psrc, "mac": i[1].hwsrc}
13         clients_list.append(clients_dict)
14     return clients_list
15
16 scan_result = scan("10.0.2.1/24")
17
18 scan_to_file = ''.join(map(str, scan_result))
19 f = open('1.txt', 'w')
20 f.write(scan_to_file)
```

4. Чтобы запустить программу (main.py), нужны права администратора, то есть в Pycharm нужно открыть терминал и ввести команду
\$sudo python3 main.py и ввести пароль kali linux
5. Чтобы проверить, что программа отработала корректно в том же терминале нужно ввести команду \$ls и посмотреть, что файл создан. Далее открыть его в редакторе nano с помощью
\$nano 1.txt



Там должен содержаться список такого вида:

GNU nano 7.2 1.txt

```
{'ip': '10.0.2.2', 'mac': '52:54:00:12:35:02'}{'ip': '10.0.2.3', 'mac':  
'52:54:00:12:35:03'}{'ip': '10.0.2.4', 'mac': '52:54:00:12:35:04'}
```

То есть в сети находится три устройства с IP адресами:

10.0.2.2

10.0.2.3

10.0.2.4

И им соответствуют MAC-адреса:

'52:54:00:12:35:02'

'52:54:00:12:35:03'

'52:54:00:12:35:04'