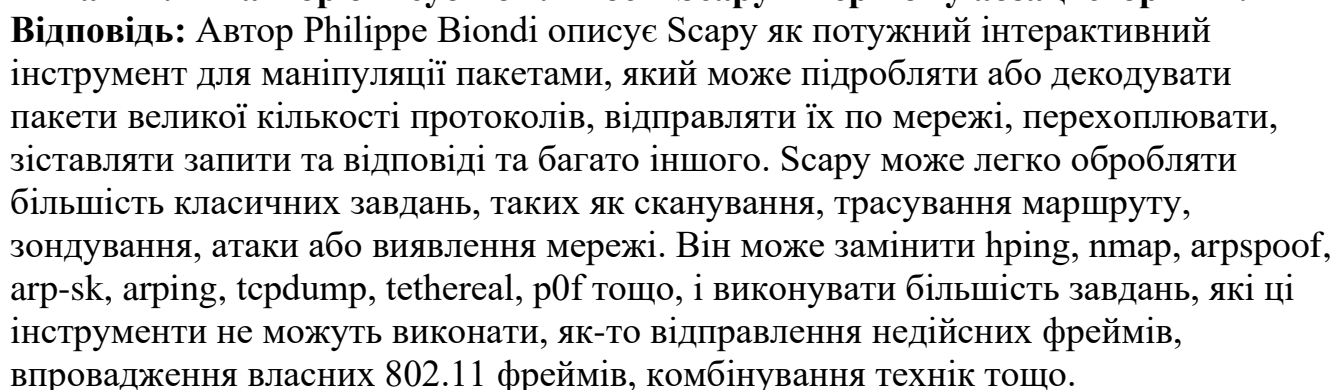


Крок 1: Дослідження документації та ресурсів Scaru



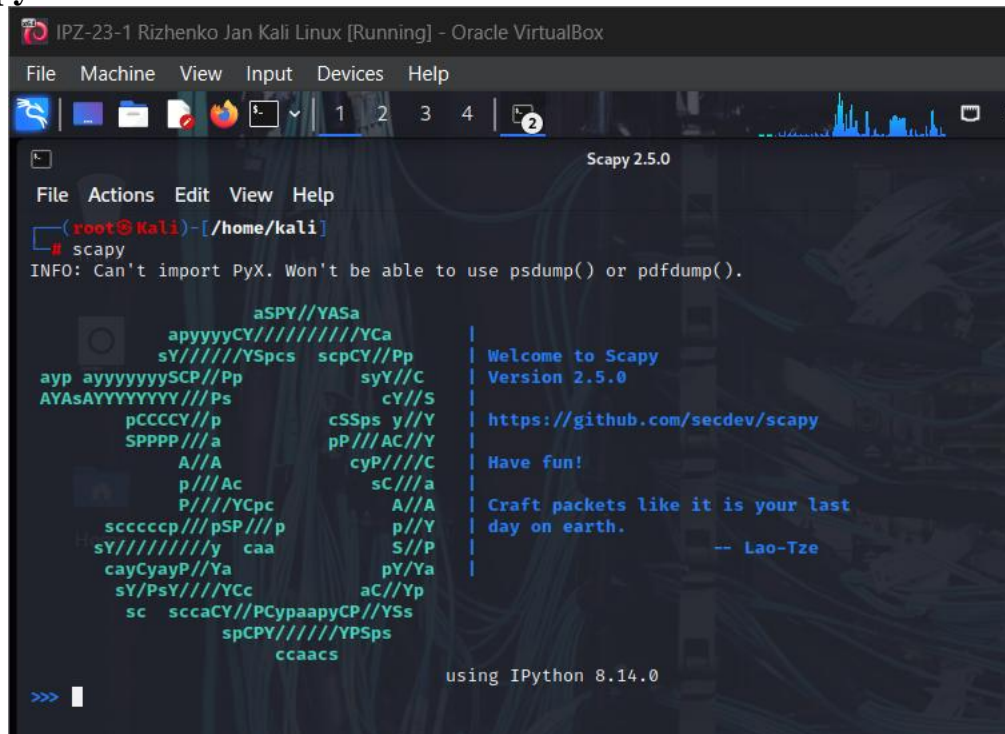
| | | | | | | | | | | | | | |
|-----------|------|---------------|--------|------|--|----------------------|--|--|------|--|---------|--|--|
| | | | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | | | | | | | | |
| | | | | | | | | | | | | | |
| Змн. | Арк. | № докум. | Підпис | Дата | | | | | | | | | |
| Розроб. | | Риженко Я.В | | | Звіт з лабораторної роботи | Літ. | | | Арк. | | Аркушів | | |
| Перевір. | | Покотило О.А. | | | | | | | 1 | | 11 | | |
| Керівник | | | | | | ФІКТ Гр. ІПЗ-23-1[2] | | | | | | | |
| Н. контр. | | | | | | | | | | | | | |
| Зав. каф. | | | | | | | | | | | | | |

Крок 2: Використання інтерактивного командного режиму Scapy

а-б. Запуск Scapy з правами root:

sudo su

scapy



```
IPZ-23-1 Rizhenko Jan Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Scapy 2.5.0
File Actions Edit View Help
(root@Kali)-[/home/kali]
# scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().

  aSPY//YASa
  apyyyyCY////////YCa
  sY////////YSPcs  scpCY//Pp
  ayp ayyyyyySCP//Pp  syY//C
  AYAsAYYYYYYYY///Ps  cY//S
  pCCCCY///p  cSSps y//Y
  SPPPP///a  pP///AC//Y
  A//A  cyP///C
  p///Ac  sC///a
  P///YCpc  A//A
  scccccp///pSP///p  p//Y
  sY////////y caa  S//P
  cayCyayP//Ya  pY/Ya
  sY/PsY///YCc  aC//Yp
  sc sccaCY//PCyaaPyCP//YSs
  spCPY////////YPSps
  ccaacs

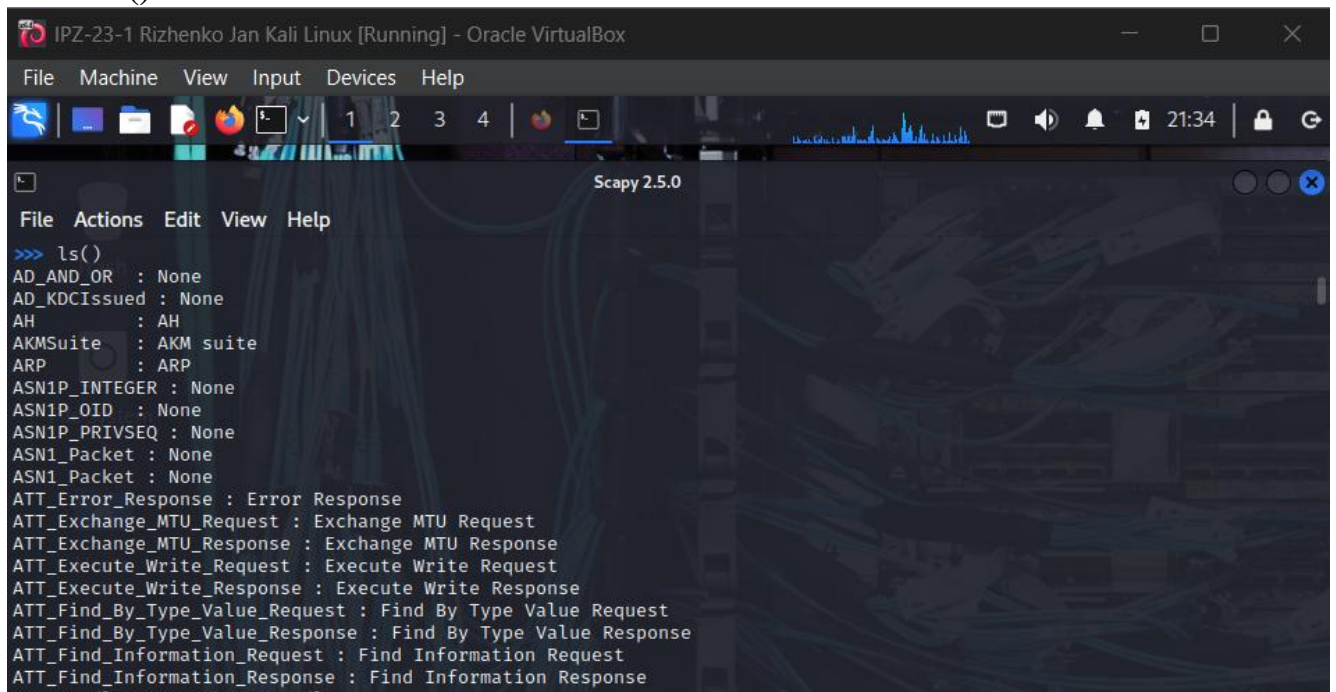
  | Welcome to Scapy
  | Version 2.5.0
  | https://github.com/secdev/scapy
  | Have fun!
  | Craft packets like it is your last
  | day on earth.
  | -- Lao-Tze

  >>> using IPython 8.14.0
```

Рис. 2. Запуск Scapy в інтерактивному режимі

с. Перегляд доступних протоколів:

ls()



```
IPZ-23-1 Rizhenko Jan Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Scapy 2.5.0
File Actions Edit View Help
>>> ls()
AD_AND_OR : None
AD_KDCIssued : None
AH : AH
AKMSuite : AKM suite
ARP : ARP
ASN1P_INTEGER : None
ASN1P_OID : None
ASN1P_PRIVSEQ : None
ASN1_Packet : None
ASN1_Packet : None
ATT_Error_Response : Error Response
ATT_Exchange_MTU_Request : Exchange MTU Request
ATT_Exchange_MTU_Response : Exchange MTU Response
ATT_Execute_Write_Request : Execute Write Request
ATT_Execute_Write_Response : Execute Write Response
ATT_Find_By_Type_Value_Request : Find By Type Value Request
ATT_Find_By_Type_Value_Response : Find By Type Value Response
ATT_Find_Information_Request : Find Information Request
ATT_Find_Information_Response : Find Information Response
ATT_Handle : ATT_Short_Handle
```

Рис. 3. Список доступних форматів пакетів та протоколів у Scapy

Питання: Скільки типів форматів пакетів TFTP перераховано?

Відповідь: У Scapy перераховано 5 типів форматів пакетів TFTP:

- TFTP - базовий клас TFTP

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Рижченко Я.В. | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. |
| | | Покотило О.А. | | | | 2 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

- TFTP_RRQ - Read Request (запит на читання)
- TFTP_WRQ - Write Request (запит на запис)
- TFTP_DATA - Data packet (пакет даних)
- TFTP_ACK - Acknowledgment (підтвердження)
- TFTP_ERROR - Error packet (пакет помилки)

Крок 3: Дослідження полів у заголовку пакета IPv4

б. Перегляд полів IP пакета:

ls(IP)

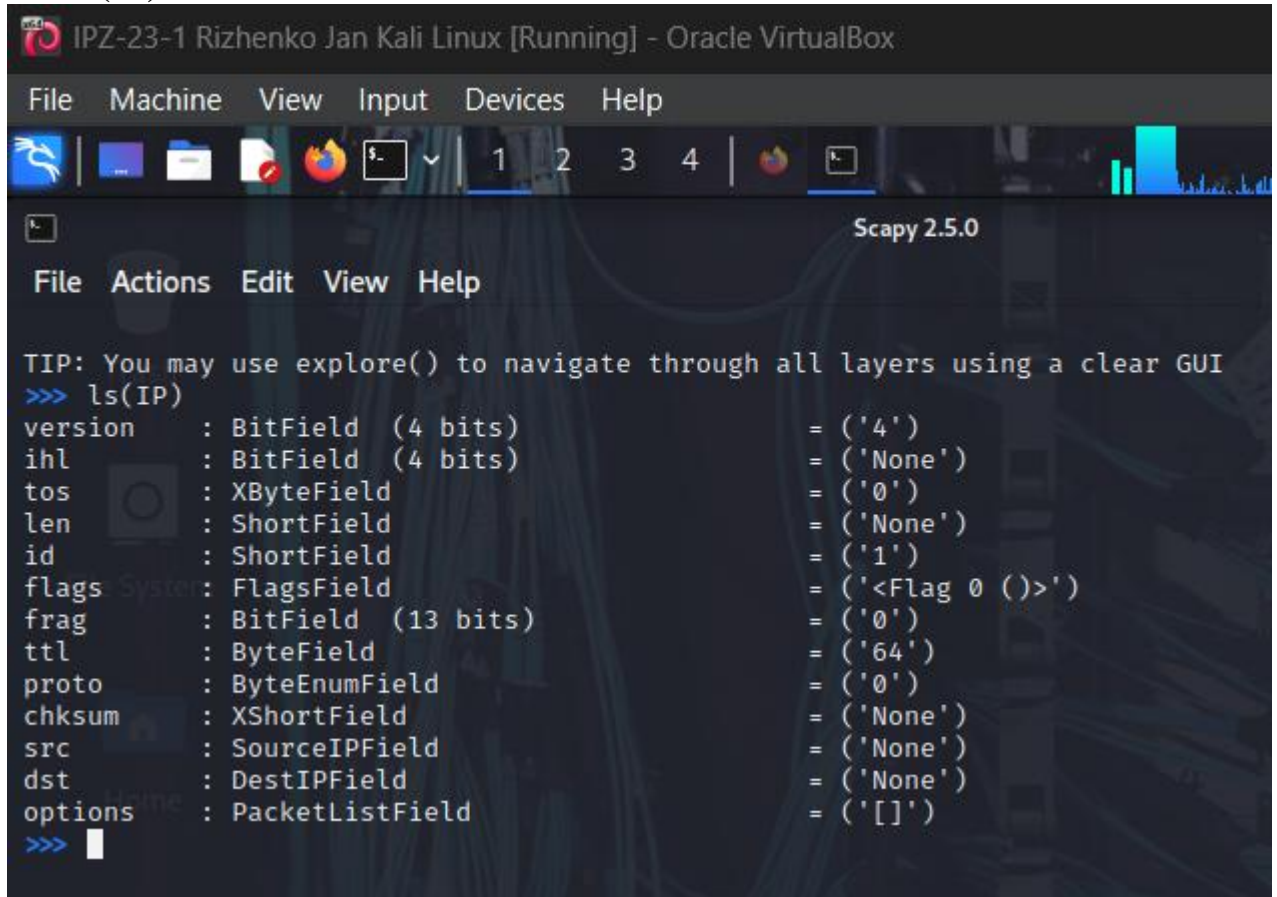


Рис. 4. Детальний список полів заголовка IP пакета в Scapy

Питання: Чи є відмінності між полями в деталях IP у Scapy та заголовком пакета, описаним у кроці 3а?

Відповідь: Так, існують деякі відмінності в термінології та представленні:

- Scapy використовує скорочення "ihl" (Internet Header Length), тоді як стандартний опис може використовувати повну назву
- Поле "tos" (Type of Service) у Scapy відповідає полю "Differentiated Services (DS)" у стандартному описі
- Scapy показує типи полів (BitField, ByteField, ShortField), чого немає в стандартному описі заголовка
- Значення за замовчуванням відображаються в Scapy, що полегшує створення пакетів
- Загалом структура та призначення полів залишаються однаковими, але Scapy надає більш програмістсько-орієнтоване представлення

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Рижченко Я.В. | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. |
| | | Покотило О.А. | | | | 3 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

Питання: Яке поле ви б змінили, щоб створити пакет, який згенерує відповідь на цільову машину, а не на машину, яка фактично відправила пакет?

Відповідь: Я б змінив поле "src" (Source IP Address) - вихідну IP-адресу. Якщо підробити (spoofing) це поле та встановити адресу цільової машини або іншого хоста, відповідь буде надіслана на підроблену адресу замість реальної машини-відправника. Це техніка, відома як IP spoofing, і вона часто використовується в атаках типу DDoS (наприклад, Smurf attack), коли зловмисник підробляє вихідну адресу, щоб спрямувати відповіді на жертву.

Частина 2: Використання Scapy для перехоплення мережевого трафіку

Крок 4: Використання функції sniff()

а-с. Захоплення трафіку:

sniff()

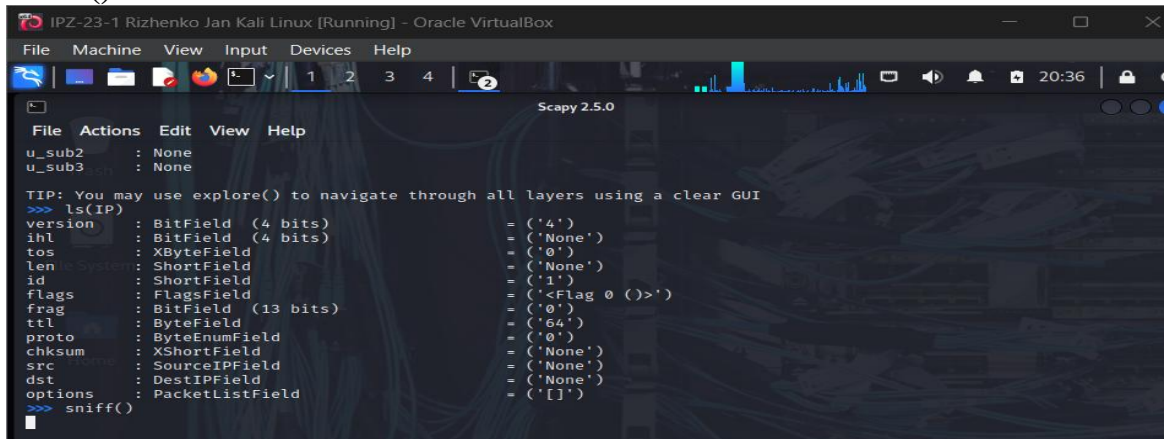


Рис. 5. Запуск захоплення пакетів за допомогою функції sniff()

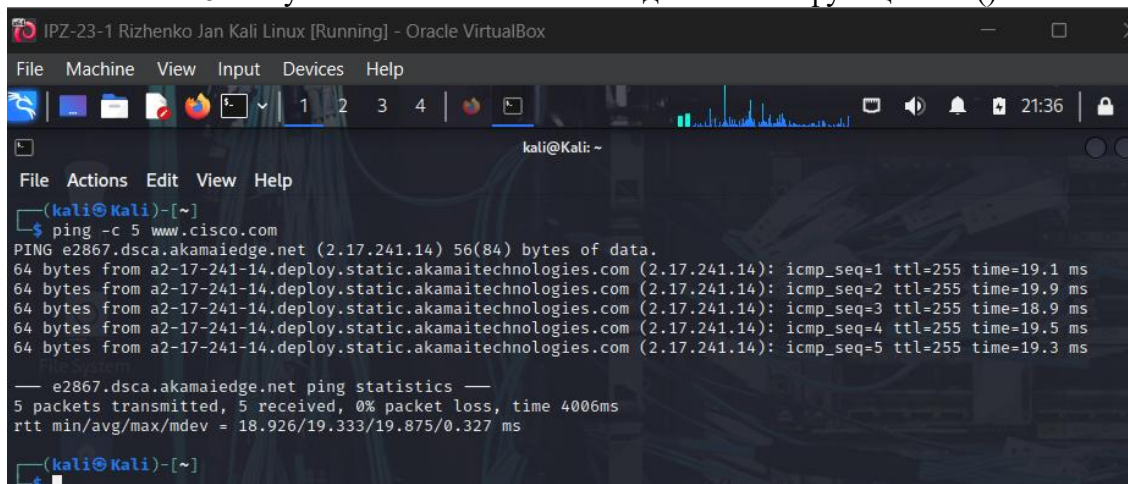


Рис. 6. Виконання команди ping до www.cisco.com з другого терміналу

д. Перегляд захопленого трафіку:

a=_

a.summary()

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Риженко Я.В | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. |
| | | Покотило О.А. | | | | 4 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

IPZ-23-1 Rizhenko Jan Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help

Scapy 2.5.0
File Actions Edit View Help

using IPython 8.14.0

>>> sniff()
^C<Sniffed: TCP:0 UDP:14 ICMP:10 Other:2>
>>> a=
>>> a.summary()
Ether / IP / UDP / DNS Qry "b'www.cisco.com.'"
Ether / IP / UDP / DNS Qry "b'www.cisco.com.'"
Ether / IP / UDP / DNS Ans "b'www.cisco.com.akadns.net.'"
Ether / IP / UDP / DNS Ans "b'www.cisco.com.akadns.net.'"
Ether / IP / ICMP 10.0.2.15 > 2.17.241.14 echo-request 0 / Raw
Ether / IP / ICMP 2.17.241.14 > 10.0.2.15 echo-reply 0 / Raw
Ether / IP / UDP / DNS Qry "b'14.241.17.2.in-addr.arpa.'"
Ether / IP / UDP / DNS Ans "b'a2-17-241-14.deploy.static.akamaitechnologies.com.'"
Ether / IP / ICMP 10.0.2.15 > 2.17.241.14 echo-request 0 / Raw
Ether / IP / ICMP 2.17.241.14 > 10.0.2.15 echo-reply 0 / Raw
Ether / IP / UDP / DNS Qry "b'14.241.17.2.in-addr.arpa.'"
Ether / IP / UDP / DNS Ans "b'a2-17-241-14.deploy.static.akamaitechnologies.com.'"
Ether / IP / ICMP 10.0.2.15 > 2.17.241.14 echo-request 0 / Raw
Ether / IP / ICMP 2.17.241.14 > 10.0.2.15 echo-reply 0 / Raw
Ether / IP / UDP / DNS Qry "b'14.241.17.2.in-addr.arpa.'"
Ether / IP / UDP / DNS Ans "b'a2-17-241-14.deploy.static.akamaitechnologies.com.'"
Ether / IP / ICMP 10.0.2.15 > 2.17.241.14 echo-request 0 / Raw
Ether / IP / ICMP 2.17.241.14 > 10.0.2.15 echo-reply 0 / Raw
Ether / IP / UDP / DNS Qry "b'14.241.17.2.in-addr.arpa.'"

kali@Kali: ~
File Actions Edit View Help

(kali@Kali)-[~]
$ ping -c 5 www.cisco.com
PING e2867.dsca.akamaiedge.net (2.17.241.14) 56(84) bytes of data:
64 bytes from a2-17-241-14.deploy.static.akamaitechnologies.com (2.17.241.14): icmp_seq=1 ttl=255 time=19.2 ms
64 bytes from a2-17-241-14.deploy.static.akamaitechnologies.com (2.17.241.14): icmp_seq=2 ttl=255 time=19.1 ms
64 bytes from a2-17-241-14.deploy.static.akamaitechnologies.com (2.17.241.14): icmp_seq=3 ttl=255 time=19.7 ms
64 bytes from a2-17-241-14.deploy.static.akamaitechnologies.com (2.17.241.14): icmp_seq=4 ttl=255 time=20.0 ms
64 bytes from a2-17-241-14.deploy.static.akamaitechnologies.com (2.17.241.14): icmp_seq=5 ttl=255 time=19.2 ms

— e2867.dsca.akamaiedge.net ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4013ms
rtt min/avg/max/mdev = 19.105/19.439/19.988/0.336 ms

(kali@Kali)-[~]
$

```

Рис. 7. Зведена інформація про захоплені пакети

Крок 5: Захоплення та збереження трафіку на конкретному інтерфейсі

а. Визначення інтерфейсу:

ifconfig

```

IPZ-23-1 Rizhenko Jan Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help

kali@Kali: ~
File Actions Edit View Help

(kali@Kali)-[~]
$ ifconfig
br-339414195aeb: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.5.5.1 netmask 255.255.255.0 broadcast 10.5.5.255
    inet6 fe80::42:10ff:fe6e:24bb prefixlen 64 scopeid 0<20<link>
    ether 02:42:10:6e:24:bb txqueuelen 0 (Ethernet)
    RX packets 75 bytes 4276 (4.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 23 bytes 3036 (2.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

br-355ee7945a88: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.1 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::42:94ff:fe8f:2e9 prefixlen 64 scopeid 0<20<link>
    ether 02:42:94:8f:02:e9 txqueuelen 0 (Ethernet)
    RX packets 80 bytes 12653 (12.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 15 bytes 2360 (2.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

br-internal: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.6.6.1 netmask 255.255.255.0 broadcast 10.6.6.255
    inet6 fe80::42:f9ff:fea2:e9cb prefixlen 64 scopeid 0<20<link>
    ether 02:42:f9:a2:e9:cb txqueuelen 0 (Ethernet)
    RX packets 2847 bytes 1015896 (992.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3039 bytes 261699 (255.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    inet6 fe80::42:caff:fe42:81d0 prefixlen 64 scopeid 0<20<link>
    ether 02:42:ca:42:81:d0 txqueuelen 0 (Ethernet)
    RX packets 84 bytes 13113 (12.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 28 bytes 3825 (3.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

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 fd00::473a:e910:bd01:3cd0 prefixlen 64 scopeid 0<0<global>
    inet6 fd00::a00:27ff:fe4a:f36e prefixlen 64 scopeid 0<0<global>
    inet6 fe80::a00:27ff:fe4a:f36e prefixlen 64 scopeid 0<20<link>
    ether 08:00:27:4a:f3:6e txqueuelen 1000 (Ethernet)
    RX packets 9758 bytes 2930913 (2.7 MiB)

```

Рис. 8. Визначення інтерфейсу з адресою 10.6.6.1 (br-internal)

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Риженко Я.В | | | | Арк. |
| | | Покотило О.А. | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | 5 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

**б-с. Захоплення трафіку на внутрішньому інтерфейсі:
sniff(iface="br-internal")**

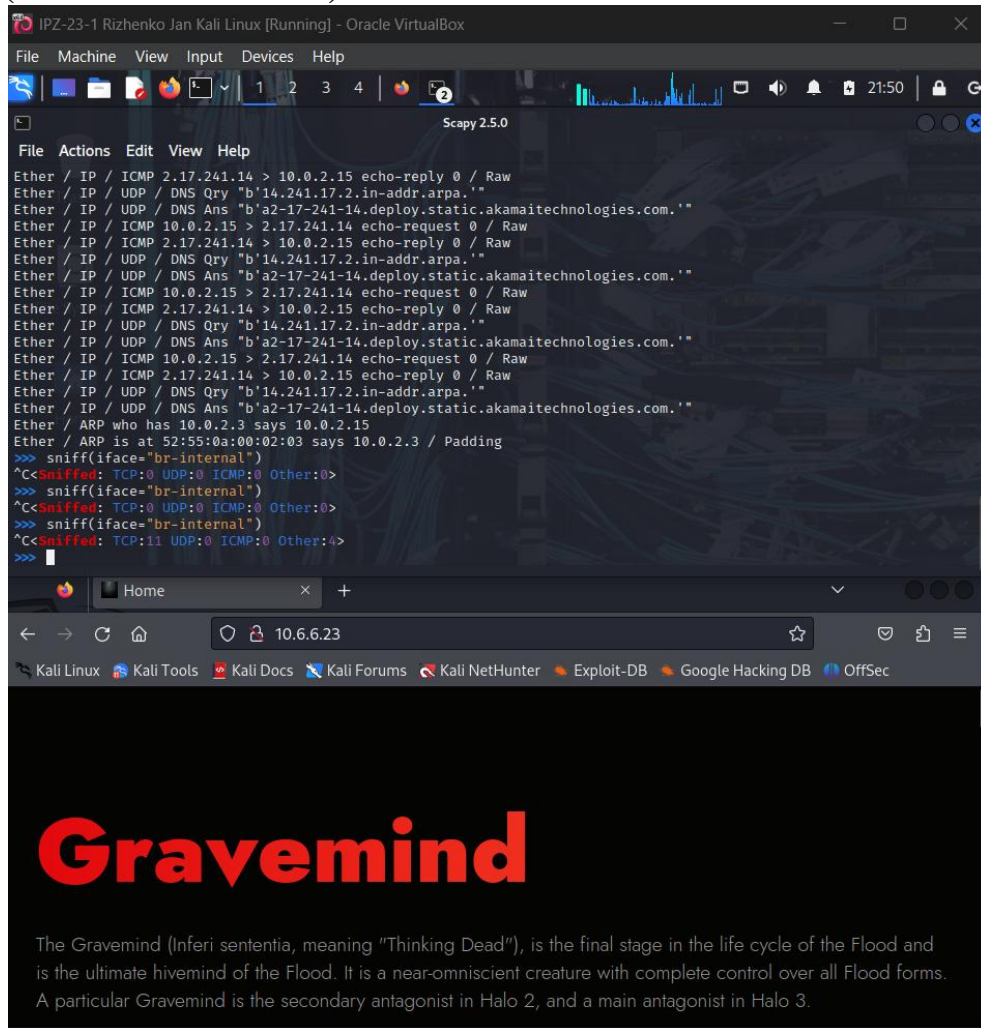


Рис. 9. Захоплення трафіку під час навігації до <http://10.6.6.23>

д. Перегляд результатів:

**a=_
a.summary()**

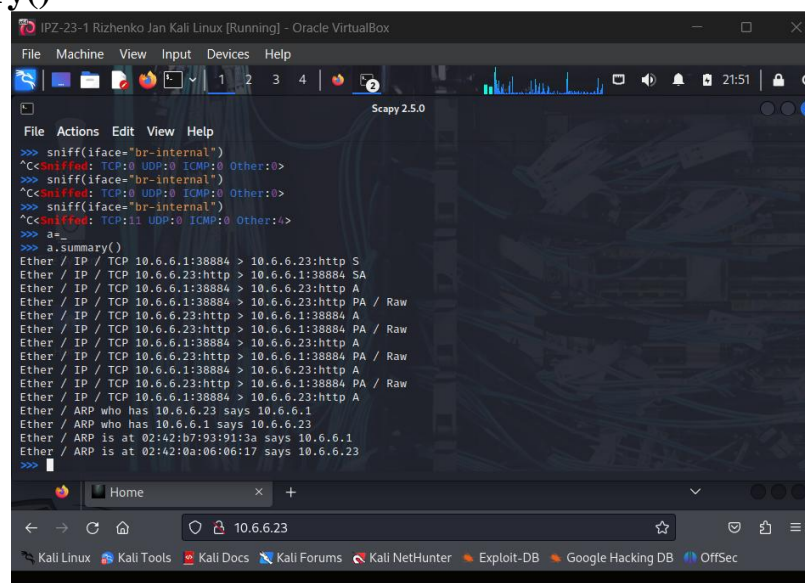


Рис. 10. Зведена інформація про захоплений HTTP трафік

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Рижченко Я.В. | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. |
| | | Покотило О.А. | | | | 6 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

Крок 6: Дослідження зібраних пакетів

а-б. Захоплення ICMP пакетів:

sniff(iface="br-internal", filter="icmp", count=10)

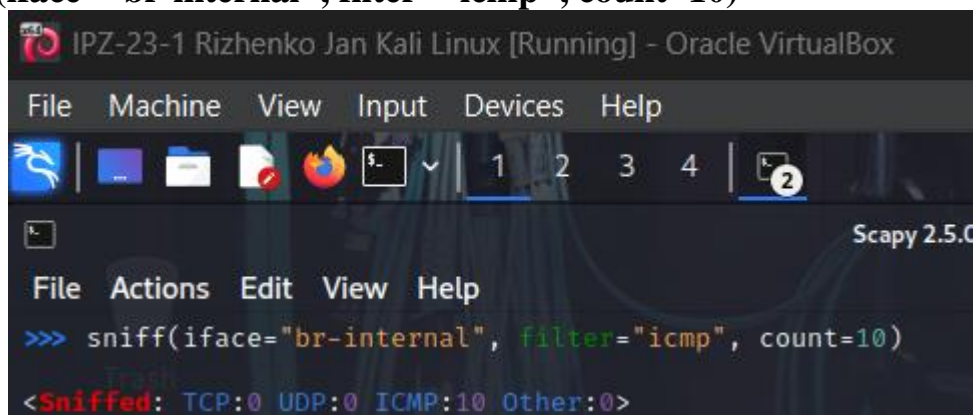


Рис. 11. Захоплення 10 ICMP пакетів

с. Перегляд із номерами рядків:

a=_

a.nsummary()

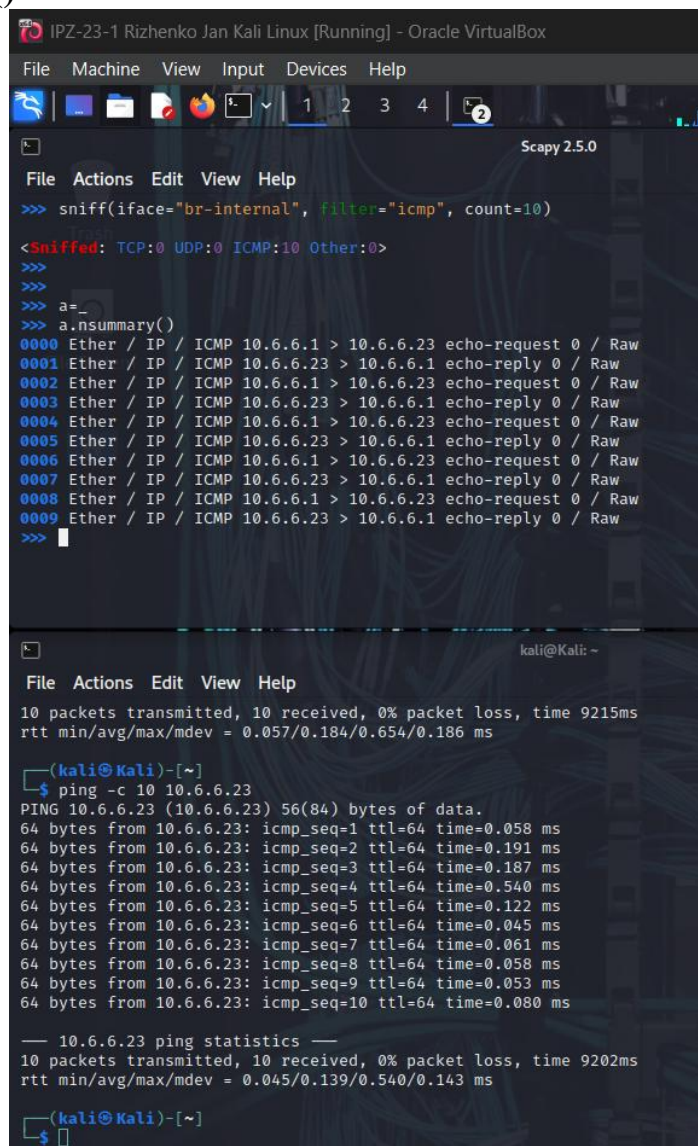


Рис. 12. Нумероване зведення захоплених ICMP пакетів

| | | | | | | |
|------|------|---------------|--------|------|--|-----------|
| | | Рижченко Я.В. | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. 7 |
| | | Покотило О.А. | | | | |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

Питання: Який трафік відображається у виводі функції nsummary()?

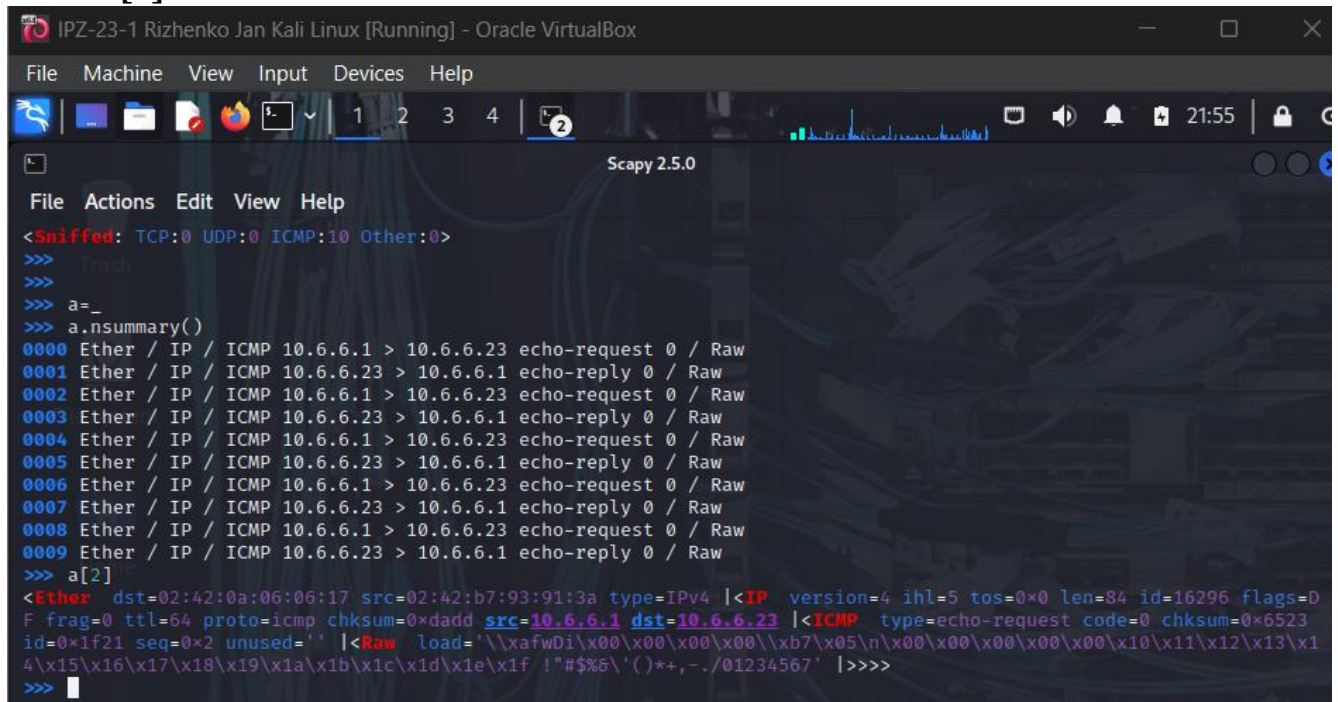
Відповідь: У виводі функції nsummary() відображаються ICMP Echo Request та ICMP Echo Reply пакети між хостом 10.6.6.1 (Kali Linux) та 10.6.6.23 (цільовий хост). Зокрема:

- Парні номери (0, 2, 4, 6, 8) - ICMP Echo Request пакети від 10.6.6.1 до 10.6.6.23
- Непарні номери (1, 3, 5, 7, 9) - ICMP Echo Reply пакети від 10.6.6.23 до 10.6.6.1

Це стандартний обмін пакетами команди ping.

d. Перегляд деталей конкретного пакета:

a[2]



```
IPZ-23-1 Rizhenko Jan Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
1 2 3 4 5
Scapy 2.5.0
File Actions Edit View Help
<Sniffed: TCP:0 UDP:0 ICMP:10 Other:0>
>>>
>>> a=
>>> a.nsummary()
0000 Ether / IP / ICMP 10.6.6.1 > 10.6.6.23 echo-request 0 / Raw
0001 Ether / IP / ICMP 10.6.6.23 > 10.6.6.1 echo-reply 0 / Raw
0002 Ether / IP / ICMP 10.6.6.1 > 10.6.6.23 echo-request 0 / Raw
0003 Ether / IP / ICMP 10.6.6.23 > 10.6.6.1 echo-reply 0 / Raw
0004 Ether / IP / ICMP 10.6.6.1 > 10.6.6.23 echo-request 0 / Raw
0005 Ether / IP / ICMP 10.6.6.23 > 10.6.6.1 echo-reply 0 / Raw
0006 Ether / IP / ICMP 10.6.6.1 > 10.6.6.23 echo-request 0 / Raw
0007 Ether / IP / ICMP 10.6.6.23 > 10.6.6.1 echo-reply 0 / Raw
0008 Ether / IP / ICMP 10.6.6.1 > 10.6.6.23 echo-request 0 / Raw
0009 Ether / IP / ICMP 10.6.6.23 > 10.6.6.1 echo-reply 0 / Raw
>>> a[2]
<Ether dst=02:42:0a:06:06:17 src=02:42:b7:93:91:3a type=IPv4 |<IP version=4 ihl=5 tos=0x0 len=84 id=16296 flags=0
F frag=0 ttl=64 proto=icmp checksum=0xdadd src=10.6.6.1 dst=10.6.6.23 |<ICMP type=echo-request code=0 checksum=0x6523
id=0x1f21 seq=0x2 unused='' |<Raw load='\\xafwDi\\x00\\x00\\x00\\x00\\xb7\\x05\\n\\x00\\x00\\x00\\x00\\x10\\x11\\x12\\x13\\x1
4\\x15\\x16\\x17\\x18\\x19\\x1a\\x1b\\x1c\\x1d\\x1e\\x1f !"#%&'()*+,-./01234567' |>>>>
```

Рис. 13. Детальна інформація про третій пакет у захопленні

Питання: Чому є два набори полів джерела та призначення?

Відповідь: Два набори полів джерела (src) та призначення (dst) присутні тому, що пакет складається з кількох рівнів інкапсуляції (моделі OSI/TCP IP):

Перший набір (Ethernet layer - каналний рівень): MAC-адреси джерела та призначення

- src: MAC-адреса відправника
- dst: MAC-адреса одержувача (або шлюзу)

Другий набір (IP layer - мережевий рівень): IP-адреси джерела та призначення

- src: IP-адреса відправника (10.6.6.1)
- dst: IP-адреса одержувача (10.6.6.23)

Це демонструє інкапсуляцію протоколів: IP пакет інкапсульовано в Ethernet фрейм. Кожен рівень має власні адреси для маршрутизації на відповідному рівні мережевої моделі.

e-g. Збереження захоплених даних:

wrpcap("capture1.pcap", a)

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Риженко Я.В. | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. |
| | | Покотило О.А. | | | | 8 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

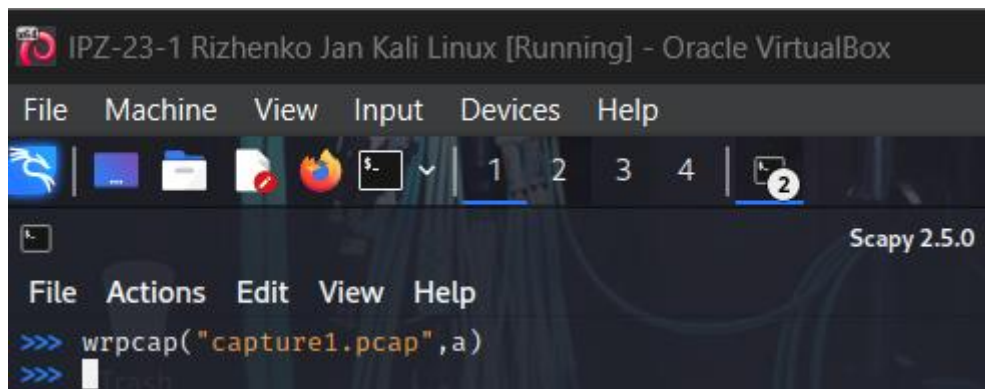


Рис. 14. Збереження захоплених пакетів у файл .pcap

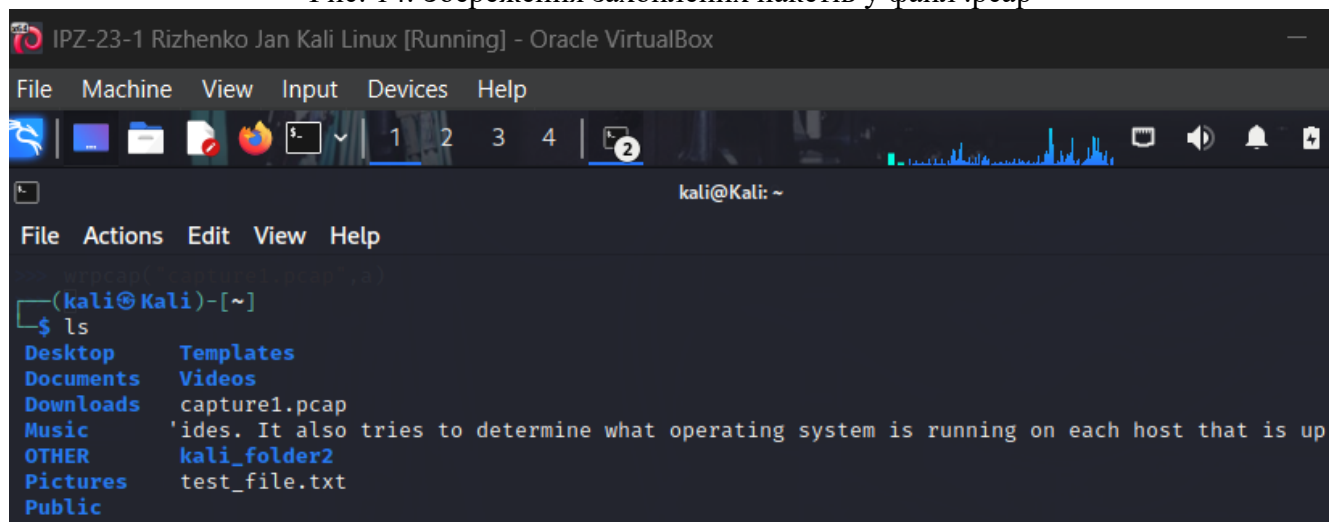


Рис. 15. Перевірка створення файлу capture1.pcap

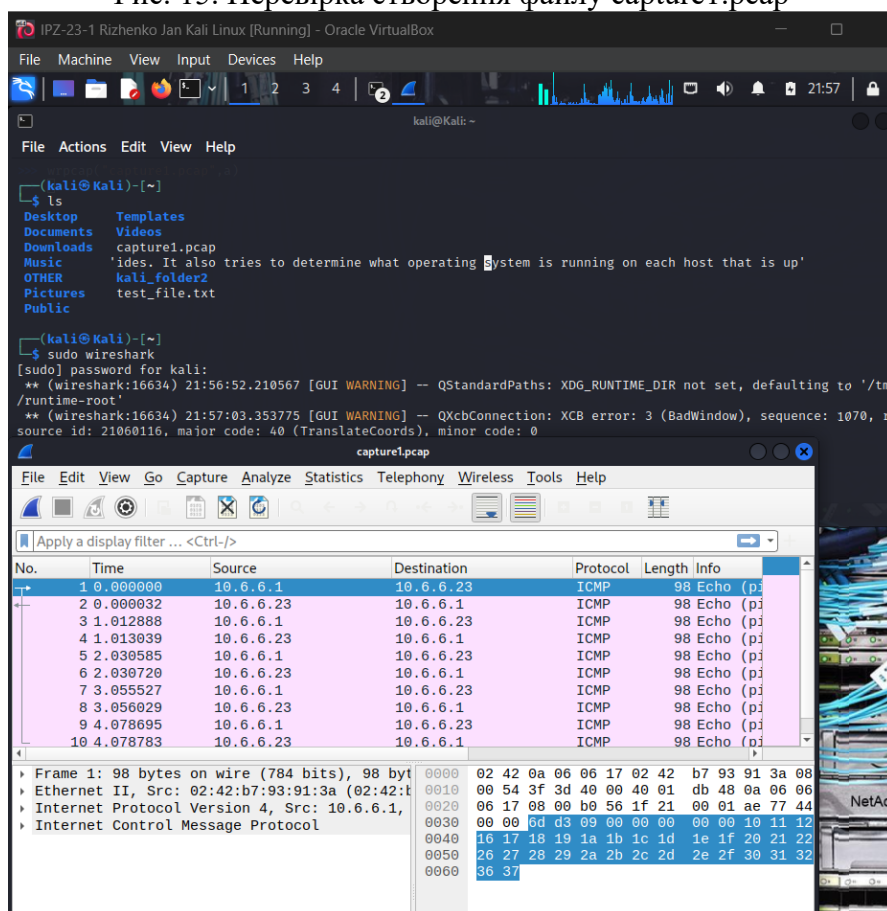


Рис. 16. Відкриття файлу capture1.pcap у Wireshark

| | | | | | | |
|------|------|---------------|--------|------|--|-----------|
| | | Рижченко Я.В | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. 9 |
| | | Покотило О.А. | | | | |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

Частина 3: Створення та відправлення ICMP пакета

Крок 1: Використання інтерактивного режиму для створення та відправлення власного ICMP пакета

a. Запуск захоплення в першому терміналі:

`sniff(iface="br-internal")`

b. Відправлення власного ICMP пакета з другого терміналу:

`send(IP(dst="10.6.6.23")/ICMP()/"This is a test")`

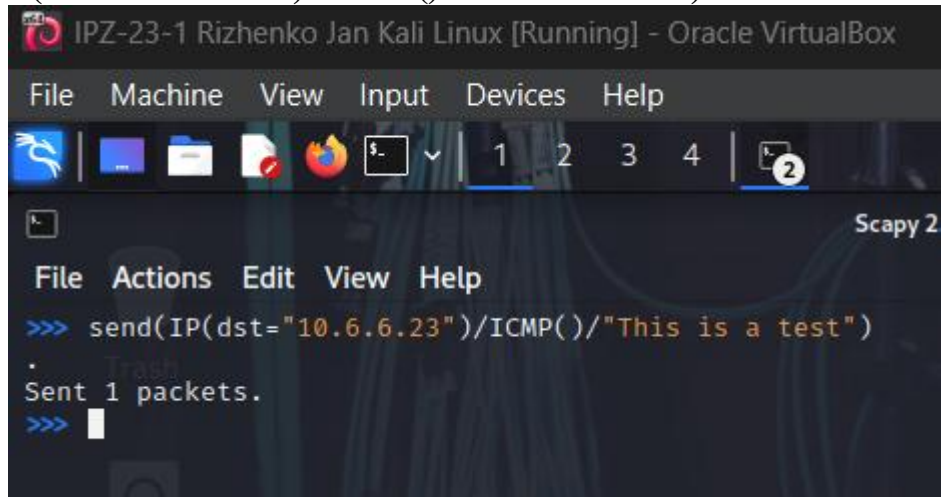


Рис. 17. Відправлення власного ICMP пакета з модифікованим payload

c-d. Перегляд результатів:

`a=_`

`a.nsummary()`

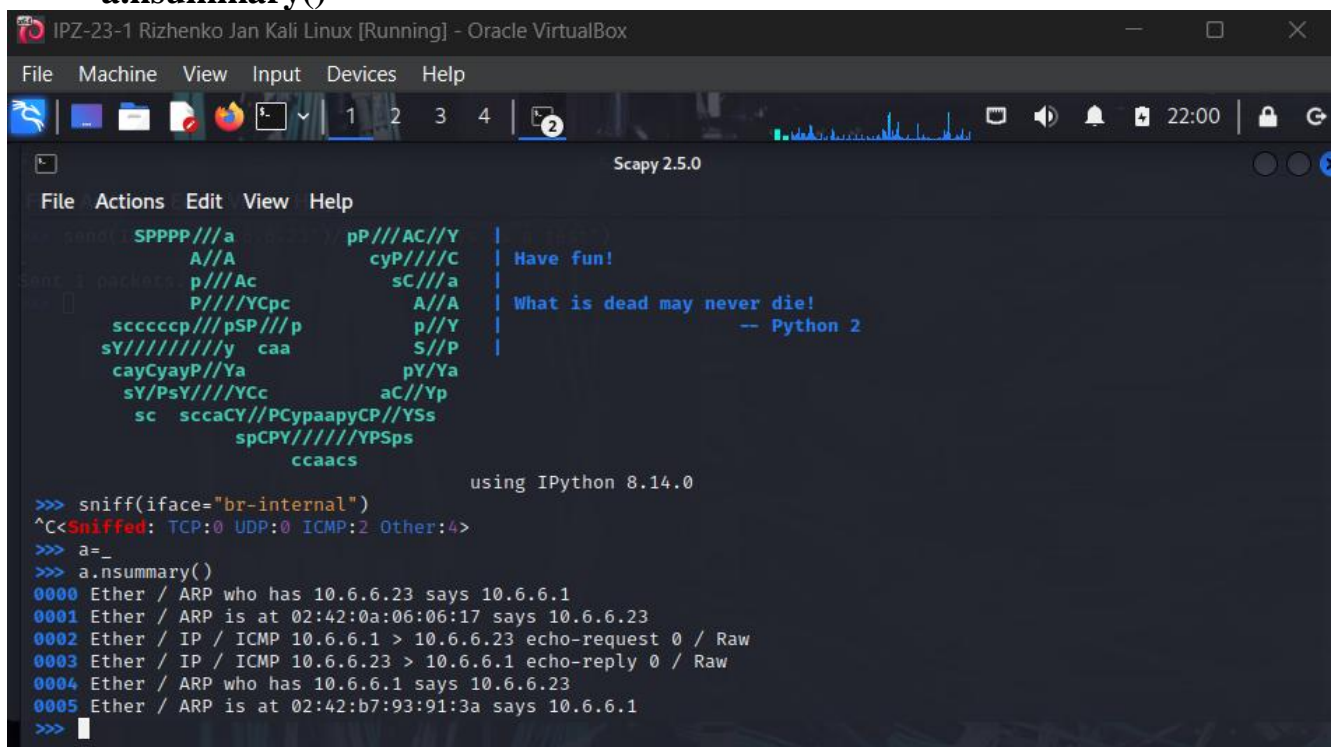


Рис. 18. Зведення захоплених пакетів після відправлення власного ICMP

Питання: Які типи пакетів показані у виводі зведення?

Відповідь: У виводі зведення показані два типи ICMP пакетів:

ICMP Echo Request (тип 8) - відправлений від 10.6.6.1 до 10.6.6.23 з власним payload "This is a test"

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Рижченко Я.В. | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. |
| | | Покотило О.А. | | | | 10 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

ICMP Echo Reply (тип 0) - відповідь від 10.6.6.23 до 10.6.6.1, яка відображає той самий payload

Крок 2: Перегляд та порівняння вмісту ICMP пакетів

a[0]

a[1]

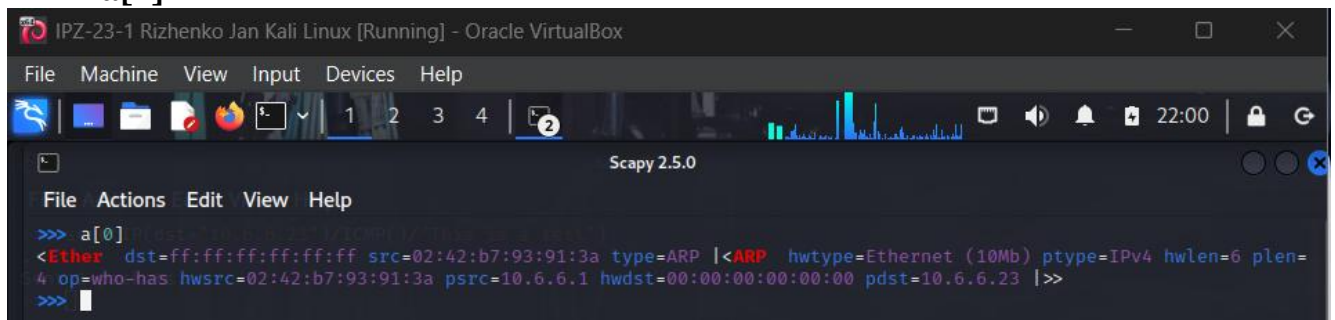


Рис. 19. Детальний вміст власного ICMP Echo Request пакета

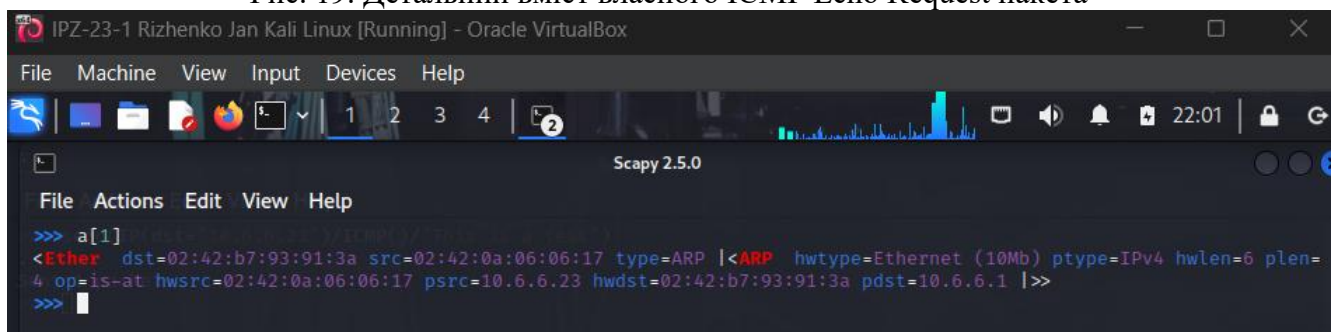


Рис. 20. Детальний вміст ICMP Echo Reply пакета

Питання: Що відрізняється між оригінальною ICMP розмовою та власною ICMP розмовою?

Відповідь: Основні відмінності між оригінальною та власною ICMP розмовою:
Payload (корисне навантаження):

- Оригінальний ring: стандартне корисне навантаження (зазвичай випадкові дані або послідовність символів)
- Власний пакет: містить текст "This is a test"

Розмір пакета:

- Оригінальний: стандартний розмір (зазвичай 64 байти)
- Власний: відрізняється залежно від довжини payload

ICMP ідентифікатор та послідовний номер:

- Оригінальний: автоматично генеровані системою значення
- Власний: значення за замовчуванням або нуль, якщо не вказано

Контрольна сума (checksum):

- Обидва мають правильну контрольну суму, але значення відрізняються через різний вміст

Це демонструє, що Scapy дозволяє повністю контролювати вміст пакетів, включаючи payload.

Частина 4: Створення та відправлення TCP SYN пакета

Крок 1: Запуск захоплення пакетів на внутрішньому інтерфейсі

a. Запуск захоплення:

sniff(iface="br-internal")

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Риженко Я.В. | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. |
| | | Покотило О.А. | | | | 11 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

b. Відправлення TCP SYN пакета:

`send(IP(dst="10.6.6.23")/TCP(dport=445, flags="S"))`

Рис. 21. Відправлення TCP SYN пакета на порт 445

Крок 2: Перегляд захоплених пакетів

a-b. Аналіз результатів:

`a=_`

`a.nsummary()`

`a[1]`

Рис. 22. Зведення захоплених TCP пакетів

Рис. 23. Детальна інформація про відповідь від цільового хоста

Питання: Що вказує прапорець SA у пакеті, повернутому від 10.6.6.23?

Відповідь: Прапорець SA (SYN-ACK) у відповідному пакеті вказує на те, що:

- Порт 445 **ВІДКРИТИЙ** на цільовому хості 10.6.6.23
- Хост готовий встановити TCP з'єднання
- SYN-ACK є другим кроком у триетапному рукошестисанні TCP (three-way handshake):

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Рижченко Я.В. | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. |
| | | Покотило О.А. | | | | 12 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

Крок 1: Клієнт надсилає SYN

Крок 2: Сервер відповідає SYN-ACK (підтверджує запит і пропонує власний SYN)

Крок 3: Клієнт відповідає ACK (не виконано в цьому прикладі)

Якби порт був закритий, хост відповів би пакетом RST (Reset). Якби порт був фільтрований firewall, відповіді не було б взагалі. Отримання SA-прапорця підтверджує, що служба SMB/CIFS активна та доступна на порту 445.

Питання для рефлексії

1. Як створення різних TCP SYN пакетів може використовуватися для проведення пасивної розвідки на цільовому хості?

Відповідь:

Створення TCP SYN пакетів є основою техніки SYN сканування (також відомого як "stealth scanning" або напівпідключення), яка використовується для виявлення відкритих портів без встановлення повного TCP з'єднання. Це дозволяє проводити розвідку наступним чином:

Методи використання:

Сканування портів: Надсилення SYN пакетів на різні порти для визначення, які сервіси працюють

Визначення стану портів на основі відповідей:

- SA (SYN-ACK) = порт відкритий
- RST (Reset) = порт закритий
- Немає відповіді = порт фільтрується firewall

Fingerprinting ОС: Аналіз специфічних характеристик відповідей (TTL, розмір вікна, опції TCP)

Виявлення firewall та IDS: Тестування різних комбінацій прапорців для обходу систем безпеки

Картографування мережі: Визначення топології та захисних механізмів

Переваги для розвідки:

- Менш помітно, ніж повне TCP підключення (не завершує handshake)
- Швидше за традиційне сканування
- Часто не реєструється в логах додатків (тільки на рівні firewall)
- Дозволяє маніпулювати полями для обходу фільтрів

Технічні можливості:

- Зміна TTL для імітації різних відстаней
- Підроблення вихідної адреси для анонімності
- Налаштування розміру вікна та опцій TCP для fingerprinting
- Використання різних послідовностей портів для уникнення виявлення

Однак варто зазначити, що це насправді активна розвідка, а не пасивна, оскільки включає відправлення пакетів до цільового хоста.

2. Як створення ICMP echo-request пакета з підробленою вихідною адресою може створити атаку відмови в обслуговуванні (DoS) проти цільового хоста?

Відповідь:

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Риженко Я.В. | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. |
| | | Покотило О.А. | | | | 13 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

Створення ICMP echo-request з підробленою вихідною адресою є основою Smurf Attack - класичної DDoS атаки з ампліфікацією. Механізм атаки працює наступним чином:

Принцип роботи Smurf Attack:

- Підроблення адреси: Зловмисник створює ICMP echo-request пакети, але встановлює поле src (вихідну IP-адресу) рівним IP-адресі жертви
- Широкомовне розсилання: Ці пакети відправляються на broadcast адресу мережі (наприклад, 192.168.1.255)
- Ампліфікація: Всі хости в цільовій мережі отримують broadcast і відповідають ICMP echo-reply
- Перевантаження жертви: Всі відповіді надсилаються до підробленої адреси (жертви), перевантажуючи її канал та ресурси

Чому це ефективна DoS атака:

- Ампліфікація трафіку: Один запит генерує десятки/сотні відповідей
- Анонімність: Справжній зловмисник залишається прихованим
- Споживання ресурсів: Жертва отримує величезну кількість трафіку
- Перевантаження мережі: Bandwidth жертви вичерпується
- Виснаження CPU: Обробка великої кількості пакетів

Варіації атаки:

- Fraggle Attack: Використання UDP замість ICMP
- DNS Amplification: Використання DNS серверів для ампліфікації
- NTP Amplification: Експлуатація NTP протоколу

Сучасний захист:

- Більшість мереж блокують directed broadcasts
 - Ingress/egress filtering запобігає IP spoofing
 - Rate limiting на ICMP трафік
- Сучасні firewall та IPS системи виявляють такі патерни

Висновок: У ході виконання лабораторної роботи було досліджено можливості Scapy як потужного Python-based інструменту для маніпуляції та створення мережевих пакетів. Освоєно інтерактивний режим роботи Scapy та вивчено структуру заголовків протоколів IP, ICMP та TCP. Практично застосовано функції sniff() для перехоплення мережевого трафіку на різних інтерфейсах з використанням фільтрів протоколів та обмеженням кількості пакетів. Виконано створення та відправлення власних ICMP пакетів з модифікованим payload, що продемонструвало можливості Scapy для повного контролю над вмістом пакетів. Реалізовано техніку SYN сканування шляхом створення та відправлення TCP SYN пакетів для визначення стану портів на цільовому хості, отримавши SYN-ACK відповідь, яка підтвердила, що порт 445 відкритий. Робота підкреслила важливість розуміння структури мережевих протоколів та продемонструвала, як інструменти створення пакетів можуть використовуватися як для легітимного тестування безпеки та розвідки, так і для потенційних зловмисних дій, включаючи DoS атаки через IP spoofing, що підкреслює необхідність належних заходів безпеки в мережевій інфраструктурі.

| | | | | | | |
|------|------|---------------|--------|------|--|------|
| | | Риженко Я.В | | | ДУ «Житомирська політехніка».23.121.26.000 – Лр11(3.2.9) | Арк. |
| | | Покотило О.А. | | | | 14 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |