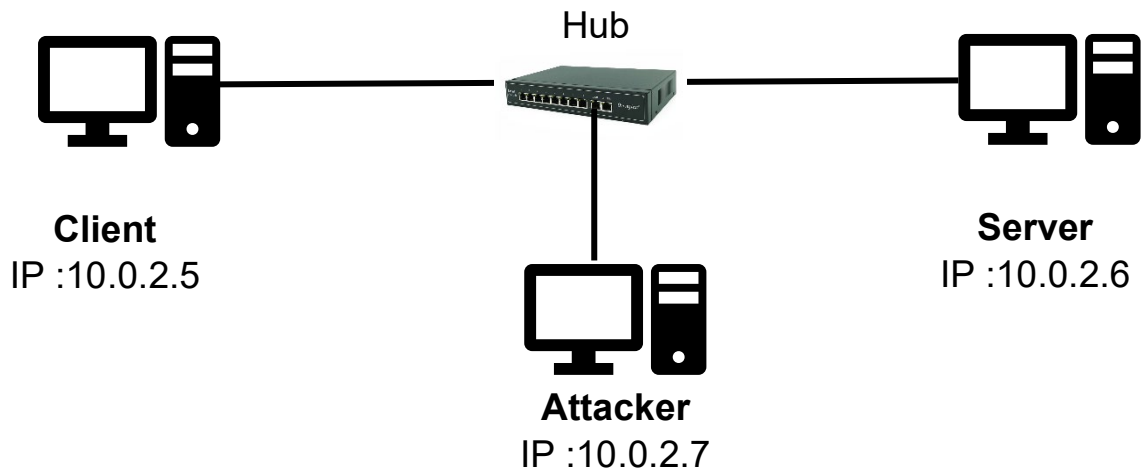


מעבדה מספר 1 – Packet Sniffing and Spoofing

Ofir Gerbi



- השרטוט מתאים לכל המעבדה.

Using Tools to Sniff and Spoof Packets :Task 1

מטרת משימה זו היא לבנות כלי שיאפשר לנו לקלוט Packets המתקבלים ברשת (Sniff) תחת פילטרים שונים, וכלי אשר יאפשר לנו לזייף Packets ברשת (Spoof).

:Task 1.1A

במשימה זו בנינו Packet Sniffer המיועד לקלוט Packets המתקבלים ברשת ולהדפיס את פרטיהם, והתבקשנו להריץ את התוכנית עם הרשאות root וללא הרשאות ולבדוק מה קורה.

תחילה, בנינו את קובץ ה-Sniffer:

```

sniff_icmp.py
#!/usr/bin/python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter='icmp', prn=print_pkt)
  
```

הקוד מיועד לקלוט Packet תחת פילטר של פרוטוקול ICMP ולהדפיס את פרטיו.

מבנה הקוד:

תחילה נקלוט את ה-Packet בעזרת פונקציית sniff, תחת פילטר של ICMP. כל פאקט הנקלטת, תשלח לפונקציית print_pkt בעזרת פרמטר prn. בפונקציה זאת נפעיל pkt.show שתדפיס לנו את פרטי הפקטה.

נריץ פקודת Ping מ-Client לשרת גוגל, שיאפשר לנו לקלוט את ה-Packets המתקבלים בפרוטוקול ICMP מה-ping.

```
[09:07:23] 26/10/21 seed@client: ping www.google.com
PING www.google.com (142.250.200.4) 56(84) bytes of data.
64 bytes from lhr48s29-in-f4.1e100.net (142.250.200.4): icmp_seq=1:
63.6 ms
64 bytes from lhr48s29-in-f4.1e100.net (142.250.200.4): icmp_seq=2:
62.7 ms
^C
--- www.google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
```

ניתן לראות כי נשלחו ונקלטו 2 Packets.

במקביל, מ-attacker, נריץ את קובץ הקוד עם הרשאות root בעזרת פקודת sudo:

```
[09:07:49] 26/10/21 seed@attacker: chmod a+x sniff_icmp.py
[09:08:06] 26/10/21 seed@attacker: sudo ./sniff_icmp.py
sudo: unable to resolve host client
###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:c5:d4:0b
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 42724
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x30c1
  src      = 10.0.2.5
  dst      = 142.250.200.4
```

Chmod a+x – פקודה שעושה את התוכנית שלנו ניתנת להרצה (executable).

Sudo (Super User Do) – פקודה שנותנת לנו הרשאות root (אדמין), הרשאות הרצה גבוהות).

ניתן לראות כי מוצג לנו פרטי הפאקט שהתקבל מה-Ping, כלומר הצלחנו לבצע קליטה של הפאקט.

כעת, ננסה להריץ את הקובץ ללא הרשאות Root:

```
[09:16:07] 26/10/21 seed@attacker sniffer_icmp.py
Traceback (most recent call last):
  File "./sniffer_icmp.py", line 5, in <module>
    pkt = sniff(filter='icmp',prn=print_pkt)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 10, in sniff
    sniffer.run(*args, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 10, in run
    **karg)] = iface
  File "/usr/local/lib/python3.5/dist-packages/scapy/arch/linux.py", line 10, in __init__
    socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))
  File "/usr/lib/python3.5/socket.py", line 134, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
```

כפי שניתן לראות, הקובץ לא ניתן להרצה ללא הרשאות מתאימות.

לסיכום, ניתן לראות כי ניסינו להריץ את קובץ הפייתון המיועד לקלוט packets עם הרשאות וללא הרשאות, וכי הקוד חייב להיות תחת הרשאות root, ובלי הרשאות אלו לא נוכל להריץ אותו. מסקנה זו הגיונית מטעמי אבטחה מאחר והתוכנית ניגשת לחלקים רגישים במערכת ההפעלה (כגון פתיחת Raw Socket ושימוש ב-Promiscuous Mode אשר מאפשרים קבלת עותק של הפאקטה).

Task 1.1B:

בד"כ נרצה לקלוט Packets מפרוטוקולים או תנאים מסוימים. במשימה זו ננסה לקלוט Packets עם פילטרים שונים של ICMP, TCP ו-SUBNET.

נעשה זאת כך שנעדכן את קובץ הקוד של ה-Sniffer כך שכל פעם יוגדר על פילטר אחר, ובצע תקשורת בהתאם לפילטר וננסה לקלוט את ה-packet.

ראשית, ננסה לקלוט packet בפרוטוקול ICMP.

נגדיר בקובץ ה-sniffer את הפילטר ל-ICMP:

```
sniff_icmp.py

#!/usr/bin/python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter='icmp', prn=print_pkt)
```

הקוד מיועד לקלוט Packet בפרוטוקול icmp ולהדפיס את פרטיו.


מ-Client נריץ Ping לשרת גוגל, שיאפשר לנו לקלוט את ה-Packets המתקבלים בפרוטוקול icmp מה-ping.

```
[09:07:23] 26/10/21 seed@client: ping www.google.com
PING www.google.com (142.250.200.4) 56(84) bytes of data.
64 bytes from lhr48s29-in-f4.1e100.net (142.250.200.4): icmp_seq=1
63.6 ms
64 bytes from lhr48s29-in-f4.1e100.net (142.250.200.4): icmp_seq=2
62.7 ms
^C
--- www.google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
```

ניתן לראות כי נשלחו ונקלטו 2 Packets.

במקביל, נריץ את ה-Sniffer ב-Attacker:

```
[09:07:49] 26/10/21 seed@attacker:~$ chmod a+x sniff_icmp.py
[09:08:06] 26/10/21 seed@attacker:~$ sudo ./sniff_icmp.py
sudo: unable to resolve host client
###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:c5:d4:0b
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 42724
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x30c1
  src      = 10.0.2.5
  dst      = 142.250.200.4
```



כפי שניתן לראות את פרטי הפאקטה תחת פרוטוקל ICMP, כלומר הצלחנו לקלוט את הפאקטות אשר נשלחו/נקלטו מ-Client.

כעת, נרצה לשנות את ה-Sniffer לקלוט פאקטה המגיעה תחת פרוטוקול TCP דרך פורט 23 מ-IP של Client, 10.0.2.5.

נשתמש בקוד הבא:

sniffer_tcp.py

```
#!/usr/bin/python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter='tcp port 23 and src host 10.0.2.5',prn=print_pkt)
```

ניתן לראות כי הגדרנו את הפילטר לפאקטה תחת פרוטוקול TCP דרך פורט 23 המגיעה ממקור IP של Client.

נריץ פקודת התקשרות telnet (אשר מתבצעת בפורט 23) ל-IP של Server, 10.0.2.6.

```
[09:37:59] 26/10/21 seed@client: telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
server login: seed
Password:
Last login: Tue Oct 26 09:33:01 IDT 2021 from 10.0.2.5 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

1 package can be updated.
0 updates are security updates.
```

החיבור התבצע בהצלחה.

במקביל, נריץ את ה-Sniffer:

```
[09:37:57] 26/10/21 seed@attacker: sudo python3 sniffer_tcp.py
sudo: unable to resolve host client
###[ Ethernet ]###
  dst      = 08:00:27:26:89:14
  src      = 08:00:27:c5:d4:0b
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x10
  len      = 60
  id       = 39612
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = tcp
  checksum = 0x87e5
  src      = 10.0.2.5
  dst      = 10.0.2.6
  \options \
###[ TCP ]###
  sport    = 34002
  dport    = telnet
  seq      = 2606969305
```

כפי שניתן לראות, מוצג לנו הפאקט, בפרוטוקול TCP, המגיע ממקור IP 10.0.2.5 תחת פורט telnet שהוא פורט 23.

עכשיו, נקלוט פאקטות מ-subnet ספציפי.

ראשית, נעדכן את ה-Sniffer:

```
sniff_subnet.py

#!/usr/bin/python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter='net 128.230.0.0/16', prn=print_pkt)
```

ה-Sniffer יקלוט פאקטות המגיעות ממקור או מיעד של subnet 128.230.0.0/16.

ב-Client נבצע Ping ל-IP מתוך ה-Subnet, למשל 128.230.0.1:

```
[00:32:52] 06/11/21 10.0.2.5@client: ping 128.230.0.1
PING 128.230.0.1 (128.230.0.1) 56(84) bytes of data.
64 bytes from 128.230.0.1: icmp_seq=1 ttl=46 time=142 ms
64 bytes from 128.230.0.1: icmp_seq=2 ttl=46 time=142 ms
64 bytes from 128.230.0.1: icmp_seq=3 ttl=46 time=142 ms
^C
--- 128.230.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
```

ניתן לראות כי 3 פאקטות נשלחו.

במקביל, נריץ את ה-Sniffer ב-attacker:

```
[00:41:30] 06/11/21 10.0.2.7@attacker: sudo python3 sniff_subnet.py
###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:c5:d4:0b
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 30936
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x34e5
  src      = 10.0.2.5
  dst      = 128.230.0.1 ←
  \options \
###[ ICMP ]###
  type     = echo-request
```

ניתן לראות כי הצלחנו לקלוט ולהציג את הפאקטה אשר הגיעה ליעד ה-Subnet אותו הגדרנו ב-Sniffer.

נמשיך לקלוט פאקטות:

```
###[ Ethernet ]###
  dst      = 08:00:27:c5:d4:0b
  src      = 52:54:00:12:35:00
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 421
  flags    =
  frag     = 0
  ttl      = 46
  proto    = icmp
  chksum   = 0xfe18
  src      = 128.230.0.1 ←
  dst      = 10.0.2.5
  \options \
###[ ICMP ]###
  type     = echo-reply
```

ניתן לראות כי הצלחנו גם לקלוט ולהציג את הפאקטה אשר ממקור ה-Subnet אותו הגדרנו ב-Sniffer.

לסיכום, ראינו ולמדנו שאנחנו יכולים לבצע קליטה (Sniffing) לפאקטות ע"פ פילטרים שונים – כגון פרוטוקולים (ICMP, TCP) וגם פורטים (23) וכן IP שונים. אפשר לראות כי הפאקטים אכן נקלטו והוצגו פרטיהם.

Spoofing ICMP Packets :Task 1.2

במשימה זו, ניצור פאקט מזויף (Spoof) בפרוטוקול ICMP, ממקור של IP מזויף. את הפאקט נשלח ל-Server.

ראשית, נבנה את התוכנית:

```
spoof.py
#!/usr/bin/python3
from scapy.all import *
a = IP(src="10.0.2.45", dst="10.0.2.6") ← 1
b=ICMP() ← 2
p=a/b ← 3
p.show() ← 4
send(p) ← 5
```

התוכנית מיועדת לזייף פאקטים ממקור IP מזויף (10.0.2.45) ליעד Server, תחת פרוטוקול ICMP. בנוסף, הקוד יציג לנו את פרטי ה- Packet לפני שישלח אותו.

מבנה הקוד:

- (1) בניית אובייקט IP, בעל מקור IP של 10.0.2.45 ובעל יעד של Server, 10.0.2.6.
- (2) בניית אובייקט פרוטוקול ICMP (בעל ערכי ברירת מחדל, ביניהם request – type=8).
- (3) אופרטור "/" מאפשר לחבר בין האובייקטים וליצור בעצם את הפאקטה שנשמרת במשתנה p.
- (4) הדפסה של פרטי הפאקטה.
- (5) שליחת הפאקטה ליעד.

כעת, נריץ את הקובץ:

```
[10:09:42] 26/10/21 seed@attacker:~$ sudo python3 spoof.py
sudo: unable to resolve host client
####[ IP ]####
version      = 4
ihl          = None
tos          = 0x0
len          = None
id           = 1
flags        = 
frag         = 0
ttl          = 64
proto        = icmp
chksum       = None
src          = 10.0.2.45
dst          = 10.0.2.6
\options     \
####[ ICMP ]####
type         = echo-request
code         = 0
chksum       = None
id           = 0x0
seq          = 0x0
unused      = ''
```

ניתן לראות כי יצירת הפאקטה התבצעה והודפסו פרטיה – פאקטה המגיעה מ-
IP 10.0.2.45 ליעד של 10.0.2.6 תחת פרוטוקול ICMP (Request).

כעת, ניגש ל-Wireshark:

Source	Destination	Protocol	Info
10.0.2.45	10.0.2.6	ICMP	Echo (ping) request id=0x0000, s
PcsCompu_26:89:14	Broadcast	ARP	Who has 10.0.2.45? Tell 10.0.2.6
PcsCompu_26:89:14	Broadcast	ARP	Who has 10.0.2.45? Tell 10.0.2.6
PcsCompu_26:89:14	Broadcast	ARP	Who has 10.0.2.45? Tell 10.0.2.6

ניתן לראות כי הפאקטה (Ping request) אכן נוצרה ונשלחה ל-Server
מהאיפי המזויף (10.0.2.45), אך מכיוון ש-10.0.2.45 אינה כתובת איפי
פעילה, לא הוחזרה תגובה (server ניסה לשלוח ARP request לכתובת ה-
MAC של 10.0.2.45 ומכיוון שלא קיבל תגובה, לא יכל לשלוח תגובה חזרה כי
לא ידע לאן).

לעומת זאת, אם נעשה אותו דבר עם כתובת אייפי פעילה, יתקבל גם ping reply.

ננסה, כעת נשנה את התוכנית לכתובת האייפי 8.8.8.8, ונריץ אותה:

```
[10:10:42] 26/10/21 seed@attacker:sudo python3 spoof.py
sudo: unable to resolve host client
####[ IP ]####
version      = 4
ihl          = None
tos          = 0x0
len          = None
id           = 1
flags        =
frag         = 0
ttl          = 64
proto        = icmp
chksum       = None
src          = 8.8.8.8
dst          = 10.0.2.6
\options     \
####[ ICMP ]####
type         = echo-request
code         = 0
chksum       = None
id           = 0x0
seq          = 0x0
unused       = ''
```

התוכנית כעת שלחה Ping request מ-IP 8.8.8.8.

ונסתכל על ה-Wireshark:

8.8.8.8	10.0.2.6	ICMP	Echo (ping) request
10.0.2.6	8.8.8.8	ICMP	Echo (ping) reply

כעת ניתן לראות כי נשלח ping request ו-server החזיר reply לכתובת (על אף שהיא כלל לא שלחה לו את ה-ping).

לסיכום, ראינו ולמדנו שאנחנו יכולים לבצע שליחה של פאקטים מזויפים (spoofing), וכן הוכחנו בעזרת Wireshark שהפאקטים נשלחו והתקבלו ואף קיבלו תגובה חזרה מ-server.

Traceroute :Task 1.3

Traceroute הוא כלי שמאפשר לנו לדעת איך הפאקט ששלחנו עובר מהמקור עד ליעד, כך שהוא מראה לנו דרך כמה ראוטרים ואילו ראוטרים הוא עובר.

נרצה לממש כלי שיעזור לנו לעשות זאת, באמצעות שליחה של פאקט בפרוטוקול ICMP עם פרמטר TTL מסוים כך שכל פעם ה-TTL יגמר והפאקט "יפול" תוחדר לנו הודעה עם הראוטר שבו זה נפל. כך נעלה את פרמטר ה-TTL עד שהפאקט יחזור ליעדו.

ראשית, בנינו תוכנית אשר מבצעת Ping לכתובת אייפי של שרתי גוגל, 8.8.8.8.

```
trace.py
#!/usr/bin/python3
from scapy.all import *
a = IP()
a.dst = '8.8.8.8'
a.ttl = 1
b = ICMP()
send(a/b)|
```

התחלנו כאשר ה-TTL הוא 1, וכל פעם העלנו אותו באחד. נריץ את התוכנית:

```
[10:50:54] 26/10/21 seed@client: sudo python3 trace.py
sudo: unable to resolve host client
.
Sent 1 packets.
[10:51:14] 26/10/21 seed@client: sudo python3 trace.py
sudo: unable to resolve host client
.
Sent 1 packets.
```

אחרי כל הרצה, נעלה את ה-TTL באחד.

במקביל, הסתכלנו על ה-Wireshark ובדקנו את הפאקטים המתקבלים מה-Ping שלנו.

Time	Source	Destination	Protocol	Info
2021-10-26 10:51:14...	10.0.2.1	10.0.2.5	ICMP	Time-to-live exceeded
2021-10-26 10:51:23...	192.168.0.1	10.0.2.5	ICMP	Time-to-live exceeded
2021-10-26 10:51:35...	192.162.0.1	10.0.2.5	ICMP	Time-to-live exceeded
2021-10-26 10:51:47...	213.57.115.197	10.0.2.5	ICMP	Time-to-live exceeded
2021-10-26 10:53:01...	142.250.160.230	10.0.2.5	ICMP	Time-to-live exceeded
2021-10-26 10:53:14...	142.250.239.67	10.0.2.5	ICMP	Time-to-live exceeded
2021-10-26 10:53:23...	172.253.71.89	10.0.2.5	ICMP	Time-to-live exceeded

האייפי של הראוטרים בהם הפאקט עבר ונפל במהלך הריצה שלו כל פעם.

לבסוף, רק לאחר 10 הרצות (TTL=10):

Wireshark					
2021-10-26 10:53:30...	PcsCompu_c5...	Broadcast	ARP	42	Who has 10.0.2.1? Te
2021-10-26 10:53:30...	RealtekU_12	PcsCompu_c5	ARP	60	10.0.2.1 is at 52:54
2021-10-26 10:53:30...	10.0.2.5	8.8.8.8	ICMP	42	Echo (ping) request
2021-10-26 10:53:30...	8.8.8.8	10.0.2.5	ICMP	60	Echo (ping) reply
2021-10-26 10:59:15...	10.0.2.5	10.0.2.5	DHCP	342	DHCP Request - Tran
2021-10-26 10:59:15...	10.0.2.5	10.0.2.5	DHCP	500	DHCP ACK Tran

קיבלנו Reply משרת גוגל 8.8.8.8, כלומר, הפאקט הגיע ליעד.

לסיכום, למדנו על כלי ה-Traceroute אשר מאפשר לנו לראות את ה"מסע" של הפאקט ודרך הראוטרים שהוא עובר, ועל פרמטר ה-TTL אשר למעשה מאפשר לו לעשות את זה. ניתן לראות שהצלחנו לממש כלי זה בעזרת שליחת פאקטים, ואכן ראינו את הראוטרים וכתובת האייפי שהוא עבר עד שהגיע ליעדו.

Sniffing and-then Spoofing :Task 1.4

במשימה זו נרצה לשלב את כלי ה-Sniff וה-Spoof מהמשימות הקודמות, כאשר נעשה Ping מ-server לכתובת IP כלשהי, ובמקביל נבנה תוכנית ב-client אשר תקלוט (sniff) את ה-Ping request ותיצור בעזרתו (spoof) פאקט reply אשר יחזור ל-server.

ראשית, השתמשנו בתוכנית הבאה:

sniff_spoof.py

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

def spoof_pkt(pkt):
    2 if pkt[ICMP].type == 8:
        3 a = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
        4 b = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
        5 data = pkt[Raw].load
        newpacket = a/b/data
        send(newpacket)

1 pkt = sniff(filter='icmp', prn=spoof_pkt)|
```

התוכנית תקלוט כל פאקט אשר מתקבל בפרוטוקול ICMP (sniff), תיקח את הפאקט ותיצור ממנו פאקט reply לכתובת השולחת, כאשר תזייף את כתובת המקור, כאילו נשלחה מהיעד המכוון (spoof).

מבנה הקוד:

- (1) התוכנית מתחילה בהפעלת פונקציית sniffer אשר קולטת פאקטות תחת פילטר של ICMP.
- (2) כל פאקטה הנקלטת נשלחת לפונקציית spoof_pkt. בפונקציה ישנו if אשר בודק אם ה-ICMP שנקלט הוא request (type=8).
- (3) לאחר מכן, אנו למעשה עושים spoofing לפאקטה חדשה, הבנויה על הפרטים של הפאקטה שהתקבלה. אנחנו בונים אובייקט IP הבנוי מהיעד והמקור של הפאקטה הפוכים (ע"מ להחזיר אותה לשולח), וה- header length יהיה זהה (ihl).
- (4) אנו בונים אובייקט ICMP הבנוי למעשה מה-ID וה-sequence number הזוהים לפאקטה המקורית, רק נשנה את ה-type=0, כלומר, אנו שולחים reply. במידה ויש data בפאקטה, נעתיק גם אותה לפאקטה החדשה, ואז נשלח את הפאקטה ליעדה.

נריץ את התוכנית ב-Attacker:

```
[19:56:34] 26/10/21 seed@attacker:~$ sudo python3 sniff_spoof.py
sudo: unable to resolve host client
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

התוכנית רצה.

במקביל, נבצע ב-Server פקודת Ping לכתובת IP שאינה קיימת – 1.2.3.4.

```
[20:02:18] 26/10/21 seed@server:~$ ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data:
64 bytes from 1.2.3.4: icmp_seq=1 ttl=64 time=11.7 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=15.7 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=25.9 ms
^C
--- 1 2 3 4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 11.770/17.821/25.970/5.983 ms
```

ניתן לראות כי אנו מקבלים תגובה מכתובת ה-IP, אף על פי שמדובר ב-IP שלא קיים ואינו פעיל.

נוכל לבדוק גם ב-Wireshark:

Time	Source	Destination	Protocol	Info
2021-10-26 ...	10.0.2.6	1.2.3.4	ICMP	Echo (ping) request
2021-10-26 ...				<Ignored>
2021-10-26 ...				<Ignored>
2021-10-26 ...	1.2.3.4	10.0.2.6	ICMP	Echo (ping) reply
2021-10-26 ...	10.0.2.6	1.2.3.4	ICMP	Echo (ping) request
2021-10-26 ...	1.2.3.4	10.0.2.6	ICMP	Echo (ping) reply
2021-10-26 ...	10.0.2.6	1.2.3.4	ICMP	Echo (ping) request
2021-10-26 ...	1.2.3.4	10.0.2.6	ICMP	Echo (ping) reply

ניתן לראות כי אנו מקבלים Reply ל-Ping שביצענו ל-IP 1.2.3.4.

כעת, נבצע Ping לכתובת אייפי אשר כן קיימת – 8.8.8.8.

```
[20:02:55] 26/10/21 seed@server: ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=9.35 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=57 time=52.0 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=6.47 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=57 time=53.6 ms (DUP!)
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates 0% packet
```

ניתן לראות כי אנו מקבלים פעמיים Reply (DUP = Duplicate)

נסתכל גם ב-Wireshark:

Time	Source	Destination	Protocol	Info
2021-10-26 ...	10.0.2.6	8.8.8.8	ICMP	Echo (ping) request
2021-10-26 ...				<Ignored>
2021-10-26 ...				<Ignored>
2021-10-26 ...	8.8.8.8	10.0.2.6	ICMP	Echo (ping) reply
2021-10-26 ...	8.8.8.8	10.0.2.6	ICMP	Echo (ping) reply
2021-10-26 ...	10.0.2.6	8.8.8.8	ICMP	Echo (ping) request
2021-10-26 ...	8.8.8.8	10.0.2.6	ICMP	Echo (ping) reply
2021-10-26 ...	8.8.8.8	10.0.2.6	ICMP	Echo (ping) reply

גם פה, ניתן לראות כי מתקבל 2 Reply.

ההסבר לזה הוא ש-reply אחד מתקבל כתגובה אמיתית מה-IP אליו נשלח ה-Ping, ו-reply שני מגיע מהתוכנית שהרצנו, שזייפה Reply.

לסיכום, ראינו כי ביכולתנו לקלוט פאקט וגם ליצור אותו, ובכך ליצור תגובה מזויפת ל-Ping request. ראינו בעזרת Wireshark וגם בעזרת ה-Ping בטרמינל עצמו כי אכן הצלחנו ליצור ולשלוח reply מזויף ל-server.

סיכום המעבדה

המעבדה הכניסה אותנו לעולם של הלינוקס, התעסקות עם הטרמינל ופקודות לינוקס. זה היה מאתגר מאחר ואף פעם לא התעסקנו בזה בעבר. למדנו קצת יותר מקרוב מה זה פאקטים הנשלחים ברשת, ואיך ניתן לקלוט פאקטים ע"פ פילטרים מסוימים, או לזייף אותם מכל IP בעזרת שפת פייתון עם החבילה של Scapy. המעבדה דרשה מאיתנו חזרה על כל נושא רשתות מחשבים ולחזור על מושגים אשר לא היו מספיק חזקים אצלנו כגון פרוטוקולים מסוימים (ICMP, TCP).

בנוסף, למדנו להתעסק עם תוכנת Wireshark המאפשרת לנו לצפות בפאקטים העוברים ברשת, מה שעזר לנו להבין ולהבחין בנעשה ברשת ולבדוק את תוצאות הניסוי שלנו.

שיטת ה-Sniffing & spoofing היא בסיס להרבה מתקפות סייבר ברשת, כמו לדוגמה מתקפת Man in the middle.