# מטלה 5

ת"ז מגישים: 209326776, 322998287

### :Task A

הסברים על הקוד נמצאים בתוך הקוד. הרצנו את הקבצי python ממטלה 2 שעובדות על tcp

ולכן בקובץ שלנו הסנפנו פקאטות TCP . והדפסנו את השדות המבוקשים .

```
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 9999, Dst Port: 32824, Seq: 1, Ack: 688, Len: 564
Source Port: 9999
Destination Port: 32824

TCP payload (564 bytes)

TCP segment data (564 bytes)
```

נתונים אלה הם מתוך פאקטה 6 (אנו התחלנו את הספירה מ0 לכן זו אותה פאקטה) וניתן לראות כי נתונים אלה תואמים להדפסה.

```
0.000033907 127.0.0.1 127.0.0.1 TCP 66 32824 - 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=444965727 Tsecr=444965727 Tsecr=444965727 TcP segment of a reassembled PDU] 0.000054959 127.0.0.1 127.0.0.1 TCP 66 9399 - 32824 [ACK] Seq=1 Ack=1 Win=65536 Len=697 Tsval=444965727 Tsecr=444965727 [TCP segment of a reassembled PDU] 0.00005490 127.0.0.1 127.0.0.1 TCP 66 9399 - 32824 [ACK] Seq=1 Ack=688 Win=65536 Len=564 Tsval=444965727 Tsecr=444965727 TCP segment of a reassembled PDU] 0.001590652 127.0.0.1 127.0.0.1 TCP 630 9399 - 32824 [PSH, ACK] Seq=1 Ack=688 Win=65536 Len=564 Tsval=444965727 Tsecr=444965727 TCP segment of a reassembled PDU] 0.001590652 127.0.0.1 127.0.0.1 TCP 630 9399 - 32824 [PSH, ACK] Seq=7 Ack=688 Win=65536 Len=564 Tsval=444965727 Tsecr=444965727 TCP segment of a reassembled PDU] 0.001590652 127.0.0.1 127.0.0.1 TCP 630 9399 - 32824 [PSH, ACK] Seq=7 Ack=688 Win=65536 Len=564 Tsval=444965729 Tsecr=444965727 TCP segment of a reassembled PDU] 0.001590652 127.0.0.1 TCP 0.001590652 127.
```

מתבצעת השליחה ממטלה 2.

<u>תשובה לשאלה:</u> אנו צריכים הרשאת מנהל כל פעם שאנו רוצים לגשת לממשק ברשת, כיוון שרק אנו יכולים להאזין promiscuous mode ורק במצב של promiscuous modeמנהל יכול לגשת ל לפאקטות שלא מיועדות לנו ברשת.

המערכת תכשל בשורה:

```
handle = pcap_open_live("lo", BUFSIZ, 1, 1000, errbuf);
```

## היכולות והמגבלות של ה:sniffer

הסניפר לוכד ומנתח פאקטות, בקוד שלנו אנו מסניפים TCP מנתח אותם ואז הוא צורב את הניתוח לתוך קובץ טקסט עם התז שלנו.

הסניפר יכול לנתח רק מה שנשלח בברטיס רשת שהוא מחובר אליו (אצלנו lo) בנוסף הוא לא יכול לנתח תעבורה שהיא מוצפנת . (הסיבה היא שההצפנה נועדה למנוע גישה)

#### :Task B

בחלק זה כתבנו את ה-spoofer.c בו זייפנו פאקטות

## אופן ההרצה של Spoofer:

נפתח טרמינל בתקייה בה נמצא הקוד ונריץ את הפקודות הבאות:

```
renana@renana: ~/Desktop/Ex5

renana@renana: ~/Desktop/Ex5$ gcc Spoofer.c -o spoofer
renana@renana: ~/Desktop/Ex5$ sudo ./spoofer

ICMP sended
renana@renana: ~/Desktop/Ex5$
```

## <u>הסבר על הקוד:</u>

וcmp() שלנו יש קריאה לפונקציה maina בפונקציית

```
int main() {
  icmp();
  printf("ICMP sended\n");

// udp();
// printf("UDP sended\n");

// tcp();
// printf("TCP sended\n");
  return 0;
}
```

source ipa ולשנות בו את ICMP header ולכן נרצה לבנות ICMP header מטרתנו בקוד היא לזייף פאקטת

בתוך הפונקציה בשלב ראשון נמלא את הheader:

: IP header בשלב שני נמלא את

לבסוף נשלח את הפאקטה המזויפת:

```
send_raw_ip_packet (ip);
```

הפונקציה in\_cksum מחשבת in\_cksum עבור

הפונקציה send\_raw\_ip\_packet פותחת send\_raw\_ip\_packet הפונקציה

## :wireshark הקלטות

ikmp									
No.	Time	Source	Destination	Protocol	Length Info				
	1 0.000000000	1.2.3.4	10.0.2.15	ICMP	44 Echo (ping) request	id=0x0000, seq=0/0, ttl=20 (rep	ly in 2)		
	2 0.000010070	10.0.2.15	1.2.3.4	ICMP	44 Echo (ping) reply	id=0x0000, seq=0/0, ttl=64 (req	uest in 1)		

Checksum: Oxf7ff [correct] -> [Checksum Status: Good]

ניתן לראות כי <mark>הsource address</mark> הוא 1.2.3.4 המזוייף שאותו אנחנו הכנסנו.

כדי לראות שקיבלנו תגובה ווליאדית שמנו את ה<mark>ip של המחשב שלי</mark> 10.0.2.15, ניתן לראות כי נשלח ping בחזרה.

#ענינו על הדרישה שהקוד יעשה spoof לפאקטות אחרות עם שינויים קטנים, בכך שעשינו שתי פונקציות נוספות(udp(); tcp()

תשובה לשאלה 1: כן אנחנו יכולים לשנות את הגודל של פאקטת הpi לערך שרירותי כיוון שאורך פקאטת הור? משתנה לגודלו המקורי, ללא קשר לגודל שאנחנו קובעים.

### תשובה לשאלה 2:

לא, אנו יוצרים socket ומבצעים bind, ואז השדות של הp header לא, אנו יוצרים socket ומבצעים טוחלם, השדות של היוצרים יכולים לתקשר עם השרת. ַומערכת ההפעלה בונה את הפאקטה כולל הchecksum עם הנתונים שלנו.

## היכולות וההגבלות של הspoofer:

המטרה של התוכנית היא לזייף את הפאקטות וה ip שלהם ברשת.

ה spoofer שלנו יכול לזייף פאקטות מסוג TCP UDP ICMP. הספופר מזייף אץ ה ip שלנו יכול לזייף פאקטות מסוג תובה חוקית.

הספופר מוגבל בכך שהוא לא יוכל לעקוף אמצעי אבטחה, ואפשר לזהות אותו ברשת.

#### :Task C

לחלק כזה כתבנו את הקוד Snooper שעושה sniff&spoof לחלק כזה כתבנו את הקוד Spoofer שאופן ההרצה של Spoofer:

attacker hostB hostA dockerם ביcontainer פתחנו שלושה

```
Container hostA-10.9.0.5 Created
Container hostB-10.9.0.6 Created
Container seed-attacker Created
```

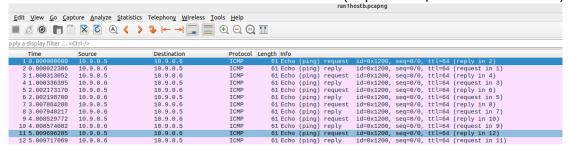
.attacker ואת hostA בעוד שתי טרמינילים שונים הרצנו את

בחלק זה עשינו sniffing וברגע שהוא קיבל gotpacet נכנסנו לפונקציה בה עשינו הדפסות של dst בחלק זה עשינו מעקב שלנו ובאותה פונקציה עשינו שליחה חוזרת לsrc ששלח את ההודעה(כלומר שליחה חוזרת למי ששלח את ההודעה, החלפנו בין ה dst לsrc בשליחה).

הרצה ראשונה: שליחת ping מA Host Bל Host B: נבחין בתמונה מעל כי הקו של hostB הוא 10.9.0.6.

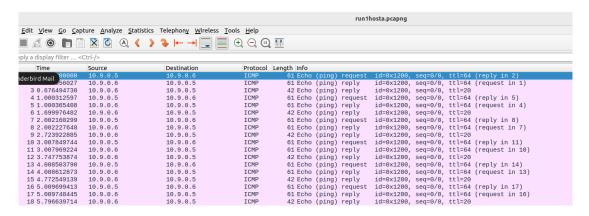
```
root@37ce81cad40a:/volumes# ./ping 10.9.0.6
                                                                                     Version: 4
ping 10.9.0.6: 19 data bytes
                                                                                    From: 10.9.0.5
                                                                                      To: 10.9.0.6
47 bytes from 10.9.0.6 icmp_seq= 0.ttl=64 RTT: 0.115000 milliseconds
                                                                                   GOT PACKET
28 bytes from 10.9.0.6 icmp_seq= 1.ttl=20 RTT: 0.111000 milliseconds
                                                                                    Version: 4
                                                                                    From: 10.9.0.5
47 bytes from 10.9.0.6 icmp_seq= 2.ttl=64 RTT: 0.154000 milliseconds
                                                                                      To: 10.9.0.6
28 bytes from 10.9.0.6 icmp_seq= 3.ttl=20 RTT: 0.173000 milliseconds
                                                                                   GOT PACKET
                                                                                    Version: 4
47 bytes from 10.9.0.6 icmp_seq= 4.ttl=64 RTT: 0.209000 milliseconds
                                                                                    From: 10.9.0.5
To: 10.9.0.6
28 bytes from 10.9.0.6 icmp_seq= 5.ttl=20 RTT: 0.096000 milliseconds
                                                                                   GOT PACKET
                                                                                     Version: 4
                                                                                    From: 10.9.0.5
                                                                                      To: 10.9.0.6
                                                                                   GOT PACKET
                                                                                    Version: 4
                                                                                    From: 10.9.0.5
To: 10.9.0.6
                                                                                   OT PACKET
```

אר מכן הרצנו את attacker ולאחר מכן הרצנו את מכן הרצנו פינג מA ל attacker הרצנו את הרצנו את הדפסנו את ip src dest

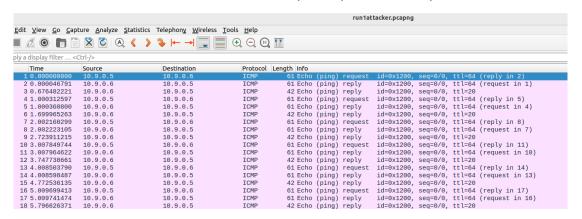


רואים שהוא מקבל בקשה ומחזיר תגובה

Version: 4 From: 10.9.0.5 To: 10.9.0.6



כמו שציפנו אנו רואים כאן שהוא שולח בקשה ומקבל 2 תגובות אחת נ host B ואחת מ-attacker.



תוך כדי ההרצה פתחנו 3 ווירשארק כאשר אחד על ממשק של הattacker אחד של הhostA ואחד על hostB hostB ניתן לראות שיש שני reply זה נובע מכך שה attacker מתערב בשליחה של הפינגים .

#### :8.8.8.8 אל hostA הרצה 2

Version: 4

Version: 4
From: 10.9.0.5

Version: 4

Version: 4

Version: 4 From: 10.9.0.5

GOT PACKET

**GOT PACKET** 

**GOT PACKET** 

ash T PACKET

From: 10.9.0.5

To: 8.8.8.8

To: 8.8.8.8

From: 10.9.0.5 To: 8.8.8.8

From: 10.9.0.5

To: 8.8.8.8

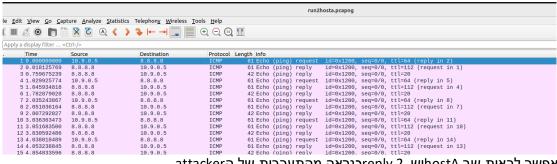
To: 8.8.8.8

. נציין שבהרצה2 בממשק של hostB לא הוסנפו שום פאקטות כפי שציפנו

```
47 bytes from 8.8.8.8 icmp_seq= 0.ttl=112 RTT: 18.240000 milliseconds
                      28 bytes from 8.8.8.8 icmp_seq= 1.ttl=20 RTT: 0.122000 milliseconds
47 bytes from 8.8.8.8 icmp_seq= 2.ttl=112 RTT: 0.164000 milliseconds
                      28 bytes from 8.8.8.8 icmp_seq= 3.ttl=20 RTT: 0.104000 milliseconds
                                                                                                                                                                run2attacker.pcapng
<u>Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help</u>
ly a display filter
                                                                                                                                                                         1d-0x1200, seq=0/0, ttl=64 (reply in 2)
id-0x1200, seq=0/0, ttl=12 (request in 1)
id-0x1200, seq=0/0, ttl=20
id-0x1200, seq=0/0, ttl=64 (reply in 5)
id-0x1200, seq=0/0, ttl=112 (request in 4)
id-0x1200, seq=0/0, ttl=112 (request in 4)
id-0x1200, seq=0/0, ttl=64 (reply in 8)
id-0x1200, seq=0/0, ttl=64 (reply in 7)
id-0x1200, seq=0/0, ttl=112 (request in 7)
id-0x1200, seq=0/0, ttl=20
id-0x1200, seq=0/0, ttl=64 (reply in 11)
id-0x1200, seq=0/0, ttl=64 (reply in 10)
id-0x1200, seq=0/0, ttl=64 (request in 10)
id-0x1200, seq=0/0, ttl=20
                                                                                                                               61 Echo (ping) request
61 Echo (ping) reply
42 Echo (ping) reply
61 Echo (ping) request
   3 0.759655426
   4 1.029925774
                               10.9.0.5
                                                                     8.8.8.8
                                                                                                           ICMP
                                                                                                                               61 Echo (ping) request
61 Echo (ping) reply
42 Echo (ping) reply
61 Echo (ping) request
61 Echo (ping) reply
42 Echo (ping) reply
61 Echo (ping) reply
61 Echo (ping) reply
42 Echo (ping) reply
   5 1.045917116
                               8.8.8.8
                                                                     10.9.0.5
                                                                                                           ICMP
   6 1.782871448
                               8.8.8.8
                                                                     10.9.0.5
                                                                                                           ICMP
     2.035243867
                               10.9.0.5
                                                                     8.8.8.8
                                                                                                           TCMP
 8 2.051020049
9 2.807283963
10 3.036363473
11 3.051654483
                                                                                                           ICMP
ICMP
ICMP
ICMP
                                                                                                                               42 Echo (ping) reply
61 Echo (ping) request
61 Echo (ping) reply
42 Echo (ping) reply
  12 3.830582016
                                                                                                                                                                           id=0x1200, seq=0/0,
                                                                                                                                                                                                              ttl=20
                                                                                                                                                                                                              ttl=64 (reply in 14)
  13 4.038018489
                               10.9.0.5
                                                                     8.8.8.8
                                                                                                           ICMP
                                                                                                                                                                           id=0x1200, seq=0/0,
 14 4.053219553 8.8.8.8
15 4.854788500 8.8.8.8
                                                                                                                                                                          id=0x1200, seq=0/0, ttl=112 (request in 13) id=0x1200, seq=0/0, ttl=20
                                                                     10.9.0.5
```

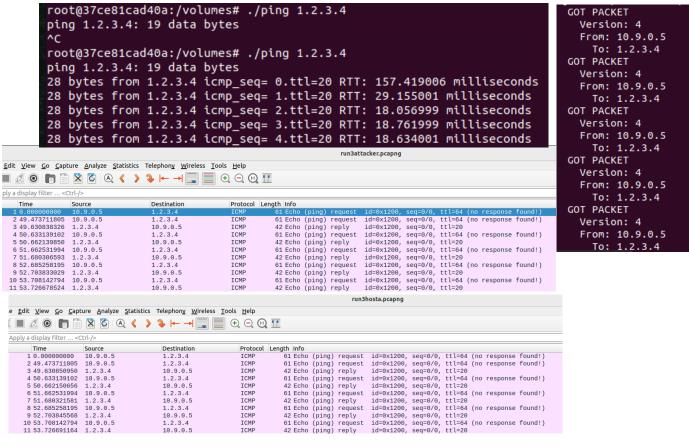
root@37ce81cad40a:/volumes# ./ping 8.8.8.8

ping 8.8.8.8: 19 data bytes



.attacker אפשר לראות שב hostAיש 2 reply כנראה מהתערבות של

## הרצה 3: מhost A לוו מזוייפת:



מה שעשינו הוא שהרצנו את hostA מבלי להריץ את הattacker מבלי להריץ את no response found וגם בשורה הראשונה שיש C^ וגם בשורה נתקע עד שעשינו לאחר מכן הרצנו את ה attacker ושוב הרצנו פינג ואז רואים שהתקבל תשובה שמראה לנו שזה הattacker שעונה.

## :task Cל

כדי להראות שהתהליך שלנו באמת התבצע כראוי שנינו בקוד את ה cכדי

host b לhost a הרצנו פינג מ

```
root@37ce81cad40a:/volumes# ./ping 10.9.0.6
ping 10.9.0.6: 19 data bytes
47 bytes from 10.9.0.6 icmp_seq= 0.ttl=64 RTT: 0.218000 milliseconds
28 bytes from 1.2.3.4 icmp_seq= 1.ttl=20 RTT: 0.134000 milliseconds
28 bytes from 1.2.3.4 icmp_seq= 2.ttl=20 RTT: 0.194000 milliseconds
```

#### : wiresharkב ניתן לראות

```
10.9.0.6
10.9.0.5
10.9.0.5
10.9.0.5
10.9.0.6
10.9.0.6
10.9.0.6
10.9.0.6
10.9.0.5
10.9.0.5
10.9.0.5
2 0.000117463
3 0.338175747
4 0.338301261
5 0.338324873
6 0.338376215
7 0.338406474
8 0.338431850
9 1.006537789
10 1.006596006
11 1.359974310
12 1.360024887
13 1.360046761
                                                                   14 1.360101403
                              1.2.3.4
14 1.369191493
15 1.369142398
16 1.369173438
17 2.919225839
18 2.382274883
19 2.382372545
20 5.938219391
21 5.938225148
22 5.938296192
                                                                                                          ICMP
ICMP
ICMP
ICMP
ARP
ARP
ARP
ARP
ARP
ARP
ARP
 24 5.549201995
 25 5.549235934
 26 5.549562850
                              02:42:0a:09:00:06
27 5.549568446
                              02:42:0a:09:00:05
                                                                    02:42:8a:3d:78:02
02:42:8a:3d:78:02
02:42:0a:09:00:05
                                                                                                                               42 Who has 10.9.0.1? Tell 10.9.0.5
42 10.9.0.1 is at 02:42:8a:3d:78:02
```

בrun1 אנחנו רואים הרבה פעמים שמתקבל no response found מכוון שה host זיהה שהפונגים מחלבה אנחנו רואים הרבה פעמים שמתקבל attacker מלו מבחינה כלשהוא עד שהוא קיבל attacker לא מתאימים לשלו מבחינה כלשהוא עד שהוא קיבל attacker שלח פונגים שכוון שאנחנו יודעים ש cattacker שולח פונגים לה host a ווידאנו זאת ע"י זה ש attacker מלו. מבוון שאנחנו יודעים שנשלח ההודעות שלו.

### attacker:נעשה הרצה של הפינג ללא

1 0.000000000	10.9.0.5	10.9.0.6	ICMP	61 Echo (p	oing) re	equest	id=0x1200,	seq=0/0,	ttl=64	(reply in 2)
2 0.000098578	10.9.0.6	10.9.0.5	ICMP							(request in 1)
3 1.012406496	10.9.0.5	10.9.0.6	ICMP							(reply in 4)
4 1.012452726	10.9.0.6	10.9.0.5	ICMP							(request in 3)
5 2.015189597	10.9.0.5	10.9.0.6	ICMP	61 Echo (p	oing) re	equest	id=0x1200,	seq=0/0,	ttl=64	(reply in 6)
6 2.015249069	10.9.0.6	10.9.0.5	ICMP	61 Echo (p	oing) re	eply	id=0x1200,	seq=0/0,	ttl=64	(request in 5)
7 3.015846167	10.9.0.5	10.9.0.6	ICMP	61 Echo (p	oing) re	equest	id=0x1200,	seq=0/0,	ttl=64	(reply in 8)
8 3.015876952	10.9.0.6	10.9.0.5	ICMP	61 Echo (p	ing) re	eply	id=0x1200,	seq=0/0,	ttl=64	(request in 7)

ניתן לראות שכך הוא עובד כראוי שולח בקשה ומקבל תשובה , ניתן להבחין בכתובת של ההוסטים.

### :Task D

הסברים על הקוד נמצאים בתוך הקוד.

נציין שאצלנו הוא מאזין ל"0.0.0.0.0" אז הוא מאזין לכל הכתובות.

ניתן לראות שזה שולח כאש<mark>ר rand>0.5</mark>

וכאשר rand<0.5 לא שולח

אצלנו ניתן לראות שנשלחו 4 פאקטות וזה תואם למה שרואים בווירשארק

```
renana@renana:~/Desktop/Ex5$ gcc Gateway.c -o gateway
renana@renana:~/Desktop/Ex5$ sudo ./gateway 127.0.0.7
in the main
host_ip 127.0.0.7
cerate host sock
cerate sock p
bind host sock
bind p sock
enter to whileloop
rand 0.840188
******SEND******
 enter to whileloop
receve
rand 0.394383
 enter to whileloop
receve
rand 0.783099
******SEND****
 enter to whileloop
rand 0.798440
*******SEND*******
enter to whileloop
rand 0.911647
******SEND******
 enter to whileloop
rand 0.197551
```

שלחנו לו פאקטות UDP בצורה הבאה-> אל P) port 8000)

```
renana@renana:~/Desktop/Ex5$ echo -n example | nc -4u -w1 10.9.0.1 8000
renana@renana:~/Desktop/Ex5$ echo -n example | nc -4u -w1 10.9.0.1 8000
renana@renana:~/Desktop/Ex5$ echo -n example | nc -4u -w1 10.9.0.1 8000
renana@renana:~/Desktop/Ex5$ echo -n example | nc -4u -w1 10.9.0.1 8000
renana@renana:~/Desktop/Ex5$ echo -n example | nc -4u -w1 10.9.0.1 8000
renana@renana:~/Desktop/Ex5$ echo -n example | nc -4u -w1 10.9.0.1 8000
renana@renana:~/Desktop/Ex5$
```

Time	Source	Destination	Protocol	Length	Info
17 15.517076570	10.0.2.15	10.9.0.1	UDP	51	57789 → 8000 Len=7
18 15.517349153	127.0.0.7	10.0.2.15	UDP	51	8001 → 57789 Len=7
20 18.405347213	10.0.2.15	10.9.0.1	UDP	51	58371 → 8000 Len=7
25 28.433202145	10.0.2.15	10.9.0.1	UDP	51	60659 → 8000 Len=7
26 28.433354467	127.0.0.7	10.0.2.15	UDP	51	8001 → 60659 Len=7
28 30.241388777	10.0.2.15	10.9.0.1	UDP	51	36972 → 8000 Len=7
29 30.241527461	127.0.0.7	10.0.2.15	UDP	51	8001 → 36972 Len=7
51 32.225851325	10.0.2.15	10.9.0.1	UDP	51	53799 → 8000 Len=7
52 32.225951333	127.0.0.7	10.0.2.15	UDP	51	8001 → 53799 Len=7
54 34.599003784	10.0.2.15	10.9.0.1	UDP	51	44479 → 8000 Len=7

שורות -1 הרנדומלי

2: אנו יודעים לפי ההדפסה שעשינו בטרמינל כי המספר 2: שהוגרל הוא 0.5<0.840188 שהוגרל הוא 10.5 <u>Dst Port: 8001</u>

ל כן אנו נכנסים אל תוך הfi ומעבירים את ה host sockh datagram שיושב על P+1. אכן:

שורה 3: עפ"י הטרמינל אנו יודעים כי המספר הרנדומלי שיצא הוא 0.5>0.3994383 כלומר לא מתבצעת שליחה ואכן אנו רואים כי בשורה הבאה הוא כבר עובר הלאה ולא שולח.

שורה 4: עפ"י הטרמינל rand=00.783099>0.5 כלומר

https://stackoverflow.com/questions/70449344/if-i-have-populated-all-fields-of-iphdr-then-how-to-calculate-the-tcphdr-doff-all-fields-of-iphdr-d

/https://www.opensourceforu.com/2015/03/a-guide-to-using-raw-sockets

https://stackoverflow.com/questions/11383497/libpcap-payload-offset-66-but-sizeofheaders-doff-62