

HW5_NetworkOfNetowrks

IDs :-

208113381

211990700

our zip contains:

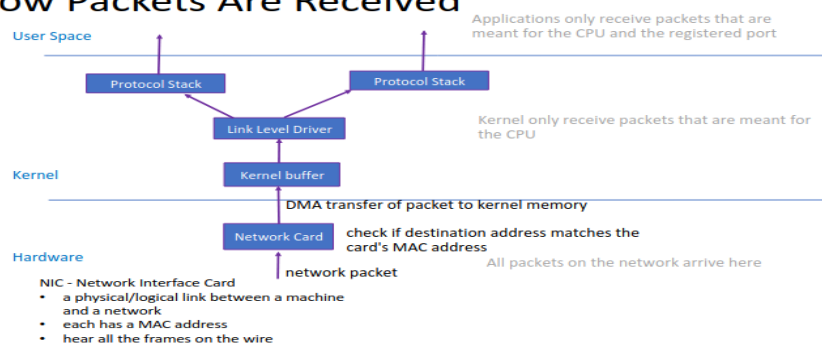
1. part a - Sniffer.c, the output .txt file from the sniffer and wireshark recording
2. part b - Spoofer.c and wireshark recording
3. part c - snoofer.c and 3 wireshark recordings of the 3 requests.
##notice - inside the dockers we compiled snoofer.c manually and we used the regular ping instead of Ex4. therefore we don't have a makefile for this part. in order to compile part c inside the docker use the next command:
"gcc -o snoofer snoofer.c -lpcap" and run it with "./snoofer" command
4. part d - Gateway.c, sender.c
5. make file that will compile all files with the command "make all"

Task A:- Sniffing.c

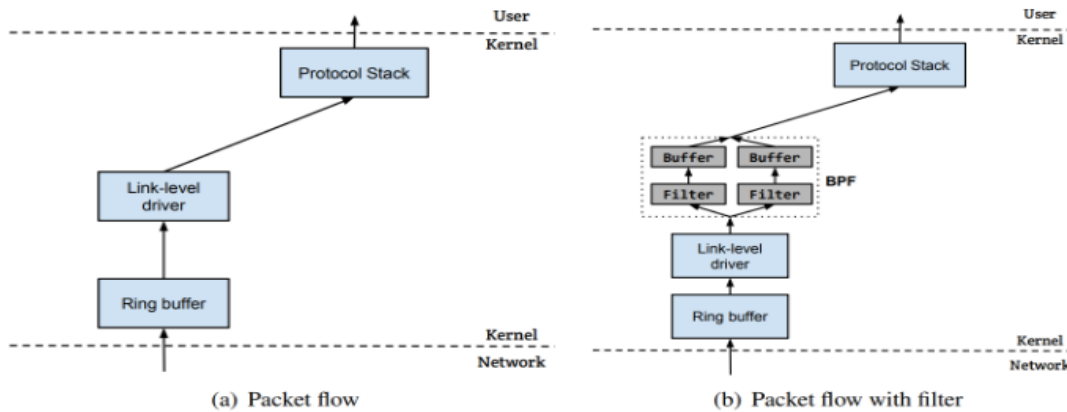


6

How Packets Are Received



Packet Flow With/Without Filters



16

Questions:-

1.) why do you need the root privilege to run a sniffer program?

Because to put the network adapter into promiscuous mode requires root privileges.

2.) Where does the program fail if it is executed the root without the root privilege?

If it didn't, any user could fully control the network adapter, then every user could see the full traffic on that controller, including all the traffic of other users.

A packet sniffer — is a software used to monitor network traffic. Sniffers work by examining streams of data packets that flow between computers on a network. These packets are intended for and addressed to specific machines, but using a packet sniffer in "promiscuous mode" allows IT professionals, end users or malicious intruders to examine any packet, regardless of destination.

Limitations

- Your system administrator might not allow the use of packet capture software due to security concerns with intercepting traffic. so, if you want to perform a “man in the middle” attack, consider being or getting rid of your current system administrator first (if someone asks, we never said that)



- On non-Windows systems, root permissions are required to access the network port in promiscuous mode, and thus to run packet capture. as a new linux user, you might never get those permissions. good luck :)
- Packet capture does not work with wireless network adapters.

איך נריץ את הקוד בחלק א?

נכניס את הפקודה "sudo ./sniffer <filter expression>"

```
codebind@codebind: ~/CLionProjects/untitled5
codebind@codebind:~/CLionProjects/untitled5$ make all
make: Nothing to be done for 'all'.
codebind@codebind:~/CLionProjects/untitled5$ sudo ./Sniffer tcp
[sudo] password for codebind:
^C
codebind@codebind:~/CLionProjects/untitled5$ make all
gcc -o Sniffer Sniffer.c -lpcap
codebind@codebind:~/CLionProjects/untitled5$ sudo ./Sniffer tcp
TCP : 31  UDP : 0  ICMP : 0  IGMP : 0  Others : 0  ^Ctotal : 30
codebind@codebind:~/CLionProjects/untitled5$
```

אפשר לראות בתמונה המסומנת, שנקלטו 31 הודעות מסוג TCP.

חשוב לציין- בקוד שלנו, לא ביצענו הסנפה מוגדרת עבור פרוטוקולים UDP ו-ICMP. השארנו שלד מוגדר המאפשר הסנפה גם בפרוטוקולים UDP ו-ICMP. כך שאם נרצה להסניף גם פקטות UDP, ICMP או פרוטוקולים אחרים, נוכל להוסיף את פונקציית ההסנפה שמגדירה בדיוק מה אנחנו רוצים להוציא מהחבילה ולהסניף דרך הקוד שלנו.

ההסנפה של פקטות ה-TCP בקוד שלנו מותאמת להודעות ה-TCP שנשלחות במטלה 2 בקורס.

איך הקוד שלנו בנוי?

1. ראשית, אנחנו פותחים קובץ חדש שאליו נדפיס את מה שהסנפנו.
2. נשתמש בפונקציה `pcap_open_live` על מנת להתחיל את ההסנפה על כרטיס הרשת שהגדרנו במשתנה `device`. בחלק הזה, המכשיר הוא `lo - loopback`, כלומר, תקשורת מהמחשב שלנו למחשב שלנו, בפורט `127.0.0.1`. (ככה עובדת מטלה 2)
3. לפי הפילטר שקיבלנו כארגומנט בתחילת ריצת התוכנית נכניס למשתנה `filter_exp` את הסוג של ההודעה אותה אנחנו רוצים להסניף.
4. בעזרת הפונקציות `pcap_compile` ו-`pcap_setfilter` נקמפל את ה-`handle` שהוא מי שמנהל את ההסנפה ל-`BPF` ונוודא שהפילטר בסדר.
5. לאחר מכן, פונקציית `pcap_loop` היא הפונקציה שקולטת את ההודעות. הגדרנו את הפונקציה כך שתקלוט מספר בלתי מוגבל של הודעות עד שאנחנו עוצרים את התוכנה (באמצעות הפרמטר -1). בנוסף, הגדרנו שכל הודעה שמוסנפת, הולכת לפונקציה `got_packet` שתדפיס את הפרטים שלה.
- # הפונקציה `got_packet` שולחת כל הודעה שהגיעה לפי סוג הפרוטוקול לפונקציית ההדפסה שמתאימה לו.
- # במקרה שלנו, הפרטים מודפסים בפונקציה `print_tcp_packet` לקובץ. ה-`data` מודפסת לקובץ על ידי הפונקציה `print_data`.
6. לבסוף, נסגור את הקובץ שאליו כתבנו את המידע, נשחרר את הזיכרון של ה-`BPF` ונסגור את מנהל ההסנפה (ה-`handle`).

נראה דוגמא לריצת הקוד:

נראה דוגמא על פקטת ה-PSH השולחת את הבקשה לשרת.

ככה נראית ההדפסה לטרמינל בזמן ההסנפה:

```
shahar@shahar-X442UQR:~/CLionProjects/matala5tikshoret$ sudo ./sniffer tcp
filter expression: tcp
TCP : 16   UDP : 0   ICMP : 0   IGMP : 0   Others : 0   Total : 15
^C
```

ככה נראית ההודעה בקובץ:

*****TCP Packet*****

Headers Data:

| -Source Port : 35056
| -Destination Port : 9999
| -Source IP : 127.0.0.1
| -Destination IP : 127.0.0.1
| -Total Length : 376
| -Unix Time Stamp : 1674035616
| -Cache Control : 65535
| -Cache Flag : 1
| -Steps Flag : 1
| -Type Flag : 1
| -Status : 0
| -psh = 1, syn = 0, ack = 1, fin = 0

DATA Dump

Data Payload

80 04 95 61 01 00 00 00 00 00 8C 0A 63 61 6C	0..a.....cal
63 75 6C 61 74 6F 72 94 8C 0A 42 69 6E 61 72 79	culator...Binary
45 78 70 72 94 93 94 29 81 94 7D 94 28 8C 0C 6C	Expr...)}.(.l
65 66 74 5F 6F 70 65 72 61 6E 64 94 68 00 8C 10	eft_operand.h...
46 75 6E 63 74 69 6F 6E 43 61 6C 6C 45 78 70 72	FunctionCallExpr
94 93 94 29 81 94 7D 94 28 8C 08 66 75 6E 63 74	...)}.(.func
69 6F 6E 94 68 00 8C 08 46 75 6E 63 74 69 6F 6E	ion.h...Function
94 93 94 29 81 94 7D 94 28 8C 04 6E 61 6D 65 94	...)}.(.name.
8C 03 6D 61 78 94 68 0A 8C 08 62 75 69 6C 74 69	..max.h...builti
6E 73 94 8C 03 6D 61 78 94 93 94 75 62 8C 04 61	ns...max...ub..a
72 67 73 94 5D 94 28 68 00 8C 08 43 6F 6E 73 74	rgs.](h...Const
61 6F 74 04 03 04 29 81 04 7D 04 8C 05 74 61 6C	ant) 1 val

וככה ההסנפה נראית ב-wireshark:

No.	Source	Destination	Protocol	Length	Info
1	127.0.0.1	127.0.0.1	TCP	74	35056 → 9999 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=384124929 TSecr=0 WS=128
2	127.0.0.1	127.0.0.1	TCP	74	9999 → 35056 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=384124929 TSecr=384124929 WS...
3	127.0.0.1	127.0.0.1	TCP	66	35056 → 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=384124929 TSecr=384124929
4	127.0.0.1	127.0.0.1	TCP	118	35056 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=384124929 TSecr=384124929 [TCP segment of a reas...
5	127.0.0.1	127.0.0.1	TCP	66	9999 → 35056 [ACK] Seq=1 Ack=377 Win=65152 Len=0 TSval=384124929 TSecr=384124929
6	127.0.0.1	127.0.0.1	TCP	126	9999 → 35056 [PSH, ACK] Seq=1 Ack=377 Win=65536 Len=60 TSval=384124929 TSecr=384124929 [TCP segment of a reas...
7	127.0.0.1	127.0.0.1	TCP	66	35056 → 9999 [ACK] Seq=377 Ack=61 Win=65536 Len=0 TSval=384124929 TSecr=384124929
8	127.0.0.1	127.0.0.1	TCP	66	35056 → 9999 [FIN, ACK] Seq=377 Ack=61 Win=65536 Len=0 TSval=384124930 TSecr=384124929
9	127.0.0.1	127.0.0.1	TCP	66	9999 → 35056 [FIN, ACK] Seq=61 Ack=378 Win=65536 Len=0 TSval=384124930 TSecr=384124930
10	127.0.0.1	127.0.0.1	TCP	66	35056 → 9999 [ACK] Seq=378 Ack=62 Win=65536 Len=0 TSval=384124930 TSecr=384124930

נתייחס באופן ספציפי לפקטת ה-PSH שמסומנת בכחול בהקלטה. בתמונה שמעל התמונה הזו, מצולמים הפרטים המודפסים שלה. נוכל לראות שאכן הודפסו הפורטים הנכונים- החבילה נשלחת מפורט 35056 לפורט 9999, הפורט של השרת. הפרטים נכונים גם בקובץ. בקובץ, גודל החבילה הוא 376, שזה בדיוק הגודל של החבילה ללא ה-headers של כל השכבות, אשר ביחד הגודל שלהם הוא 66 בייטים. ואכן, $442 = 376 + 66$. בנוסף נוכל לראות בקובץ שאכן הוא מזהה שמדובר בפקטה מסוג PSH\ACK לפי הדגלים המודפסים לפני ה-DATA. בחלק של ה-data payload, ה-DATA מודפסת בהקסאדצימלי. בנוסף, נוכל לראות ששלושת הדגלים: cache, steps, type דולקים- מאחר שבמטלה 2 בחבילת הבקשה לשרת הגדרנו שאנחנו רוצים שהשרת יחזיר לנו את השלבים של החישוב ושישמור את החבילה ב-PROXY. הדגל TYPE דלוק כי החבילה הזאת היא חבילת בקשה. הסטטוס הוא 0 מאחר שסטטוס הוא שדה שרלוונטי רק לחבילת התגובה.

נשים לב לנקודה חשובה: הסטטוס של חבילת התגובה הוא 200. אנחנו הדפסנו את כל ההודעות שהוסגפו מהתקשורת בין הלקוח לשרת במטלה 2. לפי הסטטוס, נוכל לזהות מי מהחבילות הן חבילת התגובה.

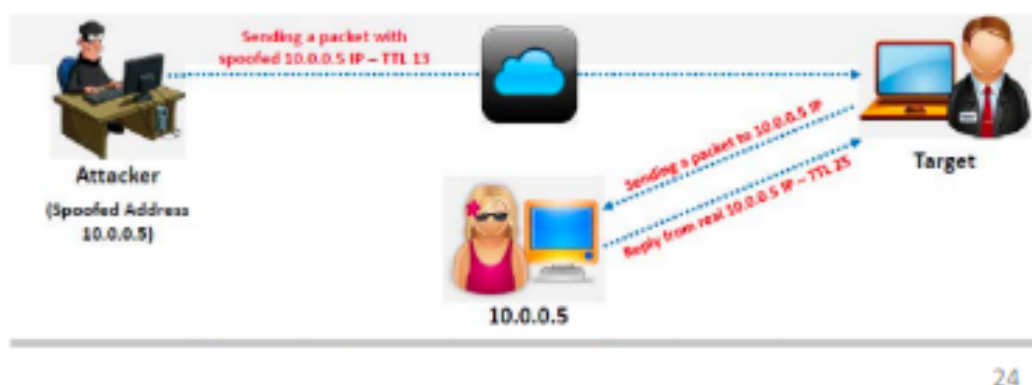
Task B:- Spoofer.c

Spoofing:-

When some critical information in the packet is forged, we refer to it as packet spoofing.

Many network attacks rely on packet spoofing.

Let's see how to send packets without spoofing.



Questions:-

1.) **Can you set the IP packet length field to an arbitrary value, regardless of how big the actual packet is?**

Yes, the length of the IP header can be set to arbitrary values.

The IP length is modified to its original size, irrespective of the size set by the programmer.

2.) **Using the raw socket programming, do you have to calculate the checksum for the IP header?**

The kernel or the underlying operating system builds the packet including the checksum for your data.

איך נריץ את התוכנה?

נוכל לראות כאן בדוגמת ההרצה:

בכניס את הפקודה לטרמינל: "sudo ./spoofer <source_ip> <victim's_ip> "

לאחר מכן, הקוד ישאל באיזה פרוטוקול אנחנו רוצים לזייף את הפקטות שלנו.

הקוד שלנו מסוגל לזייף פקטות מהסוגים - ICMP, TCP, UDP.

נוכל לראות בתמונה של הטרמינל, שלאחר שבחרנו לזייף פקטות מסוג ICMP, נשלחות פקטות

ICMP בגודל 38 בייטים, בקצב של הודעה לשנייה.

[illegible]

ובגזרת ה-WIRESHARK,

אנחנו ניסינו לעבוד על גוגל ולשלוח להם פקטות מזויפות. אבל, לצערנו ולשמחתנו גוגל קצת יותר חכמים מזה. גוגל מזהים שלא מדובר בפקטה תקנית וזורקים אותה, ולא עונים לה. נוכל לראות את זה ב-WIRESHARK, כאשר נשלחות רק פקטות בקשה שאנחנו זייפנו, ולא חוזרות תגובות. בנוסף, נשים לב שה-WIRESHARK מתריע לנו שלא נמצאה תגובה בכל פקטה.

We use `setsockopt()` to enable `IP_HDRINCL` on the socket.

```
//IP_HDRINCL to tell the kernel that headers are included  
in the packet
```

```
int enable = 1;
```

```
if (setsockopt (s, IPPROTO_IP, IP_HDRINCL, &enable,  
sizeof (enable)) < 0)
```

```
{
```

```
    perror("Error setting IP_HDRINCL");
```

```
    exit(0);
```

```
}
```

Send a packet after did a spoofer on it.

```
while (1)
```

```
{
```

```
    //Send the packet
```

```
    if (sendto (s, buffer, 1500 , 0, (struct sockaddr *)  
&sin, sizeof (sin)) < 0)
```

```
    {
```

```
        perror("sendto failed");
```

```
    }
```

```
        //Data send successfully
```

```
    else
```

```
    {
```

```
        printf ("Packet Send. Length : %d \n" ,  
iph->tot_len);
```

```
    }
```

```
        // sleep for 1 seconds
```

```
        sleep(1);
```

```
}
```

שלבי הקוד(חלק ב):

1. ראשית, ב-MAIN אנחנו ממלאים את הפרטים הקשורים ל-`ip_header`, כולל את ה-IP של המקור ושל היעד.

IP המקור - ממי אנחנו רוצים שיחשבו שהחבילה נשלחה

IP היעד - מי שאנחנו רוצים לשלוח לו פקטה מזוייפת .

2. ממלאים את הפרטים של ה-HEADER של כל אחד מהפרוטוקולים - UDP, ICMP, TCP.

3. לבסוף, אנחנו שולחים באמצעות RAWSOCKET את הפקטה המזוייפת אל היעד.

חלק ג: ה-SNOOFER:

איך עובד ה-SNOOFER:

1. כמו ב-SNIFFER, נסניף פקטה מסוג ICMP. בניגוד לחלק א, בחלק הזה אנחנו נסניף כל פעם רק פקטה אחת, כדי שנוכל לזייף את מה שהסנפנו. נעשה זאת על ידי הפרמטר 1 בפונקציה pcap_open_live.
2. בעזרת לולאה אינסופית, נקלוט כל פעם פקטה אחת ונשלח את הפקטה לפונקציה got_packet.
3. בפונקציה got_packet, נעדכן את פרטי החבילה, נחליף את IP המקור והיעד, נשנה את סוג החבילה להיות חבילת תגובה ולא בקשה על ידי שינוי הסוג ב-ICMP HEADER ל-0 במקום 8, נחשב את ה-CHECKSUM מחדש ונשלח את החבילה לפונקציה בשם send_raw_ip_packet_snoofer.
4. לבסוף, נשלח את החבילה המעודכנת ב-RAW SOCKET.

כמה נק חשובות:

- # כאשר מרימים דוקרים ורשת פנימית שהדוקרים פועלים עליה כמו במטלה, הדוקרים מחזיקים לעצמם כרטיס רשת וירטואלי. כשאנחנו הסנפנו את הפינגים שרצים בתוך הדוקרים, השתמשנו ב-INTERFACE הוירטואלי שנוצר להם. ה-INTERFACE מוגדר בתוך ה-SNOOFER בשורה 128. כרגע שורה 128 נראית ככה:
- ```
char device[] = "br-69386c74bdc9"; // Device to sniff on.
```
- אם במהלך הבדיקה הסניפר לא קולט כלום, כדאי לבדוק האם במקרה ההסנפה מתבצעת מ-INTERFACE לא נכון.

## בדיקה מספר 1:

### התהליך:

1. הדוקר HOSTA שולח לדוקר HOSTB פינג.
2. ה-SEED-ATTACKER מסניף את הפקטות בזמן שהן נשלחות.
3. ה-SEED-ATTACKER משנה את פרטי חבילת הבקשה והופך אותה לחבילת תגובה מזוייפת.
4. ה-SEED-ATTACKER שולח את החבילה חזרה ל-HOSTA.

בתהליך הזה, HOSTA מקבל 2 חבילות תגובה על כל חבילה שהוא שלח. אחת מה-SEED-ATTACKER ואחת מ-HOSTB (התשובה היא לא, לא קשרנו את HOSTB בארון עדיין)

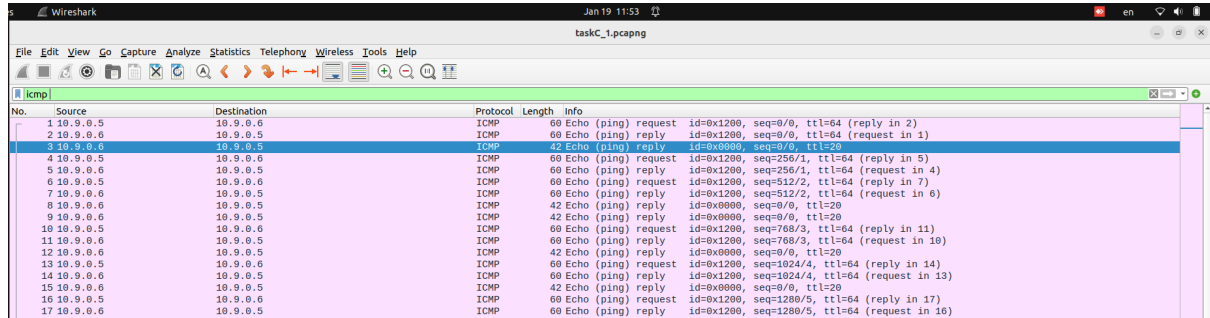
## תמונה של הטרמינל שבו הרמנו את הדוקרים:





אפשר לראות שהמבחינת הטרמינל, הוא מקבל תגובות שונות על כל חבילה שהוא שולח, מאחר שה-SEQ\_NUMBER עולה באופן רציף. אך החבילות המזוייפות שלנו מסתתרות כאן- חבילות עם TTL=20 ובגודל 46 בייטים במקום 64 הן החבילות שלנו.

## בגזרת ה-WIRESHARK,



| No. | Source   | Destination | Protocol | Length | Info                                                            |
|-----|----------|-------------|----------|--------|-----------------------------------------------------------------|
| 1   | 10.9.0.5 | 10.9.0.6    | ICMP     | 60     | Echo (ping) request id=0x1200, seq=0/0, ttl=64 (reply in 2)     |
| 2   | 10.9.0.6 | 10.9.0.5    | ICMP     | 60     | Echo (ping) reply id=0x1200, seq=0/0, ttl=64 (request in 1)     |
| 3   | 10.9.0.6 | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                    |
| 4   | 10.9.0.5 | 10.9.0.6    | ICMP     | 60     | Echo (ping) request id=0x1200, seq=256/1, ttl=64 (reply in 5)   |
| 5   | 10.9.0.6 | 10.9.0.5    | ICMP     | 60     | Echo (ping) reply id=0x1200, seq=256/1, ttl=64 (request in 4)   |
| 6   | 10.9.0.5 | 10.9.0.6    | ICMP     | 60     | Echo (ping) request id=0x1200, seq=512/2, ttl=64 (reply in 7)   |
| 7   | 10.9.0.6 | 10.9.0.5    | ICMP     | 60     | Echo (ping) reply id=0x1200, seq=512/2, ttl=64 (request in 6)   |
| 8   | 10.9.0.6 | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                    |
| 9   | 10.9.0.6 | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                    |
| 10  | 10.9.0.5 | 10.9.0.6    | ICMP     | 60     | Echo (ping) request id=0x1200, seq=768/3, ttl=64 (reply in 11)  |
| 11  | 10.9.0.6 | 10.9.0.5    | ICMP     | 60     | Echo (ping) reply id=0x1200, seq=768/3, ttl=64 (request in 10)  |
| 12  | 10.9.0.6 | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                    |
| 13  | 10.9.0.5 | 10.9.0.6    | ICMP     | 60     | Echo (ping) request id=0x1200, seq=1024/4, ttl=64 (reply in 14) |
| 14  | 10.9.0.6 | 10.9.0.5    | ICMP     | 60     | Echo (ping) reply id=0x1200, seq=1024/4, ttl=64 (request in 13) |
| 15  | 10.9.0.6 | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                    |
| 16  | 10.9.0.5 | 10.9.0.6    | ICMP     | 60     | Echo (ping) request id=0x1200, seq=1280/5, ttl=64 (reply in 17) |
| 17  | 10.9.0.6 | 10.9.0.5    | ICMP     | 60     | Echo (ping) reply id=0x1200, seq=1280/5, ttl=64 (request in 16) |
| 18  | 10.9.0.6 | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                    |

ה-WIRESHARK קולט את זה קצת מבולגן, אבל נוכל לראות שלכל בקשה מתקבלות 2 תגובות וששתיהן זהות כמעט לחלוטין. ה-WIRESHARK לא מזהה בעיה באחת מהחבילות ולא מקפיץ שגיאה או אזהרה כל שהיא.

## בדיקה מספר 2:

## התהליך:

1. הדוקר HOSTA שולח לגוגל (8.8.8.8) פינג
2. ה-SEED-ATTACKER מסניף את הפקטות בזמן שהן נשלחות
3. ה-SEED-ATTACKER משנה את פרטי חבילת הבקשה והופך אותה לחבילת תגובה מזוייפת
4. ה-SEED-ATTACKER שולח את החבילה חזרה ל-HOSTA

בתהליך הזה, HOSTA מקבל 2 חבילות תגובה על כל חבילה שהוא שלח. אחת מה-SEED-ATTACKER ואחת מגוגל.

## ככה זה נראה אצל ה-SEED-ATTACKER:

[illegible]

```
codebind@codebind: ~/CLionProjects/Labsetup

Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
Spoof done
source_ip: 8.8.8.8, dest_ip: 10.9.0.5
^C
root@codebind:/volumes#
```

וככה זה נראה ב-HOSTA:

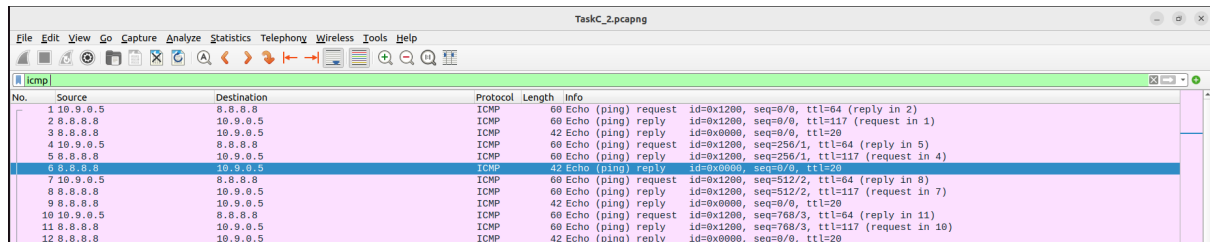
```
codebind@codebind: ~/CLionProjects/Labsetup

root@563da69eb074:/volumes# ./ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 bytes of data
64 bytes from 8.8.8.8: seq=1, ttl=117 time=5.64 ms
46 bytes from 8.8.8.8: seq=2, ttl=20 time=0.13 ms
64 bytes from 8.8.8.8: seq=3, ttl=117 time=0.17 ms
46 bytes from 8.8.8.8: seq=4, ttl=20 time=0.09 ms
64 bytes from 8.8.8.8: seq=5, ttl=117 time=0.27 ms
46 bytes from 8.8.8.8: seq=6, ttl=20 time=0.27 ms
64 bytes from 8.8.8.8: seq=7, ttl=117 time=0.27 ms
46 bytes from 8.8.8.8: seq=8, ttl=20 time=0.27 ms
64 bytes from 8.8.8.8: seq=9, ttl=117 time=0.14 ms
46 bytes from 8.8.8.8: seq=10, ttl=20 time=0.11 ms
64 bytes from 8.8.8.8: seq=11, ttl=117 time=0.27 ms
46 bytes from 8.8.8.8: seq=12, ttl=20 time=0.16 ms
64 bytes from 8.8.8.8: seq=13, ttl=117 time=0.27 ms
46 bytes from 8.8.8.8: seq=14, ttl=20 time=0.28 ms
64 bytes from 8.8.8.8: seq=15, ttl=117 time=0.25 ms
46 bytes from 8.8.8.8: seq=16, ttl=20 time=0.27 ms
64 bytes from 8.8.8.8: seq=17, ttl=117 time=0.27 ms
46 bytes from 8.8.8.8: seq=18, ttl=20 time=0.15 ms
64 bytes from 8.8.8.8: seq=19, ttl=117 time=0.23 ms
46 bytes from 8.8.8.8: seq=20, ttl=20 time=0.26 ms
64 bytes from 8.8.8.8: seq=21, ttl=117 time=0.09 ms
^C
root@563da69eb074:/volumes#
```



אפשר לראות שהמבחינת הטרמינל, הוא מקבל תגובות שונות על כל חבילה שהוא שולח, מאחר שה-SEQ\_NUMBER עולה באופן רציף. אך החבילות המזוייפות שלנו מסתתרות כאן- חבילות עם TTL=20 ובגודל 46 בייטים במקום 64 הן החבילות שלנו.

בגזרת ה-WIRESHARK,



| No. | Source   | Destination | Protocol | Length | Info                                                            |
|-----|----------|-------------|----------|--------|-----------------------------------------------------------------|
| 1   | 10.9.0.5 | 8.8.8.8     | ICMP     | 60     | Echo (ping) request id=0x1200, seq=0/0, ttl=64 (reply in 2)     |
| 2   | 8.8.8.8  | 10.9.0.5    | ICMP     | 60     | Echo (ping) reply id=0x1200, seq=0/0, ttl=117 (request in 1)    |
| 3   | 8.8.8.8  | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                    |
| 4   | 10.9.0.5 | 8.8.8.8     | ICMP     | 60     | Echo (ping) request id=0x1200, seq=256/1, ttl=64 (reply in 5)   |
| 5   | 8.8.8.8  | 10.9.0.5    | ICMP     | 60     | Echo (ping) reply id=0x1200, seq=256/1, ttl=117 (request in 4)  |
| 6   | 8.8.8.8  | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                    |
| 7   | 10.9.0.5 | 8.8.8.8     | ICMP     | 60     | Echo (ping) request id=0x1200, seq=512/2, ttl=64 (reply in 8)   |
| 8   | 8.8.8.8  | 10.9.0.5    | ICMP     | 60     | Echo (ping) reply id=0x1200, seq=512/2, ttl=117 (request in 7)  |
| 9   | 8.8.8.8  | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                    |
| 10  | 10.9.0.5 | 8.8.8.8     | ICMP     | 60     | Echo (ping) request id=0x1200, seq=768/3, ttl=64 (reply in 11)  |
| 11  | 8.8.8.8  | 10.9.0.5    | ICMP     | 60     | Echo (ping) reply id=0x1200, seq=768/3, ttl=117 (request in 10) |
| 12  | 8.8.8.8  | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                    |

נוכל לראות שלכל בקשה מתקבלות 2 תגובות וששתיהן זהות כמעט לחלוטין. ה-WIRESHARK לא מזהה בעיה באחת מהחבילות ולא מקפיץ שגיאה או אזהרה כל שהיא.

### בדיקה מספר 3:

## התהליך:

1. הדוקר HOSTA שולח ל- (1.2.3.4) פינג. נשים לב שבזמן הבדיקה לא הייתה עמדת קצה פעילה עם כתובת ה-IP הזו ואם לא היינו מזייפים לא הייתה חוזרת תגובה.
  2. ה-SEED-ATTACKER מסניף את הפקטות בזמן שהן נשלחות
  3. ה-SEED-ATTACKER משנה את פרטי חבילת הבקשה והופך אותה לחבילת תגובה מזויפת
  4. ה-SEED-ATTACKER שולח את החבילה חזרה ל-HOSTA
- #כך, HOSTA מקבל תגובה ל-PING ששלח למרות שהוא שלח פינג לעמדת קצה שאינה קיימת ולא יכולה להחזיר לו תגובה..

## כך זה נראה ב-SEED-ATTACKER:

[illegible]

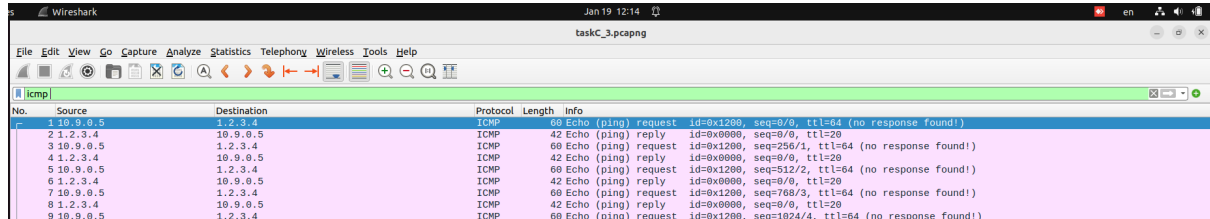
```
codebind@codebind: ~/CLionProjects/Labsetup
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
Spoof done
source_ip: 1.2.3.4, dest_ip: 10.9.0.5
^C
root@codebind:/volumes#
```

וככה זה נראה ב-HOSTA:

```
codebind@codebind: ~/CLionProjects/Labsetup
root@563da69eb074:/volumes# ./ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4): 56 bytes of data
46 bytes from 1.2.3.4: seq=1, ttl=20 time=37.18 ms
46 bytes from 1.2.3.4: seq=2, ttl=20 time=21.31 ms
46 bytes from 1.2.3.4: seq=3, ttl=20 time=23.06 ms
46 bytes from 1.2.3.4: seq=4, ttl=20 time=22.39 ms
46 bytes from 1.2.3.4: seq=5, ttl=20 time=23.69 ms
46 bytes from 1.2.3.4: seq=6, ttl=20 time=24.14 ms
46 bytes from 1.2.3.4: seq=7, ttl=20 time=22.07 ms
46 bytes from 1.2.3.4: seq=8, ttl=20 time=22.57 ms
46 bytes from 1.2.3.4: seq=9, ttl=20 time=22.39 ms
46 bytes from 1.2.3.4: seq=10, ttl=20 time=23.78 ms
46 bytes from 1.2.3.4: seq=11, ttl=20 time=23.55 ms
46 bytes from 1.2.3.4: seq=12, ttl=20 time=22.62 ms
46 bytes from 1.2.3.4: seq=13, ttl=20 time=23.98 ms
46 bytes from 1.2.3.4: seq=14, ttl=20 time=23.42 ms
46 bytes from 1.2.3.4: seq=15, ttl=20 time=22.74 ms
46 bytes from 1.2.3.4: seq=16, ttl=20 time=23.51 ms
46 bytes from 1.2.3.4: seq=17, ttl=20 time=22.02 ms
46 bytes from 1.2.3.4: seq=18, ttl=20 time=23.08 ms
46 bytes from 1.2.3.4: seq=19, ttl=20 time=21.25 ms
46 bytes from 1.2.3.4: seq=20, ttl=20 time=23.11 ms
46 bytes from 1.2.3.4: seq=21, ttl=20 time=21.79 ms
^C
root@563da69eb074:/volumes#
```

נוכל לראות כאן ש-HOSTA מקבל רק את ההודעות המזויפות שלנו, בגודל 46 בייטים ועם TTL=20.

## בגזרת ה-WIRESHARK,



| No. | Source   | Destination | Protocol | Length | Info                                                                   |
|-----|----------|-------------|----------|--------|------------------------------------------------------------------------|
| 1   | 10.9.0.5 | 1.2.3.4     | ICMP     | 60     | Echo (ping) request id=0x1200, seq=0/0, ttl=64 (no response found!)    |
| 2   | 1.2.3.4  | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                           |
| 3   | 10.9.0.5 | 1.2.3.4     | ICMP     | 60     | Echo (ping) request id=0x1200, seq=256/1, ttl=64 (no response found!)  |
| 4   | 1.2.3.4  | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                           |
| 5   | 10.9.0.5 | 1.2.3.4     | ICMP     | 60     | Echo (ping) request id=0x1200, seq=512/2, ttl=64 (no response found!)  |
| 6   | 1.2.3.4  | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                           |
| 7   | 10.9.0.5 | 1.2.3.4     | ICMP     | 60     | Echo (ping) request id=0x1200, seq=768/3, ttl=64 (no response found!)  |
| 8   | 1.2.3.4  | 10.9.0.5    | ICMP     | 42     | Echo (ping) reply id=0x0000, seq=0/0, ttl=20                           |
| 9   | 10.9.0.5 | 1.2.3.4     | ICMP     | 60     | Echo (ping) request id=0x1200, seq=1024/4, ttl=64 (no response found!) |

ראשית, נוכל לראות גם ב-WIRESHARK שיש רק חבילת תגובה אחת על כל חבילת בקשה. בנוסף, ה-WIRESHARK מתריע לנו שלא התקבלה תגובה לחבילת הבקשה ששלחנו, אבל למרות זאת מתקבלת תגובה. (SUSPICIOUS!!!)

### Task d:-

first we used a sender file ,to send a datagram packet with a message.

in order to use task d run the following command in terminal:

“./gateway <hostIP>” for gateway.c

“./sender” for sender.c

The screenshot shows an IDE with a terminal window open. The terminal displays a continuous stream of "Hello message sent." messages, indicating a loop in the execution. The IDE interface includes a menu bar at the top with options like File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, and Help. The toolbar below the menu contains various icons for file operations, running, and debugging. The left sidebar shows the Project, Bookmarks, and Structure views. The bottom status bar indicates the current file is "untitled5" and the encoding is "UTF-8".

[illegible]

```
untitled5 - Gateway.c
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
untitled5 Gateway.c
Project: Gateway.c sender.c
Terminal: Local Local (2)
codebind@codebind:~/CLionProjects/untitled5$ sudo ./Gateway 127.0.0.1
[sudo] password for codebind:
Create a sockfd
Success create a another socket
Success Bind
0.840180
rcv = 1 send =1 percent_sends = 100.00
0.394383
rcv = 2 send =1 percent_sends = 50.00
0.781099
rcv = 3 send =2 percent_sends = 66.67
0.798448
rcv = 4 send =3 percent_sends = 75.00
0.911447
rcv = 5 send =4 percent_sends = 80.00
0.197951
rcv = 6 send =4 percent_sends = 66.67
0.335223
rcv = 7 send =4 percent_sends = 57.14
0.768230
rcv = 8 send =5 percent_sends = 62.50
0.777975
rcv = 9 send =5 percent_sends = 55.56
0.553970
rcv = 10 send =6 percent_sends = 60.00
0.277775
rcv = 9 send =5 percent_sends = 55.56
0.553970
rcv = 10 send =6 percent_sends = 60.00
0.477397
rcv = 11 send =6 percent_sends = 54.55
0.628871
rcv = 12 send =7 percent_sends = 50.33
0.364784
rcv = 13 send =7 percent_sends = 53.85
0.513401
rcv = 14 send =8 percent_sends = 57.14
0.952230
rcv = 15 send =9 percent_sends = 60.00
0.916195
rcv = 16 send =10 percent_sends = 62.50
0.635712
rcv = 17 send =11 percent_sends = 64.71
0.717297
rcv = 18 send =12 percent_sends = 66.67
0.161683
rcv = 19 send =12 percent_sends = 63.16
0.686969
rcv = 20 send =13 percent_sends = 65.00
0.816301
P Version Control Python Packages TODO CMake Problems Services Terminal
Local History is broken: Local History storage file has become corrupted and will be rebuilt. (today 9:05 PM) 142:1 LF UTF-8 clang-tidy 4 spaces C:untitled5 Debug
```

```
untitled5 - Gateway.c
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
untitled5 Gateway.c
Project: Gateway.c sender.c
Terminal: Local Local (2)
0.277775
rcv = 9 send =5 percent_sends = 55.56
0.553970
rcv = 10 send =6 percent_sends = 60.00
0.477397
rcv = 11 send =6 percent_sends = 54.55
0.628871
rcv = 12 send =7 percent_sends = 50.33
0.364784
rcv = 13 send =7 percent_sends = 53.85
0.513401
rcv = 14 send =8 percent_sends = 57.14
0.952230
rcv = 15 send =9 percent_sends = 60.00
0.916195
rcv = 16 send =10 percent_sends = 62.50
0.635712
rcv = 17 send =11 percent_sends = 64.71
0.717297
rcv = 18 send =12 percent_sends = 66.67
0.161683
rcv = 19 send =12 percent_sends = 63.16
0.686969
rcv = 20 send =13 percent_sends = 65.00
0.816301
P Version Control Python Packages TODO CMake Problems Services Terminal
Local History is broken: Local History storage file has become corrupted and will be rebuilt. (today 9:05 PM) 142:1 LF UTF-8 clang-tidy 4 spaces C:untitled5 Debug
```

```
untitled5 - Gateway.c
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
untitled5 Gateway.c
Terminal Local Local (2) + v
0.884177
rcv = 24 send =14 percent_sends = 58.33
0.156679
rcv = 25 send =14 percent_sends = 56.00
0.408944
rcv = 26 send =14 percent_sends = 53.85
0.129790
rcv = 27 send =14 percent_sends = 51.85
0.188809
rcv = 28 send =14 percent_sends = 50.00
0.998924
rcv = 29 send =15 percent_sends = 51.72
0.218257
rcv = 30 send =15 percent_sends = 50.00
0.512932
rcv = 31 send =16 percent_sends = 51.61
0.839112
rcv = 32 send =17 percent_sends = 53.12
0.612640
rcv = 33 send =18 percent_sends = 54.55
0.296032
rcv = 34 send =18 percent_sends = 52.94
0.637952
rcv = 35 send =19 percent_sends = 54.29
0.524287
C
Version Control Python Packages TODO CMake Problems Services Terminal
Local History is broken Local History storage file has become corrupted and will be rebuilt. (today 9:05 PM) 142:1 LF UTF-8 clang-tidy 4 spaces C:untitled5 | Debug
```

```
untitled5 - Gateway.c
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
untitled5 Gateway.c
Terminal Local Local (2) + v
0.156679
rcv = 25 send =14 percent_sends = 56.00
0.408944
rcv = 26 send =14 percent_sends = 53.85
0.129790
rcv = 27 send =14 percent_sends = 51.85
0.188809
rcv = 28 send =14 percent_sends = 50.00
0.998924
rcv = 29 send =15 percent_sends = 51.72
0.218257
rcv = 30 send =15 percent_sends = 50.00
0.512932
rcv = 31 send =16 percent_sends = 51.61
0.839112
rcv = 32 send =17 percent_sends = 53.12
0.612640
rcv = 33 send =18 percent_sends = 54.55
0.296032
rcv = 34 send =18 percent_sends = 52.94
0.637952
rcv = 35 send =19 percent_sends = 54.29
0.524287
C
codebind@codebind:~/ClionProjects/untitled5$
Version Control Python Packages TODO CMake Problems Services Terminal
Local History is broken Local History storage file has become corrupted and will be rebuilt. (today 9:05 PM) 142:1 LF UTF-8 clang-tidy 4 spaces C:untitled5 | Debug
```

in the above pictures we see the situation:  
when get a packet:  
if the random number is greater than 0.5 then increment the send counter. and  
always the counter total\_rcv ++;  
  
in each iteration well calculate the percent of the send packets.  
  
note : we added the file sender.c to this homework.

## שלבי הקוד:

1. פתיחת סוקט לשליחה עם הנתונים (כתובת, פורט,...) המתאים לו  
כך שהוא שולח ל פורט 9999.
2. פתיחת סוקט לקבלת הודעה כך שמאזין על כל מני הוסטים , פורט 9998 .
3. ואז קבלת מספר רנדומלי בין 0 ל 1 , ואחרכך בדיקה :-
4. אם המספר הוא גדול מ 0.5 אז נשלח את הודעה לפורט 9999 (הודעה שקבלנו מ הסוקט הקבלה) , אם קטן מ 0.5 אז יחזור לקבל עוד פעם הודעה ומספר רנדומלי (בלי שליחה).