תרגיל 1 – מבוא לרשתות תקשורת

מגישות : יעל שמחיס 209009604, ולינוי דוארי 318416344.

הערה –

את חלק א' של התרגיל ביצענו על המחשב של לינוי, עליו מותקן linux mint בתצורת dual boot. לצערנו, עקב תקלת חומרה במחשב זה, נאלצנו בחלק ב' לעבור למחשב של יעל, עליו הותקנו 2 מכונות וירטואליות (lubuntu).

חלק א

2. הרצנו את השרת והלקוח והפעלנו את הwireshark, כדי לסנן את החבילות שהלקוח שלח לשרת והשרת שלח ללקוח כתבנו בשורת ה- "apply a display filter" שזה שורת החיפוש את המסננים הבאים: ip.addr == 127.0.0.1 && udp.port == 12345
שמרנו את החבילות שסיננו בקובץ trace1.pcapng שמצורף להגשה.

3. בקוד השרת יש שימוש בפורט 12345. הלקוח שולח לשרת הודעה שמכילה את שמותינו ותעודות הזהות שלנו. לאחר שהשרת מקבל את ההודעה, הוא שולח אותה כפי שהיא חזרה ללקוח (למעשה זהו שרת echo).

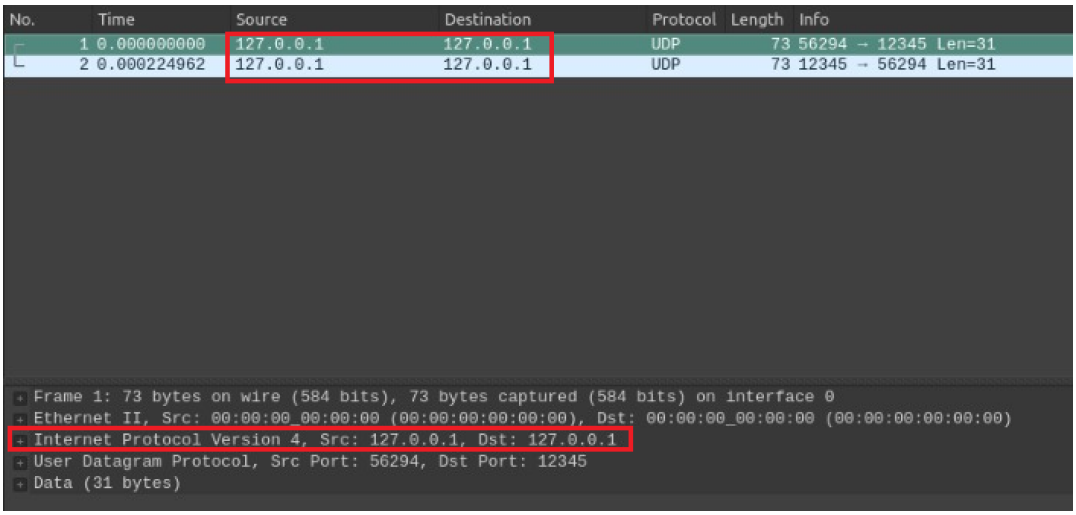
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	UDP	73	56294 → 12345 Len=31
2	0.000224962	127.0.0.1	127.0.0.1	UDP	73	12345 → 56294 Len=31

הלקוח שולח את ההודעה שלו לפורט 12345 (הוא יודע אותו מראש), זה חיוני כדי לשלוח את ההודעה לשרת הרצוי. כפי שניתן לראות בתמונה לעיל, ראשית הלקוח (פורט 56294) שולח הודעה לשרת שמאזין על פורט 12345. לאחר זמן לא רב השרת (שקיבל את ההודעה שנשלחה אליו מהלקוח) מחזיר הודעה לפורט של הלקוח.

נציין כי התקשורת (העברת החבילות) בין השרת והלקוח, קרי התקשורת בין הפורטים השונים (שממומשים על ידי תקשורת בין סוקטים) מתבצעת בשכבת התעבורה.
פורטים של TCP וגם של UDP מוגדרים בשכבת התעבורה במודל הOSI.

לפי מבחנון 1 ראינו שאחד מתפקידיה של שכבת התעבורה היא לדאוג כיצד המחשב המקבל יודע לאיזה מבין האפליקציות הרבות הרצות על המחשב החבילה מיועדת אליה. זה בדיוק מתבטא בתקשורת בין השרת והלקוח כפי שראינו בwireshark.

4. ניתן לראות בwireshark שכתובת IP המקור וכתובת IP היעד הינם localhost (127.0.0.1). ניתן לראות זאת גם בצילום מסך הבא:



ניתן לראות בצילום מסך הבא כי לאחר הרצת פקודת ifconfig כתובת הIP המשוייכת לכרטיס הרשת הינה גם 127.0.0.1:

```
(venv) linoy@LD:~/PycharmProjects/reshatot$ ifconfig
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.12  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::acfd:b1a:d9d4:448a  prefixlen 64  scopeid 0x20<link>
        ether 54:e1:ad:eb:f0:24  txqueuelen 1000  (Ethernet)
        RX packets 50213  bytes 42081971 (42.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 29940  bytes 3085766 (3.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2201  bytes 217314 (217.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2201  bytes 217314 (217.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.68.107  netmask 255.255.255.0  broadcast 192.168.68.255
        inet6 fe80::2de2:b319:f142:ca2  prefixlen 64  scopeid 0x20<link>
        ether e4:70:b8:fa:83:04  txqueuelen 1000  (Ethernet)
        RX packets 1325  bytes 84780 (84.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 256  bytes 31255 (31.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

חלק ב

ראשית, נעזרנו בקבצים שסופקו – ips.txt, parent.txt על מנת לקנפג את "בסיס הנתונים" של שרת האב ושל שרת הבן בהתאמה. ביצענו את ההרצה באופן הבא:

בשלב הראשון, הרצנו על **מכונה וירטואלית א' (ipv4 = 10.0.2.4)** את שרת האבא כך:

```
yael@Yael:~/Downloads$ python3 server.py 55555 -1 -1 parent.txt
```

בשלב השני, הרצנו על מכונה וירטואלית א' את השרת (הסטנדרטי, שרת הבן) כך:

```
yael@Yael:~/Downloads$ python3 server.py 12345 127.0.0.1 55555 ips.txt
```

לבסוף, הרצנו על **מכונה וירטואלית ב' (ipv4 = 10.0.2.15)** את הלקוח כך:

```
yael@Yael:~/Downloads$ python3 client.py 10.0.2.4 12345
```

יש לציין שהשרת שבנינו מתפקד בדומה *לשרת DNS*, וכראוי לאחד כזה, בהינתן דומיין של אתר (למשל, biu.ac.il) שהתקבל כבקשה מהלקוח, השרת שלנו צפוי להחזיר את כתובת ה-ip של האתר המבוקש. תחילה הזנו את הבקשה הבאה (ע"י הלקוח במכונה וירטואלית ב'):

```
yael@Yael:~/Downloads$ python3 client.py 10.0.2.4 12345
www.biu.ac.il
1.2.3.4
```

בשורה השלישית בתמונה הנ"ל רואים את הפלט שהתקבל ע"י קוד הלקוח והודפס לקונסולה. כעת, ע"י תוכנת ה-wireshark, ביצענו סינון בהתאם לכתובת ה-ip של השרת ובהתאם לפורט של השרת (שרת הבן). את הסינון שביצענו ואת התוצאות לאחר הסינון ניתן לראות בתמונה הבאה:

ip.addr == 10.0.2.4 && udp.port==12345 | Expression... | +

	Time	Source	Destination	Protocol	Length	Info
109	31.140489305	10.0.2.15	10.0.2.4	UDP	62	54223 → 12345
110	31.151726429	10.0.2.4	10.0.2.15	UDP	55	12345 → 54223

בשורה 109 לעיל ניתן לראות את החבילה שנשלחה מהלקוח (בפורט 54223) אל השרת (בפורט 12345). בשורה 110 ניתן לראות את התגובה – את החבילה שנשלחה מהשרת שלנו אל הלקוח.

נשים לב שלפי בניית הקוד שלנו, יכול להתקיים **אחד מהתרחישים הבאים**:

א. הלקוח הזין בקשה עם דומיין של אתר שנמצא בקובץ המיפויים של השרת ips.txt. במקרה זה, השרת שלנו יחזיר תשובה ישירות עם ה-ip המבוקש, כפי שהופיע בקובץ.

ב. הלקוח הזין בקשה עם דומיין של אתר **שאינו** נמצא בקובץ המיפויים של השרת ips.txt. במקרה זה, השרת שלנו יפנה לשרת האב (שפועל תחת API זהה כמובן) בבקשה לקבלת ה-IP של הדומיין המבוקש. שרת האב, שקונפג בעזרת קובץ המיפויים parent.txt, יחזיר את הIP המבוקש לשרת הסטנדרטי. כעת, שרת הבן

"ילמד" את המיפוי החדש (יוסיף אותו לקובץ ips.txt עד אשר יסתיים ה-TTL) ויחזיר את התשובה המתאימה ללקוח.

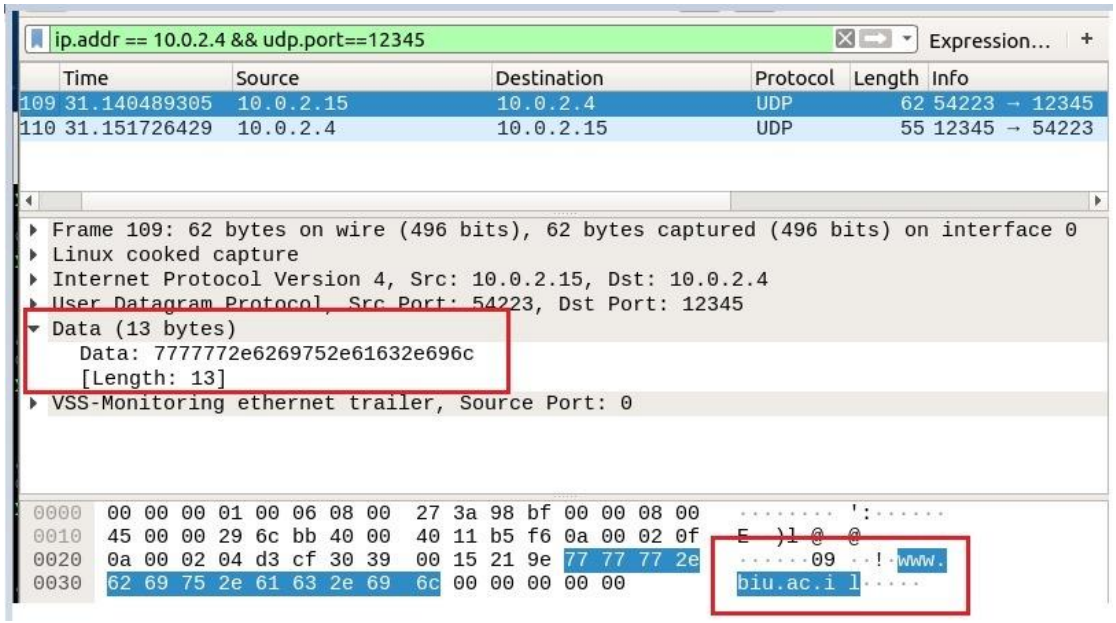
ג. הלקוח הזין בקשה עם דומיין של אתר שהשרת שלנו "למד" קודם לכן. חשוב לציין שבמקרה זה <u>עדיין לא הסתיים ה-TTL של הבקשה שנלמדה</u> (אחרת היא הייתה נמחקת מקובץ ה-ips.txt והשרת שלנו היה צריך ללמוד אותה מחדש).
במקרה זה, השרת שלנו יחזיר את כתובת ה-IP המבוקשת, בהתאם לרשומה שלמד קודם לכן משרת האב. מסיבה זו מקרה ג' דומה מאוד למקרה א'.

כעת נרצה להראות איך נבנו ההודעות שהלקוח שלח בכל אחת מהשכבות במודל שלמדנו, בהתאם לשלושת המקרים השונים שתיארנו לעיל. נראה זאת גם באופן זהה עבור ההודעות ששלחו השרת ושרת האב.

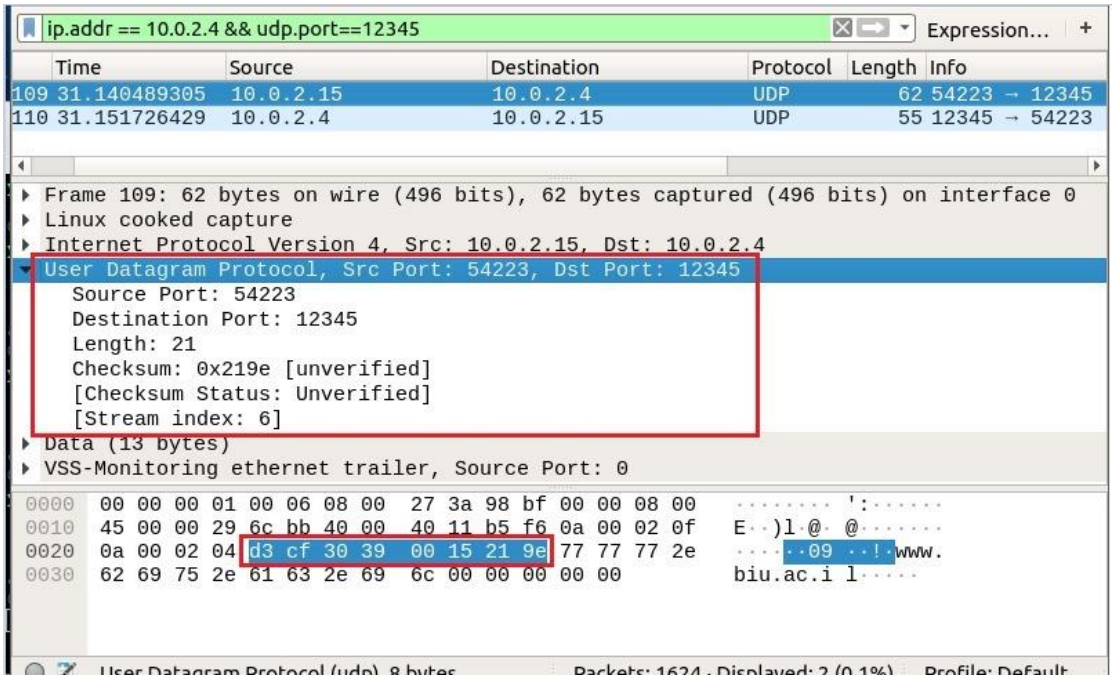
מקרה א':

1. <u>לקוח ← שרת</u>

שכבת האפליקציה: המידע שמיוצג בשכבה זו הוא ההודעה שנשלחה מהלקוח. ניתן לראות בריבועים המסומנים באדום בתמונה למטה את הבקשה, הדומיין שהזין הלקוח (www.biu.ac.il) – תחת הלשונית data בתוכנת ה-wireshark. המידע שהלקוח שלח מופיע (משמאל למטה) גם בצורה מקודדת הקסאדצימלית.



שכבת התעבורה: המידע שמיוצג בשכבה זו הוא הפורטים השונים בהם השתמשנו בהרצת השרתים והלקוח. חשוב לציין כי הפורט של הלקוח נבחר רנדומלית בהתאם לפורטים התפוסים / הפנויים במ"ה ברגע ההרצה הספציפי. מטרת הפורטים היא ליצור תקשורת מבוססת סוקטים בין הלקוח לשרת. ניתן לראות זאת בתמונה הבאה, תחת הלשונית User Datagram Protocol:



בלשונית זו ניתן לראות את 4 השדות כפי שלמדנו בכיתה –

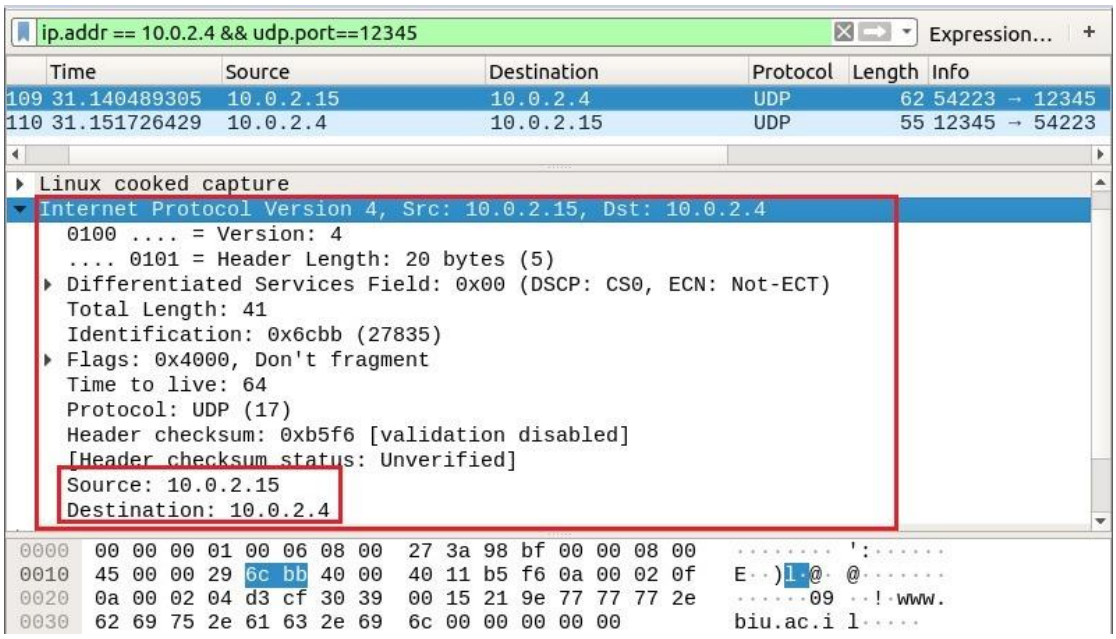
(54223) Source port – הפורט של הלקוח.

(12345) Destination port – הפורט של השרת.

(21) Length – גודל החבילה הכולל.

Checksum – שתפקידו זיהוי שגיאות.

שכבת הרשת: המידע שמיוצג בשכבה זו הוא כתובת ה-IPV4 של הלקוח (המקור) ושל השרת (היעד). בתמונה הבאה, תחת לשונית Internet Protocol Version 4:

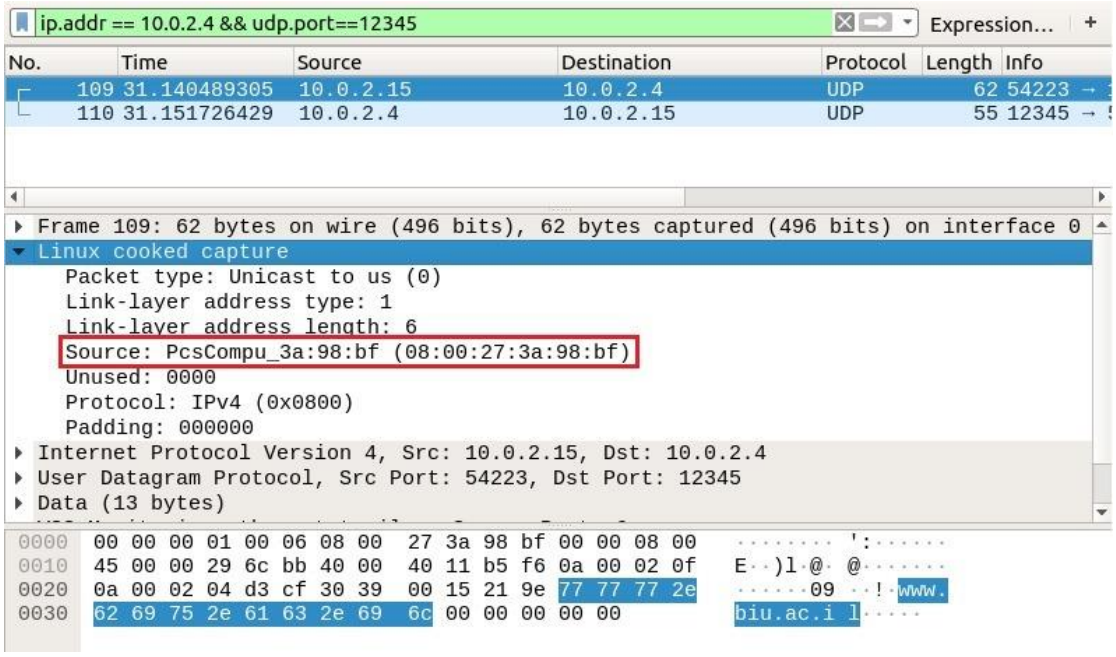


שכבת הערוץ: המידע שמיוצג בשכבה זו הוא כתובת ה-MAC של המקור, שמופיעה תחת הלשונית Linux cooked capture. ***גילוי נאות*** – חיפשנו בגוגל איך ניתן לראות

את כתובות ה-MAC ב-wireshark שהותקן על המכונה הוירטואלית (לינוקס). קריאה קצרה בלינק הבא סיפקה את התשובה עבורנו - https://wiki.wireshark.org/SLL

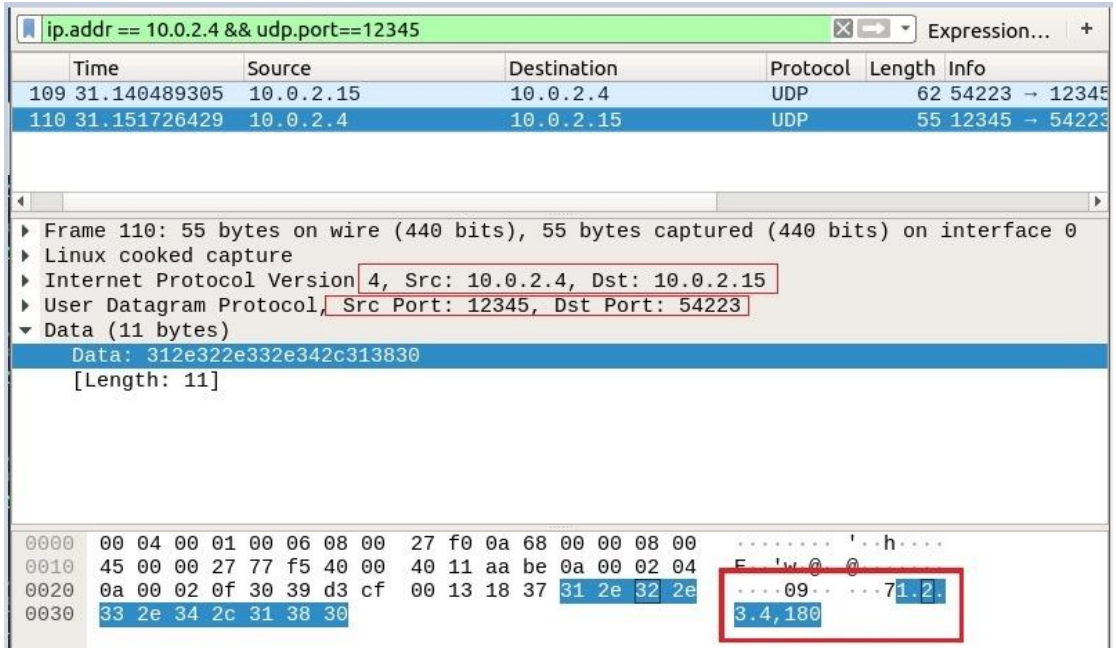
להלן צילום מסך מתוך הלינק הנ"ל:

When capturing from the "any" device, or from one of those other devices, in Linux, the libpcap doesn't supply the link-layer header for the real "hardware protocol" like Ethernet, but instead supplies a fake link-layer header for this pseudo-protocol.

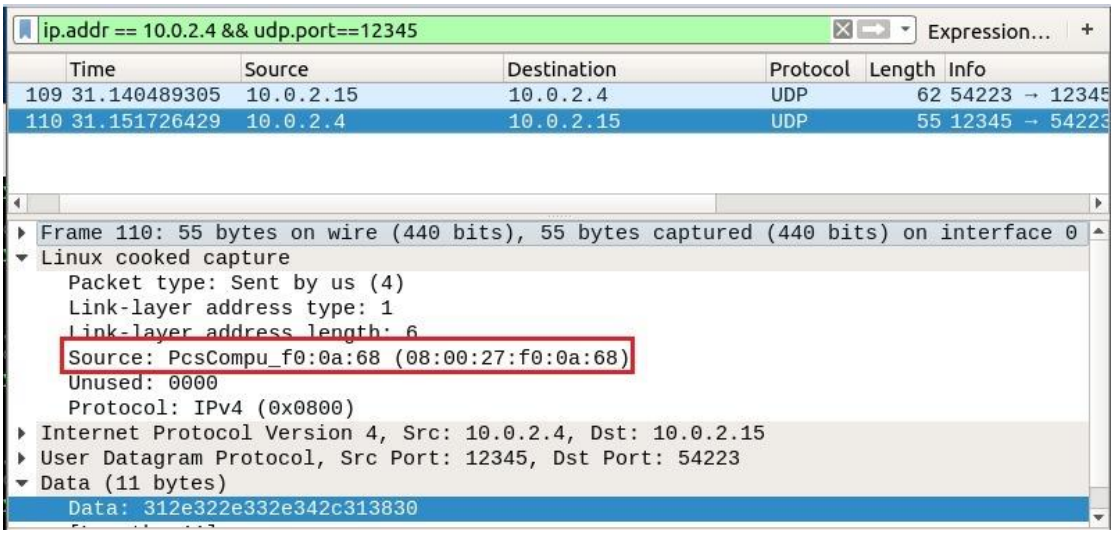


2. <u>שרת ← לקוח</u>

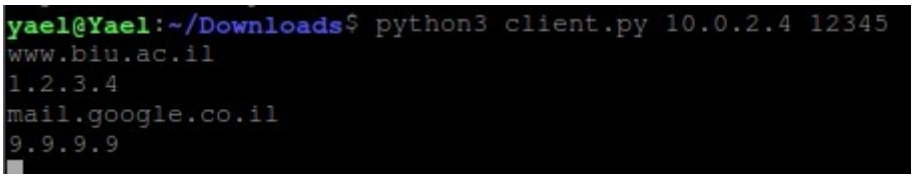
החבילה שנשלחה מהשרת ללקוח דומה לחבילה הקודמת (מהלקוח לשרת) מבחינת האופן שבו השכבות השונות מיוצגות ע"י המידע בחבילה. כלומר, המידע המיוצג בהודעה שהחזיר השרת ללקוח דומה מאוד למה שפירטנו ב-1, אך עם שינויים קלים כמפורט להלן:



שכבת האפליקציה: ניתן לראות כמסומן בריבוע האדום התחתון את תגובת השרת כפי שנשלחה ללקוח.
שכבת התעבורה: חל היפוך מבחינת פורט המקור ופורט היעד לעומת מה שראינו ב-1, כפי שניתן לראות בריבוע האדום האמצעי.
שכבת הרשת: באופן דומה, חל היפוך מבחינת כתובת ה-IP של המקור לעומת כתובת ה-IP של היעד לעומת מה שראינו ב-1, כפי שניתן לראות בריבוע האדום העליון.
שכבת הערוץ: בתמונה הבאה ניתן לראות שכתובת ה-MAC של המקור השתנתה:

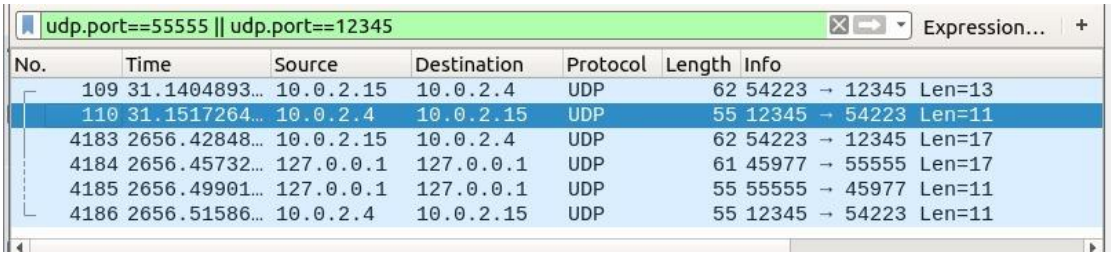


מקרה ב':



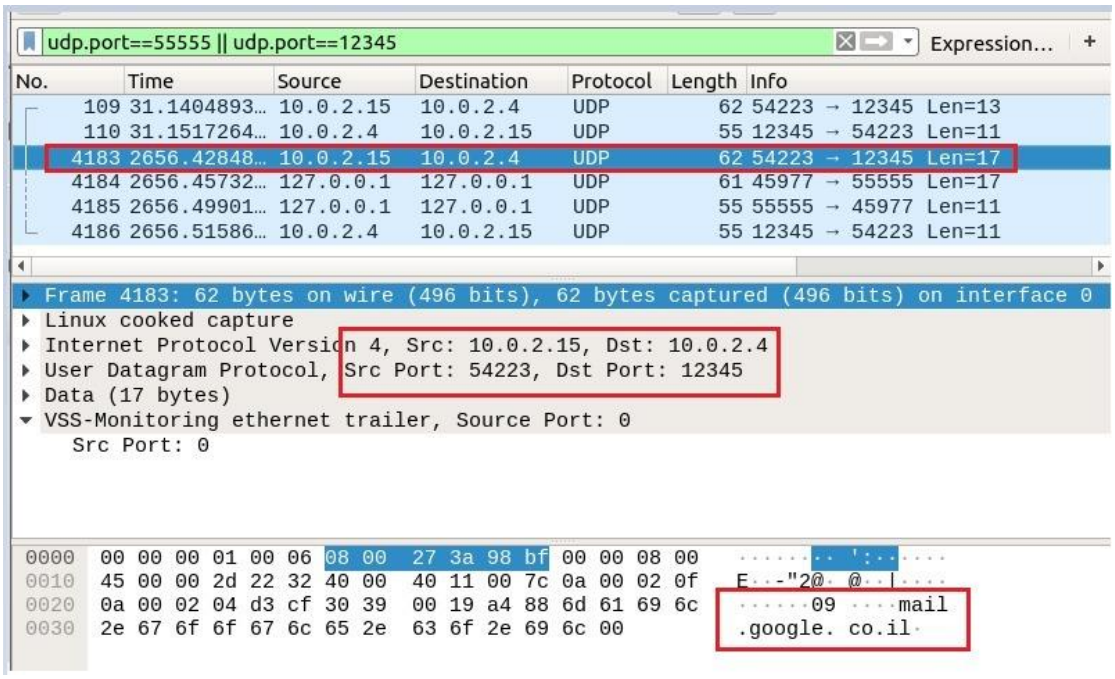
כאמור, במקרה זה הזין הלקוח בקשה לקבלת IP של דומיין (mail.google.co.il) שאינו נמצא בקובץ ips.txt. השרת שלנו פנה לשרת האב בבקשה לקבלת הרשומה המבוקשת, למד אותה, והחזיר את תשובתו ללקוח (9.9.9.9).

ניתן לראות כי התווספה לקובץ ips.txt שורה חדשה שמכילה את הרשומה החדשה שהשרת למד.

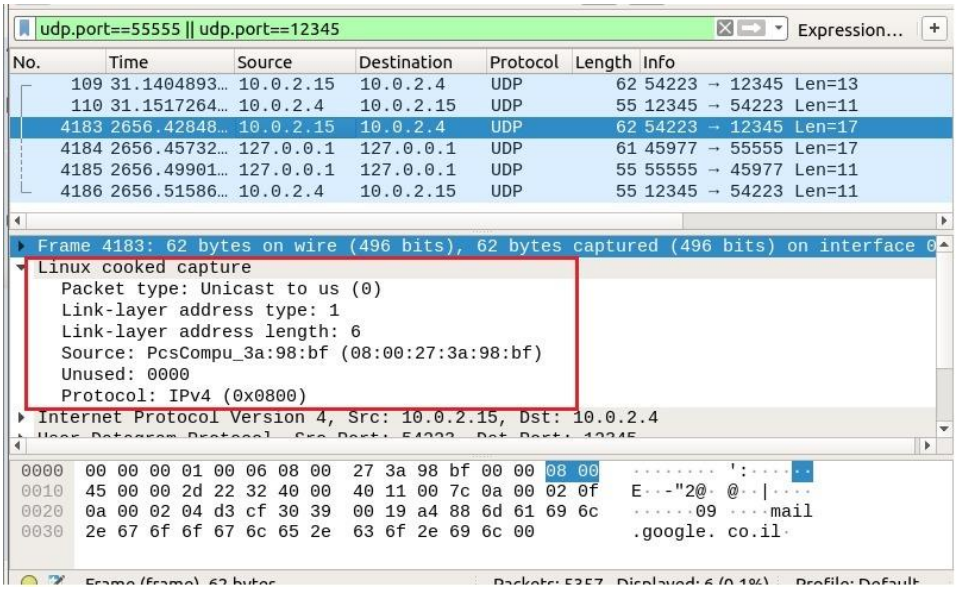


בתצלום הנ"ל ניתן לראות סינון שונה מזה שראינו קודם לכן. הסינון כעת הוא לפי הפורט של שבת האב או לפי הפורט של שרת הבן. ניתן לראות בשורות 109,110 את אותן הבקשות כפי שראינו במקרה א'. בשורה 4183 רואים את הבקשה החדשה ששלחנו (מקרה ב') מהלקוח לשרת. ניתן גם לראות את הפנייה לשרת האב (פורט 55555) ואת התשובה שהוא מחזיר (שורה 4185). לבסוף, בשורה 4186, השרת שלנו מחזיר את התשובה ללקוח.

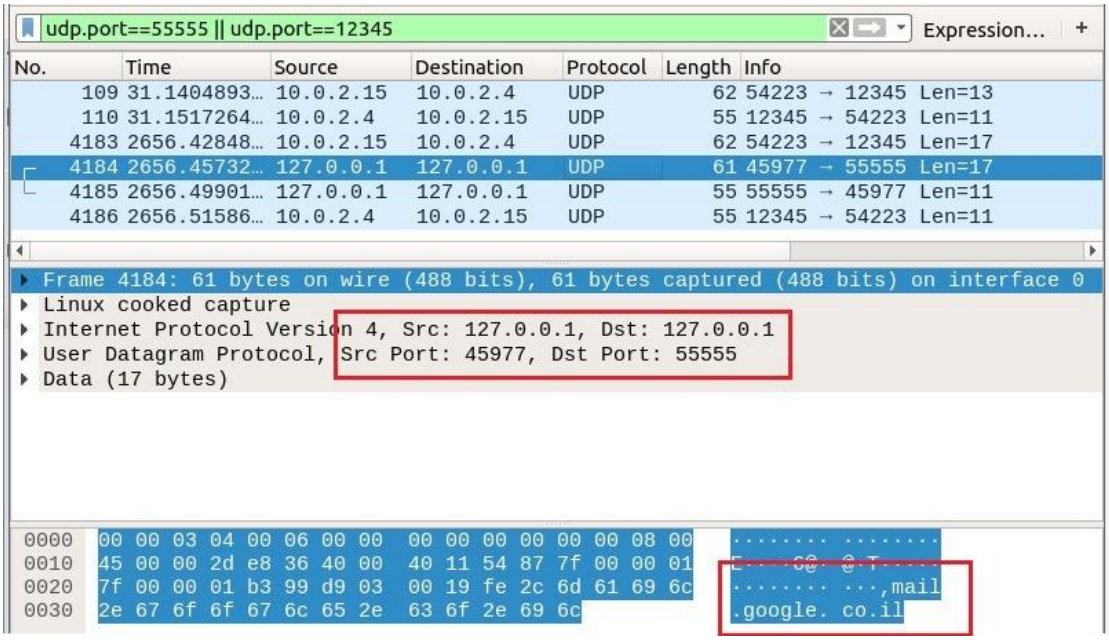
1. <u>לקוח ← שרת</u>



שכבת האפליקציה: כאמור, הלקוח שולח בקשה עבור הדומיין mail.google.co.il.
מידע זה בא לידי ביטוי כפי שניתן לראות בלשונית ה-DATA ובריבוע האדום התחתון בתמונה.
שכבת התעבורה: ניתן לראות שפורט המקור הוא הפורט של הלקוח (54223) ופורט היעד הוא הפורט של השרת (12345).
שכבת הרשת: כתובות ה-IP של הלקוח (10.0.2.15) ושל השרת (10.0.2.4).
שכבת הערוץ: כפי שראינו קודם לכן, בלשונית ה-linux cooked capture ניתן לראות את כתובת ה-MAC של המקור (הלקוח):



2. <u>שרת ← שרת אב</u>

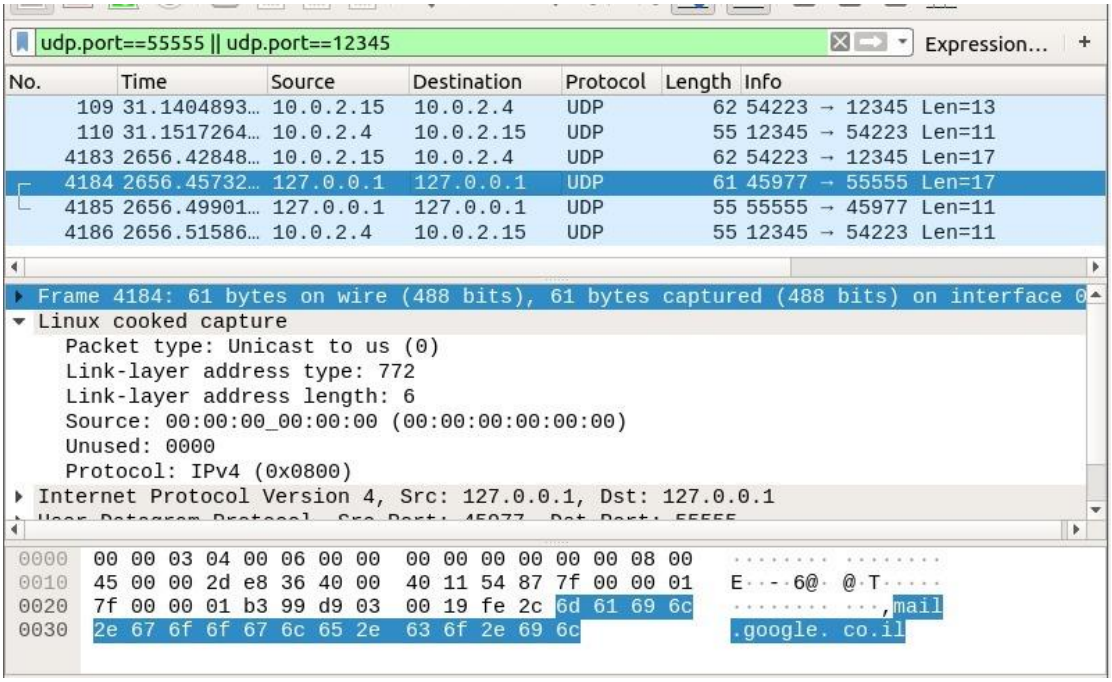


שכבת האפליקציה: בתחתית התמונה לעיל ניתן לראות את הבקשה לדומיין, כפי שמועברת מהשרת שלנו אל השרת אב.

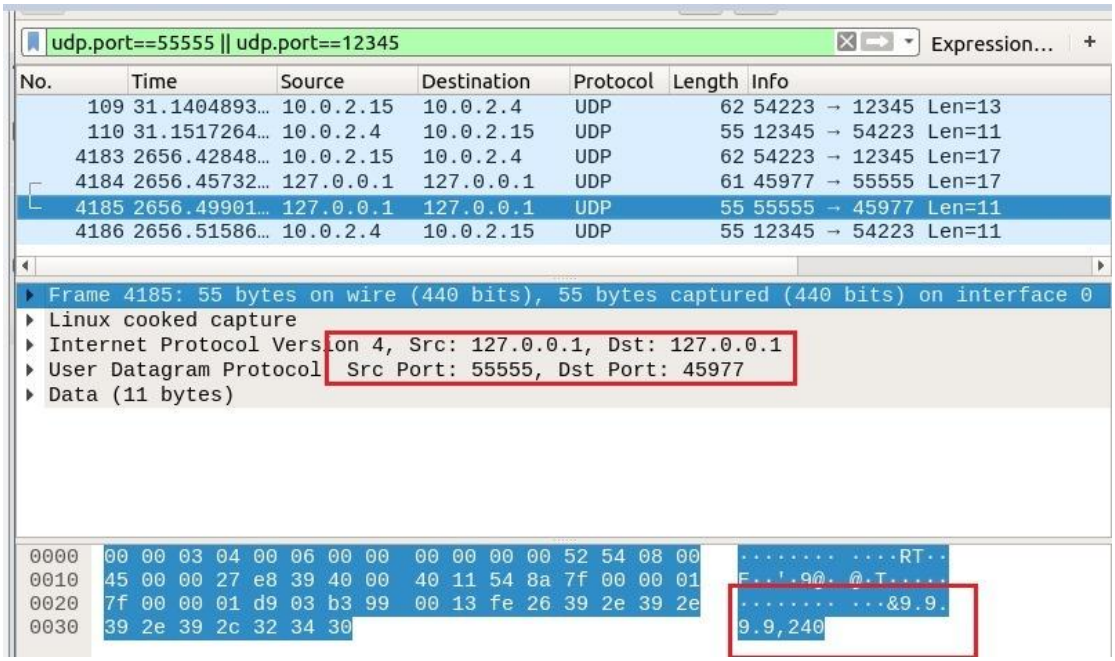
שכבת התעבורה: ניתן לראות פנייה מפורט 45977 לפורט 55555 של שרת האב.

שכבת הרשת: נשים לב שמכיוון ש-2 השרתים רצים על אותה מכונה וירטואלית, כתובת ה-IP של המקור ושל היעד זהות בשלב זה.

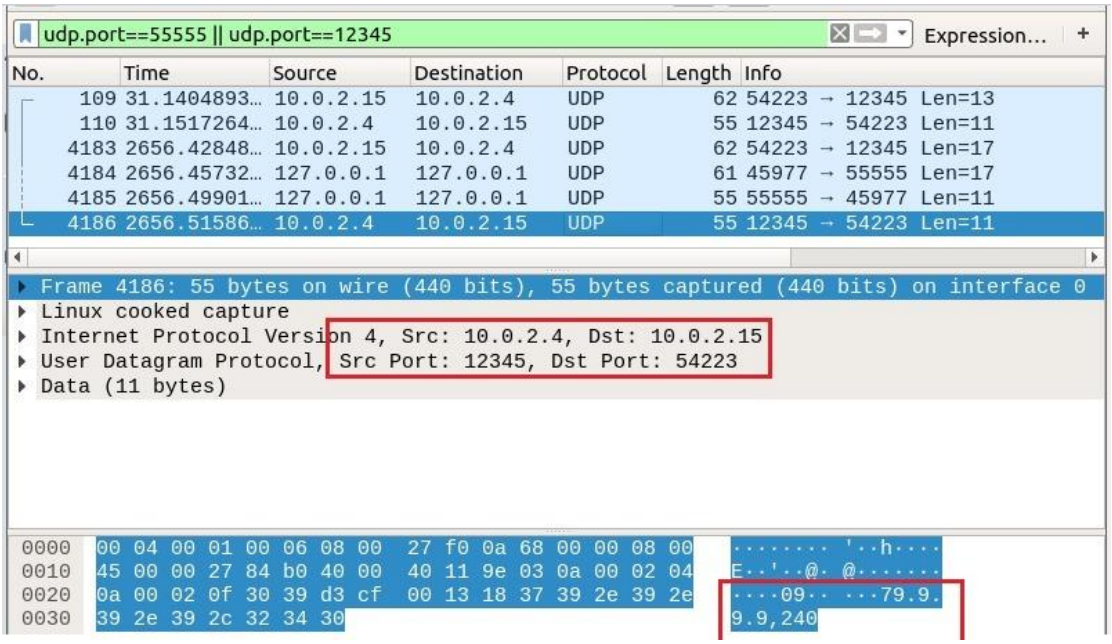
שכבת הערוץ: בדומה לסעיפים הקודמים, גם כאן ניתן לראות את כתובת הMAC של המקור. נשים לב שכתובת הMAC הפעם מופיעה כאפסים, כפי שניתן לראות בתמונה הבאה:



3. <u>שרת אב ← שרת:</u> דומה מאוד לשלב 2, רק בהיפוך של הפורטים של המקור והיעד. כמו כן, המידע בשכבת האפליקציה השתנה – שרת האב שלח את הרשומה הרלוונטית לשרת הבן, כפי שנראה בתמונה הבאה:



4. <u>שרת ← לקוח</u>



שכבת האפליקציה: ניתן לראות את התגובה שהשרת שלנו שולח ללקוח בריבוע האדום התחתון שבתמונה.

שכבת התעבורה: הפורט של השרת שלנו הוא 12345 (המקור), והפורט של הלקוח הוא 54223 (היעד).

שכבת הרשת: כתובות ה-IP של השרת ושל הלקוח - בריבוע האדום העליון.

שכבת הערוץ: בדומה לסעיפים קודמים, גם כאן ניתן לראות את כתובת הMAC של המקור (השרת).