

## פרויקט גמר

פרויקט זה עוסק בתקשורת שרת – לquo כארח התקשרות מתחילה מאפס. כלומר, מרגע שהמחשב נדלק (לפני שמתחבר ל-Wi-Fi ומקבל כתובת IP).

**בשלב הראשון** המחשב שלוח הודעת DHCP ב-Broadcast (הודעת שידור לכל שרת ה-DHCP הזמינים) ומתקשר עם שרת ה-DHCP עד שמקבל ממנו כתובת IP לשימוש.

**בשלב השני** כאשר יש למחשב כתובת IP לשימוש אשר משוויכת אליו, המחשב יוצר קשר עם שרת ה-DNS באמצעות בקשה לקבלת כתובת IP לכטובת domain אשר קיימת אצלו ומהכה לתגובה עד לקבלת כתובת ה-IP הרצiosa.

**בשלב השלישי** כאשר המחשב קיבל את כתובת ה-IP הרצiosa הוא מנסה להתחבר לשרת האפליקציה שלנו.

אנחנו בחרנו לעבוד עם שרת FTP אשר מיועד להעברת קבצים. FTP (File Transfer Protocol) הוא פרוטוקול רשות אשר משמש להעברת קבצים בין שני משתמשים דרך האינטרנט.

פרויקט זה נכתב ב-Python וצריך לרוץ עם הרשאות root על מנת שיוכל לטעוף ולנתח פאקטות בתעבורה הרשת. בנוסף, השתמשנו בספרייה Scapy על מנת לבנות ולשולח פאקטות. בכל קובץ השתמשנו ב"try" ו"-except" על מנת לטעוף שגיאות.

הפרויקט מחולק לאربעה חלקיים, DNS, DHCP ושרת FTP המכילים את הקבצים הבאים: בתיקיות של האפליקציה ישנים כמה קבצים לדוגמא אשר ניתן להוריד / למחוק.

על מנת להריץ את הפרויקט נפתח חלון טרמינל עבור כל אחד מהחלקים (אם אנחנו לא בתיקית הפרויקט ננווט לשם באמצעות הפקודה cd) ובכל חלון נריץ חלק אחר של המטלה בתוספת הפקודה sudo על מנת לתת הרשות root לתוכנית שתוכל להסניף ולנתח פאקטות.

## Client

בחלק זה כתבנו תכנית בפייתון המשתמש בספריית Scapy על מנת לבנות פאקטות ושלוח אותן. קוד זה מדמה צד לקוח.

הלקוח שלוח חבילה broadcast ב-**discovery** לכל המחשבים ברשת המקומיית על מנת לארר שרת DHCP. במידה וקיים שירות DHCP ברשת המקומיית בעלי כתובות פנויות לחלוקה, הלקוח מקבל חבילה **offer** עם כתובת IP מכל אחד מהם. (בנהנה שאין חסימה של מעבר חבילות DHCP בין הרשתות המקומיות, למשל על ידי Firewall).

הלקוח שלוח חבילה **request** עם הנתונים אותם בחר, גם כן בשידור - broadcast על מנת לעדכן את כל השירותים בכתובת שנבחרה, כך השירותים שהויצו ולא נבחרו ע"י הלקוח יודעים שהם יכולים להציג את הכתובת שלהם למחשב אחר ובנוסף יודעים מה הכתובת שהלקוח קיבל כדי שלא יקוץ אותה למחשבים אחרים.

השרת שלוח **acknowledge** (ACK) אישור שהוא קיבל את הבקשה. לאחר בקשה זו הלקוח מתחילה להשתמש בנזונים שקיבל. מעבר לכך בחבילה זו נשלחים לרוב הפרטים המאפשרים לקוחות להתנהל ברשת כמו-**subnet mask**.

תחילה מצאנו את כתובת ה-**mac** ע"י הפונקציה (**get\_mac()**) אשר משתמש בספריית subprocess כדי לקבל את כתובת ה-**mac** של הלקוח. לאחר מכן יצרנו חבילה DHCP Discovery באמצעות SCAPY בספרית.

הפקטה DHCP Discovery נוצרת ומוגדרת באמצעות ספריית Scapy. היא-**Ethernet** מוגדר לשידור הפקטות לכל המכניםים ברשת, ה-**IP** מגדרה את כתובת ה-**IP** המקורית ל-**0.0.0.0** וככתובת ה-**IP** של היעד ל-**255.255.255.255** כדי לציין הודעת שידור, ה-**UDP** מגדר את יציאות המקור והיעד ל-**68** ו-**67** בהתאם (היציאות הסטנדרטיות המשמשות את DHCP) והוא-**Bootp** מכיל את כתובת ה-**MAC** של הלקוח.

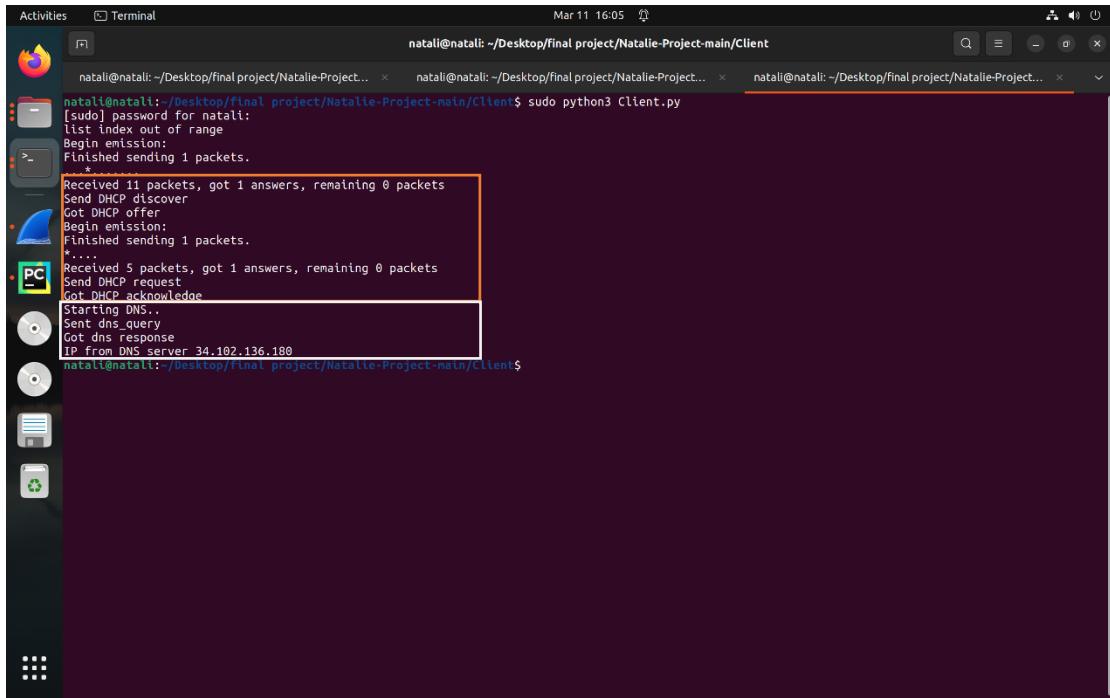
הפונקציה (**srp**) שלוחת את הפקטות DHCP Discovery ומחייב לפקטות המוצעות מכל שירות DHCP זמן. ה-**timeout** הינו הזמן המרבי בו הלקוח מחייב לתגובה משרת DHCP זמינים והמשתנה מאפשר לאסוף תשובות רבות המואחסנות בשתי רשימות **ans** ו-**unsans**, אם הלקוח לא קיבל תגובה, רשימה ה-**unsans** תכיל את פקעתה ה-**DHCP Discovery** שנשלחה.

לאחר מכן עבדנו על רשימת **ans** כדי למצוא את פקעתה "offer" של ה-**DHCP**, בדקנו את סוג ההודעה כדי לוודא שהפקטה היא הודעת "offer" של DHCP וכתובת ה-**IP** המוצעת נלקחת ממשם. יצרנו פקעתה "request" ל-**DHCP** באמצעות כתובת ה-**IP** שקבענו, והפונקציה (**srp**) נקראת שוב על מנת לשלוח את חבילת "request" ל-**DHCP** ולהמתין לתגובה אישור חזרה משרת ה-**DHCP** שנבחר.

עבדנו שוב על רשימת ה-**ans** כדי למצוא פקטות מה-**DHCP**, בדקנו שוב את סוג ההודעה הפעם כדי לוודא שהפקטה היא הודעת "ACK" משרת ה-**DHCP**.

לבסוף, פקעת DNS query נוצרת באמצעות שכבות UDP, IP ו-DNS, ונשלחת באמצעות (**srv**). כתובת ה-**IP** של שם הדומיין נשלפת מהתגובה ועודפסת למסך.

## הרצה של ה-client:



```
natali@natali:~/Desktop/final project/Natalie-Project-main$ sudo python3 Client.py
[natali@natali:~/Desktop/final project/Natalie-Project-main$ password for natali:
list index out of range
Begin emission:
Finished sending 1 packets.
*.....
Received 11 packets, got 1 answers, remaining 0 packets
Send DHCP discover
Got DHCP offer
Begin emission:
Finished sending 1 packets.
*.....
Received 5 packets, got 1 answers, remaining 0 packets
Send DHCP request
Got DHCP acknowledge
Starting DNS..
Sent dns_query
Got dns response
IP from DNS server 34.102.136.180
natali@natali:~/Desktop/final project/Natalie-Project-main$
```

בתמונה זו ניתן לראות כיצד הרצנו את ה-client שבניינו.

בריבוע הכתום ניתן לראות את התקשרות בין ה-client ל-DHCP. תחיליה הלוקו שולח ל-DHCP פאקטת "discovery" ולאחר מכן מקבל חזרה מה-DHCP פאקטת "offer" וככזה הלוקו יודע שהוא בקשר פתוח עם ה-DHCP (לאחר שקיבל פאקטת מענה ממנה, אנו רואים זאת בהדפסה למסך תחיליה בקשור פתוח עם ה-DHCP ("Got DHCP offer"- ו-"Send DHCP discover"). הלוקו שולח ל-DHCP פאקטת "request" ("Got DHCP acknowledge"- ו-"Send DHCP request"). ניתן לראות זאת בהדפס למסך, "Send DHCP acknowledge" (אקרטת ACK). (ניתן לראות זאת בהדפס למסך, "Got DHCP request"- ו-"DHCP response").

בריבוע הלבן ניתן לראות את התקשרות בין ה-client ל-DNS. תחיליה הלוקו שולח לשאילתת response ("Send dns\_query", "query") ולאחר מכן מקבל חזרה מה-DNS פאקטת מענה ("Got dns response"). השרת DNS מקבל דומיין מהлокו ומחייב לו חזרה את כתובת ה-IP בהתאם לדומיין שנשלח בשאילתת, במידה והוא קיים כМОון במאגר הכתובות של ה-DNS.

## DHCP

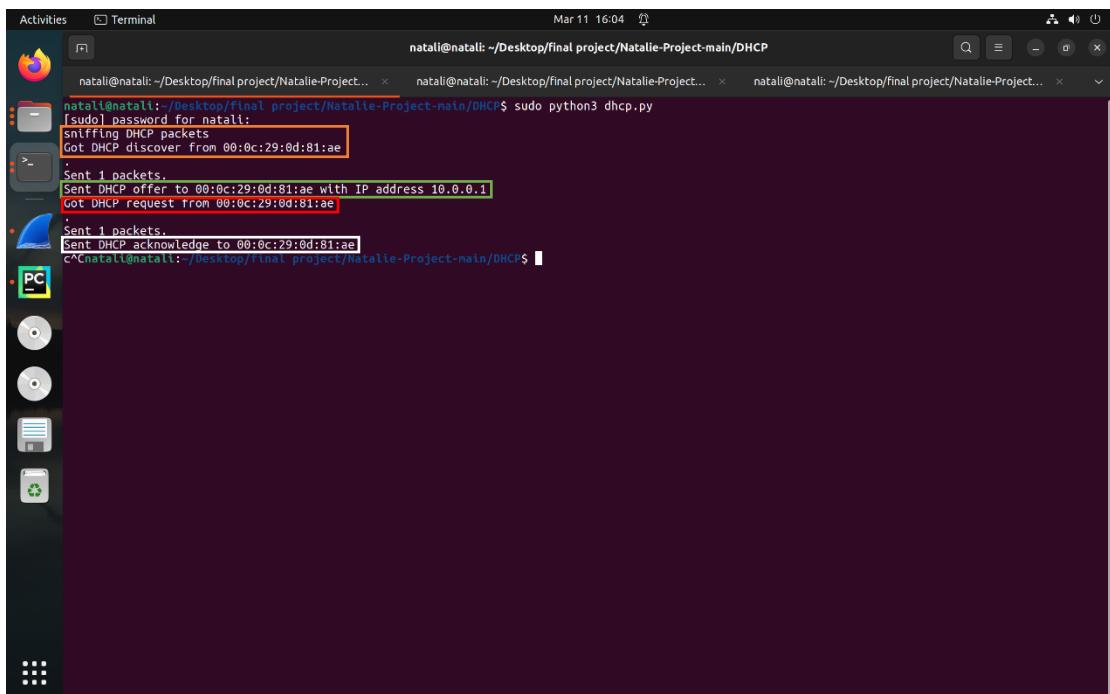
בחלק זה כתבנו תכנית בPython המממשת שרת DHCP. התכנית מסניפה פאקטות DHCP ב망 ש הרשות שצין במשתנה iface, ומגיבה בהתאם לבקשת DHCP מהлокו עם "offer" או "ACK" של DHCP. השתמשנו בספריה JSON המשמשת לקרוא קובץ בפורמט JSON על מנת לקרוא כתובות IP מהקובץ, ספריית ה- IPAddress משמשת לתמוך כתובות IP וספריית SCAPY המשמשת לבניית שליחת פאקטות. הגדרנו משתנה אשר מחזיק צימודים של IP ו-MAC שנקרא leased\_ip שנוצר כדי לעקוב אחר כתובות IP ש"הושכו" אל לקוחות (כתובות MAC).

פונקציית ה- `()load_Config()` קוראת קובץ בפורטט JSON המכיל קבוצה של מאגר כתובות IP. פונקציית `(()allocate_ip()` בוחרת כתובות IP זמינים מהמאגר, הפונקציה מחפשת כתובות IP פנויות ומחדירה את כתובות ה- IP הראשונה הזמיןה במאגר. אם אין כתובות IP זמינים, היא לא מחדירה אף אחת. פונקציה הנקראת `(()release_ip()` מוגדרת כדי לשחרר כתובות IP מלוקו על ידי מחיקתו מ- `.leased_ips`.

פונקציית ה- `(()dhcp()` היא המטפלת העיקרית עבור פאקטות DHCP. תחילה היא בודקת אם הפקטה היא "request" או "discovery", אם מדובר בפקחת discovery ההפונקציה בוחרת כתובות IP זמינים עבור הלוקו באמצעות פונקציית `(()allocate_ip()` ובונה פאקטת "offer" של DHCP "ACK" DHCP "request" בונה פאקטת "ACK" עם כתובות ה- IP המבוקשת ושולחת אותה חזרה ללוקו.

בסוף, הגדרנו וחרחן פאקטות המשמש בפונקציית `sniff()` של SCAPY על מנת להאזין לפאקטות DHCP ב망 ש הרשות שצינו. כאשר מתקבלת פאקטת DHCP פונקציית ה- `(()dhcp()` נקראת לטפל בה.

הרצת שרת DHCP:



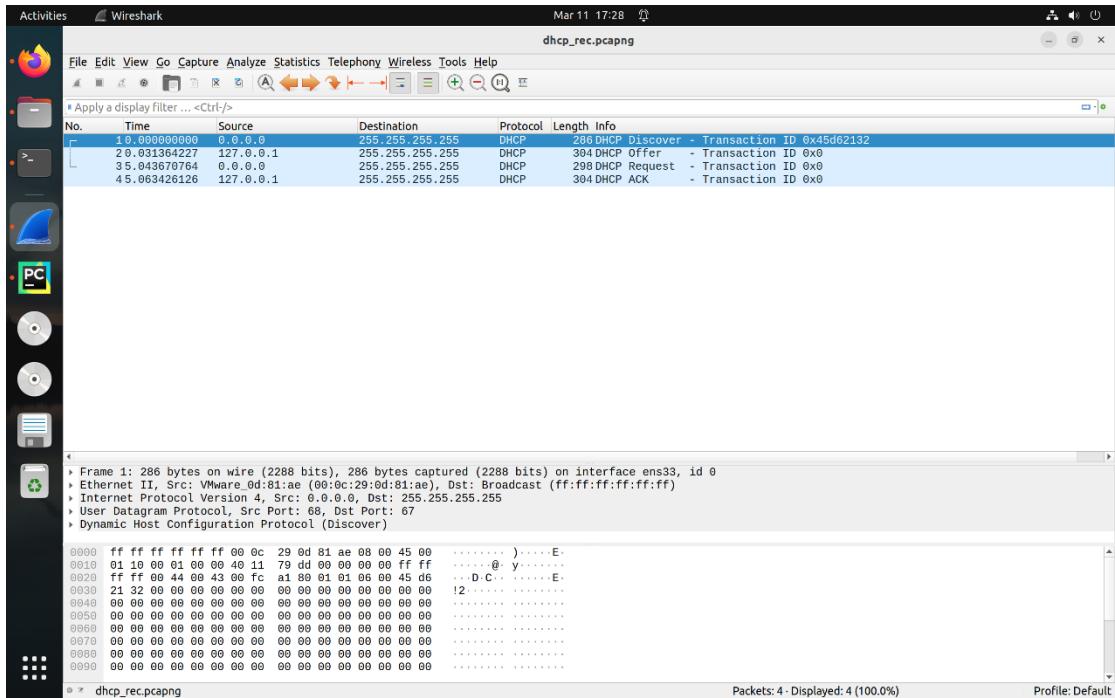
```
natali@natali:~/Desktop/final project/Natalie-Project... natali@natali:~/Desktop/final project/Natalie-Project... natali@natali:~/Desktop/final project/Natalie-Project...
[natali@natali:~/Desktop/final project/Natalie-Project...] sudo python3 dhcp.py
[sudo] password for natali:
sniffing DHCP packets
Got DHCP discover from 00:0c:29:0d:81:ae
:
Sent 1 packets.
Sent DHCP offer to 00:0c:29:0d:81:ae wth IP address 10.0.0.1
Got DHCP request from 00:0c:29:0d:81:ae
:
Sent 1 packets.
Sent DHCP acknowledge to 00:0c:29:0d:81:ae
c^Cnata@natali:~/Desktop/final project/Natalie-Project/DHCP$
```

בתמונה זו ניתן לראות כיצד הרצה את ה"`dhcp`" שבנו.

בריבוע הכתוב ניתן לראותה שה-`dhcp` מתחילה בהסנה של פאקטות ברשת ומקבל פאקטת "discovery" מהлокו, ניתן לראותה שהוא מקבל זאת מכתובת ה-MAC של הלוקו כי הלוקו רוצה לקבל משרת ה-`dhcp` את כתובות ה-IP שלו.

בריבוע הירוק הקפץ שולח פאקטת "offer" ללקוח (כתובת MAC שמננה קיבל פאקטת גילוי) עם כתובת IP של הלוקוח.

בריבוע האדום הלוקוח שולח לקפץ DHCP פאקטת "request" ולאחר מכן בריבוע הלבן הקפץ שולח פאקטת "ACK" ללקוח כגובהה על פאקטת ה-"request".



בתמונה זו ניתן לראות את הפאקטות הנשלחות בין ה-"Client"-ל-"**dhcp**".

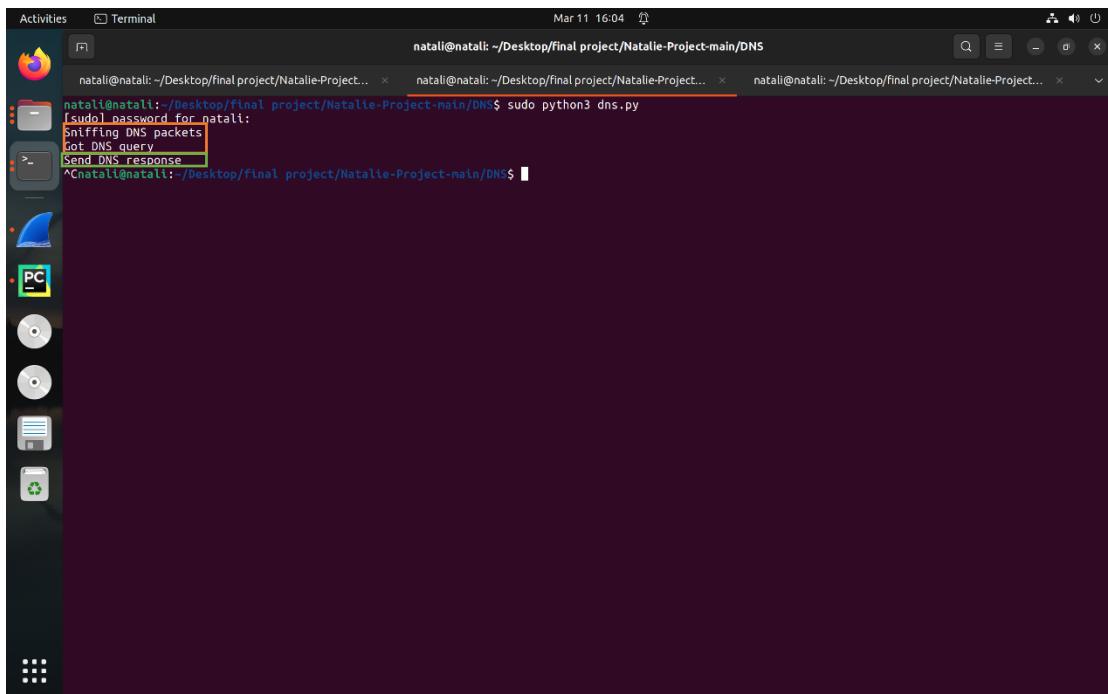
בפאקטה הראתה DHCP discovery ניתן לראות הודעה "broadcast" של הלוקוח לרשות שנשלחת מ-0.0.0.0 כיון שאין ללקוח עדיין IP. לאחר מכן בפאקטת "offer" השניה ניתן לראות מענה של dhcp ללקוח גם ע"י הודעה שידור ולאחר מכן קפץ שולח פאקטת "offer" ללקוח עם כתובת IP שלו, הלוקוח שולח חזרה פאקטת DHCP request והקפץ מחזיר לו פאקטת "ACK" חזרה.

## DNS

תכנית זו ממחשת שרת DNS באמצעות ספרית Y-SCAPY. שרת ה-DNS מקשיב לשאלות DNS בפורט UDP 53, ומגיב לשאלות עם כתובות IP מוגדרת מראש. הגדרנו את כתובות ה-IP ואת מספר הפורט של שרת ה-DNS. תחילה יבאו את ספרית Scapy בכדי להשתמש בפונקציות הדרישות לעובדה עם פאקטות DNS וכותרות UDP/IP. לאחר מכן הגדרנו צימודים של רשומות DNS, כאשר המפתחות הם שמות דומיינים והערכים הם כתובות IP. זה בסיס הנתונים של שרת ה-DNS.

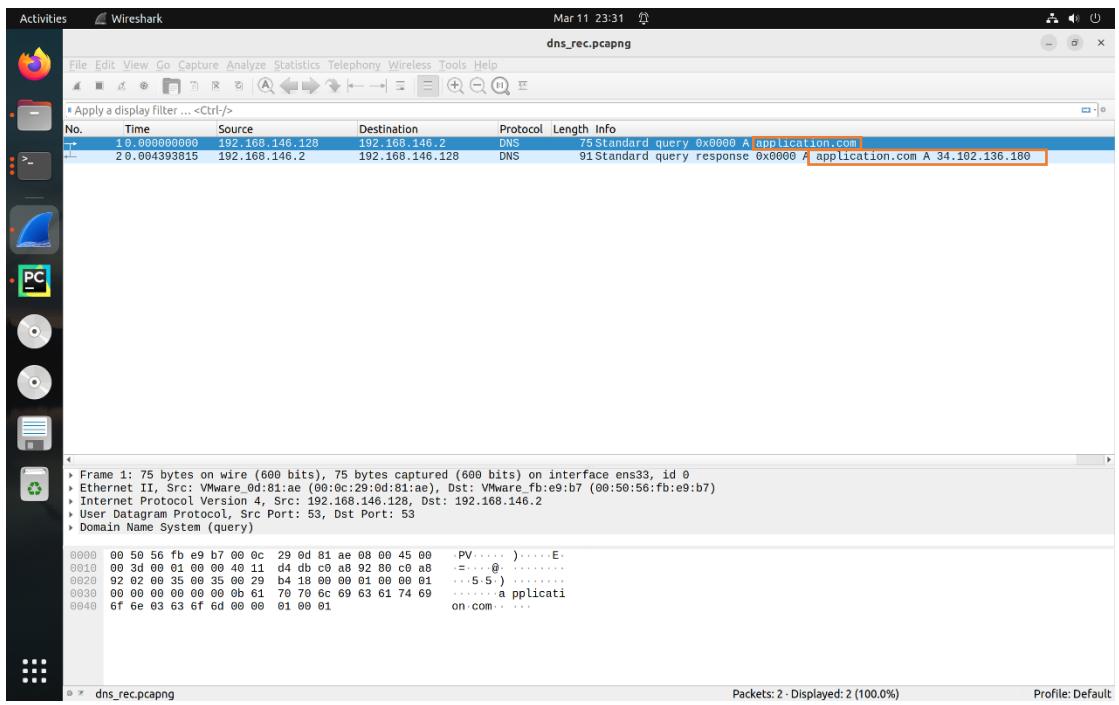
הפונקציה () dns בודקת אם הפקטה היא שאלת DNS ואם הפקטה היא שאלה DNS שלא נועתה, הפונקציה מחלצת את שם שאלת ה-DNS, בודקת אם הוא נמצא בבסיס הנתונים של ה-DNS(בצימודים שהגדכנו), ואם כן, היא בונה חבילת תגובה של DNS עם כתובות ה-IP המתאימה ושולחת אותה בחזרה ללקוח.

פונקציית () sniff משמשת לתפיסת פאקטות DNS וקוראת לפונקציה () dns כדי לטפל בהן. התוכנית מתחליה לבסוף את שרת ה-DNS על ידי קרייה לפונקציית sniff מ-Scapy.



בתמונה זו ניתן לראות כיצד הרץנו את ה"dns" שבנו.

בריבוע הכתום ניתן לראות שהdns מתחילה בהסנה של פאקטות בראש ומקבל פאקטת "query" מהלקוח ובה יש דומיין שהלקוח מחפש את כתובת ה-IP המשויכת לו ("Got DNS query"). בריבוע הירוק החם שולח פאקטת "response" בחזרה ללקוח ובה ה-IP (הערך) של הדומיין (המפתח) שקיבל מהלקוח ("Send DNS response").



בתמונה זו ניתן לראות את הפקודות הנשלחות בין ה-"Client" ל-"dns" ב-Wireshark.

בפקודה הראשונה הלקוח מקבל פקודה "query" מהלкова עם הדומין שיש לו ובפקודה השנייה הלקוח מחזיר פקודה "response" ללוקה עם צימוד הIp של הדומין שקיבל. (ניתן לראות שאלה הפקודות שלנו בריבועים הכתומים שכן מדובר בחומר application).

בנינו צד שרת אפליקציה (FTP) וצד ללקוח אשר כדי לתקשר ביניהם נדרש שלא יהיה באותו נטיב, נפתח בטרמינל בתיקייה אחת את הלקוח ובטרמינל נפרד נפתח את תיקיית-server וכך נוכל להעביר קבצים בין התקויות. מכיוון שההעברה של הקבצים מתבצעת בצורה BINARIA אין בעיה להעביר כל סוג קובץ שנרצה. צירפנו לצד הלקוח קובץ טקסט וקובץ תמונה אשר ניתן להעיבר ולבדק את הפונקציונליות של השרת והלקוח.

### Client - TCP

תכנית זו ממחשת צד ללקוח המתחבר לשרת ה-FTP באמצעות Socket. בחלק זה הגדרנו כמה פונקציות המטפלות בסוגים שונים של הודעות שניתן לשולחן לשרת:

1. **(transfer(filename))** - שולחת הודעה לשרת FTP עם שם קובץ ומורידה את הקובץ מהשרת ללקוח. הפונקציה מקבלת את שם הקובץ. בהתאם היא שולחת את שם הקובץ לשרת כמחרוזת עזרת פונקציית ()client.send, לאחר מכן היא פותחת את הקובץ באמצעות הפונקציה ()open במצב כתיבה BINARIA. לאחר מכן הפונקציה נכנסת לולאה שבה היא מקבלת חלקים מהקובץ באמצעות שיטת ()client.recv כאשר גודל המאגר מוגדר ל bytes(BUFFER\_SIZE 64). הפונקציה כותבת כל חתיכת קובץ לקובץ המלא באמצעות הפונקציית ()f.write עד שהיא מקבלת חלק מסוימת במחרוזת "DONE". כאשר זה קורה, היא סגורת את הקובץ ומדפיסה הודעה המציין שההעברה הושלמה. אם היא מקבלת חלק המכיל את המחרוזת "File not found", היא מדפיסה הודעה שגיאה, סגורת את הקובץ ומסירה את החלק שהספיק לעבר מהלקוח באמצעות הפונקציה ()os.remove .

2. **(upload(command))** - שולחת הודעה לשרת עם שם קובץ ומעלה את הקובץ מהלקוח לשרת. הפונקציה מחרוזת המכילה את השם של הקובץ שיישלח. תחילת היא שולחת את הפוקודה לשרת כמחרוזת מקודדת בפונקציית ()client.send, אם הקובץ קיים במחשב הלקוח, הפונקציה פותחת את הקובץ באמצעות הפונקציה ()open במצב קרייה BINARIA עם שם הקובץ והנתיב. לאחר מכן היא נכנסת לולאה שבה היא קוראת חלקים מהקובץ באמצעות שיטת ()f.read אשר גודל המאגר מוגדר ל- bytes(BUFFER\_SIZE 64). הפונקציה שולחת כל חתיכת קובץ לשרת באמצעות שיטת ()client.send עד שהיא מגיעה לסוף הקובץ. כאשר זה קורה, היא שולחת את המחרוזת "DONE" לשרת כדי לציין שההעברה הושלמה וסגורת את הקובץ. אם הקובץ לא קיים במחשב של הלקוח, הפונקציה שולחת את המחרוזת "File not found" לשרת.

3. **(msg\_send(path))** - פונקציה זו אחראית על שליחת הודעה רגילה מהלקוח לשרת FTP וקבלת תגובה. הפונקציה מקבלת msg כפרמטר, שהוא הודעה שיש לשולחן. ושולחת אותה לשרת באמצעות שיטת ()client.send. לאחר מכן נכנסת לולאה שבה היא מקבלת חלקים מהקובץ מהשרת באמצעות פונקציית ()client.recv אשר גודל ה-buffer מוגדר ל- bytes(BUFFER\_SIZE 64). הפונקציה משלשת כל חלק מהקובץ למשתנה עד שהיא מקבלת חלק ריק של נתונים, מה שמצויב על כך שההodata התקבלה במלואה. לאחר מכן היא מדפיסה הודעה אשר התקבלה.

הפונקציה העיקרית של התכנית מגדרה לולאה שמחכה לקלט מהשרת וקוראת לפונקציה המתאימה על סמך הקלט שהוכנס. אם המשמש מזמן פוקודה "get" או "put", פונקציית ()send או ()upload בקריאה אחרת. אחרת, פונקציית ()msg נקראת לשולחן הודעה רגילה לשרת.

הגדכנו בתחלת התכנית כמה קבועים כגון:

- **BUFFER SIZE** - מגדיר את גודל המאגר לשיליחה וקבלת נתונים דרך Socket 7-64 בתים.
- **PORT** - מגדיר את מספר היציאה ל-20941, שהוא מספר הייחודי שהשרתamazon בו.
- **DISCONNECT\_MESSAGE** - מגדיר את הודעה הניתוק "DISCONNECT!" שתישלח לשרת כאשר הלקוח ינותק.
- **SERVER** - מגדיר את כתובת ה-IP של השרת אליו הלקוח יתחבר.
- **ADDR** - יוצר tuple המכיל את כתובת ה-IP של השרת וה-PORT.

התכנית מטפלת במקרים חריגיים שיכולים להתרחש במהלך פתיחת וסירה של הקשה.

## Server - TCP

תוכנית זו מימושת שירות FTP (File Transfer Protocol) בפייתון אשר מאפשר עד 5 לקוחות להתחבר בו זמנית ולהעלות / להוריד / למחוק קבצים. בתחלת התוכנית הגדרנו כמה קבעים אשר נחוצים לנו לפעולת השירות (כגון IP של השירות).

הfonקציה ()**transfer** היא האחראית על העברת קבצים מהשרת ללקוח והfonקציה ()**upload** אחראית על העלאת קבצים מהלקוח לשרת. הfonקציה ()**client** אחראית על התקשרות עם הלקוח.

הfonקציה ()**client** מתחילה בהדפסת הודעה המציינת שלקווח חדש התחבר. לאחר מכן נכנסת לולאה שנמשכת עד שהלקוח שלוח את הודעה הניתוק. בתוך הלולאה, הfonקציה מקבלת הודעה מהלקוח באמצעות פונקציית ()**recv** של ספריית socket כדי לקבל את הקובץ הניתוק, הfonקציה יוצאת מהלולאה וסגורת את החיבור. אם הודעה היא בקשה להעברת קובץ (מסומן בקידמת "get", הfonקציה קוראת לפונקציה ()**transfer** כדי לשולח את הקובץ ללקוח. אם ההודעה היא בקשה למחיקה של קובץ הפונקציה קוראת לפונקציה ()**delete\_file** כדי למחוק את הקובץ מהשרת. אם הודעה היא בקשה להעלות קובץ (מסומן על ידי הקידמת "put"), הfonקציה קוראת לפונקציה ()**upload** כדי לקבל את הקובץ מהלקוח. אם הודעה היא בקשה לראות את הקבצים שבספרייה ללקוח. אם השירות (על ידי הודעה "ls" או "dir"), הfonקציה שולחת את רשימת הקבצים בספרייה ללקוח. אם הודעה היא כל דבר אחר, הfonקציה שולחת את תפריט האפשרויות ללקוח (MENU).

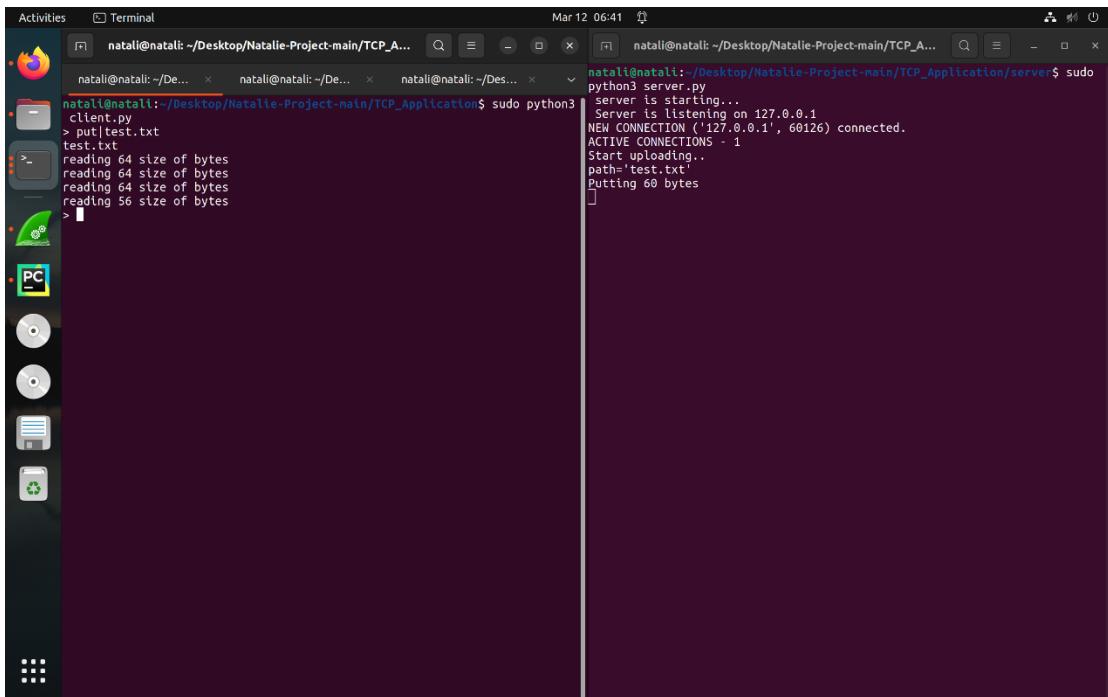
הfonקציה ()**transfer** לוקחת את שם הקובץ מההודעה על ידי פיצול בתו "." ולキחת הפרמטר שבא לאחרתו זה. לאחר מכן היא פותחת את הקובץ לקריאה במצב ביןארי ומתחילה לשולח את תוכן הקובץ ללקוח בחתיכות בגודל buffer שצוין. היא שולחת הודעה "DONE" ללקוח כאשר היא מסיימת לשולח את הקובץ.

הfonקציה ()**upload** לוקחת את שם הקובץ מההודעה כמו בfonקציה ()**transfer**. לאחר מכן היא פותחת את הקובץ לכטיבה במצב ביןארי ונכנסת לולאה שנמשכת עד שהקובץ יכול לקבל. בתוך הלולאה, הfonקציה מקבלת נתונים מהלקוח באמצעות פונקציית ()**recv** של ספריית socket וכותבת אותו לקובץ. אם העברת המידע מסתיימת בהודעת "DONE", הfonקציה סגורת את הקובץ ויוצאת מהלולאה.

הfonקציה ()**delete\_file** מקבלת את שם הקובץ ומוחקת אותו מהשרת.

הfonקציה ()**main** יוצרת socket, מחברת אותו לכתובת שצינה, ומתחילה להאזין לתקשורת כניסה. היא נכנסת לולאה שנמשכת עד שהתוכנית מופסקת על ידי המשמש (למשל על ידי לחיצה על Ctr+C). בתוך הלולאה, היא מקבלת חיבורים נוספים ויצירת thread חדש לטיפול בכל חיבור נוסף. היא מדפיסה את מספר החיבורים הפעילים לאחר כל חיבור חדש.

## הרכבת האפליקציה:



```
natali@natali:~/Desktop/Natalie-Project-main/TCP_Application$ sudo python3 client.py > put|test.txt test.txt reading 64 size of bytes reading 64 size of bytes reading 64 size of bytes reading 56 size of bytes >

natali@natali:~/Desktop/Natalie-Project-main/TCP_Application$ sudo python3 server.py server is starting... Server is listening on 127.0.0.1 NEW CONNECTION ('127.0.0.1', 60126) connected. ACTIVE CONNECTIONS - 1 Start uploading.. path='test.txt' Putting 68 bytes
```

ניתן לראות בצד ימין של התמונה את שרת האפליקציה ובצד שמאל של התמונה את הלקוח שמתחבר אליו. את שניהם נרץ עם הפקודה סודו אשר נותנת להם הרשות רוט לגשת לשכבות נמוכות יותר של התעבורה.

ברגע שנפעיל את השרת הוא ידפיס שהוא מתחילה לעבוד ולהאזין (להסניף) IP 127.0.0.1, ברגע שלקוח מתחבר אליו ניתן לראות שהוא מדפיס שלקוח חדש מתחבר עם הودעה של כמה לקוחות מחוברים כרגע. כתע הלוקו מוחובר לשרת ונitin להעלות אליו קבצים. נשתמש בפקודה המתאימה להעלאת קובץ לשרת ביחיד עם שם הקובץ ("טוק" כפי שמופיע בתמונה בצד הלקוח) וNILCHZ enter. ניתן לראות שהלקוח מדפיס למסך כל פאקטה שהוא שולח עם כמות הבטים שיש באזת אפקטה ואצל השרת שהוא מקבל ומתחיל להעלות את הקובץ. לאחר שהקובץ הועבר ניתן לראות באזת שם הוא נשמר (אותו שם של הקובץ המקורי) וכמה בתים התקבלו.

```
natali@natali:~/Desktop/Natalie-Project-main/TCP_Application$ sudo python3 client.py > get|test.txt
reading 64 size of bytes
reading 64 size of bytes
reading 64 size of bytes
reading 60 size of bytes
[+] Transfer completed
>

natali@natali:~/Desktop/Natalie-Project-main/TCP_Application$ sudo python3 server.py
server is starting...
Server is listening on 127.0.0.1
NEW CONNECTION ('127.0.0.1', 60126) connected.
ACTIVE CONNECTIONS - 1
Start uploading...
path='test.txt'
Putting 68 bytes
NEW CONNECTION ('127.0.0.1', 48308) connected.
ACTIVE CONNECTIONS - 2
Getting 64 bytes
Getting 64 bytes
Getting 64 bytes
Getting 56 bytes
```

בתמונה זו אנו שוב רואים את השרת אבל הפעם עם ליקוח חדש שמתחבר אליו, ניתן לראות שהשרת קיבל חיבור חדש על פי הודעה שהוא מדפס למסך (ניתן לשים לב להבדלים בפורט אשר מודפס ליד ה-IP) וההודעה אשר מעדכן על כמה משתמשים מחוברים כרגע (2).

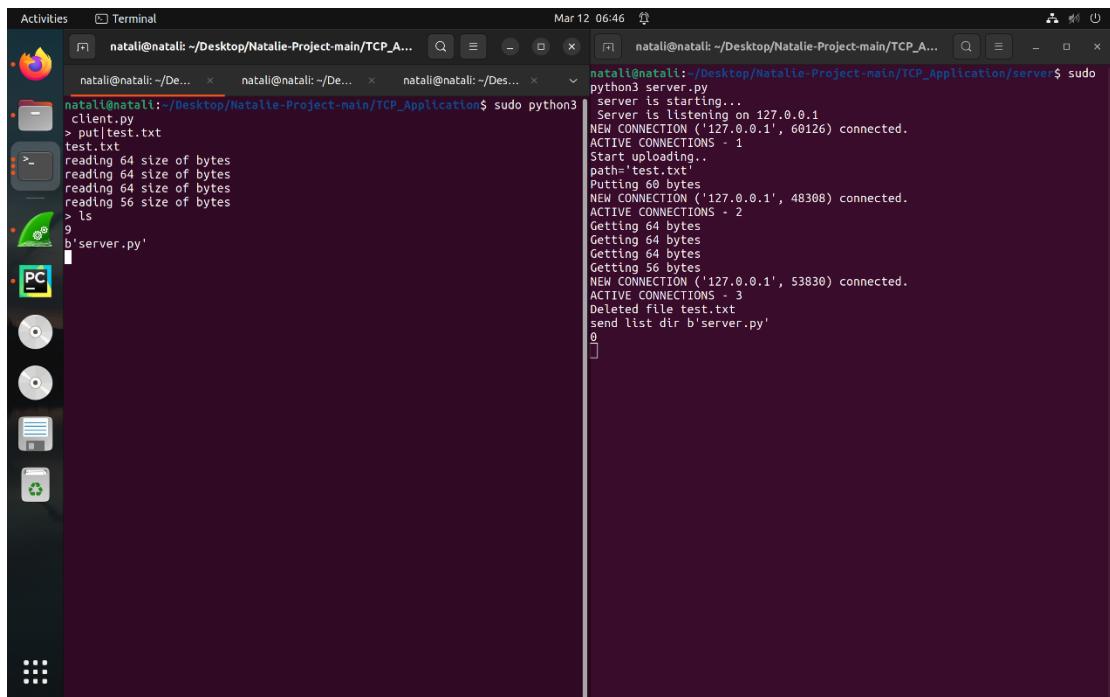
cut נססה להויריד קובץ מהשרת, השתמש בפקודה המתאימה להורדת קובץ מהשרת (get) ביחד עם שם הקובץ (כפי שופיע בתמונה בצד הליקוח) וNILICHZ enter. ניתן לראות שהשרת מדפיס למסך כל פאקטה שהוא שלוח עם כמות הבטים שיש באוטה פאקטה ואצל הלקוח שהוא מקבל כל פאקטה ומתחילה להויריד את הקובץ. לאחר שהקובץ הועבר ניתן לראות הודעה המאשר שהעברה הושלמה בהצלחה.

```
natali@natali:~/Desktop/Natalie-Project-main/TCP_Application$ sudo python3 client.py rm|test.txt
>

natali@natali:~/Desktop/Natalie-Project-main/TCP_Application$ sudo python3 server.py
server is starting...
Server is listening on 127.0.0.1
NEW CONNECTION ('127.0.0.1', 60126) connected.
ACTIVE CONNECTIONS - 1
Start uploading...
path='test.txt'
Putting 68 bytes
NEW CONNECTION ('127.0.0.1', 48308) connected.
ACTIVE CONNECTIONS - 2
Getting 64 bytes
Getting 64 bytes
Getting 64 bytes
Getting 56 bytes
Deleted file test.txt
```

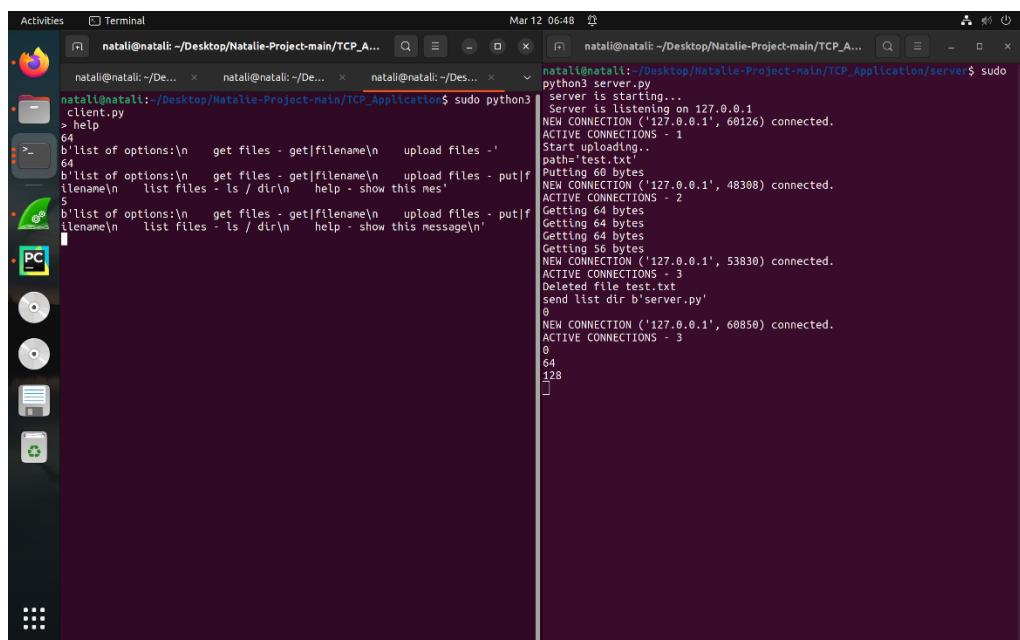
בתמונה זו אנו שוב רואים את השרת אבל הפעם עם עוד ליקוח חדש שמתחבר אליו, ניתן לראות שהשרת קיבל חיבור חדש על פי ההודעה שהוא מדפס למסך (ניתן לשים לב להבדלים בפורט אשר מודפס ליד ה-IP) וההודעה אשר מעדכן על כמה משתמשים מחוברים כרגע (3).

כעת ננסה למחוק קובץ מהשרת, נשתמש בפקודה המתאימה למחיקת קובץ מהשרת (rm) ביחד עם שם הקובץ (כפי שמופיע בתמונה בצד הליקוח) וNILCHZ enter. ניתן לראות שהשרת מדפיס הודעה שהקובץ נמחק בהצלחה.



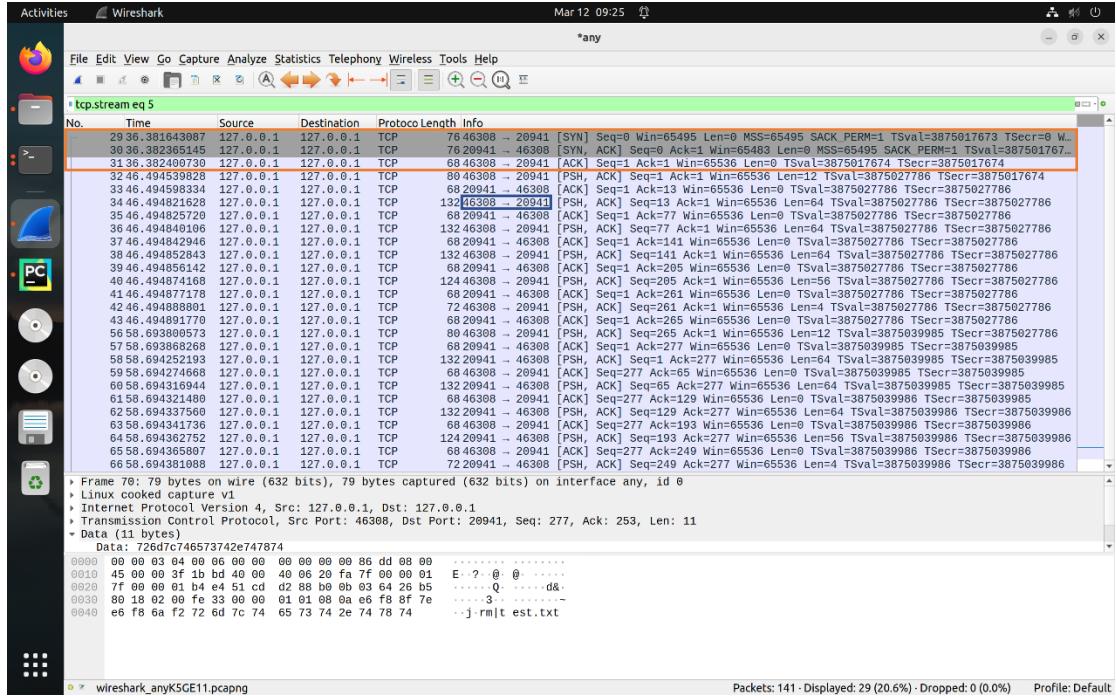
```
natali@natali:~/Desktop/Natalie-Project-main/TCP_A...$ sudo python3 server.py
server is starting...
Server is listening on 127.0.0.1
NEW CONNECTION ('127.0.0.1', 60126) connected.
ACTIVE CONNECTIONS - 1
Start uploading..
path='test.txt'
Putting 60 bytes
NEW CONNECTION ('127.0.0.1', 48308) connected.
ACTIVE CONNECTIONS - 2
Getting 64 bytes
Getting 64 bytes
Getting 64 bytes
Getting 56 bytes
NEW CONNECTION ('127.0.0.1', 53830) connected.
ACTIVE CONNECTIONS - 3
Deleted file test.txt
send list dir b'server.py'
0
```

בתמונה זו אנו שוב רואים את השרת, כאשר עדיין 3 לקוחות מחוברים אליו. נחזיר ליקוח הראשון ונבקש לקבל רשימה של כל הקבצים בשרת באמצעות הפקודה המתאימה כפי שמופיע בתמונה (זא). ניתן לראות בצד השרת שהוא שלוח את הרשימה ללקוח ואצל הלקוח מופיעעה הרשימה שcutת מכילה רק את קובץ הקוד של השרת מאחר ומחקנו את הקובץ שהעבכנו. (כנ"ל לפקודה dir (dir לפקודה rm)



```
natali@natali:~/Desktop/Natalie-Project-main/TCP_A...$ sudo python3 client.py
64
b'list of options:\n    get files - get|filename\n    upload files -\n64
b'list of options:\n    get files - get|filename\n    upload files - put|f
ilename\n    list files - ls / dir\n    help - show this mes'
5
b'list of options:\n    get files - get|filename\n    upload files - put|f
ilename\n    list files - ls / dir\n    help - show this message\n'n
0
NEW CONNECTION ('127.0.0.1', 48308) connected.
ACTIVE CONNECTIONS - 2
Start uploading..
path='test.txt'
Putting 60 bytes
NEW CONNECTION ('127.0.0.1', 53830) connected.
ACTIVE CONNECTIONS - 3
Getting 64 bytes
Getting 64 bytes
Getting 64 bytes
Getting 56 bytes
NEW CONNECTION ('127.0.0.1', 60850) connected.
ACTIVE CONNECTIONS - 3
Deleted file test.txt
send list dir b'server.py'
0
0
0
128
```

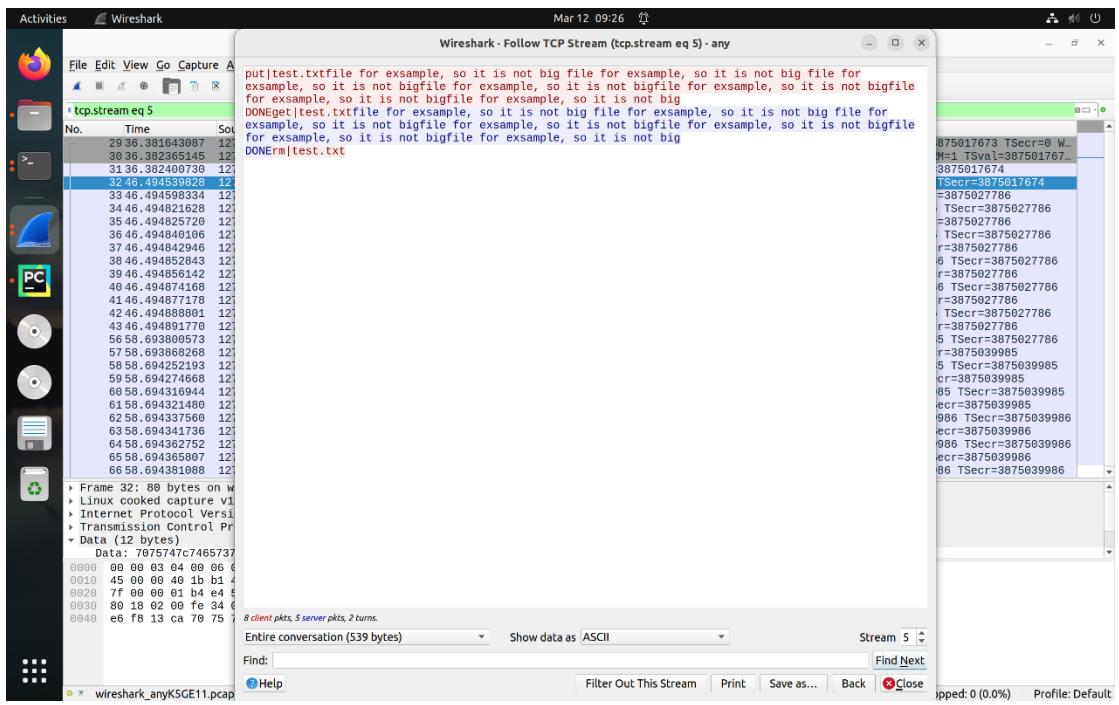
בתמונה זו אנו שוב רואים את השרת, כאשר עדין 3 ל��וחות מחוברים אליו. נלך שוב פעם ללקוק אחר שמחובר והפעם נבקש לראות את כל הפעולות האפשריות באמצעות הפקודה המתאימה כפ' שניתן לראות בצד הלוקו. ברגע שנשלחת הפקודה ניתן לראות בצד השרת את מספר הבטים של ההודעה אשר השרת שלוח וברגע שכל חלק מתתקבל הוא מוצג על המסך בתוספת החלק שהגע כבר לפניו.



בתמונה זו ניתן לראות את תעבורת הרשת בתקשורת בין האפליקציה ללקוח wireshark.

בריבוע הכתוב מעלה ניתן לראות את לחיצת היד המשולשת ואת האמין שנפתחה בין השרת ללקוח. הלקוח שלוח פקעת "SYN" לשרת, מקבל חזרה פקעת פתיחה קשור ואישור לפתחת הקשר מהשרת "SYN , ACK" ועוד הלקוח מוחזר לשרת פקעת אישור גם "ACK" וכעת הם יכולים לתקשר ביניהם.

שאר הפקעות בתמונה הם התקשרות של הלקוח עם שרת האפליקציה. ניתן לראות שכל ללקוח נשלח מפורט שונה אך כולל מגיעים לפורט האפליקציה שהוגדר להיות 20941. (ראו דוג' בריבוע החול מעלה)

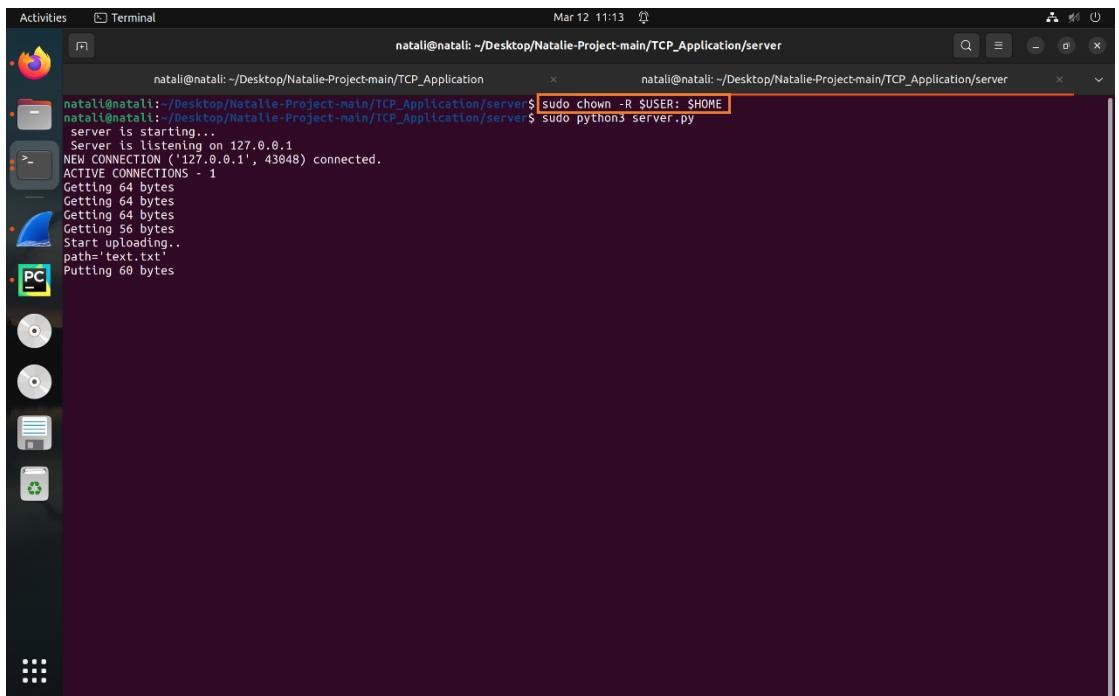


בתמונה זו לחצנו על פאקטה תקשורת כלשהי בין הלקוח לשרת האפליקציה ע"י לחיצה ימנית על העכבר <- follow TCP stream וראינו את שלושת הפקודות שנעשו בלקוח ואת הקובץ שעבר ביניהם או נמחק. בכל מקום בו כתוב DONE אנו יודעים שהפעולות התרחשו במלואן.

## איבוד פאקטות – TCP

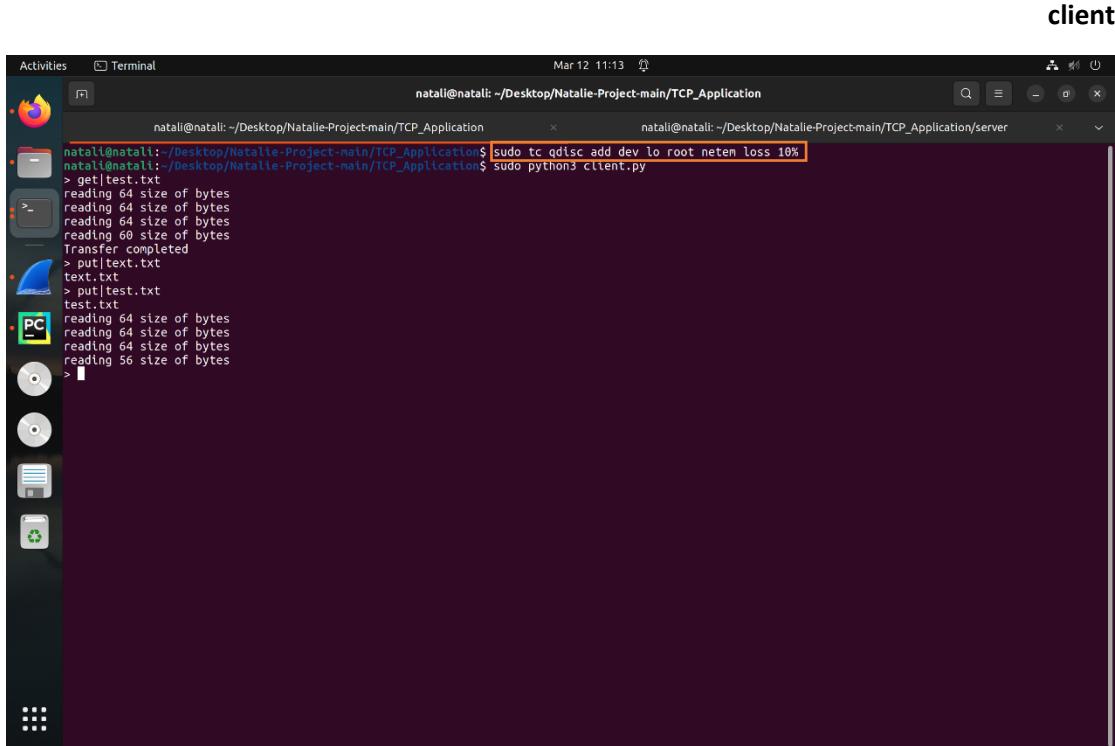
**איבוד פאקטות של 10%**

**server**



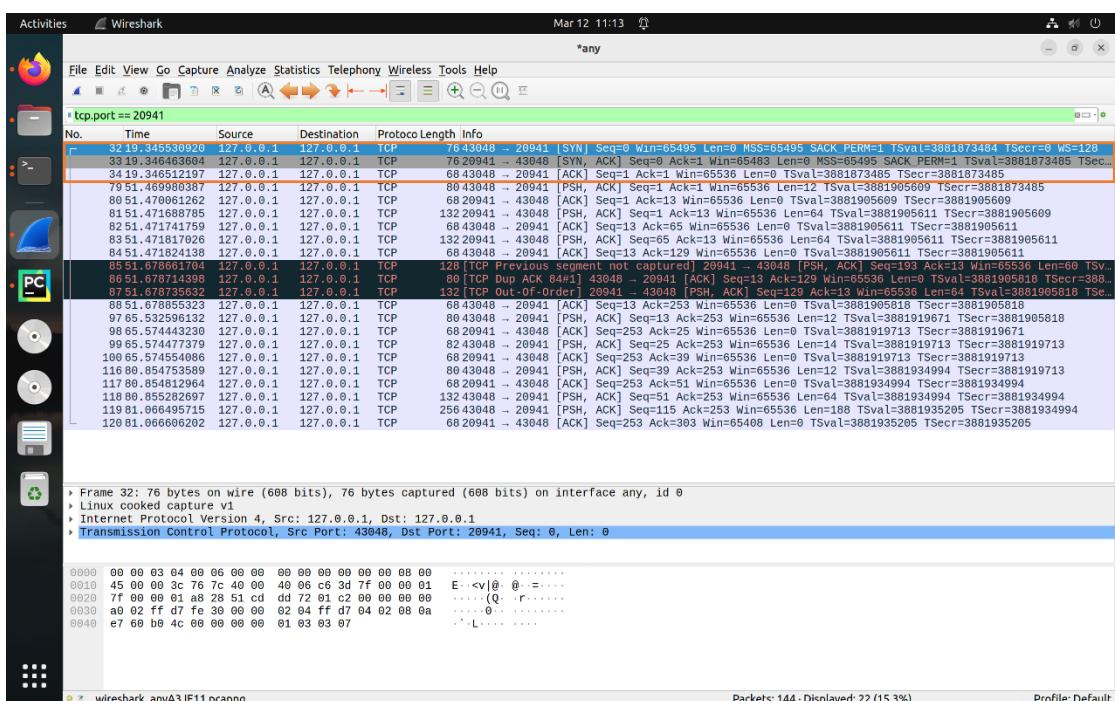
בריבוע הכתום למעלה ביצענו פקודות עם הרשאת sudo להורדת הנעה מהל הקבצים שנמצאים בשרת על מנת שנוכל להתנהל איתם.

לאחר מכן הרצינו את השרת כפי שניתן לראות בתמונה.



ניתן לראות בתמונה למעלה של ה "client" שהשתמשנו בклиן `tc` לאיבוד פאקטות רנדומלי בונח של `.sudo tc qdisc add dev lo root netem loss 10%` עם הפקודה

cut נယור לניטוח ה-



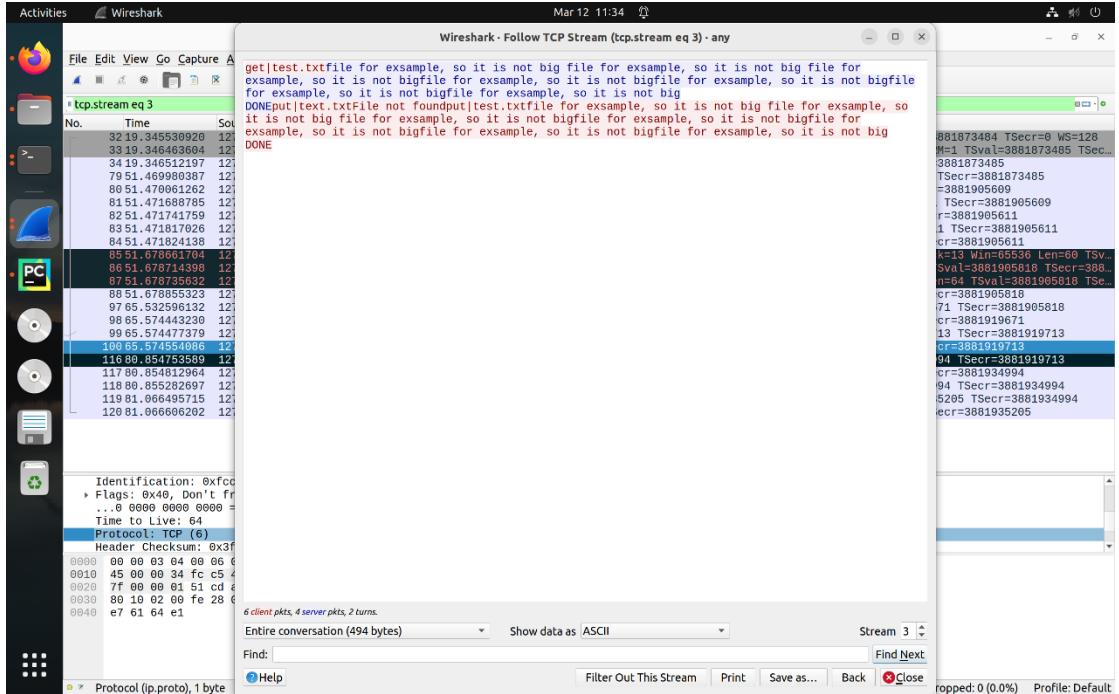
ניתן לראות בריבוע הכתום שבתמונה את פתיחת הקשר של הלקוח עם השרת ("ע" לחיצת יד משולשת).

בפקאות השחורות בתמונה ניתן לראות את האיבוד של הפאקטות בזמן העברתו.

.TCP Previous segment not captured – "התפסה" פאקטות מהשרת אפליקציה ללקוח.

TCP Dup ACK – על כל פאקטה שMOVURת ומתקבלת השרת מקבל ACK כלומר פאקטה אישור, במקורה זה השרת מקבל ACK כפול מהלקוח כדי לסמן לו שהפאקטות הללו לא הועברו ונאבדו.

TCP Out Of Order – הודיעו זאת אומרת שהחbillה המסוימת התקבלה בסדר שונה ממנה היא נשלחה (לאחר חbillה מאוחרת יותר בראוף).



בתמונה זו ניתן לראות שהפעולות בוצעו בהצלחה והקובץ הנבחר הועבר. לחצנו על פאקטה תקשורת follow TCP stream <- follow העבר soflow וראינו זאת.

## Client - RUDP

תכנית זו מימושה צד ללקוח המתחבר לשרת ה-FTP באמצעות Socket. בחלק זה השרת מכון אצלו קבצים והלקוח יכול לבחור أيזה קובץ הוא רוצה להוריד.

התכנית מייבאת את הספריות `socket` ו-`hashlib`, וקובעת משתנים קבועים מסוימים כגון כתובות ה-IP, `BUFFER SIZE` ו-`PORT`.

הfonקציה (`calc_checksum()`) מקבלת פרמטר `data`, שהוא הבטים לחישוב `checksum` באמצעות `hashlib`. ה Fonקציה מחזירה את התקציר של `hash` בתור בתים. אלגוריתם ה- MD5 hash.

הfonקציה (`udp_client()`) מתחילה ביצירת socket UDP באמצעות `(socket.socket(),` ולאחר מכן מבקשת מהמשתמש להזין את שם הקובץ שהוא רוצה להוריד.

לאחר מכן ה Fonkציה שולחת את שם הקובץ לשרת באמצעות `(().sock.sendto()`

לאחר מכן, ה Fonkציה פותחת קובץ במצב כתיבה בינהריע עם שם זהה לקובץ המבוקש, ונכנסת לולאה שבה היא קוראת נתונים מה-socket באמצעות `(().sock.recvfrom()`.

הנתונים שהתקבלו מוחלקיים לשני חלקים: 16 הבטים הראשונים הם סכום הבדיקה, ואילו הנינים הם תוכן הקובץ.

לאחר מכן, התכנית בודקת אם הנתונים ריקים. במידה וכן, זה אומר שכלי הנתונים התקבלו והלוואה מסתימה. אחרת, התכנית בודקת אם סכום הבדיקה שהתקבל תואם לסכום הבדיקה המקורי באמצעות ה Fonkציה (`calc_checksum()`). אם סכומי הבדיקה תואמים, תוכן הקובץ נכתב אצל הלוקוט. אם הם לא תואמים, התכנית מדפסה הודעה המציינת שיש בעיה בסכום בדיקה.

אם מתבצעת פעולה חריגה בעת ניסיון להוריד את הקובץ, התכנית מדפסה הודעה המציינת את ההשגיאה.

## Server - RUDP

תוכנית זו מימושה שרת P (File Transfer Protocol) בפייתון על גבי תקשורת UDP שמאזין לנתונים נוכנים מלוקחות, ולאחר מכן שולח בחזרה את התוכן של קובץ מבוקש.

התכנית יוצרת UDP socket ומקשרת אותו לכתובת IP ו-PORT שצוינו. לאחר מכן היא מזינה לנתונים שנוכנים מלוקחות.

כאשר הלוקוט שולח בקשה לקובץ, השרת קורא את הקובץ ושולח אותו ללקוח בחתיכות קטעות של נתונים, יחד עם בדיקת MD5 שלו כל חלק. הלוקוט יכול להשתמש בסיכון בדיקה אלה כדי לאמת את תקיןות הנתונים שהוא מקבל, אם הקובץ המבוקש לא נמצא, השרת שולח הודעה שגיאה ללקוח.

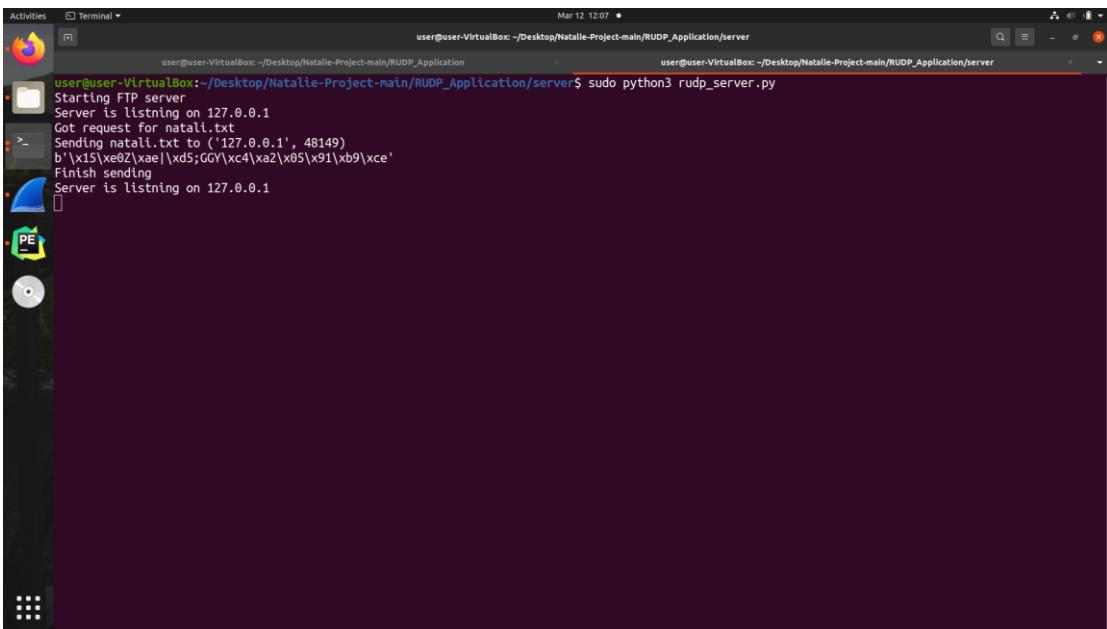
התכנית מגדרה משתנים קבועים כגון כתובות ה-IP, PORT, משתנה מסוג טאפל המכיל כתובות ה-IP ו- PORT וגם BUFFER SIZE הקבוע את כמות הנתונים המקסימלית שניתן לשולח בחבילת UDP אחת. ה Fonkציה (`calc_checksum()`) מחשבת את סכום הבדיקה MD5 של מקבץ נתונים נתון.

הfonkציה (`ftp_server()`) היא ה Fonkציה העיקרית שמקשiba לנתונים נוכנים ומגיבה לבקשת הלוקוט. הוא משתמש בלולאת while כדי להאזין ללא הרף לנתונים נוכנים.

כאשר לקוח שולח בקשה לקובץ, השרת קורא את הקובץ ושולח אותו ללקוח בחתיכות של נתונים באמצעות קריאת פונקציה `(sendto())`.

סכום הבדיקה של כל חלק מחושב באמצעות ה Fonkציה `calc_checksum()` ונסלח יחד עם הנתונים. במידה והקובץ המבוקש לא נמצא, נשלחת הודעה שגיאה ללקוח.

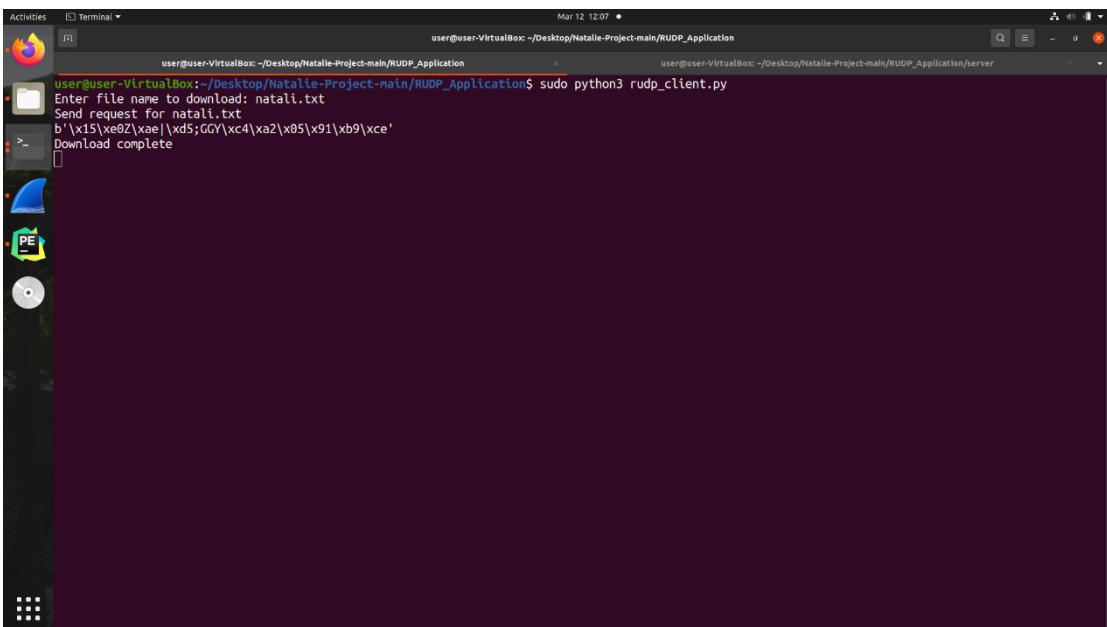
### הרצה ה-server:



```
user@user-VirtualBox:~/Desktop/Natalie-Project-main/RUDP_Application$ sudo python3 rudp_server.py
Starting FTP server
Server is listing on 127.0.0.1
Got request for natali.txt
Sending natali.txt to ('127.0.0.1', 48149)
b'\x15\x02\xae\xd5;GG\x4\x2\x05\x91\xb9\xce'
Finish sending
Server is listing on 127.0.0.1
```

מראים את השרת מהטרמינל ביצירוף הפקודה sudo על מנת לקבל הרשות root לתפיסת ושליחת תעבורת והשרת מדפס הודיעו שהוא רץ ומאזין בכתובת 127.0.0.1, מקבל בקשה לקובץ שקיים אצל ושולח אותו ללקוח (מצין את ה-IP והפורט של הלוקוח), לאחר מכן הוא מדפיס את התוצאה של checksum (כਮון שבמימוש אמיתי לא נdfs אוטו) ושולח את הקובץ מידת וה-checksum תואם. ברגע שם שמשים לשולח את הקובץ השרת מדפס הודיעו שסימן לשולח וחזר לחכות לבקשת מלוקוחות.

### הרצה ה-client:

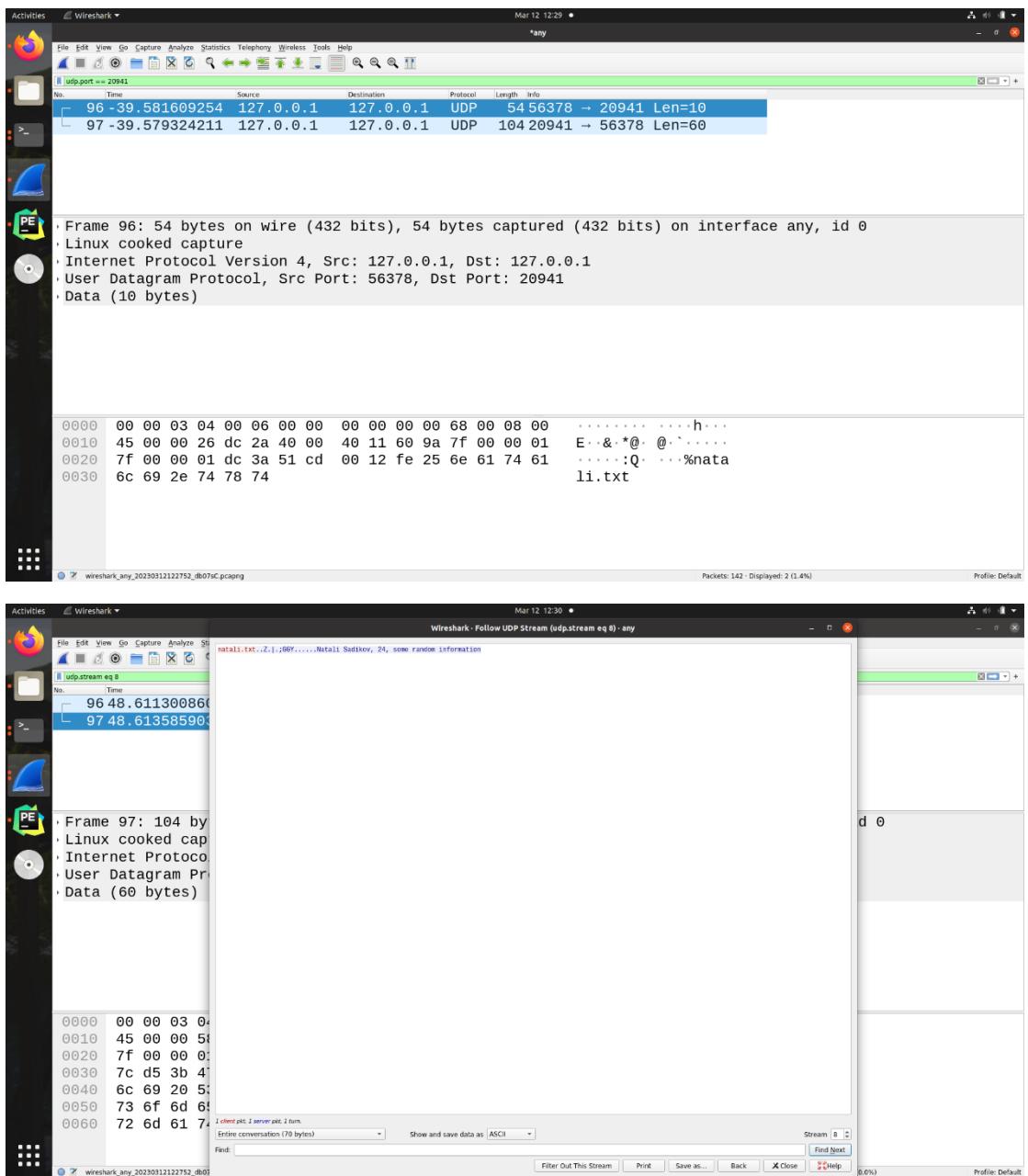


```
user@user-VirtualBox:~/Desktop/Natalie-Project-main/RUDP_Application$ sudo python3 rudp_client.py
Enter file name to download: natali.txt
Send request for natali.txt
b'\x15\x02\xae\xd5;GG\x4\x2\x05\x91\xb9\xce'
Download complete
```

מראים את הלוקוח מהטרמינל ביצירוף הפקודה sudo על מנת לקבל הרשות root לתפיסת ושליחת תעבורת והлокוח מבקש מהמשתמש להכנס את שם הקובץ שהוא רוצה לקבל מהשרת. הלוקוח שולח

את ההודעה לשרת ומדפיס על כך הודעה למסך ובנוסף מדפיס את ה-**checksum** שחייב בדיק כמו בשורת. ברגע שהקובץ סיים לרדת הוא מופיע שההורדה סיימה.

:הקלטת Wireshark



ניתן לראות שנשלחה פאקטה לשרת ה-FTP בפורט 20941 כתובות 127.0.0.1 והתקבלה תשובה שהיא בעצם הקובץ עצמו. ניתן לראות את הבקשה והתשובה אם נעשה קлик ימני-<- follow .stream.

### **טיפול בעוותים latency:**

TCP משתמש בכמה טכניקות כדי להבטיח שהנתונים מועברים בצורה יעילה - בקרת גודש (congestion control), בקרת זרימה (flow control) ושידור חוזר (retransmission). בקרת גודש מסייעת בהפחחתת עומס ברשת על ידי האטת קצב העברת הנתונים כאשר יש עומס. בקרת זרימה עוזרת למנוע עומס על ה-buffer על ידי ויסות קצב העברת הנתונים בתבסס על יכולת המקלט לטפל בפaketות. שידור חוזר משמש כדי להבטיח שפaketות אבודות יעברו מחדש עד שהן מתקבלות בהצלחה.

על מנת ש-RUDP יוכל לטפל בעוותים זו עליו למשם מספר דברים אשר יעזורו לו להימנע מבעיה זו. לדוגמה – לחת מספר סידורי לכל פאקטה וכך הוא יכול להיפטר מפaketות כפולות. לדאוג לקבל אישורי ACK חזרה על כל פאקטה וכן לדעת אם יש פאקטה שלא הגיעה. במידה והוא שם לב שלא הגיעה פאקטה בעקבות כך שלא הגיע ack \ מספר סידורי לא לפי הסדר הוא צריך לשולח שוב את הפאקטה. להשתמש בגודל חלון לשילוח של פaketות וכל פעם להגדיל את החולון כך שלא יונצ'r עומס על התקשרות \ buffer (flow control).

### **טיפול בעוותים איבוד פaketות:**

עבור איבוד חבילה, ל-TCP יש מגנון לאיתור ושילוח פאקטה אבודה מחדש. לכל פאקטה יש מספר סידורי ומספר אישור קבלה, המשמשים כדי להבטיח שהפaketות מתקבלות בסדר הנכון ולזיהוי פaketות אבודות. אם פאקטה לא אושרה על ידי המקלט תוך פרק זמן מסוים, השולח ישדר מחדש את החבילה.

על מנת שה-RUDP יוכל לטפל בעוותים זו עליו למשם מספר דברים אשר יעזורו לו להימנע מבעיה זו. לדוגמה – לדאוג לקבל אישורי ACK חזרה על כל פאקטה וכך לדעת אם יש פאקטה שלא הגיעה ובמידה והוא שם לב שלא הגיעה פאקטה בעקבות כך שלא הגיע ack הוא צריך לשולח שוב את הפאקטה. לשים לב שהוא מתחילה לשולח כמה פaketות ולא את מגדיל את כמה פaketות לפי התנועה ברשת על מנת שלא יונצ'r עומס (congection control).

### **דיאגרמת מצבים:**

