**מטלת סיום רשתות תקשורת**

שרון אלבו 314690298

קישור להקלטות wireshark, קבצי csv וגרפים:
https://drive.google.com/drive/folders/1NJY5slNTTK31hDIk8PBgssdRkfvzqUY7?usp=sharing

קישור ל github :
https://github.com/sharonelbo/Communication_Networks_final/tree/main

**חלק 1**

1. הסיבות האפשריות בשכבת בתעבורה שיכולות לגרום להעברת קבצים איטית הן:
שימוש בפרוטוקול TCP – פרוטוקול TCP משתמש בכמה כלים שיכולים להאט/להגביל את קצב העברת הנתונים, הדבר יכול לנבוע כתוצאה מאחד או שילוב של כמה מהמקרים הבאים:
   א. Maximum segment size - גודל החבילות המקסימלי שהשרת יכול לקבל קטן מידי ולכן כדי להעביר את הקבצים יש לשלוח אותם ב Segment-ים קטנים ובגלל שהפרטוקול משמש להעברת קבצים מהימנה יש לוודא עבור כל Segment שהוא הגיע בשלמותו וכאשר מחברים את ההודעה השלמה הוא נמצא במקומו.
   ב. Window size – כאשר גודל החלון קטן מידי ניתן לשלוח מספר מוגבל וקטן של הודעות בו זמנית, ולאחר שנשלחו צריך לחכות לאישור שהודעה התקבלה כדי לשלוח אחת חדשה.
   ג. Packet loss – בפרוטוק TCP שמשמש להעברת קבצים מהימנה, נבדק שכל כל חבילה חייבת להגיע ליעד במידה והחבילה לא הגיע בזמן מסויים היא תישלח שוב, לכן במקרה של תקשורת לא יציבה יכולות להיאבד הרבה חבילות בדרך דבר שיצריך שליחה מחדש של החבילות שנאבדו ובסופו של דבר יגרום להאטה של קצב העברת הקבצים.
   ד. Slow server response – בפרוטוקול TCP כשאר נשלחת חבילה יש לחכות לאישור מהשרת שהחבילה הגיעה (ACK) במידה והשרת מסיבה כזו או אחרת מגיב באיטיות לשליחת ההודעות הדבר ידחה שליחה של הודעות חדשות עד שיתקבל אישור על כך שההודעות שנשלחו התקבלו בהצלחה

כדי לפתור את הבעיות הנ"ל בפרוטוקל TCP כדאי לנסות להקליט את התקשורת בין השרת ללקוח ב wireshark ולנסות לחפש מהם הגורמים לבעיה, ניתן לבדוק כמה הודעות נשלחות לשרת ברצף לאחר יצירת הקשר (לחיצת יד משולשת) וכך לראות האם ה Window size גדול מספיק, במידה ורואים שהשרת שלח כמה הודעות ACK על אותה חבילה או שהלקוח שלח את אותה חבילה כמה פעמים אפשר להסיק שיש Packet loss בין השרת ללקוח.

2. בפרוטוקול TCP כאשר לשולח יש כוח עיבוד חזק יותר משל המקבל, הדבר יכול לגרום להעברת קבצים איטית כתוצאה משימוש של הפרוטוקול ב flow control mechanism כדי דבר שיכול ליצור צוואר בקבוק מנקודת המבט של השולח, משום שלמקבל יש כמות מוגבל של מידע שהוא יכול יקבל מהשולח לאחר שכמות המידע הזו התקבלה השולח צריך לחכות לאישור כדי להמשיך לשלוח מידע, לכן במקרה הנ"ל המקבל יכול לעבד כמות קטנה של מידע מה שגורם לשולח לשלוח מידע בקצב איטי למרות כוח העיבוד החזק שלו.
3. הניתוב קובע את המסלול ברשת שבו חבילות נשלחות, מהירות ויציבות השליחה יכולה להשתנות על פי המסלול שנבחר, לכן הגורמים שצריכים לקחת בחשבון בניתוב הם:
   א. Latency – כאשר המסלול שהחבילות צריכות לעבור קצת יותר כך גם המהירות שבה החבילות יגיעו ליעד תגדל

ב. Congestion – כאשר בוחרים מסלול אוטימלי כדאי להימנע ממסלולים בעלי גודש תנועה כדי לשפר את המהירות וכדי למנוע איבוד חבילות בדרך
ג. multiple paths – כאשר שולחים כמה חבילות במסלולים שונים אפשר להימנע מצוואר בקבוק שיכול להיווצר כאשר שולחים את כל החבילות באותו מסלול

4. MTCP - מאפשר העברת נתונים על גבי כמה ערוצי תקשורת במקביל וכל זה תחת Socket יחיד בנוסף כל ערוץ יכול לשלוח ולקבל חבילות על מסלול שונה משל הערוצים האחרים ואף להשתמש בסוגי חיבור אינטרנט שונים כמו wifi ethernet, וכו' וככה לנצל יותר מרוחב הפס של השולח
5. נחלק ל 2 מקרים:
1. גורמים אפשריים בשכבת התעבורה:
א. UDP protocol – כמשתמשים בפרוקטול UDP כאשר חבילה נאבדת הפרוקטול לא שולח אותה שוב דבר שיכול להיות מאוד מורגש
ב. Window size – כאשר משתמשים בפרוטוקול TCP כאשר החלון גודל החלון גדול מידי נשלחות הרבה חבילות בבת אחת יכול להיות שנשלחות הרבה יותר חבילות ממה שהשרת יכול לקבל, דבר שגורם לפקטות להיאבד
2. גורמים אפשריים בשכבת הרשת:
א. Congestion – כאשר ישנו מסלול שעמוס בחבילות הוא עלול לזרוק חבילות כדי לפנות מקום ולמנוע צוואר בקבוק
ב. Routing – כאשר המסלולים הנבחרים אינם אופטימליים

כדי לנסות לפתור את בעיית איבוד החבילות אפשר לנסות:

א. להשתמש ב wireshark ולבדוק מתי בדרך קורה איבוד החבילות, אולי הוא כתוצאה מכך שנשלחות יותר מידי חבילות לשרת והוא לא מצליח לעבד את כל המידע ולכן זורק חבילות, אם כן נרצה להקטין את ה window size
ב. נרצה לוודא שהחבילות נשלחות במסלולים אפטימליים ויציבים כדי למנוע איבוד חבילות
ג. במידה ומשתמשים בפרוטוקל UDP נרצה לשקול להחליף לפרוטוקול TCP, הדבר אמנם יאט את מהירות השליחה והקבלה של החבילות כי חבילות שנאבדות יצטרכו להישלח שוב, אבל בסופו של דבר כל החבילות יגיעו

## חלק 2

1. **FlowPic Encrypted Internet Traffic Classification is as Easy as Image Recognition:**

- התרומה העיקרית של המאמר היא שהוא מציג שיטה בשם FlowPick שמייצרת מטריצה דו מימדית שבה כל תא מייצג נקודת זמן ושומר בתוכו את מספר החבילות שהגיעו באותו נקודות זמן וגם את גודל החבילות שהגיעו באותה נקודת זמן, המטריצה הדו מימדית מומרת לתמונה מה שמשאפשר לסווג את סוג התעבורה בעזרת שימוש ב convolutional neural networks (CNNs) שהוא מודל למידה עמוקה שמתמחה בזיהוי תמונות ודפוסים, כך שבמקום לחקור את תוכן החבילות CNN יודע ללמוד דפוסים שונים בתמונה ולסווג אותם לסוגי אפליקציות שונות וכך הוא גם יודע להתמודד עם תעבורה מוצפנת
- הכלים ששומשו הם:
א. Packet Size Distribution – מודד את תדירות ההופעה של פקטות מגדלים שונים
ב. Inter-Arrival Times – מודד את הפרשי הזמן בין חבילות נתונים
ג. convolutional neural networks – מודל למידה עמודה שמתמחה בזיהוי תמונות ודפוסים

הכלים החדשניים הם:

א. FlowPic – השיטה שממירה תעבורת שרת מוצפנת לתמונה

ב. convolutional neural networks – שימוש במודל למידה עמיקה שמיועד לתמונות כדי לנתח ולסווג סוגי תעבורת רשת מוצפנת

- התוצאות העיקריות של המאמר הן:

| Problem | FlowPic Acc. (%) | Best Previous Result | Remark |
|---|---|---|---|
| *Non-VPN Traffic Categorization* | 85.0 | 84.0 % Pr., Gil *et al.* [15] | Different categories. [15] used unbalanced dataset |
| *VPN Traffic Categorization* | 98.4 | 98.6 % Acc., Wang *et al.* [7] | [7] Classify raw packets data. Not including browsing category |
| *Tor Traffic Categorization* | 67.8 | 84.3 % Pr., Gil *et al.* [15] | Different categories. [15] used unbalanced dataset |
| *Non-VPN Class vs. All* | 97.0 (Average) | No previous results | |
| *VPN Class vs. All* | 99.7 (Average) | No previous results | |
| *Tor Class vs. All* | 85.7 (Average) | No previous results | |
| *Encryption Techniques* | 88.4 | 99. % Acc., Wang *et al.* [7] | [7] Classify raw packets data, not including Tor category |
| *Applications Identification* | 99.7 | 93.9 % Acc., Yamansavascilar *et al.* [10] | Different classes |

א. דיוק של 85% בסיווג תעבורה ללא שימוש ב VPN

ב. דיוק של 98% בסיווג תעבורה בזמן שימוש ב VPN

ג. דיוק של 67.8% בסיווג תעבורה בזמן שימוש בדפדפן Tor

ד. דיוק של 99.7% בזיהוי אפליקציות

לסיכום FlowPick מצליח בעזרת שימוש חדשני ומקורי בהמרת המידע לתמונות וניתוחן באמצעות כלי למידה עמוקה שמיועד לתמונות לפענח בהצלחה ועם ביצועים טובים סוגי אפליקציות ולווסג תעבורת רשת מוצפנת.

## 2. Early Traffic Classification With Encrypted ClientHello A Multi-Country Study:

- המאמר מציג את שיטת Hybrid Random Forest Traffic Classifier (hRFTC) שמטרתה לסווג תעבורה ברשת לפני השלב שבו התעבורה מוצפנת, hRFTC משלבת בין שימוש במידע שאינו מוצפן שנשאר כאשר הלקוח שולח בקשה לפתיחת חיבור מאובטח עם השרת בעזרת פרוטוקול TLS של שכבת התעבורה, בעזרת שאריות מידע שנשארות לפני שנוצר חיבור מאובטח hRFTC מנצלת את חלון ההזדמנויות הקטן הזה כדי לנסות ולצפות מה יהיה סוג התעבורה, בנוסף השיטה משלבת בין ניתוח תכונות של פקטות בודדות לבין ניתוח של התנהגות הפקטה לאורך זמן
- הכלים החדשניים ששומשו הם:

א. TLS metadata – שליפת מידע כמו גרסת ה TLS וסוגי ההצפנה מחלקים לא מוצפנים של לחיצת היד בפרוטוקול TLS

ב. מאגר המידע הגדול ביותר שמכיל יותר מ 600000 דגימות של מאפייני זרימה (גודל חבילות, זמן הגעה וכו') של TLS שנדגמו מכמה מדינות שונות

ג. שילוב של שליפת המידע מה metadata בנוסף לניתוח מאפייני הזרימה

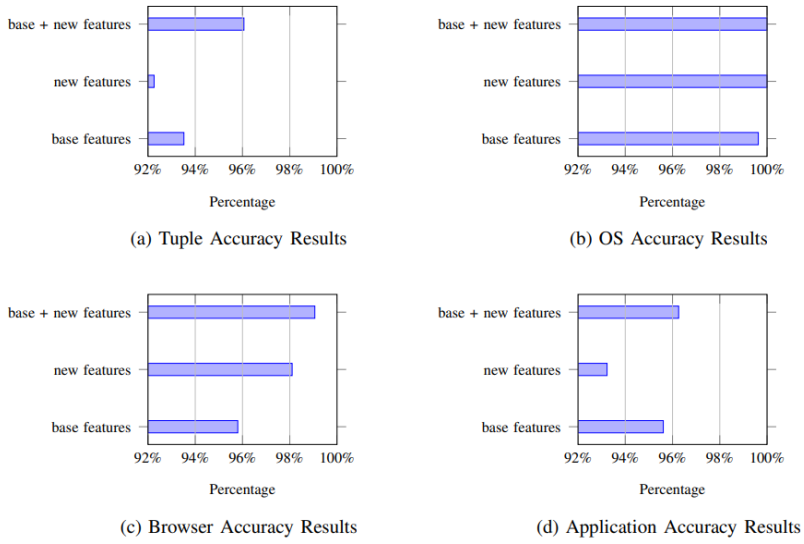
- התוצאות העיקריות הן:

א. האלגוריתם hRFTC השיג 94.6% score-F על מאגר המידע שאסף, ובכך עקף אלגוריתמים נחשבים

- אימון האלגוריתם בעזרת מידע שנאסף מיותר ממדינה אחת מאפשר השגת ביצועים גבוהים יותר מאשר איסוף מידע מאזור גיאוגרפי יחיד
- חשיבות השילוב בין שימוש במידע מבוסס זרימה, לבין השימוש החדשני של שליפת מידע מחלקי ה metadata שאינם מוצפנים

### 3. Analyzing HTTPS Encrypted Traffic to Identify User's Operating System, Browser and Application:

- המאמר מציג שיטה חדשנית לזיהוי מערכת ההפעלה, דפדפן ואפליקציות על בסיס תעבורת HTTP מוצפנת, בעזרת שימוש בזיהוי של דפוסים שונים בתעבורה כדי למצוא מאפיינים ייחודיים לכל מערכת הפעלה ודפדפן, כל זאת בעזרת מאפיינים חצוניים של חבילות שניתנים להשגה גם עם המידע מוצפן
- הכלים ששומשו הם:
  א. Packet sizes – גדלים של חבילות, גודל מקסימלי, מינימלי וגודל ממוצע
  ב. Inter-arrival time - הפרשי הזמן בין חבילות נתונים
  ג. Number of packets- מספר החבילות הכולל
- הכלים החדשניים הם:
  פרוטוקול SSL משתמש במערכת קריפטוגרפית שיוצרת קישור מוצפן בין שרת ללקוח, ומונעת מצד שלישי לקרוא או לשנות כל מידע רגיש המועבר ביניהם
  א. SSL Extension Count – מספר תעודות ה SSL שהדפדפן שולח
  ב. SSL Cipher Methods Count – בעל ערך שונה עבור כל דפדפן
  ג. SSL Cipher Methods Count – מספר שיטות הצפנה שהדפדפן מבצע
  ד. TCP Window Size – גודל החלון הקובע כמה מידע ניתן לשלוח לשרת לפני קבלת ACK
  ה. Bursty behavior - זיהוי כיצד דפדפן שולח נתונים בקפיצות ולא בצורה אחידה
- התוצאות העיקריות הן:
  א. הניתוח בעזרת השימוש בכלים הישנים ובכלים החדשניים העלה את רמת הדיוק כפי שניתן לראות בגרפים:




(a) Tuple Accuracy Results (b) OS Accuracy Results

(c) Browser Accuracy Results (d) Application Accuracy Results

  ב. זיהוי של מערכת ההפעלה, הדפדפן ואפליקציות שונות בדיוק גבוה אפילו בהינתן מידע מוצפן

## חלק 3

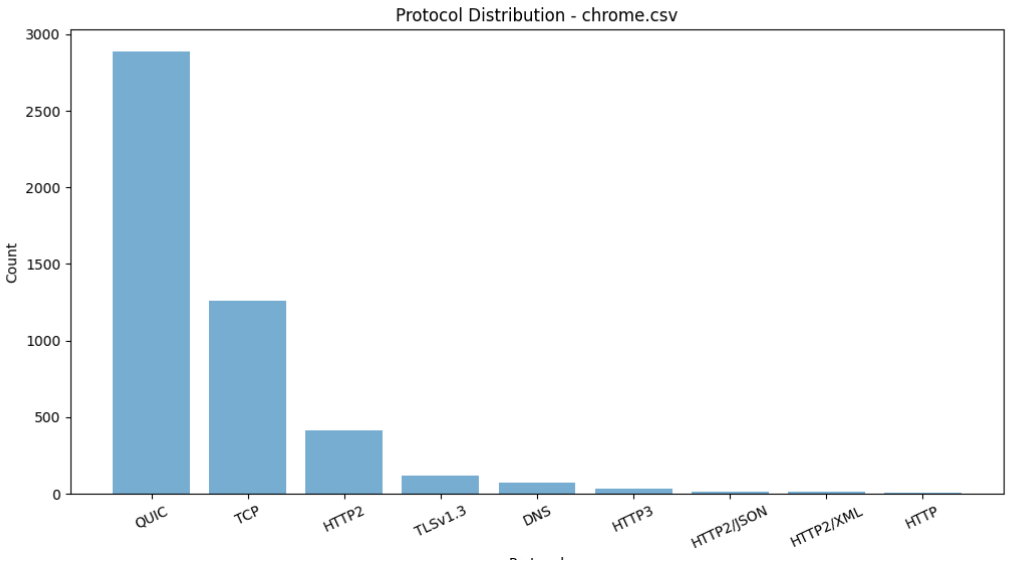
בחלק זה התבקשנו לייצר הקלטות wireshark ולאחר מכן לרשום קוד בפייתון שייצר גרפים מהקלטות ה wireshark כדי להשוות בין האפליקציות השונות

ההקלטות שיוצרו הן:

ג. גלישה בדפדפן כרום - chrome.pcapng
ד. גלישה בדפדפן פיירפוקס – firefox.pcapng
ה. האזנה למוזיקה ללא סרטון בספוטיפי – spotify.pcapng
ו. צפייה בסרטון ביוטיוב – youtube.pcapng
ז. שיחת וודיאו בדיסקורד – discord.pcapng

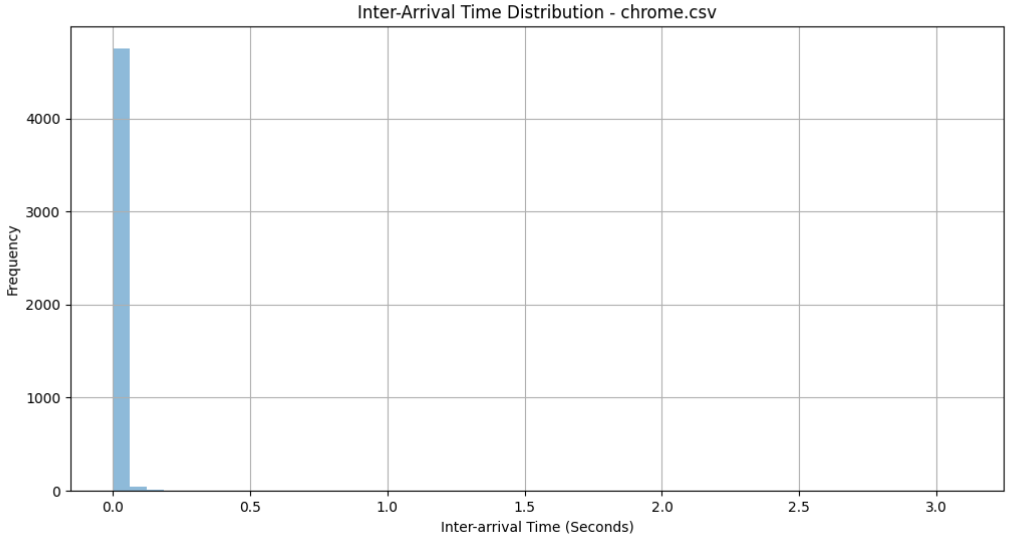
1. ניתן למצוא את קבצי הקלטות ה wireshark, קבצי ה csv ואת הגרפים בלינק הבא:
https://drive.google.com/drive/folders/1NJY5slNTTK31hDIk8PBgssdRkfvzqUY7?usp=sharing
2. הגרפים שהתוכנית מייצרת הם:
(ישנם גרפים נוספים לכל סוג גרף שמצורף, בנוסף לכל סוג גרף יש גם גרף שמשווה בין כל ההקלטות השונות, התמונת המצורפות הן רק לשם המחשה של הגרף)

- Protocol Distribution – מראה את התפלגות כל הפרוטוקולים השונים ששומשו בכל סוג אפליקציה
בציר ה X סוג הפרוטוקל, בציר ה Y מספר הפעמים שהפרוטוקול שומש באפליקציה

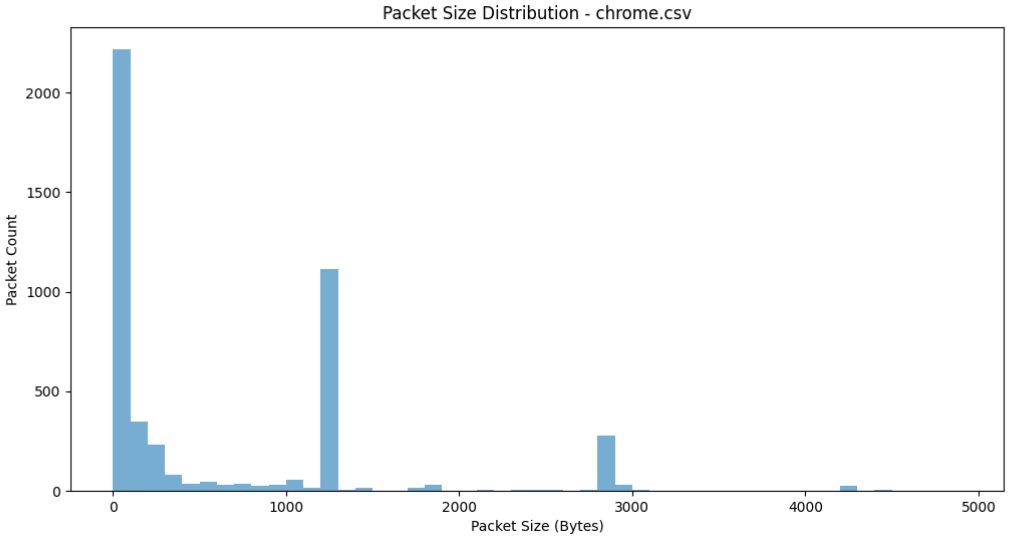



- Inter-Arrival Time Distribution – מראה את הזמן שעבר בין כל שני פקטות שהתקבלו
בציר ה X הזמן בשניות שעבר בין כל 2 פקטות, בציר ה Y מספר הפקטות שנשלחו

לאחר הזמן שמצויין בציר ה X

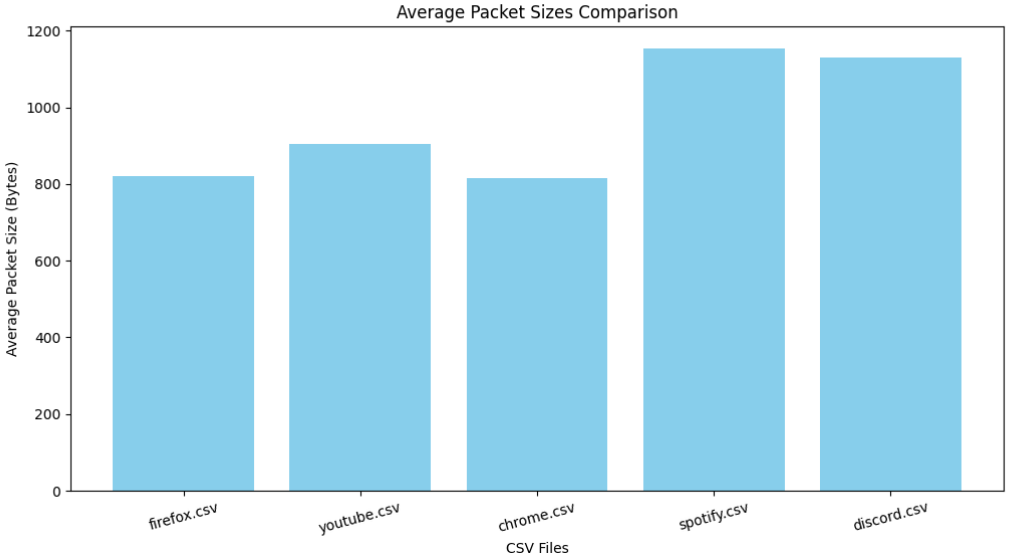



- Packet Size Distribution – מראה את ההתפלגות של גודל החבילות בכל סוג אפליקציה
בציר ה  X גודל החבילה ב Bytes, בציר ה Y מספר החבילות מאותו גודל




- Average packet size comparison – משווה בין הגודל הממוצע של חבילה בכל האפליקציותבציר ה X סוג האפליקציה, בציר ה Y הגודל הממוצע של הודעה

באפליקציה




3. המסקנות שניתן להסיק מכל אחד מהגרפים הן:

- הפרוטוקול הכי נפוץ בהקלטות שנבדקו הוא QUICK ששומש בעיקר באפליקציות firefox, chrome, spotify, הפרוטוקול משמש להעברת קבצים מהירה ויעילה לכן הגיוני שדפדפנים ישתמשו בו
- הפרוקול השני הכי נפוץ הוא פרוטוקול UDP ששומש בעיקר על ידי youtube, discord, גם פה התוצאה הגיונית משום ששתי האפלקיציות צריכות להעביר ווידיאו וגם אודיו בזמן אמת, לכן השימוש בפרוטוקל UDP כדי להעביר את החבילות כמה שיותר מהר עם אפשרות לאבד כמה חבילות מבלי שהמשתמש ישים לב לכך הגיונית לשימוש בשתי האפלקיציות הללו
- לאפליקציה spotify יש את גודל החבילות הממוצע הגבוה ביותר כנראה משום שהאפליקציה מעבירה מקטעים של יחסית גדולים של אודיו בכל פעם כדי למלא את ה buffer
- לאפליקציה chrome יש את גודל החבילות הממוצע הקטן ביותר כנראה משום שכרום מעביר מידע במהירות בכמויות קטנות
- בכלליות ניתן לראות שלאפליקציות spotify, discord, youtube שמעבירות אודיו ווידאו בזמן אמת יש גודל ממוצע של חבילות שגדול משל הדפדפנים firefox ו- chrome
- הבדל שניתן לראות בין הדפדפנים firefox ו- chrome, הוא שהדפדפן chrome כמעט ולא משתמש ב http3 ומשתמש יותר ב http2 לעומת ההפך ב firefox למרות שבהקלטה בשניהם גלשו לאותם אתרים בדיוק
- לא היה הבדל משמעותי בין זמני ההגעה של 2 חבילות עוקבות בין האפליקציות השונות

## חלק 4

בחלק זה נצטרך לדמות תוקף שמטרתו לנסות לגלות באילו אפליקציות השתמש הקורבן כדי לעשות זאת תחילה נווסג את התכונות העיקריות של כל אפליקציה שבדקנו עד כה:
chrome – משתמש בעיקר בפרוטוקלים QUICK ו TCP , חבילות קטנות יחסית, מספר גדול של חבילות מתקבל כאשר מנסים לטעון דף אינטרנט חדש, פורטים עיקריים שיהיו בשימוש 80 ו- 443 לגלישה ברשת

Firefox – מאוד דומה לכרום ההבדלים שנראה מהגרפים הם שגודל החבילת של firefox נוטה להיות גדול יותר משל כרום אך לא ברמה שאפשר להבדיל, בנוסף ב firefox היה שימוש נרחב יותר בפרוטוקול http3 לעומת כרום שהשתמש יותר ב http2

Youtube – שימוש עיקרי בפרוטוקול UDP, גודל חבילות גדול יחסית, מספר גדול של חבילות נשלחות בבת אחת כל כמה זמן כדי למלא את ה buffer

Spotify – דומה מאוד ליוטיוב אך משתמש בעיקר בפרוטוקולים QUICK ו TCP

Discord – משתמש בעיקר בפרוטוקול UDP, גודל חבילות גדול יחסית, החבילות נשלחות באופן יציב כדי להציג אודיו ווידאו בזמן אמת

**מקרה ראשון**

לתוקף יש גישה לגודל החבילות, זמני השליחה, והצפנה של ה flow ID:
במידה והתוקף יכול לפענח את המידע ב flow ID הוא יכול יחסית בקלות לזהות באיזו אפליקציה הקורבן משתמש תחילה אם ישנו שימוש רחב בפרוטוקול UDP כנראה שהקורבן משתמש ביוטיוב או בדיסקורד, אם כן ניתן לבחון את קצב קבלת החבילות במידה וחבילות מתקבלות בקצב אחיד ויציב יחסית כנראה שהקורבן משתמש באפליקציה כמו דיסקורד להעברה וקבלה של אודיו ווידאו באופן קבוע ובזמן אמת, במידה וקצב קבלת החבילות אינו אחיד וניתן לראות שמתקבל רצף גדול של חבילות כל כמה שניות ניתן להסיק שהקורבן משתמש ביוטיוב כדי לקבל אודיו ווידאו אבל החבילות מתקבלות במרווחים כדי למלא את ה buffer כל פעם.

במידה והשימש העיקרי הוא בפרוטוקולים Quick ו TCP נשווה בין גודל החבילות, אם גודל החבילות גדול יחסית כנראה שמדובר ב Spotify כיוון שמצאנו שהגודל הממצוע של חבילות הוא גדול, במידה וגודל החבילות קטן יחסית נוכל להסיק שהקורבן משתמש ב chrome או ב firefox כדי לגלוש ברשת, קשה מאוד להבדיל בין שני הדפדפנים אך אם רואים שימוש לעיתים קרובות בפרוטוקול http3 לפי המידע שאספנו ניתן להסיק שמדובר בדפדפן firefox , לעומת זאת אם נראה שימוש לעיתים קרובות יחסית בפרוטוקול http2 לפי המידע שאספנו ניתן להסיק שמדובר בדפדפן chrome

**מקרה שני**

לתוקף יש גישה לגודל החבילות וזמני השליחה שלהן:

במקרה זה יהיה לתוקף ממש קשה לזהות באיזו אפליקציה הקורבן משתמש, התוקף יכול לנסות לזהות האם הקורבן משתמש באפליקציות שמשמשות לגלישה ברשת (דפדפנים) לפי גודל החבילות הקטן יחסית, ולבין אפליקציות שמשמשות לקבלה של אודיו ווידאו עקב גודל החבילות הגדול יחסית שלהן, אבל כנראה שהתוקף לא יוכל לזהות בוודאות באיזה אפליקציה בדיוק מדובר.